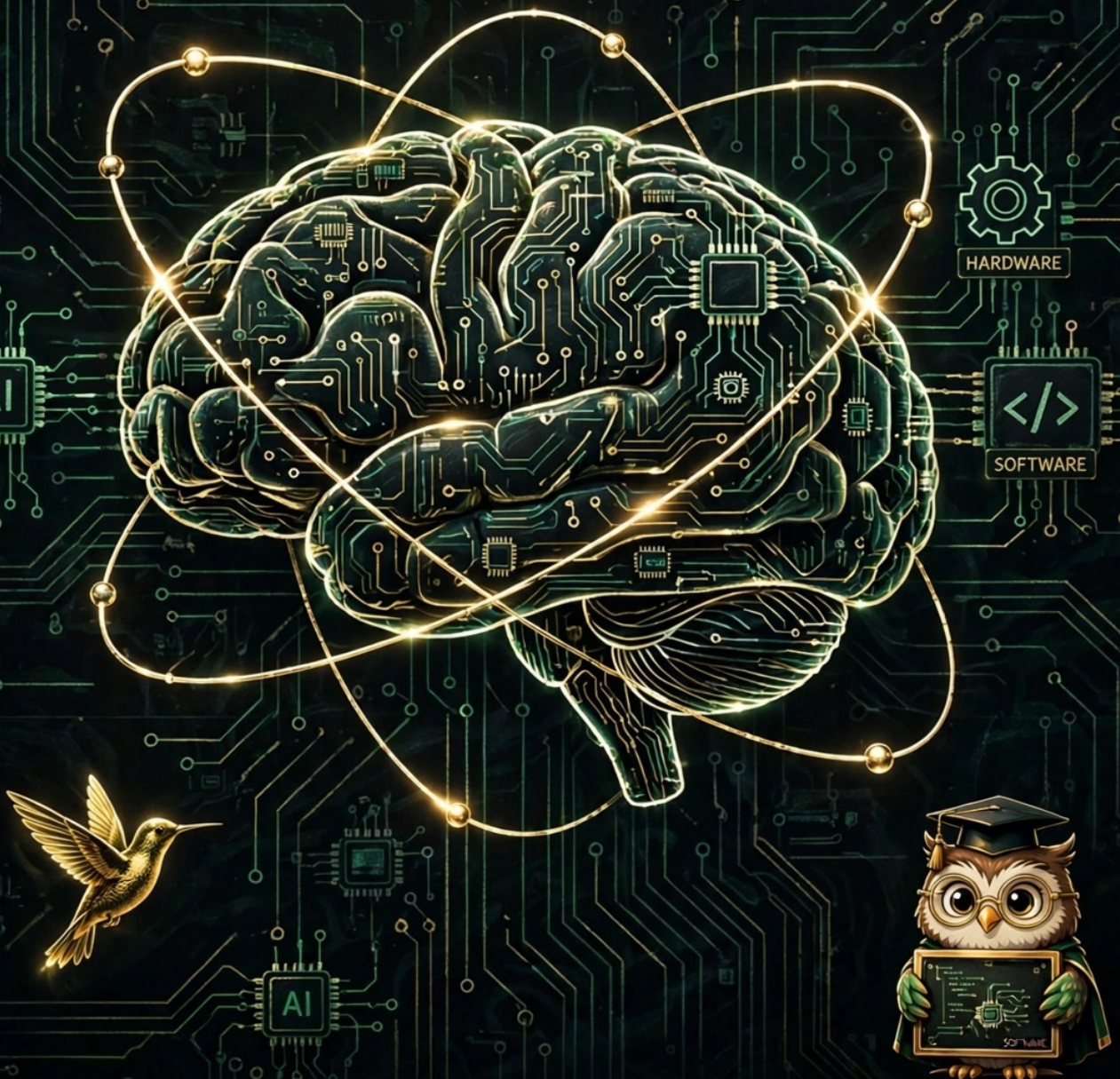


LUCAS EGA ACADEMY

Tecnología Fusión



Curso Completo de Tecnología Fusión

Autor: Esteban

lucasegaacademy.com

*Guía completa con explicaciones extensas por unidad,
prácticas y examen final.*

CRÉDITOS

Curso Completo de Tecnología Fusión

© 2026 Esteban Gutiérrez

Colibrí Editorial EGA

Hecho en México

Primera edición digital · septiembre de 2026

Sin ISBN

PRÓLOGO

Bienvenido a "Curso Completo de Tecnología Fusión", el libro número 22 de Lucas EGA Academy (lucasegaacademy.com). Si llegaste hasta aquí, es porque ya recorriste un camino largo: 21 libros anteriores te llevaron desde los fundamentos de la programación hasta la nube, la seguridad y el desarrollo móvil. Este libro es diferente a todos los demás. Aquí no vas a aprender una sola tecnología, sino a combinar varias para crear cosas que ninguna de ellas podría lograr por sí sola. Eso es la fusión tecnológica: el arte de mezclar piezas para construir algo más grande.

Piensa en un cocinero. Un cocinero no se vuelve bueno dominando un solo ingrediente: se vuelve bueno combinando ingredientes. La sal por sí sola no es un platillo, ni el jitomate, ni la carne. Pero juntos, con técnica y creatividad, se convierten en algo extraordinario. La tecnología funciona igual. La inteligencia artificial por sí sola es impresionante, pero cuando la combinas con sensores del internet de las cosas y la potencia de la nube, obtienes un invernadero que se riega solo, un sistema que detecta intrusos en tu casa o una estación meteorológica que predice el clima de tu colonia. Este libro te enseña a ser ese cocinero tecnológico.

Por qué este libro existe

Vivimos en la época de mayor cambio tecnológico de la historia. Hace veinte años, un teléfono servía para llamar y mandar mensajes de texto. Hoy, ese mismo aparato reconoce tu cara, traduce conversaciones en tiempo real, mide tu ritmo cardíaco y te dice la mejor ruta para evitar el tráfico. Nada de eso es producto de una sola invención: es la fusión de cámaras, sensores, inteligencia artificial, redes de alta velocidad y computación en la nube trabajando juntas.

El problema es que la mayoría de los cursos enseñan cada tecnología por separado, en su propia cajita. Aprendes Python en un curso, electrónica en otro, nube en otro más. Y cuando quieres construir algo real, descubres que nadie te enseñó a conectar las cajitas entre sí. Este libro cierra esa brecha. Cada unidad te presenta una tecnología emergente con la profundidad suficiente para usarla de verdad, pero siempre con la mirada puesta en la combinación: cómo se conecta con las demás, qué proyectos nacen de mezclarlas y qué errores evitar cuando lo intentas.

No necesitas ser un experto en programación para aprovechar este libro. Si completaste los libros anteriores de la academia, tienes más que suficiente. Y si llegaste aquí directamente, no

te preocupes: cada unidad empieza desde cero y avanza paso a paso, con analogías de la vida cotidiana, ejemplos comentados línea por línea y prácticas con soluciones explicadas.

Cómo está organizado el libro

El libro tiene 10 unidades y un examen final. Las unidades agrupan 30 clases en bloques de tres, siguiendo el temario del curso en video de lucasegaacademy.com:

- Unidad 1: Panorama de las tecnologías emergentes. El mapa completo del territorio: qué viene, por qué importa y cómo prepararte.
- Unidad 2: Inteligencia artificial aplicada. Cómo usar la IA hoy mismo para trabajar, estudiar y crear, sin necesidad de un doctorado.
- Unidad 3: Internet de las cosas (IoT). Sensores, microcontroladores y la magia de conectar el mundo físico con el digital.
- Unidad 4: Nube y edge computing. Dónde viven tus datos y tus programas cuando no están en tu computadora.
- Unidad 5: Ciberseguridad moderna. Cómo proteger todo lo que vas a construir (y tu vida digital en general).
- Unidad 6: Impresión 3D y fabricación digital. Cómo materializar tus ideas en objetos físicos.
- Unidad 7: Energías renovables y tecnología. Cómo alimentar tus proyectos con el sol y el viento.
- Unidad 8: Automatización del hogar inteligente. Convierte tu casa en tu primer gran proyecto fusión.
- Unidad 9: Robótica y drones. Máquinas que se mueven, ven y deciden.
- Unidad 10: Proyectos que combinan varias tecnologías. Cinco proyectos completos, de la idea al prototipo funcional.

Cada unidad sigue la misma estructura para que siempre sepas dónde estás:

1. Introducción extensa: el contexto, por qué importa el tema y qué vas a lograr.
2. Explicación detallada: los conceptos a fondo, con analogías, errores comunes y buenas prácticas.
3. Ejemplos paso a paso: código y configuraciones reales, comentados línea por línea.
4. Práctica: ejercicios con soluciones explicadas para que compruebes lo aprendido.

5. Resumen de la unidad: lo esencial en pocas líneas, ideal para repasar.

Al final encontrarás el examen final: 50 preguntas (20 de opción múltiple, 15 de verdadero o falso y 15 abiertas) con todas las respuestas explicadas. Úsalo para medir tu avance de verdad, no para memorizar.

Cómo estudiar este libro

Te recomiendo un método sencillo de tres pasos por unidad:

Primero, lee la introducción y la explicación detallada sin prisa, como si leyeras una historia. Subraya o anota las ideas que te sorprendan. Las analogías están ahí para que los conceptos se queden grabados; si una analogía no te hace clic, inventa la tuya propia con algo de tu vida diaria. Eso funciona mejor que releer.

Segundo, reproduce los ejemplos paso a paso en tu propia computadora o con tus propios componentes. La tecnología se aprende con las manos, no con los ojos. Si un ejemplo usa un sensor que no tienes, no te detengas: adapta el código, simula los datos o simplemente léelo con atención imaginando cada paso. Lo importante es entender el flujo completo: qué entra, qué se procesa y qué sale.

Tercero, haz la práctica sin mirar las soluciones primero. Es normal equivocarse; los errores son la forma más rápida de aprender. Cuando revises la solución explicada, compara tu razonamiento con el del libro, no solo el resultado final. Si fallaste, pregúntate en qué paso te desviaste: ahí está la lección real.

Un consejo importante: no intentes comprar todo el equipo de todas las unidades de golpe. Empieza con lo que tengas (una computadora e internet bastan para las primeras unidades) y ve adquiriendo componentes baratos conforme avances: un ESP32, algunos sensores y una placa de pruebas cuestan menos que una salida al cine y te sirven para media docena de proyectos. En cada unidad te digo exactamente qué necesitas y qué puedes simular.

Lo que vas a lograr

Al terminar este libro serás capaz de planear y construir proyectos que combinan inteligencia artificial, internet de las cosas, nube, impresión 3D, energías renovables y automatización. Tendrás un portafolio de proyectos reales para mostrar, una base sólida de ciberseguridad para protegerlos y la metodología para seguir aprendiendo las tecnologías que vengan en el futuro, porque vendrán muchas.

Más que eso: vas a cambiar tu forma de ver la tecnología. Dejarás de ver aparatos aislados y empezarás a ver sistemas. Cuando veas un semáforo inteligente, un reloj que mide tu sueño o un dron repartiendo paquetes, ya no pensarás "qué magia", sino "ya sé cómo funciona eso por dentro, y podría construir una versión sencilla". Esa es la verdadera meta de este libro.

Así que ponte cómodo, abre tu editor de código y tu caja de herramientas. La fusión empieza ahora. ¡Que lo disfrutes!

UNIDAD 1: Panorama de las tecnologías emergentes

Bienvenido a la Unidad 1 de tu Curso Completo de Tecnología Fusión. Antes de que te pongas a programar, a soldar cables o a entrenar tu primer modelo de inteligencia artificial, necesitas algo más importante que cualquier herramienta: el mapa del territorio. Esta unidad es ese mapa. Aquí vas a entender qué son las tecnologías emergentes, por qué todo el mundo habla de ellas, cómo se conectan entre sí para formar eso que llamamos fusión tecnológica y hacia dónde apunta la próxima década. Y lo más valioso de todo: vas a salir de aquí sabiendo cómo mantenerte actualizado sin sentir que te atropella el tren del cambio.

Te lo digo desde el inicio y con toda franqueza: la tecnología avanza tan rápido que es imposible sabérsela toda. Ni los ingenieros más veteranos de Silicon Valley la conocen completa. La diferencia entre un profesional estancado y uno que crece no es la cantidad de datos que tiene en la cabeza, sino su capacidad de orientarse, de detectar qué es ruido y qué es señal, y de aprender rápido cuando algo nuevo aparece. Esa capacidad se entrena, y esta unidad es tu primera sesión de entrenamiento.

Así que respira, ponte cómodo, y vamos a recorrer juntos el panorama. Para cuando termines esta unidad vas a poder explicarle a cualquier persona -tu abuela, tu jefe, tu compa de la escuela- qué es la cuarta revolución industrial, qué significa que las tecnologías converjan y cuáles son las diez tecnologías que van a marcar tu vida profesional en los próximos diez años. Eso ya te pone por delante del noventa por ciento de la gente.

¿Por qué este libro y por qué ahora?

Mira a tu alrededor. Es muy probable que estés leyendo esto en un teléfono que cabe en tu mano y que es millones de veces más potente que las computadoras que llevaron al hombre a la Luna. Ese teléfono se conecta sin cables a una red global, entiende tu voz, reconoce tu cara, te sugiere qué ver, qué comprar y hasta a quién escribirle. Y eso que parece normal hoy habría parecido magia hace apenas veinte años.

Vivimos un momento único en la historia: por primera vez, varias tecnologías poderosas -la inteligencia artificial, el internet de las cosas, la computación en la nube, la robótica, la biotecnología- no solo avanzan cada una por su lado, sino que se combinan y se potencian entre sí. Un coche autónomo no es "una tecnología": es inteligencia artificial más sensores más conectividad 5G más computación en la nube más robótica, todo fundido en un solo producto. Un diagnóstico médico hecho por computadora no es "un programa": es IA

entrenada con datos genómicos procesados en la nube y entregados al médico a través de una app en su celular. Esa combinación es la fusión tecnológica, y es el corazón de este libro.

Ahora bien, ¿por qué un libro y no un curso en video o una serie de tutoriales sueltos? Porque la fusión tecnológica exige pensamiento de sistema. Los tutoriales sueltos te enseñan trucos: cómo conectar un sensor, cómo llamar a una API. Eso es útil, pero sin el panorama completo terminas sabiendo muchas piezas y ninguna imagen. Este libro te da las dos cosas: el mapa (las primeras unidades) y las piezas (las unidades prácticas). Y lo hace con un orden pensado para que avances paso a paso, sin saltos de fe ni huecos que te dejen atorado a las tres de la mañana.

Te habla tu profesor, Lucas Ega. En Lucas EGA Academy (lucasegaacademy.com) creemos que la tecnología se aprende haciendo, pero también entendiendo. Por eso cada unidad de este libro combina explicación clara, ejemplos reales que puedes replicar y práctica guiada. No necesitas ser un genio de las matemáticas ni haber programado antes. Necesitas curiosidad, constancia y ganas de construir. Si traes eso, el resto lo ponemos nosotros.

¿Qué son exactamente las tecnologías emergentes?

Vamos a definirlo sin vueltas y sin palabras rimbombantes. Una tecnología emergente es una tecnología que está naciendo o creciendo muy rápido, que todavía no es parte normal de la vida cotidiana para la mayoría de la gente, pero que tiene el potencial de cambiar radicalmente cómo vivimos, trabajamos y nos relacionamos. Fíjate en las tres partes de la definición: está en desarrollo (no es una curiosidad de laboratorio nada más), crece aceleradamente (cada año es notablemente mejor o más barata que el anterior) y tiene potencial transformador (no es solo una mejora chiquita, es un cambio de reglas del juego).

Piénsalo con un ejemplo que ya viviste: el teléfono inteligente. En 2007 el primer iPhone era una tecnología emergente: caro, limitado, cosa de entusiastas. Para 2015 ya no era emergente, era la norma: todo el mundo tenía uno y las apps móviles eran una industria gigante. Las tecnologías emergentes de hoy son los iPhones de 2007: la inteligencia artificial generativa, la computación cuántica, la edición genética. Algunas se volverán tan normales como el agua; otras se quedarán en el camino. Tu trabajo como profesional del futuro no es adivinar con bola de cristal cuál triunfará, sino entender el patrón de cómo nacen, crecen y se adoptan estas tecnologías, para que puedas subirte a la ola correcta en el momento correcto.

Hay una curva famosa que describe este ciclo y que vas a ver citada en todas partes: el ciclo de sobreexpectación de Gartner. Funciona así: una tecnología nueva aparece y los medios la inflan hasta el cielo (pico de expectativas infladas); luego la realidad golpea, los proyectos fallan y llega la desilusión (abismo de la desilusión); después, los que aguantaron aprenden a

usarla bien y la tecnología madura (rampa de la consolidación); al final se vuelve productiva y normal (meseta de productividad). La inteligencia artificial pasó por su invierno en los años ochenta, por su pico de hype varias veces y hoy está en plena rampa de consolidación. La computación cuántica, en cambio, todavía anda entre el pico y el abismo. Conocer esta curva te ahorra dos errores carísimos: emocionarte de más con lo nuevo y despreciar lo que está en el abismo pero va a resurgir.

Otro concepto clave: la diferencia entre tecnología emergente y tecnología disruptiva. Toda tecnología disruptiva fue emergente alguna vez, pero no toda tecnología emergente logra ser disruptiva. Disruptiva significa que desplaza a lo que existía antes: el correo electrónico desplazó a la carta, la fotografía digital desplazó al rollo, el streaming desplazó al DVD. Como estudiante de fusión tecnológica te conviene enfocarte en las tecnologías con potencial disruptivo real, no en las modas pasajeras. En esta unidad te presento diez candidatas serias, con argumentos de por qué cada una merece tu atención.

Cómo vas a navegar esta unidad (y este libro)

Esta unidad está organizada para que la leas de corrido, pero también para que la consultes después como referencia. Primero viene la explicación detallada: ahí desmenuzamos la cuarta revolución industrial, la convergencia tecnológica y las diez tecnologías de la década, con analogías de la vida cotidiana para que nada se quede en abstracto. También encontrarás una sección honesta sobre los errores más comunes de los principiantes -los cometí yo, los cometen todos- y las buenas prácticas que te van a ahorrar meses de frustración.

Después vienen los ejemplos paso a paso: tres proyectos reales y replicables que te muestran la fusión en acción. Vas a ver comandos de terminal, archivos de configuración en JSON y YAML, y el razonamiento completo detrás de cada decisión. No te preocupes si al principio no entiendes cada línea: el objetivo aquí es que veas cómo se ve un proyecto de fusión tecnológica de verdad, no que lo domines hoy.

Luego está la sección de práctica: seis ejercicios con su solución explicada paso a paso. Hazlos de verdad, con lápiz y papel o en tu computadora. La diferencia entre leer sobre tecnología y saber tecnología está en la práctica, y estos ejercicios están diseñados para que el conocimiento se te quede grabado.

Cerramos con el resumen de la unidad: los puntos clave en forma compacta, perfectos para repasar antes de un examen o antes de explicarle el tema a alguien más. Porque aquí va un secreto de aprendizaje: si puedes explicar un tema con tus propias palabras, es que de verdad lo entendiste.

Un último consejo antes de empezar: lleva un cuaderno de notas para este libro, físico o digital. Anota las palabras que no conozcas, las ideas que te vuelen la cabeza y las preguntas que te surjan. Al final del libro ese cuaderno va a valer oro, porque será el registro de cómo tu mente se transformó de "usuario de tecnología" a "creador de tecnología". Vamos a ello.

Explicación detallada

La cuarta revolución industrial: de la máquina de vapor al dato

Para entender dónde estamos parados, hay que mirar hacia atrás. La historia de la tecnología moderna se cuenta en revoluciones industriales, y cada una cambió el mundo más que la anterior. La primera revolución industrial, a finales del siglo XVIII, fue la de la máquina de vapor y la mecanización: por primera vez las máquinas hacían el trabajo físico de los músculos humanos y los animales. Las fábricas nacieron, las ciudades crecieron y la economía dejó de depender del campo. La segunda, a finales del siglo XIX, trajo la electricidad, el acero barato y la producción en masa: la línea de ensamble, el teléfono, el automóvil. La tercera, en la segunda mitad del siglo XX, fue la revolución digital: las computadoras, internet, la automatización de la información. Cada revolución tardó décadas en desplegarse por completo, y cada una creó empleos que nadie había imaginado antes: nadie en 1750 soñaba con ser "community manager", igual que nadie en 1990 soñaba con ser "entrenador de modelos de IA".

La cuarta revolución industrial es la que estamos viviendo ahora, y su materia prima ya no es el vapor, ni la electricidad, ni siquiera la información a secas: es el dato inteligente. Se caracteriza por la fusión de los mundos físico, digital y biológico. Un sensor en un campo de cultivo mide la humedad de la tierra (mundo físico), manda el dato a la nube (mundo digital), un algoritmo decide cuánta agua necesita cada planta y un dron la riega con precisión (de vuelta al mundo físico), mientras la semilla misma fue diseñada con biotecnología (mundo biológico). Todo eso en un solo sistema. Esa es la cuarta revolución: no una tecnología, sino la conexión inteligente de todas.

¿Qué la hace distinta de la tercera? Tres cosas. Primera: la velocidad. La tercera revolución tardó unos cincuenta años en transformar la economía global; la cuarta lo está haciendo en menos de veinte. Segunda: el alcance. No está cambiando solo la industria, sino la biología (edición genética), la mente (interfaces cerebro-computadora) y hasta la noción de trabajo. Tercera: el impacto sistémico. Las tecnologías de esta revolución no mejoran procesos existentes: crean procesos imposibles antes. No es que el taxi sea más barato con una app; es que el concepto de "tener coche propio" empieza a tambalearse.

Aquí va una analogía cotidiana para que se te grabe: imagina que las revoluciones industriales son formas de cocinar. La primera te dio el fuego controlado (la estufa). La segunda te dio la estufa eléctrica con temporizador. La tercera te dio el horno de microondas con programas automáticos. La cuarta te dio una cocina inteligente que sabe qué hay en tu refri, te sugiere recetas según tu salud, pide sola los ingredientes que faltan y cocina mientras tú trabajas. Cada salto no solo fue "más rápido": cambió tu relación con la cocina. Eso mismo está pasando con la tecnología y tu vida.

Un error común de principiante es pensar que la cuarta revolución industrial es "cosa del futuro" o "cosa de países ricos". No. Si tienes un teléfono con datos móviles, ya vives dentro de ella. El agricultor en Oaxaca que consulta el pronóstico del clima en su celular y vende su cosecha por WhatsApp está participando en la cuarta revolución industrial tanto como el ingeniero de un laboratorio en Alemania. La diferencia no es de acceso a la magia, sino de comprensión: quien entiende el sistema puede construir sobre él; quien no, solo lo consume. Este libro existe para ponerte del lado de los que construyen.

Convergencia tecnológica: cuando las tecnologías se juntan, se multiplican

Aquí llegamos al concepto central de todo el libro: la convergencia tecnológica. En palabras simples, convergencia es lo que pasa cuando varias tecnologías distintas se combinan para crear algo que ninguna podría lograr sola. Y aquí está la parte emocionante: cuando se combinan, el resultado no es la suma, es la multiplicación. Dos tecnologías juntas no dan el doble de poder: dan diez veces más, porque cada una resuelve las limitaciones de la otra.

El ejemplo clásico es el teléfono inteligente. Por separado existían: la telefonía móvil, las pantallas táctiles, el GPS, las cámaras digitales, internet inalámbrico y los procesadores potentes. Ninguna de esas cosas, sola, habría cambiado el mundo. Pero alguien las metió todas en un aparatito del tamaño de tu mano y nació una plataforma sobre la cual se construyeron industrias enteras: las apps, la economía colaborativa, las redes sociales móviles, la banca digital. El iPhone no inventó casi nada; lo que inventó fue la combinación.

Ahora mira los ejemplos de hoy con ojos de convergencia. Un asistente de voz como Alexa es: micrófonos y bocinas (hardware barato) más reconocimiento de voz por IA más procesamiento de lenguaje natural más computación en la nube más internet de banda ancha en tu casa. Quítale cualquiera de esas cinco piezas y el producto se cae. Un coche autónomo es: cámaras y sensores LiDAR (IoT) más redes 5G de baja latencia más IA de visión por computadora más mapas actualizados en la nube más robótica de precisión para mover el volante. La medicina personalizada es: secuenciación genética barata (biotecnología) más IA que encuentra patrones en millones de genomas más nube para procesarlos más apps que le entregan el resultado al paciente.

¿Ves el patrón? La fórmula de la fusión tecnológica es casi siempre la misma: **SENSORES** que capturan el mundo físico (IoT), **CONECTIVIDAD** que lleva los datos a donde se procesan (5G, nube), **INTELIGENCIA** que decide qué significan esos datos (IA) y **ACTUADORES** que devuelven una acción al mundo físico (robótica). Sensor, red, cerebro, músculo. Memoriza ese ciclo: es el esqueleto de casi todos los proyectos que vas a construir en este libro y en tu carrera.

Analogía cotidiana: piensa en un equipo de fútbol. Un portero buenísimo solo no gana partidos. Pero un buen portero más una defensa sólida más un medio creativo más un delantero letal forman un equipo campeón. Cada jugador potencia a los demás: el delantero mete goles porque el medio le pone pases, y el medio se atreve a atacar porque la defensa lo respalda. Las tecnologías son igual: la IA es el delantero estrella que todos admiran, pero sin los datos del IoT (el medio que le pasa el balón) y sin la nube (la cancha donde juega), no mete ningún gol.

Y aquí va la buena práctica más importante de esta unidad: cuando veas una tecnología nueva que te emocione, no preguntes "¿qué hace?", pregunta "¿con qué se combina?". Esa segunda pregunta te revela el verdadero potencial. La impresión 3D sola es un juguete caro para hacer llaveritos; la impresión 3D combinada con IA de diseño generativo y nuevos materiales es una revolución en la manufactura que permite crear piezas de avión más ligeras y resistentes que las tradicionales. El blockchain solo es una base de datos lenta y cara; combinado con IoT y contratos inteligentes es la columna vertebral de cadenas de suministro transparentes. Entrena ese reflejo de buscar combinaciones y vas a ver oportunidades donde otros solo ven gadgets.

Las 10 tecnologías que marcarán la década (parte 1): inteligencia, conexión y cómputo

Llegó el momento de conocer a las protagonistas. Estas son diez tecnologías que, por separado y sobre todo combinadas, van a definir los próximos diez años de tu vida profesional. No son las únicas que existen, pero son las que tienen la combinación ganadora: madurez suficiente para usarse hoy y potencial suficiente para seguir creciendo mañana. Estúdialas como si fueran personajes de una serie, porque en los próximos capítulos del libro te las vas a encontrar una y otra vez.

1. **Inteligencia artificial (IA).** Es la estrella del momento y con razón: son sistemas que aprenden de datos y realizan tareas que antes requerían inteligencia humana, como entender lenguaje, reconocer imágenes, predecir comportamientos o tomar decisiones. Lo que explotó en esta década es la IA generativa, capaz de crear texto, imágenes, código, música y video. Para ti como estudiante, la IA es doblemente importante: es una

tecnología que vas a usar (como asistente) y una tecnología que vas a construir (como desarrollador). Analogía: la IA es como un aprendiz súper dotado que leyó toda la biblioteca del mundo; no entiende como tú, pero recuerda patrones a una escala imposible para un humano. Error común: creer que la IA "piensa" o "sabe" como una persona; en realidad predice patrones estadísticos, y por eso a veces se equivoca con total seguridad. Buena práctica: úsala como copiloto, no como piloto automático; verifica siempre lo importante.

2. Conectividad 5G y 6G. El 5G no es "internet más rápido en tu celular", aunque eso también. Sus tres superpoderes son: velocidad altísima, latencia ultrabaja (el tiempo de respuesta, casi instantáneo) y capacidad para conectar millones de dispositivos por kilómetro cuadrado. Eso es lo que permite coches autónomos que reaccionan en milisegundos, cirugías a distancia y fábricas llenas de robots coordinados. El 6G, que llegará hacia 2030, promete integrar la IA directamente en la red. Analogía: si el 4G era una carretera de dos carriles donde los coches hacían fila, el 5G es una autopista de veinte carriles con carriles exclusivos para ambulancias. Sin esta autopista, el IoT masivo y los vehículos autónomos simplemente no funcionan.
3. Computación cuántica. Las computadoras normales trabajan con bits (ceros y unos); las cuánticas usan cúbits, que aprovechan fenómenos de la física cuántica para explorar muchísimas posibilidades a la vez. Para ciertas tareas -descubrir medicamentos, optimizar rutas de miles de camiones, romper o crear cifrados- una computadora cuántica podría hacer en minutos lo que a todas las supercomputadoras del mundo les tomaría miles de años. Hoy está en fase temprana: son máquinas delicadísimas que necesitan frío cercano al cero absoluto. Pero los gigantes tecnológicos invierten miles de millones porque quien domine la cuántica dominará la próxima era del cómputo. Analogía: si una computadora normal es una persona que prueba llaves una por una para abrir una puerta, la cuántica es como probar miles de llaves al mismo tiempo. Error común: pensar que la cuántica va a reemplazar tu laptop; no, será una herramienta especializada para problemas específicos, como un microscopio no reemplaza tus ojos.
4. Biotecnología y edición genética. Es la tecnología aplicada a la vida: modificar organismos, leer y escribir ADN, crear medicamentos a la medida de tu genética. La herramienta estrella es CRISPR, unas "tijeras moleculares" que permiten editar genes con precisión y bajo costo. Ya se usa para curar enfermedades genéticas que antes eran sentencia de muerte, para crear cultivos resistentes a sequías y para producir materiales con bacterias. La convergencia aquí es brutal: IA que diseña proteínas, robots que hacen experimentos automáticamente e impresión 3D de tejidos. Analogía: si tu ADN es el manual de instrucciones para construirte a ti, la biotecnología moderna es por fin poder editar ese manual con un procesador de textos en lugar de hacerlo a ciegas. Dilema ético

que debes conocer: editar genes de bebés abre debates profundos; la tecnología avanza más rápido que las leyes, y como profesional tendrás que pensar en estas cosas.

5. Blockchain y Web3. Es un registro digital descentralizado: una base de datos que no vive en una sola computadora de una empresa, sino repartida en miles de computadoras que se ponen de acuerdo. Eso la hace muy difícil de falsificar o censurar. Sus usos serios van más allá de las criptomonedas: contratos inteligentes que se ejecutan solos cuando se cumplen condiciones, trazabilidad de alimentos y medicinas (saber exactamente de dónde vino lo que comes), identidad digital que tú controlas y votación electrónica auditable. Analogía: imagina un libro de contabilidad público donde cada página está pegada con pegamento a la anterior; si alguien intenta arrancar o alterar una página, se nota de inmediato porque todas las copias del libro en el mundo dejarían de coincidir. Error común: confundir blockchain con Bitcoin y con "hacerse rico rápido"; la tecnología del registro distribuido es seria y valiosa aunque las modas especulativas pasen.

Las 10 tecnologías que marcarán la década (parte 2): la frontera física y digital

6. Realidad extendida (XR): virtual, aumentada y mixta. La realidad virtual (VR) te mete en un mundo digital con un casco; la realidad aumentada (AR) pone objetos digitales sobre el mundo real a través de tu teléfono o lentes; la mixta (MR) mezcla ambas para que lo digital y lo real interactúen. Sus aplicaciones serias ya están aquí: cirujanos que practican operaciones en VR antes de tocar a un paciente real, técnicos que reparan máquinas siguiendo instrucciones flotantes en AR, arquitectos que caminan dentro de sus edificios antes de construirlos. Analogía: la VR es como meterte dentro de la película; la AR es como si los personajes de la película salieran a tu sala. En esta década los cascos se harán más ligeros, baratos y potentes, y el trabajo y la educación a distancia van a cambiar por completo.
7. Edge computing (cómputo en el borde). Tradicionalmente los datos viajan hasta un centro de datos lejano (la nube) para procesarse y la respuesta viaja de vuelta. El edge computing procesa los datos cerca de donde se generan: en el propio sensor, en la cámara, en la antena de la esquina. ¿Por qué importa? Porque en un coche autónomo no puedes esperar medio segundo a que un servidor lejano te diga "frena": necesitas decidir en milisegundos. Además ahorra ancho de banda y protege la privacidad (tus datos no tienen que viajar). Analogía: es la diferencia entre preguntarle a tu vecino (edge) y mandarle una carta a otro país (nube) cada vez que necesitas algo. La tendencia ganadora es el modelo híbrido: el borde decide lo urgente y la nube hace el análisis profundo. Cuando diseñes sistemas IoT en este libro, esta decisión -qué se procesa dónde- será una de las más importantes.

8. Impresión 3D y manufactura aditiva. En lugar de cortar material para darle forma (como un escultor), la impresión 3D construye objetos capa por capa desde un diseño digital. Ya se imprimen prótesis a la medida de cada paciente, piezas de repuesto en barcos en altamar, casas enteras en días y hasta comida. La convergencia con IA permite el diseño generativo: le dices al programa "quiero una pieza que soporte este peso con el menor material posible" y la IA diseña formas orgánicas que ningún humano habría imaginado. Analogía: es como pasar de esculpir en mármol (quitar lo que sobra) a construir con bloques de Lego (poner solo lo necesario). Error común: pensar que sirve para todo; hoy es ideal para piezas personalizadas, prototipos y series cortas, no para producir millones de botellas de plástico baratas.
9. Energías renovables y almacenamiento. La solar y la eólica ya son las fuentes de electricidad más baratas de la historia en muchos países, y su precio sigue bajando. Pero el verdadero cuello de botella era guardar esa energía para la noche o los días sin viento: ahí entran las baterías de nueva generación, el hidrógeno verde y las redes eléctricas inteligentes que usan IA para balancear oferta y demanda en tiempo real. Esta tecnología es la base silenciosa de todas las demás: los centros de datos que entrenan la IA consumen cantidades brutales de electricidad, y sin energía limpia y barata la revolución digital se frena. Analogía: las renovables son como sembrar tu propia comida en lugar de comprarla: al inicio cuesta el huerto, pero después comes casi gratis. Como profesional, entiende esto: la energía es el costo oculto de la tecnología, y quien la domine tendrá ventaja.
10. Robótica avanzada y automatización. Los robots dejaron de ser esos brazos torpes de las fábricas de autos: hoy hay robots que corren, saltan, hacen piruetas, robots blandos inspirados en pulpos, enjambres de drones que se coordinan solos y exoesqueletos que devuelven la movilidad a personas con lesiones. La fusión con IA les dio ojos (visión por computadora), cerebro (toma de decisiones) y cada vez más destreza manual. En esta década los verás en almacenes, hospitales, restaurantes, construcción y en tu casa. Analogía: si la IA es el cerebro, la robótica es el cuerpo; un cerebro sin cuerpo solo puede aconsejar, pero con cuerpo puede actuar en el mundo. Dato para tu carrera: no se trata de que "los robots nos quiten el trabajo", sino de que los trabajos se transforman: alguien tiene que diseñar, programar, mantener y supervisarse a esos robots, y ese alguien puedes ser tú.

Fíjate en algo hermoso: estas diez tecnologías no son diez islas, son diez piezas de un rompecabezas. La IA necesita datos del IoT y poder de cómputo de la nube y la cuántica; la robótica necesita IA, 5G y edge; la biotecnología necesita IA y nube; la XR necesita 5G y edge; todo necesita energía. Cuando las estudies, dibuja flechas entre ellas. Ese mapa de conexiones es exactamente lo que significa pensar en fusión tecnológica, y es la habilidad que este libro va a desarrollar en ti capítulo tras capítulo.

Errores comunes de principiantes (y cómo evitarlos)

Nadie nace sabiendo, y en tecnología los errores son parte del aprendizaje. Pero hay errores que te cuestan meses y que puedes evitar si alguien te los advierte a tiempo. Aquí van los más comunes que veo en estudiantes que entran al mundo de las tecnologías emergentes, con su remedio incluido.

Error 1: Querer aprenderlo todo al mismo tiempo. Ves las diez tecnologías de la lista anterior y quieres dominar IA, blockchain, robótica y cuántica este mismo mes. Resultado: aprendes un poquito de todo y nada bien, te frustras y abandonas. Remedio: profundiza en una o dos tecnologías base (te recomiendo programación e IA, porque potencian a todas las demás) y mantén una cultura general del resto. Es como aprender idiomas: domina bien el inglés antes de querer aprender chino, japonés y ruso a la vez.

Error 2: Enamorarse de la herramienta y olvidar el problema. Muchos principiantes eligen la tecnología primero ("quiero hacer algo con blockchain") y después buscan a fuerzas un problema donde encaje. Los profesionales hacen lo contrario: entienden un problema real y luego eligen la tecnología más simple que lo resuelva. Si un problema se resuelve con una hoja de cálculo, no necesita IA ni blockchain. Remedio: ante cada proyecto pregúntate "¿qué problema estoy resolviendo y para quién?"; si no tienes respuesta clara, todavía no empieces a programar.

Error 3: Creer que necesitas el equipo más caro para empezar. "No puedo aprender IA porque no tengo una computadora gamer", "no puedo hacer IoT porque no tengo sensores caros". Falso. Hoy existen simuladores gratuitos, servicios en la nube con capas gratuitas y placas baratas que cuestan menos que una pizza. Remedio: empieza con lo que tienes; la creatividad con recursos limitados es una de las mejores escuelas. Muchos de los ejemplos de este libro corren en cualquier computadora modesta o directo en el navegador.

Error 4: Copiar código sin entenderlo. Con la IA generativa es tentador pedirle que te escriba todo el programa y entregarlo sin leerlo. Funciona una vez, pero a la segunda falla algo, no sabes por qué, y estás atorado sin haber aprendido nada. Remedio: usa la IA como tutora, no como empleada; pídele que te explique cada parte del código que genere y modifica tú mismo

al menos una cosa en cada programa. La regla de oro: si no puedes explicar tu código con tus palabras, no es tu código.

Error 5: Estudiar solo teoría y nunca construir nada. Puedes leer veinte libros sobre natación, pero hasta que no te metes al agua no sabes nadar. En tecnología pasa igual: el conocimiento se fija cuando lo aplicas. Remedio: por cada hora de teoría, dedica al menos una hora a práctica. Termina cada unidad de este libro construyendo algo, aunque sea pequeño y feo. Un proyecto chiquito terminado vale más que tres cursos empezados.

Error 6: Compararte con los expertos y sentirte un fraude. Ves a alguien en YouTube que a los 19 años ya domina cinco tecnologías y piensas "yo nunca voy a llegar". Lo que no ves son sus cinco años de práctica ni sus cientos de errores. Remedio: compárate solo con tu yo de hace seis meses. Si hoy entiendes cosas que antes te parecían chino, vas por buen camino. La constancia le gana al talento casi siempre.

Buenas prácticas para mantenerte actualizado sin quemarte

La tecnología cambia rápido y eso genera ansiedad: el famoso FOMO (miedo a quedarse fuera). Pero mantenerse actualizado no significa leer todo lo que sale cada día; significa tener un sistema. Aquí va un sistema probado que usan muchos profesionales y que puedes adoptar desde hoy.

Primera práctica: la dieta de información. Elige pocas fuentes de calidad y consúmelas con regularidad, en lugar de picotear noticias al azar. Te sugiero: un boletín semanal de tecnología en español, un canal de YouTube educativo de tu área, y la documentación oficial de las herramientas que uses. Treinta minutos al día de lectura enfocada superan a tres horas de scroll ansioso en redes sociales. Calidad sobre cantidad, siempre.

Segunda práctica: aprende en público y en comunidad. Únete a una comunidad: un grupo de tu ciudad, un servidor de Discord, un foro, los comentarios de lucasegaacademy.com. Explica lo que aprendes, pregunta sin pena y ayuda a otros cuando puedas. Enseñar es la forma más rápida de aprender, y las comunidades te avisan de lo importante sin que tengas que buscarlo todo tú solo.

Tercera práctica: el proyecto permanente. Mantén siempre un proyecto personal vivo, algo que te emocione: un robot, una app, un sitio web, un experimento. Cada tecnología nueva que aprendas, pruébala en ese proyecto. Así el aprendizaje tiene contexto y propósito, no es teoría flotando en el aire. Tu proyecto permanente es también tu portafolio: cuando busques trabajo, vale más que cualquier certificado.

Cuarta práctica: fundamentos primero, modas después. Las modas cambian cada año, pero los fundamentos duran décadas: lógica de programación, cómo funciona internet, estadística básica, inglés técnico. Invierte el setenta por ciento de tu tiempo en fundamentos y el treinta en lo nuevo. Quien domina fundamentos aprende cualquier moda en semanas; quien solo persigue modas tiene que empezar de cero cada vez.

Quinta práctica: descansa y desconéctate. Parece consejo de abuela, pero es estrategia profesional: el cerebro consolida el aprendizaje cuando descansas, y las mejores ideas llegan cuando no estás frente a la pantalla. El agotamiento no te hace más productivo, te hace más lento y propenso a errores. Duerme bien, haz ejercicio, ten vida fuera de la tecnología. Los mejores ingenieros que conozco tienen hobbies que nada tienen que ver con computadoras.

El mapa de ruta de este libro: qué vas a construir unidad por unidad

Ya conoces el territorio; ahora mira el camino. Este libro está diseñado como una ruta de aprendizaje progresiva: cada unidad te da herramientas que usarás en la siguiente, y al final habrás construido un proyecto integrador que combina varias tecnologías. Así está organizado:

- Unidades 1 a 3: Fundamentos y panorama. Esta unidad que estás leyendo (el mapa), más una unidad sobre pensamiento computacional y lógica de programación, y otra sobre cómo funciona internet, la nube y los datos. Al terminar esta etapa sabrás leer el mundo tecnológico con ojos de creador.
- Unidades 4 a 7: Las piezas de la fusión. Programación práctica con Python, introducción a la inteligencia artificial aplicada, internet de las cosas con sensores reales y desarrollo de aplicaciones conectadas. Aquí empiezas a ensuciarte las manos: código, circuitos y datos de verdad.
- Unidades 8 a 10: La fusión en acción. Proyectos que combinan todo: un sistema IoT con IA en la nube, automatización y robótica básica, y temas de vanguardia como blockchain aplicado y realidad extendida. Cada proyecto sigue el ciclo sensor-red-cerebro-músculo que aprendiste en esta unidad.
- Unidades 11 y 12: Tu futuro profesional. Proyecto integrador final, cómo armar tu portafolio, cómo presentarte al mundo laboral o emprender, y la guía de aprendizaje continuo para que nunca te estanques.

Cada unidad mantiene la estructura que ya viste aquí: explicación detallada, ejemplos paso a paso, práctica con soluciones y resumen. Si alguna vez te sientes perdido en una unidad avanzada, regresa a esta Unidad 1: el mapa siempre te reubica. Y recuerda que en lucasegaacademy.com encontrarás cursos gratuitos que complementan cada tema del libro, con videos y ejercicios extra.

Perfiles profesionales del futuro: en qué te puedes convertir

Cerremos la explicación con la pregunta que más importa: ¿y esto para qué me sirve a mí? El mercado laboral se está transformando y los perfiles más buscados ya no son los de antes. Aquí van los perfiles que la fusión tecnológica está creando, para que elijas tu rumbo con los ojos abiertos.

- Desarrollador de IA aplicada. No diseña algoritmos desde cero en un laboratorio: toma modelos existentes y los adapta a problemas reales de empresas (atención al cliente, análisis de documentos, predicción de ventas). Es el perfil más demandado hoy y lo seguirá siendo.
- Ingeniero de IoT y sistemas embebidos. Diseña los dispositivos conectados: sensores, actuadores y el software que los hace inteligentes. Trabaja en agricultura, salud, industria y ciudades inteligentes.
- Especialista en nube y edge computing. Construye la infraestructura donde vive todo lo demás: despliega servicios, optimiza costos y decide qué se procesa en el borde y qué en la nube.
- Científico de datos. Convierte montañas de datos en decisiones: limpia, analiza y visualiza información para que las empresas entiendan qué está pasando y qué va a pasar.
- Desarrollador de realidad extendida. Crea experiencias de VR y AR para educación, medicina, industria y entretenimiento; un campo joven con poca competencia todavía.
- Especialista en ciberseguridad. Protege los sistemas de un mundo hiperconectado donde cada sensor es una posible puerta de entrada para atacantes. La demanda supera por mucho a la oferta de profesionales.
- Emprendedor tecnológico. Combina varias de estas habilidades para crear productos y empresas propias. Con las herramientas actuales, una sola persona puede construir lo que antes requería un equipo de veinte.

Lo hermoso es que no tienes que elegir hoy mismo. Este libro te da bases en todas estas áreas para que descubras cuál te apasiona. Y un consejo final de esta sección: el perfil más valioso del futuro no es el que sabe una tecnología a fondo, sino el que sabe combinar varias

para resolver problemas reales. Ese es exactamente el perfil que forma este libro: un fusionador de tecnologías. Ese puedes ser tú.

Ejemplos paso a paso

Ejemplo 1: Tu primera estación de monitoreo IoT (sensor, red, nube)

Vamos a construir el ciclo completo de la fusión tecnológica en miniatura: un sensor que mide temperatura, una conexión que envía los datos y un panel en la nube donde los ves en vivo. No necesitas comprar nada para entender el ejemplo: puedes simularlo todo en tu computadora, y cuando consigas un sensor real (una placa ESP32 cuesta muy poco), el mismo diseño funciona.

Paso 1: Define el problema. Queremos saber la temperatura de un cuarto cada minuto y recibir una alerta si supera los 30 grados. Problema claro, tecnología mínima: esa es la mentalidad correcta.

Paso 2: Elige los componentes del ciclo sensor-red-cerebro-músculo. Sensor: un DHT22 (mide temperatura y humedad) conectado a una placa ESP32. Red: la WiFi de tu casa usando el protocolo MQTT, que es un sistema de mensajería ligero diseñado para dispositivos pequeños. Cerebro: un programa en Python en tu computadora (o en la nube) que recibe los datos y decide si hay alerta. Músculo: un mensaje de alerta que llega a tu teléfono.

Paso 3: Configura el envío de datos. El sensor publica sus lecturas en un "tema" MQTT con un mensaje en formato JSON, que es el idioma estándar para intercambiar datos. Así se ve un mensaje típico:

```
{
  "dispositivo": "estacion-cocina-01",
  "temperatura_c": 28.4,
  "humedad_pct": 55,
  "marca_tiempo": "2026-09-19T10:15:00"
}
```

Paso 4: Escribe el cerebro en Python. Este programa se suscribe al tema MQTT, guarda cada lectura en un archivo y manda una alerta si la temperatura pasa el límite. Instala primero la librería necesaria desde tu terminal:

```
pip install paho-mqtt
```

Y este es el programa completo, guárdalo como `estacion.py`:

```
import json
import paho.mqtt.client as mqtt

LIMITE_TEMP = 30.0
ARCHIVO_DATOS = "lecturas.txt"

def al_recibir(cliente, datos, mensaje):
    lectura = json.loads(mensaje.payload.decode("utf-8"))
    temp = lectura["temperatura_c"]
    with open(ARCHIVO_DATOS, "a", encoding="utf-8") as f:
        f.write(json.dumps(lectura) + "\n")
    print("Recibido:", lectura["dispositivo"], temp, "C")
    if temp > LIMITE_TEMP:
        print("ALERTA: temperatura alta en", lectura["dispositivo"])

cliente = mqtt.Client()
cliente.on_message = al_recibir
cliente.connect("test.mosquitto.org", 1883, 60)
cliente.subscribe("lucas-ega/estacion/temperatura")
print("Escuchando sensores... pulsa Ctrl+C para salir")
cliente.loop_forever()
```

Paso 5: Ejecútalo y prueba. En tu terminal corre:

```
python estacion.py
```

Verás el mensaje "Escuchando sensores...". Para simular el sensor sin hardware, abre otra terminal y publica una lectura de prueba con este comando (necesitas tener instalado `mosquitto-clients`; en muchos sistemas se instala con el gestor de paquetes):

```
mosquitto_pub -h test.mosquitto.org -t lucas-ega/estacion/temperatura -m
'{"dispositivo":"estacion-cocina-01","temperatura_c":31.2,"humedad_pct":55,"marca_tiempo":"2026-09-19T10:16:00"}'
```

Verás cómo tu programa recibe el dato, lo guarda y dispara la alerta porque 31.2 supera el límite de 30. Acabas de construir un sistema IoT completo: sensor (simulado), red (MQTT), cerebro (Python) y músculo (la alerta). Cuando tengas el sensor físico, solo cambias la parte que publica los datos; el resto ya funciona.

Ejemplo 2: Tu primer asistente con IA generativa (API, nube e integración)

Ahora toca la estrella: la inteligencia artificial. Vas a construir un asistente que responde preguntas sobre las diez tecnologías de esta unidad, usando la API de un modelo de lenguaje. Aprenderás el patrón que usan miles de productos reales: tu programa manda una petición a la nube, la IA responde y tu programa presenta el resultado.

Paso 1: Entiende el patrón. No vas a entrenar un modelo (eso cuesta millones); vas a consumir un modelo ya entrenado a través de su API, que es como un mostrador donde pides y te sirven. Tú mandas texto, la IA devuelve texto. Simple y poderosísimo.

Paso 2: Prepara tu llave de acceso. Los servicios de IA te dan una clave API personal (normalmente gratis con un límite mensual). Guárdala en una variable de entorno, nunca escrita directo en el código. En tu terminal:

```
export MI_CLAVE_IA="tu-clave-aqui"
```

Paso 3: Haz tu primera llamada con curl, la herramienta de terminal para hablar con servicios web. Este ejemplo usa un formato genérico compatible con la mayoría de APIs de modelos de lenguaje:

```
curl https://api.ejemplo-ia.com/v1/chat \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer $MI_CLAVE_IA" \
  -d '{
    "modelo": "asistente-general",
    "mensajes": [
      {"rol": "sistema", "contenido": "Eres un tutor paciente de tecnologia.
Respondes en espanol mexicano, con tono claro y usando analogias cotidianas."},
      {"rol": "usuario", "contenido": "Explicame que es el edge computing como si
tuviera 12 anos."}
    ],
    "temperatura": 0.7
  }'
```

Fíjate en las partes: la URL del servicio, los encabezados (qué tipo de dato mandas y tu autorización), y el cuerpo en JSON con el modelo, la conversación y la "temperatura" (qué tan creativa será la respuesta: 0 es estricta, 1 es muy creativa).

Paso 4: Integra la IA a tu programa Python para que el asistente sea reutilizable. Guarda esto como `tutor_ia.py`:

```

import json
import os
import urllib.request

CLAVE = os.environ.get("MI_CLAVE_IA")
URL = "https://api.ejemplo-ia.com/v1/chat"

def preguntar(pregunta):
    cuerpo = {
        "modelo": "asistente-general",
        "mensajes": [
            {"rol": "sistema", "contenido": "Eres un tutor paciente de tecnologia. Respondes en espanol mexicano, claro y con analogias."},
            {"rol": "usuario", "contenido": pregunta}
        ],
        "temperatura": 0.7
    }
    peticion = urllib.request.Request(
        URL,
        data=json.dumps(cuerpo).encode("utf-8"),
        headers={"Content-Type": "application/json",
                 "Authorization": "Bearer " + CLAVE}
    )
    with urllib.request.urlopen(peticion) as respuesta:
        datos = json.loads(respuesta.read().decode("utf-8"))
        return datos["opciones"][0]["mensaje"]["contenido"]

if __name__ == "__main__":
    print("Tutor IA listo. Escribe 'salir' para terminar.")
    while True:
        q = input("Tu pregunta: ")
        if q.strip().lower() == "salir":
            break
        print("\n" + preguntar(q) + "\n")

```

Paso 5: Ejecútalo y conversa:

```
python tutor_ia.py
```

Pregúntale sobre cualquiera de las diez tecnologías y observa cómo el mensaje de "sistema" moldea su personalidad de tutor. Cambia ese mensaje y el asistente cambia de carácter: así de fácil se personaliza la IA. Lección clave de fusión: tu programa (lógica local) más la IA (cerebro en la nube) crean un producto que ninguno lograría solo.

Ejemplo 3: Despliegue edge con un archivo de configuración YAML

El tercer ejemplo te enseña una habilidad profesional real: describir un sistema completo en un archivo de configuración, sin programar cada detalle. Usaremos YAML, el formato favorito para configuraciones porque es legible para humanos. Imagina que vas a instalar diez estaciones de monitoreo como la del Ejemplo 1 en un invernadero, y quieres que cada una procese sus datos localmente (edge) y solo mande alertas a la nube.

Paso 1: Entiende qué describes. Un despliegue edge típico tiene: dispositivos (qué sensores y dónde), reglas de procesamiento local (qué calcular en el borde), y política de nube (qué subir y cuándo). Todo eso cabe en un solo archivo YAML.

Paso 2: Escribe el archivo de configuración. Guárdalo como invernadero.yaml:

```
proyecto: invernadero-lucas-ega
version: 1

dispositivos:
  - id: estacion-norte-01
    ubicacion: "sector norte"
    sensores: [temperatura, humedad, luz]
    intervalo_segundos: 60
  - id: estacion-sur-02
    ubicacion: "sector sur"
    sensores: [temperatura, humedad]
    intervalo_segundos: 60

borde:
  modo: local_primerio
  reglas:
    - nombre: alerta_calor
      condicion: "temperatura_c > 32"
      accion: notificar
    - nombre: resumen_cada_hora
      condicion: "cada 3600 segundos"
      accion: calcular_promedios

nube:
  subir: solo_alertas_y_resumenes
  panel_url: "https://panel.ejemplo.com/invernadero"
  retencion_dias: 90
```

Lee el archivo en voz alta: casi se entiende solo, ¿verdad? Esa es la magia de YAML: un técnico en el campo puede leer y modificar la configuración sin ser programador.

Paso 3: Escribe un validador en Python que lea la configuración y verifique que todo esté en orden antes del despliegue real. En la industria, validar antes de desplegar evita desastres. Guarda esto como validar.py:

```
import sys

try:
    import yaml
except ImportError:
    import json as yaml
    print("Aviso: usando JSON como alternativa")

def validar(ruta):
    with open(ruta, encoding="utf-8") as f:
        conf = yaml.safe_load(f)
        errores = []
        if "dispositivos" not in conf:
            errores.append("Falta la seccion 'dispositivos'")
        for d in conf.get("dispositivos", []):
            if "id" not in d:
                errores.append("Un dispositivo no tiene 'id'")
            if d.get("intervalo_segundos", 0) < 10:
                errores.append("Intervalo muy corto en " + d.get("id", "?"))
        if "borde" not in conf:
            errores.append("Falta la seccion 'borde': sin reglas edge no hay despliegue")
        return errores

errores = validar("invernadero.yaml")
if errores:
    print("Configuracion INVALIDA:")
    for e in errores:
        print(" -", e)
    sys.exit(1)
print("Configuracion valida. Listo para desplegar.")
```

Paso 4: Valida desde la terminal (instala PyYAML primero con pip install pyyaml):

```
pip install pyyaml
python validar.py
```

Deberías ver "Configuración válida. Listo para desplegar." Prueba romper algo a propósito -por ejemplo, cambia `intervalo_segundos` a 5- y corre de nuevo: verás cómo el validador te protege. Esa es una buena práctica profesional: nunca despliegues sin validar.

Con estos tres ejemplos ya viste la fusión en acción tres veces: IoT con MQTT y Python, IA consumida desde la nube, y configuración declarativa para edge computing. Cada ejemplo por separado es útil; juntos son el vocabulario con el que están escritos los capítulos prácticos de este libro.

Práctica

Ejercicio 1: Define con tus palabras qué es una tecnología emergente

Enunciado: Escribe en tu cuaderno una definición de "tecnología emergente" con tus propias palabras, de no más de tres líneas, y luego pon dos ejemplos actuales y uno que ya dejó de ser emergente para convertirse en tecnología normal.

Solución explicada: Una buena definición debe incluir las tres ideas clave de la unidad: que está en desarrollo o crecimiento acelerado, que todavía no es de uso masivo cotidiano, y que tiene potencial de transformar la vida o el trabajo. Por ejemplo: "Una tecnología emergente es una tecnología nueva que mejora muy rápido, que aún poca gente usa, pero que podría cambiar cómo vivimos o trabajamos". Paso 1 para verificar tu definición: subraya en ella las palabras que hablen de novedad, de crecimiento y de impacto; si falta alguna de las tres, complétala. Paso 2, ejemplos actuales válidos: inteligencia artificial generativa, computación cuántica, edición genética con CRISPR, realidad aumentada. Paso 3, ejemplo de tecnología que ya maduró: el teléfono inteligente (emergente en 2007, normal en 2015), el GPS en el celular, o las videollamadas. Si tus ejemplos cumplen ese antes y después, tu comprensión es correcta.

Ejercicio 2: Ubica tecnologías en el ciclo de sobreexpectación

Enunciado: Toma estas cinco tecnologías: IA generativa, computación cuántica, impresión 3D, 5G y metaverso. Para cada una, decide en qué fase del ciclo de sobreexpectación de Gartner crees que está hoy (pico de expectativas, abismo de la desilusión, rampa de consolidación o meseta de productividad) y escribe una frase justificando tu elección.

Solución explicada: No hay respuestas únicas porque el ciclo se interpreta, pero tu justificación debe ser coherente con las señales reales. Paso 1: IA generativa. Señales: millones de usuarios diarios, empresas invirtiendo fuerte, pero también proyectos que fracasan por

expectativas irreales. Ubicación razonable: entre el pico y la rampa de consolidación, porque ya hay usos productivos reales pero sigue el hype. Frase modelo: "Está bajando del pico porque ya hay productos que sí funcionan, aunque muchos proyectos aún prometen más de lo que cumplen". Paso 2: computación cuántica. Señales: avances de laboratorio impresionantes pero casi ningún uso comercial real. Ubicación: entre el pico de expectativas y el abismo. Frase modelo: "Los titulares prometen revolución, pero todavía no resuelve problemas de negocio cotidianos". Paso 3: impresión 3D. Señales: pasó su moda hace años, hoy se usa en serio en medicina e industria sin tanto ruido. Ubicación: rampa de consolidación o meseta de productividad. Frase modelo: "Ya no sale en las noticias, pero las fábricas y hospitales la usan todos los días". Paso 4: 5G. Señales: desplegado en la mayoría de ciudades, útil pero sin haber cumplido las promesas más locas. Ubicación: rampa de consolidación. Paso 5: metaverso. Señales: gran hype alrededor de 2021-2022, luego silencio y recortes. Ubicación: abismo de la desilusión. Si tus cinco justificaciones mencionan señales observables (adopción, inversión, titulares, usos reales), el ejercicio está bien resuelto aunque tu fase exacta difiera un poco.

Ejercicio 3: Detecta la convergencia en un producto real

Enunciado: Elige un producto tecnológico que uses o conozcas bien (puede ser un asistente de voz, un reloj inteligente, una app de navegación con tráfico en vivo o un videojuego con IA). Identifica al menos cuatro tecnologías de las diez de la unidad que se combinan en ese producto y explica qué aporta cada una y qué pasaría si faltara.

Solución explicada: Tomemos como modelo un reloj inteligente con medición de salud. Paso 1: lista las tecnologías presentes. IoT y sensores: el reloj mide tu pulso, oxígeno y movimiento con sensores en miniatura; sin ellos no hay datos. Conectividad (5G/WiFi/Bluetooth): manda los datos a tu teléfono y a la nube; sin ella el reloj sería una calculadora de muñeca aislada. Nube: guarda tu historial de meses y lo procesa; sin ella perderías todo al cambiar de reloj. IA: detecta patrones, como un ritmo cardíaco irregular, y te avisa; sin ella solo verías números crudos sin interpretación. Paso 2: para cada una escribe la consecuencia de quitarla, como hicimos arriba. Paso 3: verifica que tu análisis mencione el ciclo sensor-red-cerebro-músculo: sensor (el reloj mide), red (conectividad), cerebro (IA en la nube), músculo (la alerta que vibra en tu muñeca). Si tu producto elegido permite mapear esas cuatro etapas, demostraste que entiendes la convergencia, que era el objetivo del ejercicio.

Ejercicio 4: Diseña en papel un sistema de fusión para un problema local

Enunciado: Piensa en un problema real de tu colonia, escuela o negocio familiar (por ejemplo: se desperdicia agua, hay inseguridad en una calle, un negocio pierde ventas por no tener control de inventario). Diseña en papel un sistema que lo resuelva combinando al menos tres tecnologías de la unidad. Dibuja el ciclo sensor-red-cerebro-músculo y nombra cada componente.

Solución explicada: Usemos como modelo el desperdicio de agua en el riego de un jardín o huerto. Paso 1: define el problema con números si puedes: "se riega todos los días a la misma hora aunque haya llovido, desperdiciando agua". Paso 2: elige tres tecnologías. IoT: sensores de humedad en la tierra. IA sencilla: un programa que decide si hace falta regar según humedad, pronóstico del clima y hora del día. Nube más app móvil: el dueño ve todo en su teléfono y recibe alertas. Paso 3: dibuja el ciclo. Sensor: los medidores de humedad entierran sus sondas y leen cada 30 minutos. Red: una placa WiFi manda las lecturas por MQTT a la nube. Cerebro: el programa compara la humedad con el umbral y consulta el pronóstico; si la tierra está seca y no va a llover, ordena regar. Músculo: una electroválvula abre el riego solo en la zona que lo necesita. Paso 4: justifica por qué es fusión y no solo "un aparato": ninguna tecnología sola resuelve el problema (el sensor sin decisión no sirve, la IA sin datos no decide, la válvula sin orden no actúa). Si tu diseño conecta al menos tres tecnologías en un ciclo completo con un problema real al inicio, el ejercicio está perfecto.

Ejercicio 5: Escribe tu plan personal contra la obsolescencia

Enunciado: Basándote en la sección de buenas prácticas, escribe tu plan personal de actualización tecnológica para los próximos seis meses: tus fuentes de información (máximo tres), tu proyecto permanente, qué fundamento vas a reforzar y cómo vas a cuidar tu descanso. Sé específico: nada de "voy a leer más", sino qué, cuándo y cuánto.

Solución explicada: Un buen plan es específico, medible y realista. Paso 1: fuentes. Ejemplo válido: "Boletín semanal de tecnología en español, 20 minutos cada domingo en la mañana; canal de YouTube de programación, dos videos por semana; documentación oficial de Python, 15 minutos diarios". Lo importante es que sean pocas y con horario. Paso 2: proyecto permanente. Ejemplo: "Voy a construir la estación de monitoreo del Ejemplo 1, primero simulada y luego con un sensor real; le agregaré una mejora cada mes". Paso 3: fundamento. Ejemplo: "Voy a reforzar lógica de programación resolviendo un ejercicio diario de 20 minutos durante tres meses". Paso 4: descanso. Ejemplo: "No pantallas después de las 10 de la noche, domingo libre de estudio". Paso 5: revisión. Marca en tu calendario una fecha dentro de tres meses para revisar si cumpliste el plan y ajustarlo. Si tu plan cabe en media cuartilla, tiene horarios concretos y cubre los cinco puntos, tienes un sistema real, no un propósito de año nuevo.

Ejercicio 6: Elige tu perfil profesional y traza tu ruta

Enunciado: Relee la lista de perfiles profesionales del futuro al final de la explicación. Elige el que más te atraiga y escribe: por qué te atrae (dos razones), qué dos tecnologías de la unidad son su base principal, y qué tres pasos concretos darás en el próximo año para acercarte a ese perfil.

Solución explicada: Tomemos como modelo el perfil de desarrollador de IA aplicada. Paso 1: razones. Ejemplo: "Me atrae porque combina programación con impacto real en empresas, y porque puedo empezar con herramientas gratuitas desde mi casa". Tus razones deben ser personales y honestas, no lo que suena impresionante. Paso 2: tecnologías base. Para IA aplicada: inteligencia artificial (obvio) y computación en la nube (donde viven los modelos y los datos). Justifica cada una en una línea: sin IA no hay producto, sin nube no hay dónde correrlo a escala. Paso 3: tres pasos concretos y ordenados. Ejemplo: "1) Terminar este libro y dominar Python en seis meses. 2) Construir tres proyectos que usen APIs de IA y publicarlos en mi portafolio. 3) Conseguir un trabajo junior o freelance donde aplique IA a problemas reales, aunque empiece con proyectos pequeños". Verifica que tus pasos vayan de lo que controlas hoy (estudiar) a lo que harás después (construir) y luego a lo externo (trabajar): ese orden es el que funciona. Si tu ruta tiene esa progresión, está bien trazada.

Glosario esencial de la unidad: habla como un profesional desde el día uno

Cada campo tiene su vocabulario, y la tecnología no es la excepción. Si quieres que te tomen en serio en una entrevista, en una comunidad o en un proyecto, necesitas manejar estos términos con naturalidad. Aquí va el glosario de la unidad: léelo completo una vez y luego úsalo como referencia cada que una palabra te suene a chino.

- Algoritmo: una receta paso a paso para resolver un problema. Cuando sigues una receta de cocina al pie de la letra, estás ejecutando un algoritmo. Los programas de computadora no son más que algoritmos escritos en un lenguaje que la máquina entiende.
- API (interfaz de programación de aplicaciones): un mostrador donde un programa le pide servicios a otro. Cuando tu app del clima muestra la temperatura, le está pidiendo el dato a la API de un servicio meteorológico. Aprender a usar APIs es como aprender a pedir en un restaurante: no necesitas saber cocinar el platillo, solo cómo pedirlo bien.
- Big data: conjuntos de datos tan grandes que las herramientas normales no pueden procesarlos. El truco no es el tamaño por sí solo, sino lo que puedes descubrir en él: patrones de compra, epidemias nacientes, fraudes bancarios.
- Bit y byte: el bit es la unidad mínima de información (un 0 o un 1); un byte son 8 bits. Todo lo digital -tus fotos, tus mensajes, este libro- está hecho de bits.
- Blockchain: registro digital repartido en miles de computadoras que es muy difícil de falsificar. Piensa en un libro de contabilidad público donde todas las copias deben coincidir.
- Chatbot: programa que conversa con personas usando lenguaje natural. Los modernos usan IA generativa y pueden ayudar en atención al cliente, educación y soporte técnico.
- Ciberseguridad: el arte y la ciencia de proteger sistemas, datos y personas de ataques digitales. En un mundo donde todo está conectado, es una de las profesiones con más demanda.
- Computación en la nube: usar computadoras lejanas (de empresas como Amazon, Google o Microsoft) a través de internet, en lugar de comprar tus propios servidores. Es como rentar un departamento en lugar de construir una casa: pagas por lo que usas.
- CRISPR: herramienta de edición genética que funciona como unas tijeras moleculares capaces de cortar y modificar el ADN con precisión. Una de las tecnologías más prometedoras y debatidas de la biotecnología.
- Cúbit: la unidad de información de la computación cuántica. A diferencia del bit, puede aprovechar fenómenos cuánticos para explorar muchas posibilidades a la vez.
- Dato: un hecho crudo sin interpretar, como "28.4". Información es el dato con contexto: "la temperatura del cuarto es 28.4 grados". Conocimiento es saber qué hacer con esa información. Esta escalera (dato, información, conocimiento) es la base de todo lo que harás con IA.

- Edge computing: procesar los datos cerca de donde se generan (en el sensor, la cámara, el dispositivo) en lugar de mandarlos a un centro de datos lejano. Clave cuando cada milisegundo cuenta.
- Firmware: el programa grabado dentro de un dispositivo de hardware, como el que hace funcionar tu control remoto o una placa ESP32. Está entre el hardware y el software.
- Hardware: las partes físicas de la tecnología: placas, sensores, cables, carcasas. Si lo puedes patear (no lo hagas), es hardware.
- Inteligencia artificial: sistemas que realizan tareas que normalmente requieren inteligencia humana: entender lenguaje, reconocer imágenes, predecir o decidir. La IA generativa, además, crea contenido nuevo: texto, imágenes, código, audio y video.
- Internet de las cosas (IoT): la red de objetos físicos conectados a internet que capturan y comparten datos: sensores, focos inteligentes, relojes, tractores. Son los "sentidos" de la fusión tecnológica.
- Latencia: el tiempo que tarda un dato en ir y volver. En un videojuego en línea, la latencia alta es el famoso "lag". El 5G la reduce a milisegundos, lo que permite coches autónomos y cirugía a distancia.
- Machine learning (aprendizaje automático): la rama de la IA donde las máquinas aprenden patrones a partir de datos, en lugar de seguir reglas escritas a mano por un programador. Es como enseñar con ejemplos en vez de con instrucciones.
- MQTT: protocolo de mensajería ligero muy usado en IoT para que dispositivos pequeños envíen datos. Funciona con un modelo de publicación y suscripción: unos publican mensajes en "temas" y otros se suscriben a esos temas para recibirlos.
- Prototipo: una versión temprana e imperfecta de un producto, hecha para probar una idea rápido y barato. En tecnología, un prototipo feo que funciona vale más que una idea bonita en papel.
- Realidad aumentada (AR): tecnología que coloca objetos digitales sobre el mundo real, como los filtros de tu cámara o las instrucciones flotantes para reparar una máquina.
- Realidad virtual (VR): tecnología que te sumerge en un mundo completamente digital usando un casco. Se usa para entrenar cirujanos, pilotos y técnicos sin riesgo real.
- Red neuronal: modelo de IA inspirado (muy a grandes rasgos) en cómo se conectan las neuronas del cerebro. Aprende ajustando millones de conexiones internas hasta acertar. Es la base de la IA moderna.

- Robótica: la disciplina que diseña, construye y programa robots: máquinas que perciben su entorno y actúan en él. Combinada con IA, pasa de repetir movimientos a tomar decisiones.
- Sensor: dispositivo que convierte una magnitud del mundo físico (temperatura, luz, movimiento, sonido) en una señal digital. Son los ojos, oídos y piel de los sistemas inteligentes.
- Servidor: una computadora que ofrece servicios a otras (páginas web, bases de datos, APIs). Cuando "subes algo a la nube", lo estás guardando en servidores de alguien más.
- Software: los programas e instrucciones que le dicen al hardware qué hacer. Si no lo puedes patear, es software.
- 5G: la quinta generación de redes móviles, con alta velocidad, latencia ultrabaja y capacidad para conectar millones de dispositivos. La autopista que necesitan el IoT masivo y los vehículos autónomos.
- Startup: empresa joven que busca crecer rápido con un producto innovador y escalable. Muchas de las tecnologías de esta unidad nacieron en startups antes de llegar a las grandes empresas.
- Token: en IA de lenguaje, un fragmento de texto (una palabra o parte de ella) que el modelo procesa. Los servicios de IA suelen cobrar por tokens, así que escribir instrucciones concisas también ahorra dinero.
- Web3: la visión de un internet descentralizado donde los usuarios controlan sus datos e identidad usando tecnologías como blockchain, en lugar de depender de unas pocas empresas gigantes.
- Actuador: el dispositivo que convierte una orden digital en una acción física: un motor que gira, una válvula que se abre, un foco que se enciende. Si el sensor es el sentido del sistema, el actuador es su músculo.
- Ancho de banda: la cantidad de datos que una conexión puede transportar por segundo. Piensa en una tubería: entre más ancha, más agua (datos) pasa al mismo tiempo. El video en alta definición necesita mucho ancho de banda; un mensaje de texto, casi nada.
- Computación cuántica: paradigma de cómputo que usa cúbits y fenómenos cuánticos para resolver ciertos problemas muchísimo más rápido que las computadoras clásicas. Hoy es una tecnología en fase temprana, ideal para investigación en fármacos, materiales y optimización.

- Despliegue (deploy): el acto de poner un sistema o programa a funcionar en su ambiente real, ya sea un servidor, la nube o un dispositivo. Validar la configuración antes de desplegar, como hiciste en el Ejemplo 3, es una buena práctica profesional.
- Gemelo digital: una réplica virtual de un objeto o sistema físico que se actualiza con datos reales en tiempo real. Una fábrica puede tener el gemelo digital de su línea de producción para probar cambios sin detener las máquinas de verdad.
- Hidrógeno verde: hidrógeno producido con energía renovable, propuesto como combustible limpio para industrias y transporte pesado. Parte de la revolución energética que sostiene a todas las demás tecnologías.
- Latencia ultrabaja: retardos de apenas unos milisegundos en la comunicación, característicos del 5G. Es lo que hace posible que un coche autónomo frene a tiempo o que un cirujano opere a distancia.
- Prompt: la instrucción o pregunta que le das a un modelo de IA para obtener una respuesta. Escribir buenos prompts -claros, con contexto y con ejemplos- es una habilidad práctica que usarás todos los días.
- Sostenibilidad tecnológica: el esfuerzo por que el avance tecnológico no destruya el planeta: centros de datos eficientes, hardware reciclable y energía limpia. La tecnología del futuro tiene que ser poderosa y responsable al mismo tiempo.

Un consejo para memorizar vocabulario técnico: no intentes aprenderte el glosario de memoria como si fuera un examen de historia. Úsalo. Cada vez que leas o escuches uno de estos términos en un video, un artículo o una conversación, regresa aquí, relea la definición y úsala en una frase propia. Las palabras se aprenden usándolas, igual que los músculos se fortalecen usándolos.

Resumen de la unidad

Llegaste al final de la Unidad 1. Vamos a compactar todo lo importante en los puntos clave que debes llevarte grabados.

- Una tecnología emergente es una tecnología en desarrollo acelerado, aún no masiva, con potencial de transformar la vida y el trabajo. No todas llegan a la meta: el ciclo de sobreexpectación de Gartner (pico, abismo, rampa, meseta) explica por qué algunas se inflan, caen y luego triunfan en serio.
- Estamos en la cuarta revolución industrial, caracterizada por la fusión de los mundos físico, digital y biológico. Su materia prima es el dato inteligente y su velocidad de cambio no tiene precedente: lo que antes tomaba cincuenta años hoy toma menos de veinte.

- La convergencia tecnológica es la idea central del libro: las tecnologías combinadas se multiplican, no se suman. Casi todo sistema moderno sigue el ciclo sensor-red-cerebro-músculo: el IoT captura, la conectividad transporta, la IA decide y la robótica actúa.
- Las diez tecnologías que marcarán la década son: inteligencia artificial, conectividad 5G/6G, computación cuántica, biotecnología, blockchain, realidad extendida, edge computing, impresión 3D, energías renovables y robótica avanzada. Estúdialas como un sistema conectado, no como una lista aislada.
- Los errores que más frenan a los principiantes son: querer aprenderlo todo a la vez, enamorarse de la herramienta antes que del problema, esperar a tener equipo caro, copiar sin entender, solo estudiar teoría y compararse con expertos. Todos tienen remedio y ahora conoces cada uno.
- Mantenerse actualizado es un sistema, no un esfuerzo heroico: pocas fuentes de calidad, comunidad, un proyecto permanente, fundamentos primero y descanso real. El setenta por ciento de tu tiempo a fundamentos, el treinta a lo nuevo.
- Este libro es tu mapa de ruta: fundamentos, piezas de la fusión, fusión en acción y tu futuro profesional, con proyecto integrador al final. Cada unidad combina explicación, ejemplos, práctica y resumen.
- Los perfiles profesionales del futuro (IA aplicada, IoT, nube y edge, datos, realidad extendida, ciberseguridad, emprendimiento) premian a quien combina tecnologías para resolver problemas reales. Ese fusionador de tecnologías es el perfil que este libro forma, y ya diste el primer paso.

Guarda bien este resumen y tu cuaderno de notas: en las próximas unidades vas a necesitar este mapa más de una vez. Nos vemos en la Unidad 2, donde pasamos del mapa al territorio y empiezas a pensar como programador. ¡A construir!

UNIDAD 2: Inteligencia artificial aplicada

Bienvenido a la unidad más emocionante de este libro. Hasta aquí ya sabes qué es la tecnología y cómo se organiza el mundo digital, pero ahora entramos a la materia que está moviendo todo: la inteligencia artificial. Si alguna vez le pediste a tu teléfono que te pusiera una canción, si viste cómo Netflix te recomienda una serie que te encanta, o si te sorprendió una aplicación que escribe textos casi como una persona, ya probaste la inteligencia artificial sin darte cuenta. En esta unidad vas a entender qué hay detrás de esa magia, qué puede hacer de verdad, qué no puede hacer, y sobre todo cómo puedes usarla hoy mismo para trabajar mejor, estudiar más rápido y hasta programar con ayuda.

Te hablo de tú porque este libro es para ti, lector de carne y hueso, no para un salón de clases. No necesitas saber matemáticas avanzadas ni haber programado antes para seguir esta unidad. Si programaste algo en la Unidad 1 o en tus cursos, mejor, pero no es obligatorio. Lo que sí necesitas es curiosidad y ganas de experimentar, porque la inteligencia artificial se aprende usándola. Cada concepto que veas aquí viene acompañado de ejemplos de la vida real, analogías que te van a hacer sentido y prácticas que puedes hacer con herramientas gratuitas desde tu computadora o tu celular.

Antes de arrancar, déjame ponerte en contexto con un dato importante: la inteligencia artificial no es cosa del futuro, es cosa de tu presente. Cuando llegas a casa y le dices a tu bocina inteligente que apague las luces, hay IA. Cuando el banco detecta un cargo raro en tu tarjeta y te manda un aviso, hay IA. Cuando subes una foto y la aplicación te sugiere etiquetas para las caras, hay IA. La diferencia entre la gente que aprovecha esta tecnología y la que solo la mira pasar es una sola: saber pedirle las cosas bien y entender sus límites. Al terminar esta unidad, tú vas a estar del lado de los que la aprovechan.

Por qué la inteligencia artificial cambió las reglas del juego

Piensa en cómo era trabajar en una computadora hace veinte años. Si querías hacer una tabla, tenías que saber dónde estaba cada botón. Si querías buscar información, tenías que escribir las palabras exactas. La computadora era una herramienta obediente pero torpe: hacía exactamente lo que le decías, ni más ni menos, y si te equivocabas en una coma, todo fallaba. La inteligencia artificial cambió esa relación de raíz. Hoy puedes escribirle a un asistente "hazme un resumen de este documento de diez páginas en lenguaje sencillo" y lo hace. Puedes decirle "escribeme un correo para pedirle a mi jefe vacaciones la próxima semana, en tono amable pero formal" y te entrega un borrador listo para enviar. Eso no es un truco de magia, es el resultado de décadas de investigación que por fin llegaron a las manos de todo el mundo.

El cambio de reglas se nota en tres cosas. Primera: la barrera de entrada bajó muchísimo. Antes, para automatizar una tarea necesitabas programar; hoy, muchas veces basta con describir lo que quieres en español normal. Segunda: la velocidad. Tareas que tomaban horas, como clasificar cien correos, transcribir una entrevista de una hora o generar ideas para un negocio, ahora toman minutos. Tercera: la personalización. La IA puede adaptar una explicación a tu nivel, traducir un texto a tu idioma, o ayudarte a estudiar justo los temas donde vas más flojo. Por eso se dice que la IA es una tecnología de propósito general, como la electricidad o internet: no sirve para una sola cosa, sirve para cambiar la forma en que se hacen casi todas las cosas.

Pero ojo, que aquí viene la primera lección importante de esta unidad: que algo sea poderoso no significa que sea perfecto. La IA se equivoca, inventa datos con total seguridad, repite sesgos que aprendió de internet y a veces te da una respuesta que suena profesional pero está mal. Quien no entiende esto termina confiando ciegamente y metiendo la pata en el trabajo o en la escuela. Quien sí lo entiende usa la IA como lo que es: un asistente rapidísimo y muy capaz, pero que necesita supervisión humana. Esa mentalidad, la de piloto que revisa los instrumentos en vez de pasajero dormido, es lo que te va a separar del montón.

En Lucas EGA Academy creemos que la tecnología se aprende haciéndola tuya. Así que en esta unidad no solo vas a leer definiciones: vas a escribir prompts, vas a comparar respuestas buenas y malas, vas a correr un pequeño programa en Python que simula cómo "aprende" una máquina, y vas a salir con una lista de herramientas gratuitas que puedes usar hoy mismo. Al final, en la sección de práctica, encontrarás seis ejercicios con su solución explicada paso a paso para que compruebes lo que aprendiste.

Qué vas a aprender en esta unidad

Esta unidad está organizada para llevarte de cero a usuario competente de IA aplicada. Empezamos con lo fundamental: qué es la inteligencia artificial y qué no es, para que nadie te venda humo. Vas a conocer la diferencia entre la IA de las películas, esa que tiene conciencia y quiere dominar el mundo, y la IA real, que es buenísima para tareas específicas pero no "piensa" como tú. Esa distinción te va a ahorrar muchos malentendidos y te va a dar una base sólida para todo lo demás.

Después nos metemos al corazón del asunto: el aprendizaje automático, mejor conocido como machine learning. Te lo voy a explicar sin fórmulas raras, con una analogía que no se te va a olvidar: enseñarle a una máquina es parecido a enseñarle a un niño a distinguir frutas, mostrándole muchos ejemplos hasta que agarra la onda. Vas a entender las ideas de datos de entrenamiento, modelo y predicción, que son las tres patas de casi toda la IA moderna. También vas a conocer los tipos principales de IA que te vas a encontrar en la vida real: la IA predictiva, que adivina qué va a pasar; la IA generativa, que crea textos, imágenes y hasta música; y la visión por computadora, que le permite a las máquinas "ver" e interpretar imágenes.

La tercera parte es la más práctica: los modelos de lenguaje y la ingeniería de prompts. Los modelos de lenguaje son esos asistentes con los que platicas por texto, y los prompts son las instrucciones que les das. Aquí está el secreto que casi nadie te dice: la calidad de lo que obtienes depende enormemente de la calidad de lo que pides. Te voy a enseñar técnicas concretas, con ejemplos de prompts malos y prompts buenos, para que tus resultados mejoren desde el primer día. Verás también cómo usar asistentes de IA en tu trabajo y en tus estudios, y cómo la IA te ayuda a programar: autocompletado de código, explicación de errores y depuración asistida.

Cerramos con dos temas que todo usuario responsable debe conocer: los sesgos y la ética de la IA, y cómo la IA se combina con otras tecnologías como el internet de las cosas y la nube para crear soluciones todavía más potentes. Para cuando termines, no solo sabrás usar la IA: sabrás usarla bien, con criterio, y con los pies en la tierra.

Explicación detallada

Qué es la inteligencia artificial y qué no es

Vamos a empezar por lo básico pero bien explicado, porque aquí es donde la mayoría se confunde. La inteligencia artificial es la rama de la computación que busca crear sistemas capaces de hacer tareas que normalmente requieren inteligencia humana: entender lenguaje, reconocer imágenes, tomar decisiones, resolver problemas, aprender de la experiencia. Fíjate que la definición habla de "tareas que requieren inteligencia", no de "ser inteligente como un humano". Esa diferencia es clave. Un programa que detecta si un correo es spam está haciendo una tarea que a ti te tomaría leer y juzgar; el programa lo hace rapidísimo y bastante bien, pero no "entiende" el correo como tú lo entiendes.

Para que te quede clarísimo, piensa en la IA como un alumno súper especializado. Imagina que contratas a una persona que es la mejor del mundo detectando billetes falsos, pero que no sabe cocinar, ni manejar, ni sumar. ¿Dirías que es inteligente en general? No, dirías que es buenísima en su especialidad. Pues así es casi toda la IA que existe hoy: se le llama IA estrecha o IA débil, y significa que cada sistema es experto en una tarea específica. El asistente que te recomienda canciones no sabe diagnosticar enfermedades; el programa que reconoce placas de autos no sabe escribir poemas. Cada uno es un campeón en su cancha y un inútil fuera de ella.

Entonces, ¿qué no es la IA? Primero, no es la IA de las películas. Esos robots con conciencia propia, emociones y planes de dominación mundial pertenecen a la ciencia ficción. En la vida real no existe una máquina que "despierte" y empiece a tener deseos propios. Segundo, la IA no comprende de verdad. Cuando un asistente te responde algo coherente, no es porque haya entendido tu problema como lo haría un amigo; es porque calculó, con base en millones de textos que leyó, cuál es la respuesta más probable y útil. A veces esa respuesta es brillante y a veces es un disparate dicho con mucha seguridad. Tercero, la IA no tiene sentido común. Un niño de cinco años sabe que no debe meter un tenedor al enchufe; una IA potentísima puede fallar en cosas igual de obvias si nunca vio ejemplos parecidos. Cuarto, la IA no es infalible ni neutral: aprende de datos creados por humanos, así que hereda nuestros errores, nuestros prejuicios y nuestras lagunas.

Un error común de principiantes es tratar a la IA como si fuera una persona sabia que siempre tiene la razón. La tratan como oráculo: le preguntan algo, copian la respuesta y la entregan sin revisar. Eso es peligroso. La buena práctica es tratarla como un becario muy veloz y muy leído, pero al que hay que supervisar: le pides un borrador, lo revisas, lo corriges y lo mejoras. Otro error común es el opuesto: despreciarla porque "se equivoca a veces". Claro que se equivoca, pero también te ahorra horas de trabajo mecánico. La postura ganadora está en medio: úsala para acelerar, verifica lo importante, y decide tú.

Analogía para que no se te olvide: la IA es como una calculadora científica avanzada. Una calculadora no "sabe" matemáticas como tu maestro; aplica reglas rapidísimo. Nadie le reclama a la calculadora que no entienda el problema de física, pero todos la usan para no hacer cuentas a mano. Con la IA pasa igual: no le pidas conciencia ni sabiduría, pídele velocidad, borradores, clasificaciones, resúmenes y segundas opiniones. Ahí es donde brilla.

Aprendizaje automático explicado sin matemáticas complicadas

El aprendizaje automático, o machine learning, es la técnica que está detrás de casi toda la IA moderna. Y aunque suena a tema de genios con pizarrón lleno de fórmulas, la idea central es sencilla y te la voy a explicar con una historia.

Imagina que quieres enseñarle a tu sobrinito de cuatro años a distinguir entre un plátano y un pepino. No le das una definición de diccionario ni le explicas la botánica. ¿Qué haces? Le muestras un plátano y le dices "esto es un plátano". Le muestras otro plátano, más chiquito, y repites. Le muestras un pepino y le dices "esto es un pepino, no es plátano". Después de ver veinte ejemplos de cada uno, el niño agarra la onda: se fija en el color, la forma, la textura. Un día le muestras una fruta que nunca había visto y te dice "es plátano", y le atina. El niño aprendió de ejemplos, no de reglas escritas en piedra. Pues el machine learning es exactamente eso, pero con una computadora como alumna y con miles o millones de ejemplos en vez de veinte.

Vamos a ponerle nombre a las piezas, porque estos términos los vas a ver en todas partes. Los ejemplos que le muestras a la máquina se llaman datos de entrenamiento. En nuestro cuento, las fotos de plátanos y pepinos con su etiqueta ("plátano" o "pepino") son los datos de entrenamiento. El "cerebro" que se forma en la máquina después de ver esos ejemplos se llama modelo. El modelo no es una lista de reglas que alguien programó a mano; es un conjunto de ajustes internos que la máquina fue calibrando sola para acertar lo más posible. Y cuando le muestras algo nuevo y el modelo te dice qué cree que es, eso se llama predicción. Datos de entrenamiento, modelo, predicción: esas tres palabras te abren la puerta a entender el 80 por ciento de la IA actual.

Ahora, ¿cómo "calibra" la máquina esos ajustes internos sin que nadie le diga las reglas? Con un proceso de prueba y error masivo. Al principio el modelo adivina casi al azar y se equivoca muchísimo. Cada vez que se equivoca, el algoritmo ajusta un poquito sus parámetros internos para equivocarse menos la próxima vez. Repite eso millones de veces y el modelo termina acertando muy bien. Es como si el niño, cada vez que confunde un pepino con un plátano, ajustara tantito su idea mental de "qué hace que algo sea plátano". Nadie le programó la regla "amarillo y curvo igual a plátano"; él la descubrió solo viendo ejemplos.

Hay tres formas principales en que las máquinas aprenden, y te conviene conocerlas porque explican por qué cada IA sirve para lo que sirve. El aprendizaje supervisado es el del cuento del plátano: le das ejemplos ya etiquetados y la máquina aprende a etiquetar cosas nuevas. Sirve para detectar spam, aprobar o rechazar créditos, diagnosticar con base en síntomas. El aprendizaje no supervisado es cuando le das un montón de datos sin etiquetas y le pides que encuentre grupos o patrones sola. Es como darle al niño una canasta de frutas revueltas sin decirle nada y que él solito las separe por parecidos. Sirve para segmentar clientes, detectar comportamientos raros o descubrir temas en documentos. El aprendizaje por refuerzo es aprender a base de premios y castigos, como entrenar a un perro: la máquina prueba acciones y recibe puntos cuando lo hace bien. Así se entrenan IAs que juegan ajedrez, controlan robots o deciden la mejor ruta de reparto.

Un error común aquí es creer que "más datos siempre es mejor" sin fijarse en la calidad. Si le enseñas al niño con fotos borrosas o mal etiquetadas, va a aprender mal. Con las máquinas pasa igual: datos sucios, incompletos o sesgados producen modelos malos, aunque sean millones. Por eso los expertos dicen que el 80 por ciento del trabajo en IA es conseguir y limpiar datos, no programar algoritmos elegantes. La buena práctica, si algún día trabajas con datos, es esta: revisa de dónde vienen, busca errores y huecos, y pregúntate a quién representan y a quién dejan fuera. Un modelo entrenado solo con fotos de un tipo de personas va a fallar con las demás, y eso no es un detalle técnico, es un problema real que ya ha causado discriminaciones.

Otra idea importante: un modelo solo sabe lo que vio en su entrenamiento. Si entrenas un clasificador de frutas solo con plátanos y pepinos, y un día le muestras una sandía, va a decir "plátano" o "pepino" con toda seguridad, porque no conoce otra opción. Las máquinas no dicen "no sé" a menos que alguien las programe para dudar. Este es el origen de muchas metidas de pata famosas de la IA: responde con seguridad sobre cosas que nunca vio. Grábatelo: la IA no sabe lo que no sabe. Por eso tú, el humano, tienes que ser el filtro final.

Tipos de IA que te vas a encontrar en la vida real

Ya que entiendes la idea del aprendizaje automático, vamos a ordenar el zoológico. La IA no es una sola cosa: hay familias distintas, cada una buena para lo suyo. Conocerlas te sirve para dos cosas: elegir la herramienta correcta para cada tarea y no dejarte apantallar por la mercadotecnia, porque hoy todo producto dice tener "IA" aunque a veces solo tenga una fórmula sencilla con buen maquillaje.

La IA predictiva es la veterana de la familia. Su trabajo es responder "qué va a pasar" o "a qué grupo pertenece esto". Ejemplos que ya usas: el correo que separa el spam, la app del banco que te avisa de un cargo sospechoso, la tienda en línea que te dice "los clientes que

compraron esto también compraron...", el pronóstico del clima de tu teléfono. Detrás de cada uno hay un modelo entrenado con datos históricos que aprendió patrones. Si en los últimos cinco años cada vez que subió la humedad y bajó la presión llovió, el modelo aprende esa correlación y la usa para predecir. La IA predictiva es silenciosa y trabajadora: no platica contigo, pero toma millones de microdecisiones al día. Su límite es claro: predice con base en el pasado, así que cuando el mundo cambia de golpe, sus predicciones se vuelven menos confiables. Y correlación no es causa: que dos cosas pasen juntas no significa que una provoque la otra, algo que los modelos a veces "olvidan".

La IA generativa es la estrella del momento. En vez de clasificar o predecir, crea: textos, imágenes, música, código, videos. Los asistentes con los que conversas, los generadores de imágenes donde describes una escena y te la dibujan, las herramientas que componen melodías, todo eso es IA generativa. ¿Cómo crea? Aprendió de cantidades enormes de ejemplos cómo "se ven" los textos o las imágenes del mundo, y genera piezas nuevas que siguen esos patrones. Es como un chef que probó diez mil platillos y ahora inventa recetas nuevas combinando sabores que aprendió. El resultado puede ser impresionante, pero tiene un defecto famoso: a veces alucina, es decir, inventa datos, citas, fechas o hechos con total seguridad. Si le pides las fuentes de un ensayo, te puede dar referencias que suenan reales pero no existen. Por eso la regla de oro con la IA generativa es: úsala para crear borradores e ideas, y verifica los datos importantes por tu cuenta.

La visión por computadora es la IA que "ve". Le das imágenes o video y ella identifica objetos, personas, textos, gestos o situaciones. La usas cuando desbloqueas tu teléfono con tu cara, cuando subes una foto y te sugiere etiquetar a tus amigos, cuando el estacionamiento lee tu placa automáticamente, o cuando una app traduce un letrero con solo apuntar la cámara. En la industria se usa para revisar que los productos salgan sin defectos, en medicina para ayudar a detectar anomalías en radiografías, en agricultura para contar frutos o detectar plagas con drones. Su gran reto es el contexto: reconocer un objeto en una foto es uno, entender la escena completa es otro. Un sistema puede identificar perfectamente "persona con cuchillo" en una cocina y no distinguir si está cocinando o amenazando a alguien. La visión ve píxeles, no intenciones.

Hay más familias, como la IA para robótica o los sistemas de recomendación, pero con estas tres ya tienes el mapa esencial. Fíjate en algo: casi todo lo que hoy se vende como "IA" cae en una de estas tres canastas. Cuando veas un producto nuevo, pregúntate: ¿predice, genera o ve? Esa pregunta te convierte de consumidor pasivo en usuario crítico. Y un consejo de buenas prácticas: para tareas donde un error sale caro (salud, dinero, seguridad), prefiere sistemas predictivos bien probados y auditados sobre soluciones generativas improvisadas. La novedad no siempre es la mejor herramienta.

Modelos de lenguaje y cómo escribir buenos prompts

Llegamos a la parte que más vas a usar en tu día a día: los modelos de lenguaje. Son sistemas de IA generativa entrenados con cantidades gigantescas de texto: libros, artículos, páginas web, código, conversaciones. Gracias a eso aprendieron cómo se estructura el lenguaje humano y pueden continuar, resumir, traducir, explicar y transformar textos con una fluidez sorprendente. Cuando platicas con un asistente de IA, estás hablando con un modelo de lenguaje. No es una persona, no tiene opiniones propias ni recuerdos de su "vida": es un predictor súper avanzado que calcula cuál es la mejor continuación para tu mensaje.

Piensa en el autocompletado de tu teclado, ese que te sugiere la siguiente palabra cuando escribes un mensaje. Un modelo de lenguaje es como ese autocompletado llevado al extremo: en vez de sugerir una palabra, genera párrafos enteros, y en vez de basarse solo en tus últimos mensajes, se basa en patrones aprendidos de miles de millones de textos. Por eso a veces parece que "entiende": si le escribes el inicio de un correo formal, continúa en tono formal porque aprendió cómo suenan los correos formales. Pero recuerda la lección anterior: predice texto probable, no verdades garantizadas. Puede escribirte una explicación de física impecable y al mismo tiempo inventarse una ley que no existe, todo en el mismo párrafo y con la misma seguridad.

Aquí entra el concepto estrella de esta sección: el prompt. Un prompt es la instrucción que le das al modelo: tu mensaje, tu petición, tu pregunta. Y aquí va el secreto que separa a los usuarios promedio de los usuarios avanzados: el modelo responde según lo que le pides, así que pedir bien es una habilidad que se entrena. A esto se le llama ingeniería de prompts, y no es brujería, es comunicación clara aplicada a una máquina. Las mismas reglas que usarías para pedirle algo a un compañero de trabajo aplican aquí: sé específico, da contexto, di qué formato quieres y pon ejemplos si puedes.

Mira la diferencia con un ejemplo de la vida real. Prompt malo: "hazme un texto de ventas". ¿De qué producto? ¿Para quién? ¿Qué tono? ¿Qué largo? El modelo va a adivinar y probablemente te dé algo genérico que no te sirva. Prompt bueno: "Escribe un texto de ventas de 120 palabras para unos audífonos con cancelación de ruido que cuestan 899 pesos, dirigido a estudiantes universitarios que viajan en transporte público. Tono joven y directo, sin exagerar. Incluye un llamado a la acción al final." ¿Ves la diferencia? El segundo prompt le da al modelo rol implícito, audiencia, producto, precio, tono, extensión y estructura. El resultado va a estar años luz por encima del primero, y no te tomó más de un minuto extra escribirlo.

Te dejo las técnicas de prompting que más resultado dan, con su explicación. Primera, el contexto: dile al modelo quién eres y para qué necesitas el texto. "Soy maestro de secundaria y necesito explicar la fotosíntesis a alumnos de 13 años" produce algo muy distinto a

"explícame la fotosíntesis". Segunda, el formato de salida: pide exactamente cómo quieres la respuesta. "Dámelo en una tabla con tres columnas", "en cinco viñetas", "como guion de video de 60 segundos". Tercera, el rol: "Actúa como un tutor paciente de matemáticas" o "como un editor exigente". El rol ajusta el tono y el enfoque. Cuarta, los ejemplos: si le muestras uno o dos ejemplos del estilo que quieres, el modelo los imita muy bien. Quinta, la descomposición: para tareas grandes, divídelas. En vez de "hazme un plan de negocios", pide primero el resumen, luego el análisis de mercado, luego las finanzas. Sexta, la iteración: la primera respuesta rara vez es la final. Responde "más corto", "con ejemplos", "menos formal", y refina hasta que quede. Los profesionales no escriben un prompt perfecto a la primera: conversan con el modelo.

Errores comunes de principiantes con los prompts: hacer preguntas ambiguas y enojarse con respuestas ambiguas; pedirle al modelo que adivine información que no le diste; usarlo para datos exactos y recientes sin verificar (los modelos tienen fecha de corte y no navegan siempre); copiar y pegar sin leer, sobre todo en trabajos escolares o correos importantes; y compartir información confidencial, como contraseñas, datos bancarios o documentos privados de tu empresa, porque lo que escribes en un servicio gratuito puede usarse para mejorar el servicio. Buenas prácticas: sé específico y verifica; itera en vez de rendirte al primer intento; guarda los prompts que te funcionaron, como recetas; y nunca le pidas al modelo que tome por ti decisiones que requieren tu juicio, como diagnósticos médicos o decisiones legales. Úsalo para informarte y redactar, decide tú.

Asistentes de IA en el trabajo y el estudio, e IA para programar

Vamos a lo concreto: ¿cómo se usa la IA un martes cualquiera a las nueve de la mañana? En el trabajo, los asistentes de IA brillan en las tareas que son necesarias pero repetitivas. Redactar correos, minutas de reuniones, descripciones de productos, reportes semanales, publicaciones para redes sociales: todo eso sale en minutos con un buen prompt y una revisión tuya. También sirven para procesar información: pégale la transcripción de una junta de una hora y pídele los acuerdos y los pendientes en una tabla. Súbele un contrato largo y pídele que te explique en lenguaje sencillo las cláusulas más importantes. Ojo: aquí aplica la regla de verificación, porque un dato mal resumido en un contrato sí te puede meter en problemas. Úsalo para entender rápido, pero lee tú mismo lo crítico.

Para generar ideas son buenísimos. Atorado con el nombre de tu negocio, con diez opciones de títulos para tu video, con actividades para una clase, con el menú de la semana: el modelo te da veinte propuestas en segundos y tú eliges y combinas. Para aprender temas nuevos funcionan como tutor personal disponible a toda hora: "explícame qué es el interés compuesto como si tuviera 15 años, con un ejemplo de 1000 pesos", y si no entiendes, le dices "no

entendí la parte del ejemplo, explícamela de otra forma". Esa paciencia infinita es oro para estudiar. También te ayudan a practicar idiomas: pídele que converse contigo en inglés y te corrija, o que te haga un examen de vocabulario con tus palabras débiles.

En el estudio hay una línea que no debes cruzar: usar la IA para que haga tu tarea y entregarla como si fuera tuya. Eso no es aprender, es autoengañarte, y los maestros cada vez detectan mejor los textos generados. La buena práctica es usarla como compañera de estudio: que te explique, que te haga preguntas de repaso, que revise tu redacción y te sugiera mejoras, que te ayude a organizar tus apuntes. Si tú escribes el ensayo y la IA te ayuda a pulirlo, estás aprendiendo. Si ella lo escribe y tú solo pones tu nombre, no aprendiste nada y el examen te lo va a recordar.

Ahora, la IA para programar merece su sección porque está cambiando el oficio. Las herramientas de autocompletado de código sugieren líneas o funciones completas mientras escribes, aprendidas de millones de programas. Es como tener un compañero que ya vio casi todos los errores y soluciones del mundo. Además, le puedes pegar un mensaje de error rarísimo y preguntarle qué significa y cómo arreglarlo; muchas veces te explica la causa y te propone el código corregido. También le puedes pedir que te explique un código que no entiendes línea por línea, que te genere pruebas para tus funciones, o que te traduzca un programa de un lenguaje a otro.

Pero aquí va la advertencia importante para programadores: la IA escribe código que parece correcto y a veces no lo es. Puede inventar funciones de librerías que no existen, usar versiones viejas de una herramienta, o producir código con huecos de seguridad. El error clásico del principiante es copiar el código generado, ver que "corre" y darlo por bueno sin entenderlo. La buena práctica es la misma que con los textos: úsala para acelerar, entiende lo que te da, Pruébalo y quédate con la decisión final. Un programador que usa IA bien produce el doble; uno que la usa mal produce el doble de errores. La diferencia no está en la herramienta, está en el criterio.

Analogía de cierre para esta sección: un asistente de IA es como una bicicleta eléctrica. No pedalea por ti todo el camino, pero multiplica tu esfuerzo: llegas más lejos, más rápido y menos cansado. El que sabe andar en bici le saca provecho; el que nunca aprendió, ni con motor eléctrico avanza. Por eso este libro insiste en que aprendas los fundamentos: la IA potencia lo que ya sabes, no reemplaza el saber.

Sesgos y ética de la IA

Este tema es menos divertido que generar imágenes, pero es el que te convierte en usuario responsable. Un sesgo en IA significa que el sistema trata de forma injusta o inexacta a ciertos grupos de personas. ¿De dónde sale? De los datos. Si entrenas un sistema de contratación con historiales de una empresa donde casi todos los gerentes fueron hombres, el modelo puede aprender a preferir currículums de hombres, no porque sean mejores, sino porque eso vio en los datos. Si entrenas un reconocedor de caras mostly con fotos de un solo grupo de personas, va a fallar más con los demás grupos. La máquina no es malintencionada: simplemente repite los patrones del pasado, incluidos los injustos.

Esto ya pasó en la vida real y con consecuencias serias. Ha habido sistemas que negaron préstamos con más frecuencia a ciertos barrios, herramientas de contratación que penalizaban currículums con palabras asociadas a mujeres, y sistemas de vigilancia que identificaban mal a personas de ciertos grupos con más frecuencia. Ninguno de esos sistemas fue programado para discriminar; discriminaron porque aprendieron de un mundo que ya discrimina. La lección es doble: quienes crean IA tienen la responsabilidad de usar datos representativos y probar sus sistemas con grupos diversos, y quienes usamos IA debemos desconfiar de sus decisiones cuando afectan a personas, y exigir supervisión humana en esos casos.

La ética de la IA abarca más que los sesgos. Está el tema de la privacidad: los modelos se entrenan con datos de internet que incluyen información personal, y lo que tú escribes en un asistente gratuito puede almacenarse. Buena práctica: no pegues datos sensibles, ni tuyos ni de tu trabajo. Está el tema de la desinformación: la IA generativa permite crear textos, imágenes, audios y videos falsos muy convincentes, desde noticias inventadas hasta imitaciones de voces. Ya existen estafas donde clonan la voz de un familiar para pedir dinero por teléfono. Tu defensa es el escepticismo sano: verifica por otro canal antes de actuar, sobre todo si hay dinero o urgencia de por medio. Está el tema del trabajo: la IA va a automatizar tareas y a transformar empleos, como antes lo hicieron la máquina de vapor o la computadora. La respuesta sensata no es el pánico ni la negación, sino la adaptación: aprender a trabajar con estas herramientas te vuelve más valioso, no menos.

Y está el tema de la autoría y la honestidad. Si usas IA para generar contenido, sé transparente cuando importe: en la escuela, en el trabajo, con tus clientes. Pasar texto generado como 100 por ciento propio sin decirlo puede violar reglas académicas o laborales. Además, respeta los derechos de otros: no uses IA para suplantar a alguien, crear imágenes comprometedoras falsas de personas reales, o copiar el estilo de un artista para venderlo como original sin permiso. La tecnología es neutral; el uso que le das habla de ti.

Un error común es pensar "la ética es problema de las grandes empresas, no mío". Falso. Cada vez que decides qué prompt escribir, qué respuesta compartir, qué dato pegar en un asistente o si verificas antes de reenviar, estás tomando decisiones éticas con IA. La buena práctica se resume en tres preguntas antes de actuar: ¿esto es verdad o lo verifiqué?, ¿esto daña a alguien o es injusto con alguien?, ¿estaría cómodo si todos supieran que usé IA aquí? Si las tres se responden bien, vas por buen camino.

Cómo la IA se combina con otras tecnologías: IA más IoT, IA más nube

La IA sola ya es potente, pero su verdadero poder explota cuando se combina con otras tecnologías. Esta es la idea central de este libro, "tecnología fusión": las revoluciones no vienen de una tecnología aislada, sino de mezclarlas. Vamos a ver las dos combinaciones más importantes.

Primero, IA más internet de las cosas (IoT). El IoT son los objetos conectados: sensores, cámaras, relojes inteligentes, focos, termostatos, máquinas industriales, autos. Esos dispositivos generan cantidades brutales de datos: temperatura, movimiento, consumo de energía, ubicación, ritmo cardíaco. ¿Y quién analiza esos datos mejor que nadie? La IA. Juntas forman un ciclo perfecto: los sensores capturan el mundo real, la IA lo interpreta y decide, y los dispositivos actúan. Ejemplo en casa: tu reloj detecta un patrón raro en tu ritmo cardíaco durante varios días, la IA lo compara con millones de registros y te sugiere ir al médico antes de que sea grave. Ejemplo en una fábrica: sensores en las máquinas detectan vibraciones anormales, la IA predice qué pieza va a fallar y en cuántos días, y se programa el mantenimiento antes de que la línea se detenga. A eso se le llama mantenimiento predictivo y ahorra millones. Ejemplo en la ciudad: cámaras y sensores de tráfico alimentan una IA que ajusta los semáforos en tiempo real para reducir embotellamientos. Sin IoT, la IA no tendría ojos ni oídos en el mundo físico; sin IA, el IoT sería solo un montón de datos que nadie alcanza a revisar.

Segundo, IA más nube. Entrenar y correr modelos de IA potentes requiere computadoras carísimas que casi nadie tiene en casa. La nube resuelve eso: empresas como Google, Microsoft y Amazon rentan ese poder de cómputo por internet, y gracias a eso tú puedes usar un asistente potentísimo desde un celular modesto. Cuando le hablas a tu bocina inteligente, tu voz viaja a un centro de datos, allá un modelo gigante la procesa, y la respuesta regresa en un segundo. Ese modelo gigante vive en la nube porque ningún teléfono podría cargarlo. La nube también democratizó la creación: hoy un estudiante puede rentar por unas horas la potencia para entrenar su propio modelo sin comprar hardware. Y al revés, la IA está mejorando la nube: optimiza el consumo de energía de los centros de datos, detecta ataques de seguridad y organiza recursos automáticamente.

Hay más fusiones dando vueltas: IA con robótica (robots que aprenden a moverse solos), IA con blockchain (verificar la autenticidad de contenido), IA con realidad aumentada (lentes que reconocen lo que miras y te dan información). Pero el patrón es siempre el mismo: una tecnología captura o mueve, otra piensa. Cuando entiendes ese patrón, empiezas a ver oportunidades en todas partes: ¿qué pasaría si le pongo sensores a esto y una IA que lo analice? Esa pregunta, hecha en tu negocio, en tu escuela o en tu comunidad, es la semilla de la innovación. Y ya tienes el conocimiento de esta unidad para empezar a responderla.

Cómo se entrena un modelo en la vida real (el panorama completo)

Ya viste la versión de juguete con el clasificador de spam. Ahora veamos cómo se ve el proceso en un proyecto real, para que entiendas el trabajo que hay detrás de cada IA que usas y por qué no es magia.

Todo empieza con un problema bien definido. Nadie entrena una IA "para todo"; se entrena para algo concreto, como "detectar si este correo es spam" o "predecir cuánta energía va a consumir este edificio mañana". Definir mal el problema es el primer error caro: si pides predecir "clientes felices" sin definir qué significa feliz en datos medibles, el proyecto nace muerto.

Después viene la recolección de datos, que es la etapa más larga y menos glamorosa. Se juntan miles o millones de ejemplos: correos marcados como spam, fotos etiquetadas, historiales de ventas. Luego viene la limpieza: quitar duplicados, corregir etiquetas equivocadas, rellenar huecos. Los expertos bromean con que un científico de datos pasa el 80 por ciento de su tiempo limpiando datos y el 20 por ciento presumiendo el modelo. No es broma, es real.

Sigue el entrenamiento: se le muestran los datos al algoritmo, que ajusta sus parámetros internos millones de veces hasta acertar lo más posible. Después se prueba con datos nuevos que el modelo nunca vio, para medir qué tan bien generaliza. Si acierta mucho con los datos viejos pero falla con los nuevos, se dice que memorizó en vez de aprender, y hay que corregirlo. Finalmente se despliega: el modelo se pone a trabajar en el mundo real, pero el trabajo no termina ahí, porque el mundo cambia y los modelos se desactualizan. Un modelo de predicción de tráfico entrenado antes de una pandemia se vuelve inútil cuando cambian los hábitos de movilidad. Por eso los sistemas serios se reentrenan periódicamente con datos frescos.

La lección para ti como usuario: cuando una IA falle, casi siempre la causa está en alguna de estas etapas: problema mal definido, datos malos o viejos, o un mundo que cambió después del entrenamiento. Y como futuro creador: el algoritmo elegante es lo de menos; lo que define el éxito es un buen problema, buenos datos y pruebas honestas con datos nuevos.

Herramientas gratuitas de IA que puedes usar hoy

Una cosa es entender la teoría y otra tener las herramientas instaladas. Aquí va un mapa de herramientas gratuitas, organizadas por lo que hacen, para que empieces hoy mismo sin gastar un peso. Ojo: los servicios gratuitos cambian sus planes seguido, así que si alguno limita funciones mañana, busca su alternativa; la categoría es lo que importa, no la marca.

Para conversar y redactar están los asistentes de lenguaje: ChatGPT, Claude, Gemini y Copilot tienen versiones gratuitas que te dejan platicar, pedir resúmenes, redactar correos y generar ideas. Mi recomendación: prueba dos o tres con el mismo prompt y quédate con el que te dé mejores respuestas para tu tipo de tareas. Para buscar información con fuentes, Perplexity responde citando de dónde sacó cada dato, lo cual te ahorra el paso de verificación. Para estudiar, puedes pedirle a cualquiera de estos que te explique temas, te haga exámenes de práctica o te organice apuntes.

Para crear imágenes, hay generadores donde describes la escena con palabras y te la dibujan: las versiones gratuitas de varios servicios te dan un número de imágenes al mes. Sirven para portadas, ilustraciones de tus proyectos o ideas visuales para tu negocio. Para audio, hay herramientas gratuitas de transcripción que convierten voz a texto: graba una junta o una entrevista, súbela y obtén el texto para resumirlo después. También hay voces sintéticas gratuitas para narrar tus videos, como las que usamos en los cursos de la academia.

Para programar, GitHub Copilot tiene plan gratuito para estudiantes y ofrece autocompletado en tu editor; también puedes simplemente pegarle errores a un asistente de lenguaje y pedirle explicación. Para organizar tu trabajo, Notion y otras apps ya traen IA integrada que resume notas y redacta borradores. Y para experimentar como creador, Google Colab te da una computadora gratuita en la nube donde puedes correr código Python, incluidos pequeños modelos de machine learning, sin instalar nada.

Buenas prácticas con herramientas gratuitas: lee qué hacen con tus datos antes de pegar información de tu trabajo; no subas documentos confidenciales; guarda en una nota los prompts que te funcionaron en cada herramienta; y combina herramientas en vez de casarte con una sola: una para redactar, otra para buscar con fuentes, otra para imágenes. El usuario avanzado no es el que paga más, es el que sabe qué herramienta usar para cada tarea.

Ejemplos paso a paso

Ejemplo 1: un prompt malo convertido en un prompt bueno

Vamos a ver con lujo de detalle cómo la ingeniería de prompts cambia el resultado. Imagina que tienes una pequeña tienda en línea de playeras y quieres publicar en redes sociales. Este es el proceso completo, como lo haría un profesional.

Paso 1: el prompt malo. Escribes esto en tu asistente de IA favorito:

```
Hazme una publicacion para vender playeras.
```

¿Qué obtienes? Algo genérico como: "Playeras de excelente calidad al mejor precio. ¡Compra ahora!" No menciona tu tienda, ni el diseño, ni a quién le hablas, ni dónde comprar. Es texto de relleno que podrías haber escrito tú en diez segundos. El problema no es el modelo, es la instrucción: no le diste nada con qué trabajar.

Paso 2: analiza qué le faltó. Hazte estas preguntas: ¿de qué playeras hablo? ¿Quién es mi cliente? ¿Qué tono usa mi marca? ¿Qué acción quiero que hagan? ¿Qué tan largo debe ser el texto? Cada respuesta que agregues es una mejora directa en el resultado.

Paso 3: escribe el prompt bueno, con contexto, audiencia, tono, formato y llamado a la acción:

```
Actua como un redactor publicitario experto en redes sociales.
```

```
Contexto: tengo una tienda en linea llamada "Playeras del Barrio" en la Ciudad de Mexico. Vendo playeras de algodón con diseños de cultura mexicana (lucha libre, tacos, alebrijes). Precio: 249 pesos.
```

```
Tarea: escribe una publicacion para Instagram de maximo 60 palabras para jovenes de 18 a 30 años. Tono divertido y orgulloso de lo mexicano, sin sonar cursi. Incluye 3 hashtags relevantes y un llamado a la acción para visitar la tienda en linea.
```

```
Formato: solo el texto de la publicacion, sin explicaciones extra.
```

Paso 4: itera sobre el resultado. Supón que el modelo te da algo bueno pero un poco largo. No empieces de cero, refina: "está bien, pero hazlo más corto, de 40 palabras, y cambia el llamado a la acción por algo más urgente". O si el tono no te convence: "menos formal, más como hablan los chavos en la calle pero sin groserías". Cada iteración te acerca al texto final.

Paso 5: verifica y personaliza. Lee el texto, ajusta los detalles que solo tú conoces (una promoción real, tu dirección, tu estilo), y publícalo. La IA te dio el 80 por ciento; tu toque final es el 20 por ciento que hace la diferencia entre un anuncio genérico y uno que suena a tu negocio.

Lección del ejemplo: un buen prompt tiene cinco ingredientes: rol, contexto, tarea específica, formato y restricciones. Memoriza esos cinco y tus resultados van a mejorar desde hoy.

Ejemplo 2: un mini clasificador en Python que "aprende" de ejemplos

Ahora vamos a ver con tus propios ojos cómo una máquina aprende de ejemplos. Vas a escribir un programa pequeñito en Python que simula un clasificador como los de machine learning real, pero simplificado para que se entienda sin matemáticas. La idea: le damos mensajes de ejemplo etiquetados como "spam" o "no spam", el programa cuenta qué palabras aparecen en cada grupo, y luego clasifica mensajes nuevos viendo a qué grupo se parecen más. Es una versión de juguete de la técnica real que usan los filtros de spam.

Paso 1: abre tu editor o la terminal y crea el archivo. En la terminal:

```
mkdir -p ~/ia-practica
cd ~/ia-practica
```

Paso 2: crea el archivo clasificador.py con este código. Fíjate que todo el código usa solo caracteres ASCII para que no tengas problemas al correrlo:

```
# Mini clasificador de spam: version simplificada de machine learning
# Idea: aprende que palabras aparecen mas en spam y cuales en mensajes normales

# Paso 1: datos de entrenamiento (ejemplos ya etiquetados)
entrenamiento = [
    ("gana dinero rapido sin esfuerzo haz clic aqui", "spam"),
    ("oferta increible solo por hoy compra ya", "spam"),
    ("felicidades ganaste un premio reclama ahora", "spam"),
    ("descuento exclusivo en todos los productos", "spam"),
    ("hola como estas nos vemos manana en la escuela", "no spam"),
    ("te mando la tarea de matematicas por correo", "no spam"),
    ("la junta es a las tres en la sala principal", "no spam"),
    ("feliz cumpleaños te deseo lo mejor siempre", "no spam"),
]

# Paso 2: entrenamiento - contar palabras por categoria
conteo_spam = {}
conteo_normal = {}
```

```

for texto, etiqueta in entrenamiento:
    for palabra in texto.split():
        if etiqueta == "spam":
            conteo_spam[palabra] = conteo_spam.get(palabra, 0) + 1
        else:
            conteo_normal[palabra] = conteo_normal.get(palabra, 0) + 1

print("Palabras aprendidas como spam:", sorted(conteo_spam))
print("Palabras aprendidas como normales:", sorted(conteo_normal))

# Paso 3: prediccion - clasificar un mensaje nuevo
def clasificar(mensaje):
    puntos_spam = 0
    puntos_normal = 0
    for palabra in mensaje.lower().split():
        puntos_spam += conteo_spam.get(palabra, 0)
        puntos_normal += conteo_normal.get(palabra, 0)
    if puntos_spam > puntos_normal:
        return "SPAM"
    elif puntos_normal > puntos_spam:
        return "NO SPAM"
    else:
        return "NO SE (empate)"

# Paso 4: probar con mensajes nuevos que el modelo nunca vio
pruebas = [
    "gana un descuento increible haz clic",
    "nos vemos manana en la escuela",
    "hola te mando un saludo",
]

for mensaje in pruebas:
    print("Mensaje:", mensaje)
    print("Clasificacion:", clasificar(mensaje))
    print("---")

```

Paso 3: corre el programa:

```
python3 clasificador.py
```

Paso 4: interpreta el resultado. Verás que el primer mensaje se clasifica como SPAM porque contiene palabras como "gana", "descuento", "increible" y "clic", que aparecieron en los ejemplos de spam. El segundo se clasifica como NO SPAM porque sus palabras aparecieron

en mensajes normales. El tercero dice "NO SE" porque sus palabras no aparecieron en ningún ejemplo. ¡Eso es exactamente lo que te expliqué en la teoría! El modelo solo sabe lo que vio en su entrenamiento, y cuando no reconoce nada, no debería inventar.

Paso 5: experimenta como un científico de datos. Agrega más ejemplos de entrenamiento y vuelve a correr: verás que clasifica mejor. Luego prueba con un mensaje tramposo como "gana dinero en la escuela mañana": tiene palabras de ambos grupos y el resultado depende de los conteos. Así entiendes en carne propia por qué la calidad y cantidad de los datos de entrenamiento lo es todo. Los sistemas reales usan millones de ejemplos y matemáticas más finas, pero la idea es esta misma: aprender patrones de ejemplos etiquetados y predecir con base en ellos.

Ejemplo 3: consultar una API de IA gratuita desde la terminal

Muchos servicios de IA se usan a través de APIs, que son como ventanillas por internet donde tu programa pide cosas y recibe respuestas. Vamos a practicar el flujo completo con un ejemplo que no necesita clave ni registro: consultar una API pública de chistes para programadores y luego simular cómo se vería una petición a una API de IA. Esto te enseña la mecánica que usarías con cualquier servicio de IA real.

Paso 1: verifica que tienes Python y la librería requests:

```
python3 --version  
pip install requests
```

Paso 2: crea el archivo api_demo.py:

```

import requests

# Paso 1: pedir datos a una API publica (sin clave)
url = "https://official-joke-api.appspot.com/jokes/programming/random"
respuesta = requests.get(url, timeout=10)

print("Codigo de estado:", respuesta.status_code)

if respuesta.status_code == 200:
    datos = respuesta.json()
    chiste = datos[0]
    print("Pregunta:", chiste["setup"])
    print("Respuesta:", chiste["punchline"])
else:
    print("La API no respondio bien. Revisa tu conexion.")

# Paso 2: asi se veria una peticion a una API de IA (ejemplo ilustrativo)
# Los servicios reales de IA (OpenAI, Anthropic, Google, etc.) funcionan
# con este mismo patron: URL + clave + JSON con tu prompt.

ejemplo_peticion_ia = {
    "url": "https://api.ejemplo-ia.com/v1/chat",
    "headers": {"Authorization": "Bearer TU_CLAVE_AQUI"},
    "body": {
        "model": "modelo-ejemplo",
        "messages": [
            {"role": "user", "content": "Explica que es un prompt en 2 lineas"}
        ],
    },
}

print("Estructura de una peticion real a una API de IA:")
print(ejemplo_peticion_ia)

```

Paso 3: corre el programa:

```
python3 api_demo.py
```

Paso 4: observa lo que pasó. Tu programa habló con un servidor en internet, pidió un chiste, recibió datos en formato JSON y los mostró ordenados. Esa es la mecánica básica. Cuando uses una API de IA real, el flujo es idéntico, solo cambia el contenido: en vez de pedir un chiste, mandas tu prompt en el campo "messages", y en vez de un chiste recibes el texto generado. La única diferencia importante es que las APIs de IA piden una clave de acceso (API key) que obtienes registrándote en el servicio, y muchas cobran por uso.

Paso 5: buenas prácticas con APIs. Nunca escribas tu clave directamente en el código que compartes; guárdala en una variable de entorno. Revisa los precios antes de hacer miles de peticiones, porque cada una cuesta. Y maneja los errores: si la API no responde, tu programa no debería tronar, debería mostrar un mensaje amable. Con este ejemplo ya entiendes el puente entre tus programas y los servicios de IA en la nube, que es justo la fusión de IA más nube que vimos en la teoría.

Práctica

Ejercicio 1: distingue qué es y qué no es inteligencia artificial

Enunciado: clasifica cada caso como "es IA", "no es IA" o "usa IA pero no es IA en sí", y explica por qué en una línea. Los casos: a) una calculadora que suma y resta; b) el filtro de spam de tu correo; c) un video donde un robot de película tiene sentimientos; d) el autocompletado del teclado de tu celular; e) una hoja de cálculo con fórmulas que tú escribiste; f) un asistente que resume documentos.

Solución paso a paso. Paso 1: recuerda la definición: IA es un sistema que hace tareas que requieren inteligencia humana, y que típicamente aprende de datos en vez de seguir solo reglas escritas a mano. Paso 2: aplica el criterio caso por caso. a) La calculadora no es IA: solo aplica reglas aritméticas fijas que alguien programó, no aprende nada. b) El filtro de spam sí es IA: aprende de millones de correos etiquetados qué patrones indican spam, es aprendizaje supervisado. c) El robot con sentimientos de la película no es IA real: es ficción, ningún sistema actual tiene conciencia ni emociones. d) El autocompletado sí es IA: predice la siguiente palabra con base en patrones aprendidos de muchísimo texto. e) La hoja de cálculo con tus fórmulas no es IA: ejecuta exactamente las reglas que tú escribiste, no aprende. f) El asistente que resume sí es IA: usa un modelo de lenguaje entrenado para comprender y generar texto. Paso 3: verifica tu lógica con la pregunta clave: ¿el sistema aprende patrones de datos o solo obedece reglas fijas? Si aprende, es IA; si solo obedece, no lo es. Conclusión: b, d y f son IA; a y e no son IA; c es ficción.

Ejercicio 2: identifica el tipo de aprendizaje automático

Enunciado: para cada situación, di si se trata de aprendizaje supervisado, no supervisado o por refuerzo, y justifica. a) Un banco entrena un sistema con 100,000 solicitudes de crédito ya marcadas como "pagó" o "no pagó" para decidir nuevas solicitudes. b) Una tienda en línea agrupa a sus clientes en segmentos sin decirle al sistema cuántos grupos hay ni quién pertenece a cuál. c) Un programa aprende a jugar videojuegos probando movimientos y ganando puntos cuando sobrevive más tiempo. d) Un hospital entrena un modelo con radiografías etiquetadas por doctores como "sana" o "con anomalía".

Solución paso a paso. Paso 1: recuerda las tres definiciones. Supervisado: datos con etiquetas, el sistema aprende a etiquetar lo nuevo. No supervisado: datos sin etiquetas, el sistema encuentra grupos o patrones solo. Por refuerzo: el sistema prueba acciones y recibe premios o castigos. Paso 2: analiza cada caso. a) Supervisado: las solicitudes vienen etiquetadas ("pagó"/"no pagó") y el sistema aprende a clasificar solicitudes nuevas. b) No supervisado: no hay etiquetas ni número de grupos definido; el sistema descubre los segmentos por sí solo. c) Por refuerzo: no hay ejemplos etiquetados, hay un sistema de puntos que premia sobrevivir; aprende a base de prueba y premio. d) Supervisado: las radiografías vienen etiquetadas por expertos y el modelo aprende a repetir ese etiquetado. Paso 3: comprueba con la pregunta de oro: ¿había etiquetas en los datos? Si sí, supervisado; si no y buscaba grupos, no supervisado; si aprendía de premios, refuerzo. Conclusión: a y d son supervisado, b es no supervisado, c es por refuerzo.

Ejercicio 3: reescribe tres prompts malos y mejóralos

Enunciado: estos prompts son malos. Reescríbelos aplicando los cinco ingredientes (rol, contexto, tarea específica, formato y restricciones). a) "escribe sobre perros". b) "hazme una dieta". c) "corrige mi texto: ayer fui al mercado y compre muchas cosas".

Solución paso a paso. Paso 1: detecta qué le falta a cada uno. Al a) le falta todo: ¿qué tipo de texto, para quién, qué largo? Al b) le falta contexto personal: una dieta depende de tu cuerpo, tus gustos y tu salud. Al c) le falta decir qué tipo de corrección quieres y el texto completo a corregir. Paso 2: reescribe el a): "Actúa como veterinario. Escribe un artículo de 200 palabras para dueños primerizos sobre los cuidados básicos de un cachorro en sus primeros 3 meses: vacunas, alimentación y entrenamiento. Tono amable y claro, sin tecnicismos." Paso 3: reescribe el b): "Actúa como nutriólogo. Soy hombre de 30 años, peso 85 kilos, no tengo enfermedades, no me gusta el pescado. Dame un plan de comidas de 5 días para bajar de peso de forma saludable, con desayuno, comida y cena, usando ingredientes baratos de México. Formato de tabla por día. Aclara al inicio que esto no sustituye una consulta médica." Nota la buena práctica: pedir el aviso de salud. Paso 4: reescribe el c): "Actúa como corrector

de estilo. Corrige la ortografía y la puntuación del siguiente texto sin cambiar su significado ni su tono. Devuélveme el texto corregido y después una lista de los errores que encontraste: [aquí pegas tu texto completo]". Paso 5: compara antes y después. Los prompts reescritos le dan al modelo audiencia, formato, tono y límites; el resultado será útil a la primera en vez de requerir cinco intentos. Conclusión: un prompt bueno responde por adelantado las preguntas que el modelo tendría que adivinar.

Ejercicio 4: experimenta con el mini clasificador de Python

Enunciado: usando el programa clasificador.py del Ejemplo 2, haz lo siguiente: a) agrega 4 ejemplos nuevos de entrenamiento (2 spam y 2 no spam) y vuelve a correrlo; b) prueba el mensaje "felicidades ganaste dinero rapido" y anota la clasificación; c) explica con tus palabras por qué el modelo clasifica así ese mensaje.

Solución paso a paso. Paso 1: abre clasificador.py y agrega tus ejemplos a la lista de entrenamiento, por ejemplo: ("gana premios increíbles cada semana entra ya", "spam"), ("oferta solo hoy aprovecha el descuento", "spam"), ("manana hay examen estudia el capitulo tres", "no spam"), ("te invito a mi fiesta el sabado en la tarde", "no spam"). Paso 2: corre el programa con python3 clasificador.py y observa que ahora las listas de palabras aprendidas son más largas. Paso 3: agrega a la lista de pruebas el mensaje "felicidades ganaste dinero rapido" y corre de nuevo. El resultado será SPAM. Paso 4: explica el porqué: las palabras "felicidades", "ganaste", "dinero" y "rapido" aparecieron en los ejemplos etiquetados como spam durante el entrenamiento, así que acumulan más puntos de spam que de mensaje normal. El modelo no "entiende" que es un engaño; solo reconoce que esas palabras suelen aparecer en spam. Paso 5: reflexiona sobre la lección: con más ejemplos de entrenamiento, el modelo distingue mejor; con pocos ejemplos, cualquier palabra nueva lo confunde. Eso mismo pasa en los sistemas reales: la calidad del entrenamiento define la calidad de las predicciones. Conclusión: acabas de hacer machine learning de verdad en versión miniatura: datos etiquetados, entrenamiento por conteo de patrones y predicción sobre datos nuevos.

Ejercicio 5: detecta el sesgo en un sistema de IA

Enunciado: una empresa usa una IA para filtrar currículums. La entrenaron con los currículums de las personas contratadas en los últimos 10 años, cuando el 90 por ciento de los contratados fueron hombres. Ahora la IA rechaza con más frecuencia currículums de mujeres. Responde: a) ¿qué tipo de problema es este?; b) ¿de dónde viene el problema, del algoritmo o de los datos?; c) propón dos medidas concretas para corregirlo.

Solución paso a paso. Paso 1: identifica el problema: es un sesgo en IA, porque el sistema trata de forma injusta a un grupo de personas (las mujeres) sin que haya una razón válida relacionada con su capacidad. Paso 2: localiza el origen. El problema viene de los datos, no del algoritmo en sí: el modelo aprendió el patrón "los contratados suelen ser hombres" porque eso es lo que vio en 10 años de historial. El algoritmo solo repitió el patrón del pasado, incluidos sus prejuicios. Recuerda: la máquina no es malintencionada, pero sí es fiel a datos injustos. Paso 3: propón medidas. Medida 1: balancear y limpiar los datos de entrenamiento, incluyendo currículums de mujeres calificadas que fueron contratadas o que debieron serlo, y quitar del entrenamiento datos que reflejen decisiones discriminatorias del pasado; además, ocultar al modelo datos como el nombre o el género que no aportan al mérito. Medida 2: auditar el sistema con pruebas de equidad, es decir, pasar currículums equivalentes cambiando solo el género y verificar que la decisión no cambie, y mantener siempre a una persona revisando las decisiones finales en vez de dejar que la IA decida sola. Paso 4: formula la lección general: todo sistema que afecte a personas debe probarse con grupos diversos y tener supervisión humana. Conclusión: los sesgos no se arreglan con mejor código solamente, se arreglan con mejores datos, pruebas de equidad y humanos responsables al mando.

Ejercicio 6: diseña una fusión de IA con IoT para un problema real

Enunciado: en tu colonia el agua se desperdicia porque los tinacos se llenan de más y se derraman, y nadie se da cuenta hasta que ya se tiró mucha. Diseña una solución que combine IoT e IA: a) describe qué sensores usarías y qué datos capturan; b) explica qué haría la IA con esos datos; c) describe qué acción automática tomaría el sistema; d) menciona un riesgo o límite de tu solución.

Solución paso a paso. Paso 1: elige los sensores (IoT). Un sensor de nivel de agua dentro del tinaco que mida cada 5 minutos qué tan lleno está, y un sensor de flujo en la tubería de entrada que mida cuánta agua está entrando. Esos son los "ojos" del sistema en el mundo físico. Paso 2: define el trabajo de la IA. Con semanas de datos, un modelo predictivo aprende el patrón de consumo de la casa: a qué horas se usa más agua, cuánto tarda en llenarse el tinaco, cuándo suele haber fugas (por ejemplo, si el nivel baja de noche cuando nadie usa agua, hay fuga). La IA predice cuándo el tinaco llegará al tope y detecta comportamientos anormales. Paso 3: define la acción automática. Cuando la IA predice que el tinaco se llenará en 20 minutos, manda cerrar una válvula inteligente conectada; si detecta posible fuga, manda una alerta al celular del dueño con los datos. El ciclo se cierra: sensor captura, IA decide, dispositivo actúa. Paso 4: reconoce riesgos y límites con honestidad. Si se va el internet, el sistema no puede avisar al celular, así que la válvula debe poder cerrarse sola sin depender de la nube; los sensores necesitan mantenimiento y se pueden descomponer; y la IA puede equivocarse al inicio cuando tiene pocos datos, cerrando la válvula antes de tiempo. Buena

práctica: el sistema debe fallar de forma segura, es decir, ante la duda, cerrar el agua es mejor que derramarla. Conclusión: acabas de diseñar tecnología fusión de verdad, y lo más valioso no fue la idea, sino pensar también en qué puede salir mal.

Resumen de la unidad

Llegaste al final de la unidad y es momento de amarrar todo lo aprendido. Aquí van los puntos clave, directo y sin rodeos, para que los repases cuando quieras.

La inteligencia artificial es la rama de la computación que crea sistemas capaces de hacer tareas que requieren inteligencia humana: entender lenguaje, reconocer imágenes, predecir, decidir y aprender de ejemplos. Casi toda la IA actual es IA estrecha: buenísima en su tarea específica e inútil fuera de ella. No tiene conciencia, no comprende de verdad y no tiene sentido común. Úsala como un asistente rapidísimo que necesita supervisión, no como un oráculo infalible.

El aprendizaje automático es la técnica detrás de casi toda la IA moderna y su idea es simple: en vez de programar reglas a mano, le muestras a la máquina muchos ejemplos etiquetados y ella descubre los patrones sola. Los tres conceptos que debes dominar son datos de entrenamiento (los ejemplos), modelo (lo que la máquina construye con ellos) y predicción (lo que el modelo dice ante algo nuevo). Hay tres formas de aprender: supervisado (con etiquetas, como el filtro de spam), no supervisado (sin etiquetas, encontrando grupos solo) y por refuerzo (a base de premios y castigos). Y la lección de oro: la IA solo sabe lo que vio en su entrenamiento, y datos malos producen modelos malos aunque sean millones.

Los tipos de IA que te vas a encontrar son tres. La IA predictiva responde qué va a pasar o a qué grupo pertenece algo: spam, fraudes, pronósticos, recomendaciones. La IA generativa crea contenido nuevo: textos, imágenes, música, código; es la más famosa hoy, pero a veces alucina e inventa datos con seguridad, así que verifica lo importante. La visión por computadora le permite a las máquinas interpretar imágenes y video: desbloqueo facial, lectura de placas, control de calidad. Ante un producto nuevo, pregúntate: ¿predice, genera o ve?

Los modelos de lenguaje son IA generativa entrenada con cantidades enormes de texto; predicen la mejor continuación para tu mensaje con fluidez sorprendente, pero no garantizan verdades. El prompt es tu instrucción y la ingeniería de prompts es la habilidad de pedir bien. Un buen prompt lleva cinco ingredientes: rol, contexto, tarea específica, formato y restricciones. Itera en vez de rendirte al primer intento, guarda tus mejores prompts y nunca pegues información confidencial en servicios gratuitos.

En el trabajo y el estudio, la IA acelera redacción, resúmenes, ideas, análisis de documentos y tutorías personalizadas. Para programar, ayuda con autocompletado, explicación de errores y depuración, pero el código generado debe revisarse y probarse siempre: el programador que usa IA con criterio produce el doble, el que la usa sin criterio produce el doble de errores. En el estudio, úsala como compañera que explica y corrige, no como quien hace tu tarea.

Los sesgos aparecen cuando los datos de entrenamiento reflejan injusticias del pasado y el modelo las repite: aprende de un mundo que discrimina y discrimina. La ética también exige cuidar tu privacidad, no crear ni difundir desinformación, ser transparente sobre el uso de IA cuando importe y no usarla para dañar o suplantar a otros. Tres preguntas antes de actuar: ¿lo verifiqué?, ¿daña o es injusto con alguien?, ¿me sentiría cómodo si todos supieran que usé IA aquí?

La tecnología fusión es donde todo se potencia: IA más IoT crea el ciclo sensor captura, IA decide, dispositivo actúa, como en el mantenimiento predictivo o las casas inteligentes. IA más nube pone modelos gigantes al alcance de tu celular y permite que cualquiera rentee poder de cómputo para crear. El patrón es siempre el mismo: una tecnología captura o mueve, otra piensa. Cuando veas un problema, pregúntate qué pasaría si le pones sensores y una IA que los analice: esa es la mentalidad fusión.

Tu siguiente paso es practicar. Abre hoy mismo un asistente de IA gratuito, escribe un prompt con los cinco ingredientes sobre algo de tu trabajo o tu estudio, itera dos veces y compara el resultado con lo que habrías hecho sin ayuda. Corre el mini clasificador de Python y juega con más ejemplos. Y cuando estés listo, sigue con la Unidad 3, donde esta IA que ya entiendes se va a conectar con el mundo físico de frente. Nos vemos ahí.

UNIDAD 3: Internet de las cosas (IoT)

Introducción

Imagina que despiertas por la mañana y tu casa ya hizo la mitad del trabajo: el cafetero empezó a preparar el café justo a la hora que te levantas, el termostato ajustó la temperatura para que no sientas frío al salir de la cama, las persianas se abrieron solas para que entre la luz del sol y tu teléfono te avisa que el nivel de la batería de tu coche eléctrico ya está completo. Nada de esto es magia ni ciencia ficción. Es Internet de las cosas, y hoy tú vas a aprender cómo funciona desde adentro, con las manos en los cables y el código corriendo en un microcontrolador real.

Internet de las cosas, que en inglés se dice Internet of Things y por eso lo abreviamos IoT, es una idea sencilla pero poderosa: tomar objetos cotidianos (focos, sensores, cerraduras, regaderas, macetas) y conectarlos a internet para que puedan medir cosas, tomar decisiones y comunicarse entre ellos y contigo. Cuando tu refrigerador "habla" con tu teléfono, cuando un sensor en el campo le dice al sistema de riego que la tierra está seca, cuando una lámpara se enciende sola porque detectó que ya es de noche, estás viendo IoT en acción. En esta unidad vas a construir sistemas así desde cero.

Lo mejor de todo es que no necesitas ser ingeniero ni tener un laboratorio caro. Con un microcontrolador ESP32 que cuesta menos que una comida corrida, un sensor de temperatura de unos pocos pesos, unos cables y tu computadora, ya tienes todo lo necesario para empezar. De hecho, esa es una de las razones por las que el IoT explotó en todo el mundo: el hardware se volvió barato, el software se volvió abierto y gratuito, y la información está disponible para quien quiera aprender. Tú estás justo en el momento correcto para subirte a este tren.

En esta unidad recorrerás el camino completo de un proyecto IoT real. Empezarás entendiendo qué es el IoT y cuál es la arquitectura típica de un sistema de este tipo: sensor, microcontrolador, red, nube y aplicación. Luego conocerás los componentes físicos: sensores que miden el mundo (temperatura, humedad, movimiento, luz) y actuadores que actúan sobre él (motores, relevadores, luces). Después trabajarás con los cerebros del sistema: los microcontroladores ESP32 y Arduino. Aprenderás los protocolos con los que estos aparatos se comunican: WiFi, MQTT y HTTP. Construirás tu primer circuito leyendo un sensor y mostrando el valor, enviarás datos a la nube para verlos en un tablero (dashboard), y aprenderás seguridad básica para que tus dispositivos no sean una puerta abierta. Al final, con los proyectos de riego automático, monitor de calidad del aire y medidor de consumo eléctrico, habrás armado cosas que mucha gente paga caro en tiendas.

Qué es el Internet de las cosas y por qué te importa

Vamos a definirlo de forma clara: el Internet de las cosas es una red de objetos físicos equipados con sensores, software y conectividad que les permite recolectar e intercambiar datos a través de internet. La palabra clave es "cosas": no hablamos de computadoras ni de teléfonos, que ya nacieron conectados, sino de objetos que antes eran "tontos" y ahora se vuelven "inteligentes" al conectarse.

Piensa en una maceta normal. Una maceta tradicional no te dice nada: si la planta se está secando, te enteras cuando las hojas ya están marchitas. Una maceta IoT, en cambio, tiene un sensor de humedad en la tierra, un microcontrolador que lee ese sensor cada cierto tiempo, conexión WiFi para mandar los datos a la nube y una app en tu teléfono que te avisa: "la tierra de tu planta está seca, riégala". La maceta sigue siendo una maceta, pero ahora tiene ojos (el sensor), cerebro (el microcontrolador), voz (la conexión a internet) y memoria (la nube donde se guardan los datos).

¿Por qué te importa aprender esto? Por tres razones. Primera: el mercado laboral. Las empresas buscan gente que sepa de IoT para agricultura, industria, ciudades inteligentes, salud y hogar. Segunda: tus proyectos personales. Con IoT puedes automatizar tu casa, cuidar tus plantas, monitorear tu negocio o crear productos que vender. Tercera: porque es la puerta de entrada perfecta a la tecnología. El IoT junta electrónica, programación, redes y datos en un solo paquete; lo que aprendas aquí te servirá para robótica, inteligencia artificial y desarrollo de software.

Y ojo con algo importante: el IoT no es solo para expertos. Los proyectos que vas a hacer en esta unidad usan los mismos principios que usan las grandes empresas, solo que a escala de escritorio. La diferencia entre tu sensor de temperatura casero y el sistema de monitoreo de un edificio inteligente es de tamaño y de presupuesto, no de concepto. Cuando entiendas el concepto, entenderás los dos.

La arquitectura de un sistema IoT: el camino de los datos

Todo sistema IoT, desde el más sencillo hasta el más complejo, sigue el mismo camino de datos. Memoriza esta cadena porque la vas a ver una y otra vez: sensor, microcontrolador, red, nube, aplicación. Vamos a recorrerla con un ejemplo concreto: un termómetro inteligente que mide la temperatura de tu cuarto y la muestra en tu teléfono.

Primero está el sensor. El sensor es el que "siente" el mundo físico. En nuestro ejemplo, un sensor de temperatura como el DHT11 o el DHT22 mide qué tan caliente o frío está el aire y convierte esa medición en una señal eléctrica. Los sensores son los ojos y oídos del sistema: sin ellos, el microcontrolador estaría ciego. Existen sensores para casi todo: temperatura, humedad, luz, movimiento, distancia, presión, calidad del aire, sonido, nivel de agua.

Después viene el microcontrolador. El microcontrolador es un pequeño chip programable que lee al sensor, procesa la información y decide qué hacer con ella. En nuestro ejemplo, el ESP32 lee la temperatura del sensor cada 10 segundos. Piensa en el microcontrolador como el cerebro local: es pequeño y barato, pero suficiente para leer sensores, hacer cálculos sencillos y mandar mensajes. Los dos microcontroladores más populares para IoT son el Arduino y el ESP32, y los vas a conocer a fondo en esta unidad.

Luego está la red. La red es el camino por donde viajan los datos. El microcontrolador se conecta a tu WiFi casero (o a otras redes como LoRa o red celular, según el proyecto) y envía la temperatura usando un protocolo de comunicación. Los protocolos son las "reglas del idioma" que hablan los aparatos: en esta unidad aprenderás MQTT, que es ligero y perfecto para IoT, y HTTP, que es el mismo que usan las páginas web.

Después está la nube. La nube es un servidor en internet que recibe los datos, los guarda y los organiza. En nuestro ejemplo, un servicio en la nube recibe la temperatura cada 10 segundos y la almacena en una base de datos. La nube es la memoria del sistema: gracias a ella puedes ver el historial de temperaturas de toda la semana, aunque tu microcontrolador se apague.

Al final está la aplicación. La aplicación es lo que tú ves y tocas: un tablero (dashboard) en una página web o una app en tu teléfono que muestra la temperatura actual con gráficas bonitas. Es la cara visible del sistema. Detrás de ese tablero hay todo el camino que acabamos de recorrer, pero el usuario final solo ve el resultado.

Esta arquitectura de cinco pasos es tu mapa. Cada vez que veas un producto IoT en una tienda, pregúntate: ¿cuál es el sensor? ¿cuál es el microcontrolador? ¿por qué red viajan los datos? ¿dónde está la nube? ¿cómo se ve la aplicación? Cuando puedas responder esas cinco preguntas, ya entiendes el sistema. Y en esta unidad vas a construir cada una de esas cinco partes con tus propias manos.

Lo que necesitas para esta unidad

Antes de seguir, hablemos de materiales. Para aprovechar al máximo esta unidad te conviene tener el hardware real, pero si no lo tienes todavía, no te detengas: todo el código y los conceptos se pueden estudiar y simular primero, y cuando consigas los componentes, ya sabrás exactamente qué hacer con ellos.

La lista básica es corta y barata: un ESP32 (placa de desarrollo, la consigues en tiendas de electrónica o en línea), un sensor DHT11 o DHT22 (temperatura y humedad), un sensor PIR (movimiento), una fotorresistencia LDR (luz), unos LED, resistencias, un protoboard (placa de pruebas sin soldadura) y cables dupont. Con eso haces el 90 por ciento de los ejercicios. Si además consigues un relevador (relay) y un sensor de humedad de suelo, podrás armar el proyecto de riego automático completo.

En cuanto a software, necesitas: el IDE de Arduino o Thonny (para programar el ESP32 en MicroPython), Python instalado en tu computadora, una cuenta gratuita en un servicio de MQTT en la nube (te explicaremos cuáles) y un navegador web. Todo es gratuito. En la unidad 1 y 2 de este libro ya instalaste Python y viste conceptos de programación; si vienes de ahí, estás más que listo. Si es tu primer contacto, no te preocupes: el código de esta unidad está explicado línea por línea.

Un consejo de alguien que ya se quemó las pestañas con esto: compra un ESP32 extra si puedes. No porque se vayan a descomponer (son bastante resistentes), sino porque cuando tengas dos podrás hacer proyectos donde uno manda datos y el otro los recibe, y eso te enseña muchísimo sobre comunicación entre dispositivos. Además, siempre es bueno tener repuesto cuando experimentas. Dicho eso, con uno solo es suficiente para toda la unidad.

Explicación detallada

Sensores: los sentidos del sistema

Un sensor es un dispositivo que convierte una magnitud del mundo físico (temperatura, luz, movimiento) en una señal eléctrica que un microcontrolador puede leer. Así de simple y así de poderoso. Sin sensores, tu sistema IoT no sabría nada del mundo real; sería como una persona con los ojos vendados y los oídos tapados tratando de describir un cuarto.

Los sensores se dividen en dos grandes familias: digitales y analógicos. Un sensor digital entrega valores claros y discretos: por ejemplo, un sensor de movimiento PIR te dice "hay movimiento" o "no hay movimiento", un 1 o un 0. Un sensor analógico entrega un rango continuo de valores: una fotorresistencia te dice qué tan intensa es la luz en una escala, por ejemplo de 0 a 4095 en el ESP32. Esta diferencia es importante porque cambia cómo lees el sensor en el código, como verás en los ejemplos.

Vamos a conocer a los cuatro protagonistas de esta unidad. El DHT11 (y su hermano mayor, el DHT22) mide temperatura y humedad del aire. Es digital, barato y facilísimo de usar: con un solo cable de datos le preguntas "¿qué temperatura hace?" y te responde. El DHT11 mide de 0 a 50 grados con precisión de más o menos 2 grados; el DHT22 mide de -40 a 80 grados con precisión de medio grado. Para proyectos caseros el DHT11 basta; para algo más serio, usa el DHT22. Un detalle importante: estos sensores son lentos, solo puedes leerlos una vez cada 2 segundos. Si los lees más rápido, te dan datos basura. Este es un error clásico de principiante y ya te lo advertimos aquí para que no te pase.

El sensor PIR (infrarrojo pasivo) detecta movimiento de personas y animales. Funciona detectando cambios en la radiación infrarroja, es decir, el calor corporal. Es digital: su salida es ALTA cuando detecta movimiento y BAJA cuando no. Lo has visto en acción en las lámparas de los pasillos que se encienden solas cuando pasas. Tiene dos potenciómetros (tornillitos) para ajustar la sensibilidad y el tiempo que la señal se queda activa. Buena práctica: cuando lo conectes por primera vez, déjalo "calentar" unos 30 a 60 segundos sin moverte enfrente; necesita ese tiempo para calibrarse con la temperatura del ambiente.

La fotorresistencia LDR mide la intensidad de la luz. Es analógica: su resistencia eléctrica baja cuando hay mucha luz y sube cuando hay oscuridad. Se conecta en un divisor de voltaje con una resistencia fija, y el microcontrolador lee el voltaje resultante. Con ella puedes hacer que una lámpara se encienda sola cuando oscurece, o medir cuánta luz recibe tu planta. Es el sensor más barato de todos y uno de los más divertidos para experimentar: tapa la LDR con el dedo y mira cómo cambian los valores en tiempo real.

El sensor de humedad de suelo mide cuánta agua hay en la tierra de una maceta. Tiene dos electrodos que se clavan en la tierra; entre más húmeda está la tierra, mejor conduce la electricidad. Es analógico y será la estrella del proyecto de riego automático. Advertencia de

buena práctica: los modelos baratos con electrodos expuestos se corroen con el tiempo por la electrólisis. Para un proyecto permanente, busca la versión capacitiva (no se corroe) o acepta que tendrás que reemplazar el sensor cada algunos meses.

Analogía cotidiana: piensa en los sensores como los sentidos de tu cuerpo. El DHT11 es como tu piel sintiendo el calor y la humedad del ambiente. El PIR es como tu oído detectando que alguien entró al cuarto. La LDR es como tus ojos ajustándose a la luz. Y el microcontrolador es tu cerebro recibiendo todas esas señales. Cuando programas un sistema IoT, en el fondo estás construyendo un pequeño organismo artificial con sentidos, cerebro y la capacidad de hablar por internet.

Errores comunes de principiantes con sensores: primero, no revisar el voltaje. Muchos sensores funcionan a 3.3V y otros a 5V; conectar un sensor de 3.3V a 5V puede quemarlo. El ESP32 trabaja a 3.3V en sus pines, así que verifica siempre la hoja de datos. Segundo, cables flojos en el protoboard: si tus lecturas saltan de forma loca, el 90 por ciento de las veces es un cable mal conectado, no un sensor descompuesto. Tercero, leer el DHT11 más de una vez cada 2 segundos. Cuarto, no calibrar: los sensores baratos tienen un margen de error; compara tu sensor con un termómetro confiable una vez y anota la diferencia para corregirla en el código.

Actuadores: las manos del sistema

Si los sensores son los sentidos, los actuadores son las manos: son los dispositivos que actúan sobre el mundo físico cuando el microcontrolador se los ordena. Un sistema que solo mide pero nunca actúa es un monitor; un sistema que mide Y actúa es un sistema de control, y eso es mucho más interesante.

El actuador más sencillo es el LED. Ya lo conoces: se enciende y se apaga con una señal digital. Pero no lo subestimes: con LEDs aprendes el principio fundamental de todo actuador (recibir una orden y convertirla en acción física) y sirven para prototipos, indicadores de estado y alarmas visuales. En esta unidad los usarás para representar lámparas, alarmas y estados del sistema antes de conectar cosas más potentes.

El relevador (relay) es el actuador que te abre la puerta al mundo real de alto voltaje. Un relevador es un interruptor controlado eléctricamente: con una señal pequeña de 3.3V del ESP32 puedes encender y apagar aparatos de 110V o 220V, como una lámpara de tu casa, un ventilador o una bomba de agua. Por dentro tiene una bobina que al energizarse mueve un contacto metálico; escucharás un "clic" característico cada vez que se activa. Los módulos de relevador para Arduino/ESP32 ya vienen con el circuito de protección incluido y se conectan directo a un pin digital.

Aquí va una advertencia seria, y no es opcional: trabajar con voltaje de casa (110V/220V) puede matarte si lo haces mal. Reglas de oro: nunca manipules cables de corriente alterna con el circuito energizado; usa relevadores en módulos cerrados y bien aislados; si no estás seguro de lo que haces, practica primero con LEDs y aparatos de bajo voltaje (una bomba de agua de 5V por USB, por ejemplo). En los proyectos de esta unidad usaremos relevadores con cargas pequeñas y seguras, pero quiero que interiorices el respeto por la corriente alterna desde hoy.

Otro actuador muy útil es el buzzer (zumbador), que produce sonido. Es perfecto para alarmas: si el sensor de movimiento detecta a alguien de noche, el buzzer suena. También está el servomotor, un motorcito que puede girar a ángulos precisos (0 a 180 grados), ideal para abrir una compuerta, mover una cámara o simular la apertura de una puerta. Y el motor de corriente directa con una bomba de agua pequeña es el corazón del proyecto de riego automático.

Analogía cotidiana: si el sistema IoT fuera una persona, los sensores serían sus ojos y oídos, el microcontrolador su cerebro, y los actuadores sus manos y su voz. Cuando el cerebro decide "hace calor", las manos prenden el ventilador (relevador) y la voz avisa con un pitido (buzzer). Programar actuadores es darle a tu sistema la capacidad de cambiar el mundo, no solo de observarlo.

Buenas prácticas con actuadores: primero, nunca conectes un motor o relevador directo a un pin del microcontrolador si consume más corriente de la que el pin puede dar (los pines del ESP32 dan unos 12 mA como máximo seguro). Usa los módulos diseñados para esto, que ya traen transistores y diodos de protección. Segundo, alimenta los actuadores potentes con una fuente externa y une las tierras (GND) de la fuente con la del microcontrolador. Tercer error común: el relevador "al revés". Muchos módulos de relevador se activan con señal BAJA (LOW), no con ALTA; si tu lámpara hace lo contrario de lo que esperas, revisa este detalle antes de pensar que algo está quemado.

Microcontroladores: ESP32 y Arduino

El microcontrolador es el cerebro del sistema IoT. Es un chip que integra procesador, memoria y pines de entrada/salida en un solo paquete, diseñado para hacer una tarea específica de forma eficiente y barata. A diferencia de tu computadora, que hace muchas cosas a la vez, un microcontrolador hace una sola cosa muy bien: leer sensores, tomar decisiones y comunicarse.

Arduino es la plataforma con la que la mayoría empieza. Técnicamente "Arduino" es varias cosas: una familia de placas (como el Arduino UNO), un lenguaje de programación basado en

C++ y un entorno de desarrollo (el IDE de Arduino). El Arduino UNO usa el chip ATmega328P, tiene 14 pines digitales y 6 analógicos, y se programa por USB. Es robusto, tiene una comunidad gigantesca y miles de tutoriales. Su limitación para IoT: el UNO clásico no trae WiFi integrado; para conectarlo a internet necesitas un módulo extra (como el ESP8266) o usar placas Arduino más modernas.

El ESP32 es el rey actual del IoT casero y profesional de bajo costo. Es un chip de la empresa Espressif que integra WiFi y Bluetooth, dos núcleos de procesador, y una buena cantidad de pines digitales y analógicos, todo por un precio muy bajo. Las placas de desarrollo ESP32 (como la ESP32 DevKit) se programan por USB igual que el Arduino, y lo mejor es que puedes programarlas de dos formas: con el IDE de Arduino usando C++ (aprovechando todo lo que ya sabes de Arduino) o con MicroPython, que es una versión ligera de Python para microcontroladores. En esta unidad usaremos ambas opciones para que elijas la que más te guste.

Comparémoslos de forma práctica. Si tu proyecto necesita WiFi y quieres gastar poco, el ESP32 gana por goleada: WiFi integrado, más rápido, más memoria, más barato que un Arduino UNO más módulo WiFi. Si estás empezando desde cero en electrónica y quieres la mayor cantidad de tutoriales y shields (placas que se montan encima), el Arduino UNO sigue siendo una gran escuela. Para los proyectos de esta unidad usaremos el ESP32 como placa principal, porque el WiFi integrado simplifica todo: una sola placa, un solo cable USB, y ya estás en internet.

Conceptos clave que debes dominar del ESP32: los pines GPIO (General Purpose Input/Output) son los pines que puedes configurar como entrada (para leer sensores) o salida (para controlar actuadores). El ESP32 tiene ADC (convertidor analógico a digital) para leer sensores analógicos como la LDR, con resolución de 12 bits (valores de 0 a 4095). También tiene PWM para "simular" salidas analógicas, útil para atenuar un LED o controlar la velocidad de un motor. Y tiene modos de bajo consumo (deep sleep) que permiten que un sensor funcione meses con una batería: el chip se duerme, despierta, mide, envía y se vuelve a dormir.

Errores comunes de principiantes con microcontroladores: primero, no instalar los drivers o la definición de la placa. Si tu computadora no reconoce el ESP32, casi siempre es el driver del chip USB-serial (CP2102 o CH340) o que no agregaste la URL de Espressif en el IDE de Arduino. Segundo, usar pines "problemáticos" del ESP32: algunos pines se usan durante el arranque (strapping pins) y pueden causar que la placa no inicie si los conectas mal; quédate con pines seguros como el 13, 14, 25, 26, 27 y 32 para empezar. Tercero, alimentar el ESP32 solo por USB cuando conectas muchos actuadores: el USB da corriente limitada; si tu proyecto se reinicia solo, probablemente le falta corriente. Cuarto, confundir 3.3V con 5V: el ESP32 es

un dispositivo de 3.3V; el pin VIN acepta 5V del USB, pero los pines GPIO no toleran 5V en sus entradas.

Buena práctica fundamental: antes de conectar todo el circuito final, prueba cada componente por separado con un programa mínimo. Primero el LED parpadeante, luego el sensor solo, luego el WiFi solo. Cuando cada parte funciona aislada, las unes. Esta estrategia de "divide y vencerás" te ahorrará horas de frustración buscando un error en un circuito completo cuando en realidad solo un cable estaba mal desde el inicio.

Protocolos de comunicación: WiFi, MQTT y HTTP

Los dispositivos IoT necesitan hablar entre ellos y con la nube, y para eso usan protocolos: conjuntos de reglas que definen cómo se empaquetan, envían y reciben los mensajes. Es como el idioma y la etiqueta de una conversación: no basta con gritar datos al aire, hay que hacerlo de forma que el otro entienda. En esta unidad trabajarás con tres: WiFi, MQTT y HTTP.

WiFi no es exactamente un protocolo de mensajes, sino la tecnología de red que conecta tu ESP32 a internet sin cables. Cuando programas el ESP32, le das el nombre de tu red (SSID) y la contraseña, y él se conecta como lo hace tu teléfono. Para proyectos IoT caseros, el WiFi es la opción más cómoda porque ya tienes el router en casa. Sus limitaciones: consume bastante energía (malo para proyectos con batería) y su alcance es el de tu casa u oficina. Existen alternativas para otros casos: Bluetooth para distancias cortas, LoRa para kilómetros con poca energía, y red celular (4G/5G) para lugares sin WiFi. Pero para aprender y para la mayoría de proyectos caseros, WiFi es perfecto.

HTTP es el protocolo de la web: el mismo que usa tu navegador para pedir páginas. Funciona con un modelo simple de pregunta y respuesta: el cliente pide ("GET dame la temperatura") y el servidor responde ("aquí está: 25 grados"). El ESP32 puede actuar como cliente HTTP: cada cierto tiempo le manda los datos del sensor a un servidor con una petición POST, o le pregunta a un servicio web con un GET. Ventajas: es universal, fácil de depurar (puedes probar las peticiones desde el navegador) y todos los servicios en la nube lo hablan. Desventajas: es "pesado" para un microcontrolador (las cabeceras HTTP son largas) y el modelo pregunta-respuesta no es ideal cuando quieres que el servidor le avise algo al dispositivo al momento.

MQTT es el protocolo favorito del IoT y el que más usarás en esta unidad. Fue diseñado para conexiones inestables y aparatos con pocos recursos. Su modelo es publicación/suscripción con un intermediario llamado broker. Funciona así: el ESP32 se conecta al broker y publica mensajes en "temas" (topics), que son como canales con nombre, por ejemplo

"casa/sala/temperatura". Otro dispositivo (o tu teléfono, o un programa en tu PC) se suscribe a ese tema y el broker le reenvía cada mensaje que llegue. El que publica no necesita saber quién lo escucha, y el que escucha no necesita saber quién publica: el broker hace de cartero.

Analogía cotidiana: MQTT es como un grupo de WhatsApp con un administrador. Tú mandas un mensaje al grupo "casa/sala/temperatura" (publicas en el tema) y todos los miembros del grupo (suscriptores) lo reciben al instante. No necesitas saber quién está en el grupo ni mandar el mensaje uno por uno. HTTP, en cambio, es como llamar por teléfono: marcas, preguntas, te responden y cuelgas. Para mandar la temperatura cada 10 segundos a varios interesados, el grupo de WhatsApp (MQTT) es mucho más eficiente que llamar a cada uno por teléfono (HTTP).

MQTT tiene además tres niveles de calidad de servicio (QoS): 0 (lo mando una vez y no me importa si se pierde), 1 (me aseguro de que llegue al menos una vez) y 2 (me aseguro de que llegue exactamente una vez). Para lecturas de sensores donde el siguiente dato llega en 10 segundos, QoS 0 basta. También tiene "mensajes retenidos" (el broker guarda el último mensaje del tema para quien se suscriba después) y "última voluntad" (un mensaje que el broker publica automáticamente si tu dispositivo se desconecta de golpe, ideal para alarmas de "sensor desconectado").

En la práctica usarás un broker público gratuito para tus pruebas (como los de prueba de Mosquitto o HiveMQ) o instalarás Mosquitto en tu propia computadora. Más adelante, servicios en la nube como los tableros IoT ya traen broker MQTT integrado. Buena práctica: en tus temas usa una estructura jerárquica clara, por ejemplo "lucas-iot/casa/sala/temperatura". Así, cuando tengas 20 sensores, podrás suscribirte a "lucas-iot/casa/#" y recibir todo lo de tu casa de un jalón (el "#" es un comodín).

Errores comunes con protocolos: primero, olvidar que el ESP32 y tu programa de PC deben conectarse al MISMO broker para verse. Segundo, temas mal escritos: "Casa/Sala/Temperatura" y "casa/sala/temperatura" son temas distintos (MQTT distingue mayúsculas). Tercero, publicar mensajes enormes o rapidísimo (cada 100 ms) y saturar la conexión: para sensores ambientales, cada 5 a 30 segundos es más que suficiente. Cuarto, no manejar reconexiones: el WiFi se cae a veces; tu código debe intentar reconectarse solo, no quedarse colgado para siempre.

De la nube al tablero: cómo ver tus datos

Ya tienes el sensor midiendo, el microcontrolador procesando y la red transportando. Ahora, ¿a dónde llegan los datos y cómo los ves? Aquí entran la nube y el tablero (dashboard).

La nube, en el contexto IoT, es simplemente una computadora en internet que siempre está encendida, esperando tus datos. Puede ser un servidor que rentas, una computadora en tu casa (como una Raspberry Pi), o un servicio especializado en IoT. Sus trabajos son tres: recibir los datos (por MQTT o HTTP), guardarlos (en una base de datos, para tener historial) y mostrarlos o reenviarlos (al tablero o a tu teléfono). Cuando mandas la temperatura cada 10 segundos durante un mes, la nube acumula más de 250,000 lecturas; con ese historial puedes ver gráficas, detectar patrones y tomar decisiones.

El tablero (dashboard) es la página web donde ves todo bonito: gráficas de temperatura, medidores de humedad, botones para prender y apagar actuadores, mapas y alertas. Existen plataformas IoT gratuitas que te dan el broker MQTT, la base de datos y el tablero todo junto, sin que programes el servidor. En los ejemplos de esta unidad aprenderás el flujo completo y verás cómo conectar tu ESP32 a un tablero para ver tus datos en tiempo real desde cualquier parte del mundo.

Analogía cotidiana: la nube es como la bodega y la oficina de una empresa. Los repartidores (tus ESP32) llegan todo el día a dejar paquetes (datos). La bodega los guarda ordenados (base de datos) y la oficina atiende a los clientes (el tablero) que llegan a preguntar "¿cómo va la temperatura de hoy?". Tú, como dueño, no necesitas estar en la bodega: entras a la oficina (abres el tablero en tu teléfono) y ahí está todo resumido y graficado.

Buenas prácticas con datos en la nube: primero, no mandes datos más seguido de lo necesario; cada lectura cuesta energía, ancho de banda y espacio. Segundo, manda los datos en un formato estándar como JSON (verás ejemplos en esta unidad), que casi todas las plataformas entienden. Tercero, incluye siempre una marca de tiempo o un identificador del dispositivo en tus mensajes, para que cuando tengas varios sensores sepas cuál dato es de cuál. Cuarto, piensa en qué pasa si se va el internet: un buen sistema IoT sigue funcionando localmente (por ejemplo, el riego automático debe regar aunque no haya WiFi) y sincroniza con la nube cuando vuelve la conexión.

Seguridad básica en dispositivos IoT

Aquí viene un tema que muchos tutoriales se saltan y que tú no te vas a saltar: la seguridad. Un dispositivo IoT conectado a internet es una computadora más en tu red, y si la dejas desprotegida, alguien más puede controlarla. No es paranoia: existen botnets (redes de aparatos secuestrados) formadas por cámaras y focos inteligentes con contraseñas de fábrica. Tus proyectos deben nacer seguros desde el primer día.

La regla número uno es: cambia todas las credenciales por defecto. Muchos ejemplos de internet usan usuario "admin" y contraseña "admin" o redes WiFi abiertas en el código de

ejemplo. Eso está bien para una prueba de 5 minutos en tu escritorio, pero jamás para un dispositivo que se queda conectado. Usa contraseñas largas y únicas para el WiFi, el broker MQTT y los tableros. Y nunca subas tu código con contraseñas a GitHub o a foros públicos: si compartes código, borra las credenciales o usa un archivo de configuración separado que no compartas.

La regla número dos es: usa conexiones cifradas cuando sea posible. MQTT puede funcionar sobre TLS (puerto 8883 en vez de 1883), que cifra los mensajes como lo hace el candadito de tu navegador. HTTP tiene su versión segura, HTTPS. Para tus primeros proyectos en red local puedes usar conexiones sin cifrar, pero en cuanto publiques datos en internet o controles actuadores remotamente, activa el cifrado. La mayoría de brokers en la nube gratuitos ya ofrecen MQTT con TLS; úsalo.

La regla número tres es: dale a cada dispositivo los permisos mínimos necesarios. Si tu sensor de temperatura solo necesita publicar en el tema "casa/sala/temperatura", configúrale un usuario MQTT que solo pueda publicar ahí y nada más. Así, si alguien consigue esa credencial, no puede suscribirse a tus cámaras ni publicar comandos a tus relevadores. Es el principio del menor privilegio, y aplica igual en sistemas gigantes.

La regla número cuatro es: mantén actualizado el firmware. Los fabricantes del ESP32 y las librerías que usas corrigen vulnerabilidades con el tiempo; revisa cada pocos meses si hay versiones nuevas del core de ESP32 o de tus librerías MQTT y actualiza tus dispositivos. Y desactiva lo que no uses: si tu ESP32 no necesita Bluetooth, apágalo; cada servicio activo es una puerta potencial.

Analogía cotidiana: asegurar tu dispositivo IoT es como asegurar tu casa. Cambiar la contraseña por defecto es cambiar la chapa que venía de fábrica (todos los ladrones saben cuál es). El cifrado TLS es como mandar tus cartas en sobre sellado en vez de en postal abierta. Los permisos mínimos son como darle al jardinero solo la llave del jardín, no la de toda la casa. Y actualizar el firmware es como reparar la ventana que se rompió: si la dejas rota, alguien entrará por ahí.

Errores comunes de seguridad en principiantes: dejar el ESP32 creando una red WiFi abierta de configuración permanentemente; exponer el tablero a internet sin contraseña "porque nadie va a encontrar la dirección" (los bots escanean todo internet automáticamente, sí la van a encontrar); usar la misma contraseña en todos los dispositivos y servicios; y olvidar que los temas MQTT sin protección los puede leer cualquiera conectado al mismo broker público. En los brokers públicos de prueba, asume que todo lo que publiques lo puede leer un desconocido: úsalos solo para aprender, con datos inventados o sin importancia, y con nombres de tema únicos y difíciles de adivinar.

Ejemplos paso a paso

Ejemplo 1: leer un sensor DHT22 y una fotorresistencia con MicroPython en el ESP32

Este es tu "hola mundo" del IoT: el primer circuito real. Vas a conectar un sensor DHT22 (temperatura y humedad) y una fotorresistencia LDR al ESP32, y el programa leerá ambos cada 5 segundos y mostrará los valores en la consola. Aquí aprendes el flujo sensor -> microcontrolador, la base de todo lo demás.

Material: ESP32 DevKit, sensor DHT22, fotorresistencia LDR, resistencia de 10k ohm, protoboard y cables dupont.

Conexiones: el DHT22 tiene 3 pines útiles (algunos módulos traen 4, ignora el no conectado). Conecta VCC del DHT22 a 3V3 del ESP32, GND a GND, y el pin de datos al GPIO 14 del ESP32. Para la LDR arma un divisor de voltaje: conecta un extremo de la LDR a 3V3, el otro extremo al GPIO 34 y a una resistencia de 10k ohm, y el otro extremo de la resistencia a GND. Así, el GPIO 34 lee un voltaje que cambia con la luz.

Instala el firmware de MicroPython en tu ESP32 (desde micropython.org, usando la herramienta esptool o Thonny) y abre Thonny. Escribe este programa y guárdalo como main.py en la placa:

```

from machine import Pin, ADC
from time import sleep
import dht

# Configuración de sensores
sensor_temp = dht.DHT22(Pin(14)) # DHT22 en el GPIO 14
sensor_luz = ADC(Pin(34))        # LDR en el GPIO 34 (entrada analógica)
sensor_luz.atten(ADC.ATTN_11DB)  # Rango completo 0-3.3V
sensor_luz.width(ADC.WIDTH_12BIT) # Resolución 0-4095

print("Estación de monitoreo iniciada")
print("Lee temperatura, humedad y luz cada 5 segundos")
print("-" * 40)

while True:
    try:
        sensor_temp.measure()          # Pide la medición al DHT22
        temp = sensor_temp.temperature() # Grados Celsius
        hum = sensor_temp.humidity()    # Porcentaje de humedad
        luz = sensor_luz.read()         # Valor 0-4095

        # Clasifica la luz con palabras simples
        if luz > 3000:
            estado_luz = "muy iluminado"
        elif luz > 1500:
            estado_luz = "iluminado"
        elif luz > 500:
            estado_luz = "tenue"
        else:
            estado_luz = "oscuro"

        print("Temp: {:.1f} C | Humedad: {:.1f} % | Luz: {} ({}).format(
            temp, hum, luz, estado_luz))
    except OSError:
        # El DHT22 a veces falla una lectura; no es grave, se reintenta
        print("Lectura del DHT22 fallida, reintentando...")

    sleep(5) # Espera 5 segundos (el DHT22 necesita 2 s mínimo)

```

Qué hace cada parte: importamos las librerías de hardware (machine), tiempo y el driver del DHT. Configuramos el DHT22 en el pin 14 y el ADC en el pin 34 con rango completo de voltaje y 12 bits de resolución. En el ciclo infinito medimos, leemos los tres valores, clasificamos la luz en palabras y los imprimimos. El try/except evita que el programa muera si el DHT22 falla una lectura ocasional, algo normal en este sensor.

Prueba: corre el programa y abre la consola (shell) de Thonny. Deberías ver líneas como "Temp: 24.5 C | Humedad: 55.0 % | Luz: 2800 (iluminado)". Tapa la LDR con el dedo y verás cómo el valor de luz baja y el estado cambia a "oscuro". Respira cerca del DHT22 (sin tocarlo) y verás cómo suben la temperatura y la humedad. Si ves "Lectura fallida" de vez en cuando, es normal; si es siempre, revisa las conexiones y que el pin de datos sea el correcto.

Ejemplo 2: publicar lecturas por MQTT con Arduino C++ en el ESP32

Ahora subimos de nivel: el ESP32 se conecta a tu WiFi y publica la temperatura y humedad a un broker MQTT, para que cualquier otro aparato o programa suscrito las reciba en tiempo real. Este es el flujo sensor -> microcontrolador -> red. Usa el mismo circuito del ejemplo 1 (DHT22 en GPIO 14).

En el IDE de Arduino, instala desde el gestor de librerías: "DHT sensor library" de Adafruit, "Adafruit Unified Sensor" y "PubSubClient" de Nick O'Leary. En el gestor de placas instala el soporte "esp32" de Espressif. Elige tu placa (ESP32 Dev Module) y el puerto correcto.

Para el broker usaremos uno público de prueba. Escribe tu nombre de tema único (por ejemplo con tu nombre y un número al azar) para no mezclarte con otros estudiantes. El código:

```
#include <WiFi.h>
#include <PubSubClient.h>
#include <DHT.h>

// --- Configuracion: cambia estos valores ---
const char* WIFI_SSID = "NOMBRE_DE_TU_WIFI";
const char* WIFI_PASS = "CONTRASENA_DE_TU_WIFI";
const char* MQTT_BROKER = "test.mosquitto.org";
const int   MQTT_PORT = 1883;
const char* TOPIC_TEMP = "lucas-iot/tu-nombre-unico/sala/temperatura";
const char* TOPIC_HUM  = "lucas-iot/tu-nombre-unico/sala/humedad";

#define DHT_PIN 14
#define DHT_TYPE DHT22

DHT dht(DHT_PIN, DHT_TYPE);
WiFiClient wifiClient;
PubSubClient mqtt(wifiClient);
unsigned long ultimaPublicacion = 0;

void conectarWiFi() {
  Serial.print("Conectando a WiFi ");
  WiFi.begin(WIFI_SSID, WIFI_PASS);
  while (WiFi.status() != WL_CONNECTED) {
```

```

    delay(500);
    Serial.print(".");
}
Serial.println("\nWiFi conectado. IP: " + WiFi.localIP().toString());
}

void conectarMQTT() {
    while (!mqtt.connected()) {
        Serial.print("Conectando al broker MQTT... ");
        // ID de cliente unico: usa la MAC para no chocar con otros
        String clientId = "esp32-" + String((uint32_t)ESP.getEfuseMac(), HEX);
        if (mqtt.connect(clientId.c_str())) {
            Serial.println("conectado");
        } else {
            Serial.print("fallo, reintentando en 5 s (codigo ");
            Serial.print(mqtt.state());
            Serial.println(")");
            delay(5000);
        }
    }
}

void setup() {
    Serial.begin(115200);
    dht.begin();
    mqtt.setServer(MQTT_BROKER, MQTT_PORT);
    conectarWiFi();
}

void loop() {
    if (WiFi.status() != WL_CONNECTED) {
        conectarWiFi(); // Si se cae el WiFi, reconecta solo
    }
    if (!mqtt.connected()) {
        conectarMQTT(); // Si se cae MQTT, reconecta solo
    }
    mqtt.loop();

    // Publica cada 10 segundos sin bloquear el programa
    if (millis() - ultimaPublicacion >= 10000) {
        ultimaPublicacion = millis();
        float t = dht.readTemperature();
        float h = dht.readHumidity();

        if (isnan(t) || isnan(h)) {

```



```

    Serial.println("Error leyendo el DHT22");
    return;
}

char bufTemp[8], bufHum[8];
dtostrf(t, 4, 1, bufTemp);    // Convierte float a texto "24.5"
dtostrf(h, 4, 1, bufHum);

mqtt.publish(TOPIC_TEMP, bufTemp);
mqtt.publish(TOPIC_HUM, bufHum);
Serial.print("Publicado -> temp: ");
Serial.print(bufTemp);
Serial.print(" C, hum: ");
Serial.println(bufHum);
}
}

```

Puntos clave del código: conectarWiFi y conectarMQTT se aseguran de que el equipo siempre intente reconectarse solo; un buen dispositivo IoT nunca se queda colgado en silencio. El ID de cliente usa la MAC del chip para ser único. Publicamos cada 10 segundos usando millis() en vez de delay(), para que el programa siga atendiendo la conexión MQTT entre publicaciones. Los valores se convierten a texto porque MQTT transporta bytes y el texto es lo más compatible.

Prueba: sube el programa y abre el monitor serial a 115200 baudios. Verás la conexión al WiFi, la conexión al broker y las publicaciones cada 10 segundos. Para comprobar que los mensajes salen a internet, en tu computadora instala un cliente MQTT (como MQTT Explorer) o usa una página web de prueba, conéctate al mismo broker y suscríbete a tu tema: verás llegar los valores en vivo. Ese momento en que el dato viaja del sensor de tu escritorio a tu pantalla por internet es cuando el IoT "hace clic" en tu cabeza.

Ejemplo 3: script Python que se suscribe al MQTT y guarda los datos (la "mini nube")

El ESP32 ya publica. Ahora necesitas quien escuche, guarde y muestre: ese es el trabajo de la nube. En este ejemplo escribes en tu computadora un programa Python que se suscribe a tus temas MQTT, guarda cada lectura en un archivo CSV (tu mini base de datos) y muestra un resumen en pantalla. Es el flujo red -> nube simplificado, y te enseña exactamente qué hace un servidor IoT por dentro.

Instala la librería con: `pip install paho-mqtt`. Crea el archivo `estacion_nube.py`:

```
import csv
```

```

import json
from datetime import datetime
from collections import deque
import paho.mqtt.client as mqtt

BROKER = "test.mosquitto.org"
PUERTO = 1883
TOPIC_BASE = "lucas-iot/tu-nombre-unico/sala" # MISMO prefijo del ESP32
ARCHIVO_CSV = "lecturas_sala.csv"

# Guardamos las ultimas 10 lecturas de cada variable para el resumen
historial = {"temperatura": deque(maxlen=10), "humedad": deque(maxlen=10)}

def guardar_csv(fecha, variable, valor):
    primera_vez = False
    try:
        with open(ARCHIVO_CSV, "r"):
            pass
    except FileNotFoundError:
        primera_vez = True
    with open(ARCHIVO_CSV, "a", newline="") as f:
        w = csv.writer(f)
        if primera_vez:
            w.writerow(["fecha_hora", "variable", "valor"])
        w.writerow([fecha, variable, valor])

def al_conectar(cliente, userdata, flags, rc, props=None):
    print("Conectado al broker, codigo:", rc)
    cliente.subscribe(TOPIC_BASE + "/"#") # # = todo lo de este prefijo
    print("Suscrito a:", TOPIC_BASE + "/"#")

def al_recibir(cliente, userdata, msg):
    variable = msg.topic.split("/")[-1] # "temperatura" o "humedad"
    try:
        valor = float(msg.payload.decode("utf-8"))
    except ValueError:
        print("Mensaje ignorado (no es numero):", msg.payload)
        return
    fecha = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    guardar_csv(fecha, variable, valor)
    if variable in historial:
        historial[variable].append(valor)
        datos = list(historial[variable])
        promedio = sum(datos) / len(datos)
        print("[{}] {} = {} (promedio ultimas {}: {:.1f})".format(

```

```

        fecha, variable, valor, len(datos), promedio))

cliente = mqtt.Client(mqtt.CallbackAPIVersion.VERSION2)
cliente.on_connect = al_conectar
cliente.on_message = al_recibir

print("Mini nube iniciada. Escuchando datos del ESP32...")
print("Presiona Ctrl+C para detener.")
cliente.connect(BROKER, PUERTO, 60)
cliente.loop_forever()

```

Qué hace: al conectarse se suscribe a todo lo que publique tu ESP32 bajo tu prefijo (el comodín #). Cada mensaje que llega lo convierte a número, le pone fecha y hora, lo guarda en el CSV y actualiza un promedio de las últimas 10 lecturas que muestra en pantalla. El CSV es tu historial permanente: ábrelo en Excel o en cualquier hoja de cálculo y podrás graficar la temperatura del día.

Prueba completa del sistema: primero corre este script en tu PC, luego enciende el ESP32 del ejemplo 2. Verás cómo los datos empiezan a llegar solos a tu "nube". Acabas de construir el flujo completo: sensor -> microcontrolador -> red (WiFi + MQTT) -> nube (tu script). Solo falta el tablero bonito, que verás en la sección de práctica y proyectos usando una plataforma IoT gratuita donde este mismo flujo se conecta a gráficas web en minutos.

Archivo de configuración JSON: para no dejar tus credenciales regadas en el código, es buena práctica guardarlas en un archivo config.json que no compartes con nadie:

```

{
  "wifi_ssid": "NOMBRE_DE_TU_WIFI",
  "wifi_password": "CONTRASENA_DE_TU_WIFI",
  "mqtt_broker": "test.mosquitto.org",
  "mqtt_port": 1883,
  "topic_base": "lucas-iot/tu-nombre-unico",
  "intervalo_segundos": 10
}

```

En MicroPython lo leerías así al inicio de tu programa:

```
import json

with open("config.json") as f:
    cfg = json.load(f)

WIFI_SSID = cfg["wifi_ssid"]
WIFI_PASS = cfg["wifi_password"]
BROKER = cfg["mqtt_broker"]
```

Y en Python de PC igual de fácil. Si subes tu proyecto a GitHub o lo compartes, sube un archivo `config_ejemplo.json` con valores falsos y agrega `config.json` a tu `.gitignore`. Tus contraseñas nunca salen de tu máquina. Este hábito, que parece exagerado en un ejercicio, es exactamente lo que hacen los profesionales, y te evitará disgustos cuando tus proyectos crezcan.

Práctica

Ejercicio 1: alarma de movimiento con LED y buzzer

Enunciado: conecta un sensor PIR y un buzzer al ESP32 para crear una alarma: cuando el PIR detecte movimiento, el buzzer suena y un LED se enciende durante 5 segundos. Usa MicroPython. El PIR va al GPIO 13, el LED al GPIO 25 y el buzzer al GPIO 26.

Solución paso a paso:

Paso 1: conecta el PIR. VCC a 3V3, GND a GND y OUT al GPIO 13. Conecta el LED con una resistencia de 220 ohm en serie al GPIO 25 (la resistencia protege al LED y al pin). Conecta el buzzer activo al GPIO 26 y a GND (los buzzers activos suenan con solo darles voltaje; si el tuyo es pasivo necesitarías generar un tono con PWM, revisa tu modelo).

Paso 2: deja que el PIR se calibre. Al energizar el circuito, no te muevas frente al sensor durante 60 segundos. Si lo pruebas antes, dará falsas alarmas y pensarás que está descompuesto.

Paso 3: escribe el programa:

```

from machine import Pin
from time import sleep

pir = Pin(13, Pin.IN)      # Sensor PIR como entrada
led = Pin(25, Pin.OUT)     # LED como salida
buzzer = Pin(26, Pin.OUT)  # Buzzer como salida

print("Alarma activada. Calibrando 60 s, no te muevas...")
sleep(60)
print("Sistema vigilando.")

while True:
    if pir.value() == 1:    # Hay movimiento
        print("Movimiento detectado!")
        led.on()
        buzzer.on()
        sleep(5)           # Alarma 5 segundos
        led.off()
        buzzer.off()
        sleep(2)           # Pausa antes de volver a vigilar
        sleep(0.1)         # Revisa el sensor 10 veces por segundo

```

Paso 4: prueba caminando frente al sensor. Ajusta los potenciómetros del PIR si es muy sensible o muy sordo: uno controla la sensibilidad (alcance) y el otro el tiempo que la salida se queda en alto.

Qué aprendiste: a leer un sensor digital, a controlar dos actuadores a la vez y la importancia de la calibración. Este mismo patrón (sensor digital -> condición -> actuadores) sirve para lámparas automáticas, alarmas de puerta y sistemas de presencia.

Ejercicio 2: lámpara que se enciende sola cuando oscurece

Enunciado: usa la LDR y un relevador para encender una lámpara de 5V (o un LED potente que la represente) automáticamente cuando la luz ambiental baje de cierto nivel, y apagarla cuando amanezca. Agrega histéresis: el umbral de encendido debe ser más bajo que el de apagado, para que la lámpara no parpadee cuando la luz está justo en el límite. Usa Arduino C++.

Solución paso a paso:

Paso 1: conecta la LDR con su divisor de voltaje al GPIO 34 (como en el ejemplo 1). Conecta el pin IN del módulo relevador al GPIO 27, VCC del módulo a 5V (VIN del ESP32) y GND a GND. Conecta tu lámpara de 5V al lado de carga del relevador siguiendo el diagrama de tu módulo (terminal COM y NO para que encienda al activarse).

Paso 2: primero mide. Sube un programa que solo imprima el valor de la LDR cada segundo, y anota qué valores ves con luz de día, con luz tenue y a oscuras. Supongamos que ves: día = 3000, tenue = 1200, oscuro = 200. Elige umbral de encendido = 800 y de apagado = 1500.

Paso 3: escribe el programa con histéresis:

```
const int PIN_LDR = 34;
const int PIN_RELAY = 27;
const int UMBRAL_ENCENDER = 800;    // Si baja de aqui, prende
const int UMBRAL_APAGAR = 1500;    // Si sube de aqui, apaga

bool lamparaEncendida = false;

void setup() {
  Serial.begin(115200);
  pinMode(PIN_RELAY, OUTPUT);
  digitalWrite(PIN_RELAY, HIGH);    // Muchos modulos se activan en LOW:
                                     // HIGH = relevador apagado al inicio
}

void loop() {
  int luz = analogRead(PIN_LDR);

  if (!lamparaEncendida && luz < UMBRAL_ENCENDER) {
    digitalWrite(PIN_RELAY, LOW);    // Activa el relevador
    lamparaEncendida = true;
    Serial.println("Oscurecio: lampara ENCENDIDA");
  }
  else if (lamparaEncendida && luz > UMBRAL_APAGAR) {
    digitalWrite(PIN_RELAY, HIGH);    // Desactiva el relevador
    lamparaEncendida = false;
    Serial.println("Amanecio: lampara APAGADA");
  }

  delay(1000);
}
```

Paso 4: prueba tapando la LDR con la mano (la lámpara debe encender) y alumbrándola con la lámpara de tu celular (debe apagarse). Verifica que cerca del umbral no parpadee: esa es la

histéresis trabajando. Si tu módulo de relevador se activa al revés (hace lo contrario), invierte los HIGH y LOW: es el detalle del que te advertimos en la teoría.

Qué aprendiste: control automático con histéresis, una técnica que se usa en termostatos, refrigeradores y climas de verdad. También practicaste el uso seguro del relevador con cargas pequeñas antes de pensar en voltaje de casa.

Ejercicio 3: tablero en la nube con tu temperatura en vivo

Enunciado: lleva los datos de tu ESP32 a un tablero web gratuito para ver la temperatura y humedad en gráficas desde tu teléfono. Usa la plataforma gratuita de tu elección que soporte MQTT (hay varias; elige una con plan gratuito y tablero arrastrable). El ESP32 publicará a ese servicio en vez del broker de prueba.

Solución paso a paso:

Paso 1: crea tu cuenta gratuita en la plataforma IoT y crea un "dispositivo" nuevo. La plataforma te dará: dirección del broker MQTT, puerto (casi siempre 1883, o 8883 con TLS), usuario y contraseña (un token), y los temas o el formato exacto donde espera los datos. Anótalos.

Paso 2: modifica el código del ejemplo 2: cambia MQTT_BROKER, MQTT_PORT, agrega usuario y contraseña en `mqtt.connect()`, y cambia los temas a los que pide la plataforma. Muchas plataformas aceptan JSON en un solo tema, así que puedes publicar algo como `{"temperatura": 24.5, "humedad": 55.0}` en vez de dos temas separados. Para eso, arma el texto JSON con `snprintf`.

Paso 3: en el tablero de la plataforma, agrega dos widgets: una gráfica de línea para temperatura y un medidor (gauge) para humedad. Configura cada widget para que lea el campo correspondiente de tu dispositivo. Guarda el tablero.

Paso 4: enciende el ESP32 y abre el tablero en tu teléfono. Deberías ver los datos llegando en vivo. Configura una alerta: "avísame por correo si la temperatura pasa de 30 grados". Sopla aire caliente cerca del sensor (con cuidado, sin quemarlo) y comprueba que la alerta llega.

Paso 5 (buena práctica): activa la conexión TLS si tu plataforma la ofrece (puerto 8883) y guarda tus credenciales en el `config.json` como viste en el ejemplo 3, no escritas directo en el código que compartes.

Qué aprendiste: el flujo completo hasta la aplicación, que es lo que el usuario final ve. Entendiste que la "nube" no es magia: es un broker MQTT más una base de datos más una página web, y que hay servicios que te dan las tres cosas para que te concentres en tu dispositivo. Con esto ya puedes presumir un proyecto IoT real y funcional.

Ejercicio 4: riego automático para una maceta

Enunciado: arma un sistema que riega tu planta solo. Un sensor de humedad de suelo clavado en la maceta mide la tierra; cuando está seca, el ESP32 activa 5 segundos una mini bomba de agua de 5V (por USB) mediante un relevador; cuando está húmeda, no hace nada. Además publica el nivel de humedad por MQTT cada 30 segundos para verlo en tu tablero. Debe seguir regando aunque se caiga el WiFi.

Solución paso a paso:

Paso 1: conecta el sensor de humedad de suelo. Los módulos típicos tienen salida analógica (AO): conéctala al GPIO 35, VCC a 3V3 y GND a GND. Clava los electrodos en la tierra de la maceta, sin que toquen el fondo. Conecta la mini bomba: su cable positivo pasa por el relevador (COM y NO) y se alimenta de una fuente USB de 5V independiente; el pin IN del relevador al GPIO 27 del ESP32. Une el GND de la fuente USB con el GND del ESP32.

Paso 2: calibra. Sube un programa que imprima el valor analógico cada segundo. Riega la tierra y anota el valor en húmedo (por ejemplo 1200); deja secar un día y anota el valor en seco (por ejemplo 2800). Ojo: en estos sensores el valor SUBE cuando la tierra se SECA (menos conductividad). Elige el umbral en medio, por ejemplo 2000.

Paso 3: escribe el programa. La lógica de riego va en el loop principal y NO depende del WiFi; la publicación MQTT va aparte y si falla, no detiene el riego:


```

#include <WiFi.h>
#include <PubSubClient.h>

const int PIN_SUELO = 35;
const int PIN_RELAY = 27;
const int UMBRAL_SECO = 2000;      // Arriba de esto = tierra seca
const unsigned long TIEMPO_RIEGO = 5000; // 5 segundos de bomba

// ... (configuracion WiFi/MQTT igual que el ejemplo 2) ...

void loop() {
  // 1) Control local: funciona SIEMPRE, con o sin internet
  int suelo = analogRead(PIN_SUELO);
  if (suelo > UMBRAL_SECO) {
    digitalWrite(PIN_RELAY, LOW); // Activa bomba
    delay(TIEMPO_RIEGO);
    digitalWrite(PIN_RELAY, HIGH); // Apaga bomba
    Serial.println("Riego realizado");
    delay(60000); // Espera 1 min antes de volver a evaluar
  }

  // 2) Reporte a la nube: solo si hay conexion
  if (WiFi.status() == WL_CONNECTED && mqtt.connected()) {
    mqtt.loop();
    static unsigned long ultimo = 0;
    if (millis() - ultimo >= 30000) {
      ultimo = millis();
      char buf[8];
      itoa(suelo, buf, 10);
      mqtt.publish("lucas-iot/tu-nombre-unico/maceta/humedad_suelo", buf);
    }
  }
  delay(1000);
}

```

Paso 4: prueba con la maceta seca: la bomba debe activarse 5 segundos. Verifica que el agua llegue a la tierra (manguerita bien colocada en un vaso con agua). Importante: pon la bomba más abajo que el vaso o usa válvula antirretorno casera para que no se vacíe por sifón. Revisa el tablero para ver la gráfica de humedad bajando después del riego.

Qué aprendiste: el patrón sense-decide-actuate completo, la regla de oro de que el control crítico debe ser local (sin depender de internet), y la calibración de un sensor analógico contra la realidad. Este proyecto, bien presentado en una caja, es algo que se puede vender.

Ejercicio 5: monitor de calidad del aire con alertas

Enunciado: con un sensor MQ-135 (gases y calidad del aire) y el ESP32, mide la calidad del aire de tu cuarto, publícala por MQTT en formato JSON y haz que tu script Python de la "mini nube" mande una alerta cuando la calidad sea mala por más de 5 minutos seguidos.

Solución paso a paso:

Paso 1: conecta el MQ-135. Estos sensores tienen salida analógica (AO) y digital (DO). Conecta AO al GPIO 32, VCC a 5V (VIN, porque el calentador interno del sensor necesita 5V) y GND a GND. Déjalo precalentar 24 horas la primera vez para lecturas estables (sí, 24 horas; es un sensor químico y necesita estabilizarse). Para pruebas rápidas basta con 30 minutos, aceptando menos precisión.

Paso 2: en el ESP32, lee el valor analógico y publícalo como JSON cada 15 segundos en el tema "lucas-iot/tu-nombre-unico/cuarto/calidad_aire":

```
int aire = analogRead(32);
char msg[32];
snprintf(msg, sizeof(msg), "{\"calidad_aire\": %d}", aire);
mqtt.publish("lucas-iot/tu-nombre-unico/cuarto/calidad_aire", msg);
```

Primero mide tu ambiente normal y anota el valor base (por ejemplo 400). Define "aire malo" como base + 300 (en el ejemplo, 700). Cada sensor y cada cuarto son distintos: calibrar contra tu realidad es obligatorio, no opcional.

Paso 3: modifica el script Python del ejemplo 3 para la alerta con persistencia de 5 minutos:

```

UMBRAL_MALO = 700
rachas_malas = 0

def al_recibir(cliente, userdata, msg):
    global rachas_malas
    datos = json.loads(msg.payload.decode("utf-8"))
    valor = datos["calidad_aire"]
    print("Calidad del aire:", valor)
    if valor > UMBRAL_MALO:
        rachas_malas += 1
    else:
        rachas_malas = 0
    # 15 s por mensaje x 20 mensajes = 5 minutos
    if rachas_malas == 20:
        print("ALERTA: mala calidad del aire por 5 minutos. Ventila el cuarto.")
        rachas_malas = 0 # Para no spamear, reinicia el conteo

```

Paso 4: prueba acercando (sin tocar) un marcador o un poco de humo de cocina al sensor: el valor debe subir y, si se mantiene, la alerta debe dispararse a los 5 minutos. Aleja la fuente y verifica que el contador se reinicia cuando el aire mejora.

Qué aprendiste: a trabajar con sensores que necesitan calibración y precalentamiento, a publicar JSON estructurado, y a programar alertas con persistencia temporal (una sola lectura mala no dispara la alarma; solo un problema sostenido). Esta lógica anti-falsas-alarmas se usa en sistemas de seguridad reales.

Ejercicio 6: medidor de consumo eléctrico casero

Enunciado: mide cuánta corriente consume un aparato (por ejemplo tu cargador de laptop o una lámpara) usando un sensor ACS712 con el ESP32, calcula la potencia aproximada y publícala por MQTT. Muestra en el tablero el consumo en watts y lleva la cuenta de watts-hora consumidos en el día.

Solución paso a paso:

Paso 1: entiende el sensor. El ACS712 mide corriente y entrega un voltaje proporcional: sin corriente entrega 2.5V (la mitad de 5V), y varía 185 mV por cada ampere (en la versión de 5A). Conéctalo en SERIE con el aparato a medir, del lado de bajo voltaje para practicar de forma segura: por ejemplo, entre una fuente de 5V USB y tu aparato USB. VCC del módulo a 5V, GND a GND, OUT al GPIO 33 del ESP32. Advertencia: para medir aparatos de 110V necesitas el sensor en el cable de corriente alterna con aislamiento adecuado; si no tienes experiencia, practica solo con el circuito USB de 5V.

Paso 2: el programa toma muchas muestras y promedia, porque la corriente real fluctúa:

```
const int PIN_ACS = 33;
const float SENSIBILIDAD = 0.185; // 185 mV por ampere (modulo de 5A)

float leerCorriente() {
    long suma = 0;
    for (int i = 0; i < 1000; i++) {
        suma += analogRead(PIN_ACS);
        delayMicroseconds(200);
    }
    float promedioADC = suma / 1000.0;
    float voltaje = (promedioADC / 4095.0) * 3.3; // ADC del ESP32 a 3.3V
    // OJO: el ACS712 entrega hasta 5V; usa un divisor de voltaje
    // para no meter mas de 3.3V al ESP32 (ver paso 3)
    float corriente = (voltaje - 2.5) / SENSIBILIDAD;
    if (corriente < 0.05 && corriente > -0.05) corriente = 0; // ruido
    return corriente;
}
```

Paso 3: protege la entrada. Como el ACS712 puede entregar hasta 5V y el ESP32 solo tolera 3.3V, pon un divisor de voltaje (dos resistencias, por ejemplo 20k y 10k) entre la salida del sensor y el GPIO 33, y ajusta la fórmula multiplicando por el factor del divisor (en el ejemplo, x1.5). Si omites esto, puedes dañar el pin. Mide con multímetro antes de conectar al ESP32 si tienes uno.

Paso 4: calcula potencia y energía, y publica:

```
float I = leerCorriente(); // Amperes
float V = 5.0; // Voltaje de tu fuente USB (conocido)
float P = V * I; // Watts
energiaWh += P * (15.0 / 3600.0); // Suma cada 15 s: Wh del dia

char msg[64];
snprintf(msg, sizeof(msg),
    "{\"corriente_a\": %.2f, \"potencia_w\": %.2f, \"energia_wh\": %.2f}",
    I, P, energiaWh);
mqtt.publish("lucas-iot/tu-nombre-unico/energia/medidor", msg);
```

Paso 5: en el tablero agrega un medidor de watts en vivo y una gráfica acumulada de watts-hora. Conecta y desconecta el aparato y mira cómo la potencia sube y baja en tiempo real. Al final del día tendrás el consumo total: multiplícalo por el costo del kWh de tu recibo de luz y sabrás cuánto te cuesta usar ese aparato. Ese número, calculado por ti con tu propio sensor, vale más que mil explicaciones sobre por qué el IoT importa.

Qué aprendiste: a medir una magnitud eléctrica real con un sensor analógico, a promediar muestras para limpiar el ruido, a proteger las entradas del microcontrolador con un divisor de voltaje, y a convertir mediciones crudas en información útil (de amperes a pesos en tu recibo de luz). Con este ejercicio cierras el círculo: ya sabes sensar, actuar, comunicar, guardar, visualizar y asegurar. Eres oficialmente una persona que hace IoT.

Resumen de la unidad

Llegaste al final de la unidad 3 y ya no ves un foco "inteligente" de la misma forma: ahora ves un sensor, un microcontrolador, una red, una nube y una aplicación trabajando juntos. Vamos a repasar los puntos clave para que queden bien grabados.

Primero: el Internet de las cosas es la conexión de objetos físicos a internet para que midan, decidan y actúen. No es magia ni es solo para grandes empresas: con un ESP32 barato y sensores de pocos pesos puedes construir sistemas reales que automatizan tu casa, cuidan tus plantas y miden tu mundo.

Segundo: todo sistema IoT sigue la misma arquitectura de cinco pasos: sensor, microcontrolador, red, nube y aplicación. El sensor siente el mundo físico y lo convierte en señal eléctrica. El microcontrolador lee esa señal, la procesa y decide. La red (típicamente WiFi) transporta los datos. La nube los recibe, guarda y organiza. La aplicación (tablero o app) los muestra de forma útil. Cuando analices cualquier producto IoT, identifica estas cinco partes y lo habrás entendido.

Tercero: los sensores son los sentidos y los actuadores son las manos. Aprendiste a usar el DHT11/DHT22 para temperatura y humedad, el PIR para movimiento, la LDR para luz y el sensor de humedad de suelo para riego. Del lado de los actuadores, dominaste el LED, el buzzer, el servomotor y sobre todo el relevador, que te permite controlar aparatos reales con una señal pequeña. Recuerda las reglas de seguridad: verifica voltajes, no excedas la corriente de los pines, une las tierras de fuentes externas y trata la corriente alterna de casa con respeto absoluto.

Cuarto: el ESP32 es tu mejor aliado para IoT: trae WiFi y Bluetooth integrados, se programa en MicroPython o en C++ con el IDE de Arduino, y cuesta poco. Arduino (como el UNO) sigue

siendo una gran escuela de electrónica, pero para proyectos conectados el ESP32 gana por precio y simplicidad. Aprendiste sus conceptos esenciales: pines GPIO, entradas analógicas con ADC de 12 bits, y la importancia de probar cada componente por separado antes de armar el circuito completo.

Quinto: los protocolos son el idioma de los aparatos. WiFi conecta tu dispositivo a la red local e internet. HTTP funciona con pregunta y respuesta, es universal y fácil de depurar, pero pesado para microcontroladores. MQTT, con su modelo de publicación y suscripción a través de un broker y sus temas jerárquicos, es el protocolo ideal para IoT: ligero, eficiente y en tiempo real. Aprendiste a estructurar temas como "lucas-iot/casa/sala/temperatura", a usar comodines y a programar reconexiones automáticas para que tus dispositivos nunca se queden colgados.

Sexto: la nube y el tablero cierran el ciclo. La nube recibe, guarda y organiza los datos; el tablero los convierte en gráficas, medidores y alertas que ves en tu teléfono. Construíste tu propia "mini nube" en Python que se suscribe por MQTT y guarda historiales en CSV, y conectaste tu ESP32 a una plataforma IoT gratuita con tablero real. También aprendiste la regla de oro del diseño IoT: el control crítico (como el riego) debe funcionar localmente aunque se caiga el internet; la nube es para monitoreo y análisis, no para lo esencial.

Séptimo: la seguridad no es opcional. Cambia todas las credenciales por defecto, usa MQTT con TLS y HTTPS cuando publiques en internet, dale a cada dispositivo solo los permisos mínimos que necesita, mantén actualizado el firmware y nunca compartas código con contraseñas. Los bots escanean todo internet buscando dispositivos desprotegidos: el tuyo no será uno de ellos.

Octavo: los errores de principiante ya los conoces de antemano, así que no te van a detener: leer el DHT11 más de una vez cada 2 segundos, cables flojos en el protoboard, confundir 3.3V con 5V, temas MQTT con mayúsculas distintas, publicar datos rapidísimo sin necesidad, relevadores que se activan en LOW, y olvidar la calibración de los sensores. Cada uno de estos te habría costado horas; ahora te costará cero.

Noveno: construiste cosas de verdad. Tu primer circuito leyendo sensores, tu ESP32 publicando por MQTT, tu mini nube guardando datos, una alarma de movimiento, una lámpara automática con histéresis, un tablero en vivo, un riego automático que funciona sin internet, un monitor de calidad del aire con alertas inteligentes y un medidor de consumo eléctrico que traduce amperes a pesos de tu recibo de luz. Cada proyecto usa el mismo esqueleto de cinco pasos, así que el siguiente que imagines te tomará la mitad del tiempo.

Hacia dónde seguir: con esta base puedes atacar proyectos más ambiciosos. Un sistema de seguridad para tu casa con cámaras ESP32-CAM y notificaciones al teléfono. Un invernadero automatizado que controle riego, ventilación y luz. Una estación meteorológica completa que

suba datos a internet. Y cuando quieras, el siguiente paso natural es combinar IoT con inteligencia artificial: que tus datos alimenten modelos que predigan, no solo que reporten. Eso lo verás más adelante en este libro.

Un último consejo de colega a colega: documenta tus proyectos. Tómales foto, guarda el código con comentarios, anota qué funcionó y qué no. Dentro de seis meses, cuando quieras replicar tu riego automático o mejorarlo, tu yo del futuro te lo va a agradecer. Y si publicas tus proyectos en lucasegaacademy.com o en tus redes, estarás haciendo exactamente lo que hacen los profesionales: aprender en público y construir un portafolio que habla por ti.

Nos vemos en la unidad 4. Sigue construyendo.

Recuerda: cada experto en IoT empezó exactamente donde estás tú ahora, con un sensor en un protoboard y muchas preguntas. La diferencia la hace la constancia. Arma, mide, rompe, arregla y repite.

UNIDAD 4: Nube y edge computing

Piensa en tu computadora de escritorio o en tu laptop. Ahí tienes tu disco duro, tu memoria RAM, tu procesador. Todo lo que haces -escribir documentos, ver fotos, correr programas- sucede físicamente dentro de esa caja. Si quieres más potencia, compras una máquina más cara. Si se te descompone el disco, pierdes tus archivos. Si quieres que otra persona use tu programa, tienes que darle la computadora o instalarle el software. Durante décadas así fue el mundo del cómputo: cada quien con su caja, cada empresa con su cuarto de servidores, cada proyecto limitado por el fierro que podía comprar.

La nube cambió todo eso. La idea es tan sencilla que parece obvia una vez que la entiendes: en lugar de comprar y cuidar tus propias computadoras, rentas computadoras de alguien más, a través de internet, por el tiempo que las necesites. Esos "alguien más" son los proveedores de nube: empresas gigantescas que tienen edificios enteros -los centros de datos- llenos de miles de servidores, con electricidad de respaldo, aire acondicionado industrial, conexión a internet rapidísima y equipos de ingenieros vigilando todo las 24 horas. Tú pides una máquina virtual, un espacio de almacenamiento o una base de datos desde tu navegador, y en minutos -a veces en segundos- la tienes lista para usar. Cuando terminas, la apagas o la borras y dejas de pagar por ella.

Esa sencillez es engañosa, porque detrás hay una de las transformaciones tecnológicas más grandes de la historia. La nube es lo que permitió que una startup de dos personas compita con una empresa de miles de empleados: ya no necesitas millones de pesos para montar tu infraestructura. Es lo que hizo posible que Netflix le sirva video a millones de personas al mismo tiempo, que tu correo electrónico esté disponible desde cualquier dispositivo, que los modelos de inteligencia artificial se entrenen con miles de computadoras trabajando en paralelo. Sin la nube, el mundo digital que conoces hoy simplemente no existiría.

Pero la nube también trajo un primo nuevo a la familia: el edge computing, o cómputo en el borde. Si la nube es "todo sucede en los centros de datos lejanos", el edge dice "mejor que algunas cosas sucedan aquí cerquita, donde se generan los datos". Un coche autónomo no puede esperar a que sus datos viajen hasta un servidor en otro país para decidir si frena: necesita procesar la información en el mismo carro. Una cámara de seguridad que detecta movimiento, un sensor de fábrica que detecta una falla, un videojuego que necesita responder en milisegundos: todos se benefician de procesar cerca del origen. En esta unidad vas a entender ambos mundos -la nube y el borde- y, sobre todo, cuándo conviene cada uno.

Por qué la nube cambió las reglas del juego

Antes de la nube, montar un servicio en internet era un proyecto pesado. Imagina que en 2005 querías lanzar una página web que esperabas que visitaran miles de personas. Primero comprabas un servidor físico: una caja cara, de varios miles de dólares. Luego necesitabas un lugar donde ponerlo: un cuarto con aire acondicionado, electricidad con respaldo, conexión a internet dedicada. Instalabas el sistema operativo, configurabas la red, ponías la seguridad. Todo esto tomaba semanas y costaba mucho dinero, antes de que un solo usuario visitara tu página. Y lo peor: tenías que adivinar cuánta capacidad ibas a necesitar. Si comprabas de más, desperdiciabas dinero en hierro ocioso. Si comprabas de menos y tu página se volvía popular, se caía justo en el momento más importante.

A ese problema se le llamaba "planear la capacidad", y era una pesadilla. Las empresas compraban servidores para el peor escenario posible -el día con más tráfico del año- y el resto del tiempo esas máquinas caras estaban medio dormidas, consumiendo luz y espacio. Era como comprar un camión de carga para mudarte una vez al año y usarlo como coche familiar el resto del tiempo.

La nube eliminó ese problema con una palabra mágica: elasticidad. En la nube puedes pedir más capacidad cuando la necesitas y soltarla cuando ya no. Si tu página recibe diez veces más visitas por un video viral, pides diez servidores más con unos clics -o hasta se agregan solos, automáticamente- y cuando pasa la ola, los sueltas. Pagas solo por lo que usas, como la luz eléctrica: no compras la planta generadora, pagas el recibo según tu consumo.

Este cambio de "comprar" a "rentar" tuvo consecuencias enormes. Primero, democratizó la tecnología: hoy un estudiante con una tarjeta y la capa gratuita de un proveedor puede experimentar con la misma infraestructura que usan las empresas grandes. Segundo, aceleró la innovación: probar una idea ya no requiere meses de compras y configuraciones; en una tarde puedes tener un prototipo funcionando. Tercero, cambió los trabajos de tecnología: surgieron roles como ingeniero de nube, arquitecto de soluciones y especialista en DevOps, todos muy bien pagados y con mucha demanda.

También cambió la forma de pensar el software. Los programas dejaron de diseñarse para "una computadora específica" y empezaron a diseñarse para "cualquier cantidad de computadoras". Nació la idea de los microservicios -dividir una aplicación en piezas pequeñas e independientes- y de los contenedores, que son como cajitas que llevan tu programa con todo lo que necesita para correr igual en cualquier máquina. Todo eso lo verás en esta unidad con ejemplos que puedes probar tú mismo.

Lo que vas a aprender en esta unidad

Esta unidad está organizada para llevarte de cero a desplegar cosas reales. No necesitas haber usado la nube antes; sí necesitas las bases de las unidades anteriores: saber qué es un sistema operativo, qué es una red, qué es una dirección IP y cómo funciona el modelo cliente-servidor de internet. Si esos conceptos te suenan, estás listo.

El recorrido es el siguiente. Primero, en la explicación detallada, vas a entender qué es exactamente la nube, los tres modelos de servicio -IaaS, PaaS y SaaS- con ejemplos de la vida real, quiénes son los grandes proveedores -AWS, Google Cloud y Azure- y qué te regalan en sus capas gratuitas para que practiques sin pagar. Vas a ver cómo crear tu primera máquina virtual paso a paso, cómo subir una página web estática y cómo desplegar una API sencilla. Luego entra el almacenamiento y las bases de datos en la nube: dónde guardar archivos y datos sin tener tu propio servidor.

Después viene el edge computing: qué es, en qué se diferencia de la nube y en qué casos conviene -latencia baja, internet de las cosas, privacidad-. Vas a ver ejemplos concretos, desde coches autónomos hasta fábricas inteligentes. También hablaremos de dinero, porque la nube tiene una trampa famosa: es facilísimo gastar de más si no pones atención. Te voy a enseñar cómo leer una factura de nube y cómo evitar sorpresas. Y cerraremos con la combinación ganadora: nube más inteligencia artificial más internet de las cosas, que es donde están los proyectos más emocionantes de hoy.

Al final de la unidad, en la sección de práctica, vas a hacer seis ejercicios con solución explicada: desde crear recursos con comandos de terminal hasta escribir un archivo de configuración para desplegar una aplicación. La idea es que termines la unidad no solo entendiendo la teoría, sino con las manos en la masa.

Un aviso honesto antes de empezar: la nube es un tema enorme. Hay personas que se dedican profesionalmente solo a esto y aun así siguen aprendiendo. Esta unidad no pretende convertirte en experto certificado en una sentada; pretende darte los fundamentos sólidos y la confianza para seguir por tu cuenta. Cada concepto viene con analogías de la vida cotidiana para que se te quede grabado, con los errores típicos de principiantes para que no los cometes, y con buenas prácticas que usan los profesionales. Vamos paso a paso, como siempre: tú puedes con esto.

Una analogía para llevar en la cabeza

Guarda esta imagen mental durante toda la unidad, porque te va a servir una y otra vez: la nube es como la red eléctrica. Hace más de cien años, las fábricas tenían su propio generador de electricidad. Era caro, requería un ingeniero que lo cuidara, y si se descomponía, la fábrica

se detenía. Luego llegaron las compañías eléctricas: tú solo conectas tus aparatos al enchufe y pagas por lo que consumes. No te importa dónde está la planta generadora ni cómo funciona; solo quieres que la luz prenda cuando la necesitas.

La nube es exactamente eso, pero con cómputo. Antes, cada empresa tenía su "generador": su cuarto de servidores. Ahora se conecta al "enchufe" de internet y consume cómputo, almacenamiento y servicios por demanda. Y el edge computing sería como tener un panel solar en tu techo: para ciertas cosas te conviene generar tu propia energía cerquita, sin depender de la red lejana. Con esa imagen en mente, todo lo que sigue va a tener sentido.

Explicación detallada

Qué es realmente la nube: definición sin humo

La palabra "nube" suena etérea, como si tus datos flotaran en el cielo. La realidad es mucho más terrenal: la nube son computadoras de alguien más, en edificios de alguien más, que usas a través de internet. Cada vez que guardas una foto y aparece en tu otro teléfono, cada vez que ves una serie en streaming, cada vez que usas el correo desde el navegador, estás usando la nube. Tus archivos no están "en el aire": están en discos duros dentro de centros de datos, que son naves industriales llenas de racks con servidores, con sistemas de enfriamiento que consumen tanta energía como una ciudad pequeña.

El instituto de estándares de Estados Unidos, el NIST, definió la nube con cinco características esenciales, y vale la pena conocerlas porque son la base de todo:

1. Autoservicio bajo demanda: puedes pedir recursos -una máquina virtual, almacenamiento- cuando quieras, sin hablar con un vendedor ni esperar días. Todo es automático, desde un panel web o con comandos.
2. Acceso amplio por red: los servicios están disponibles a través de internet y los usas desde cualquier dispositivo: tu laptop, tu celular, una tablet.
3. Agrupación de recursos: el proveedor junta sus miles de servidores en un gran alberca común y reparte pedazos a cada cliente según lo pida. Tú no sabes -ni te importa- en qué máquina física exacta están tus cosas.
4. Elasticidad rápida: puedes subir y bajar tu capacidad en minutos. Esta es la característica estrella, la que resolvió el problema de "planear la capacidad" del que hablamos en la introducción.

5. Servicio medido: todo se mide -horas de cómputo, gigabytes almacenados, datos transferidos- y pagas según lo que consumiste. Como el recibo de la luz.

Hay tres formas de desplegar la nube. La nube pública es la que ofrecen AWS, Google Cloud y Azure a cualquier persona: compartes la infraestructura con miles de clientes, pero tus datos están aislados y seguros. La nube privada es infraestructura dedicada a una sola organización -puede estar en sus oficinas o rentada en exclusiva- y se usa cuando hay requisitos estrictos de control o regulación. La nube híbrida combina ambas: mantienes lo sensible en tu nube privada y usas la pública para lo demás o para picos de demanda. Para ti, que vas empezando, la nube pública es el camino: es la más barata para aprender y la que usan la mayoría de las empresas para sus proyectos nuevos.

Analogía cotidiana: la nube pública es como el transporte público. No compras el camión, no lo manejas, no lo llevas al taller: pagas tu pasaje y te lleva. La nube privada es como tener tu propio coche: más control, más costo, tú te encargas del mantenimiento. La híbrida es tener coche propio pero a veces tomar taxi cuando te conviene.

Error común de principiante: pensar que "subir a la nube" significa que ya no hay que cuidar nada. La nube se encarga del fierro, pero la seguridad de tus datos, tus contraseñas y tu código siguen siendo tu responsabilidad. Hay un dicho famoso en el medio: "la nube es el centro de datos de alguien más, pero tus errores siguen siendo tuyos". Más adelante verás el modelo de responsabilidad compartida, que aclara exactamente quién cuida qué.

Buena práctica: desde el primer día, trata los recursos de nube como desechables. Las máquinas virtuales se crean y se destruyen; lo valioso es tu código y tus datos, que deben vivir en repositorios y respaldos, no atrapados dentro de una máquina que puedes borrar por accidente.

Los tres modelos de servicio: IaaS, PaaS y SaaS

Esta es la clasificación más importante de la unidad. Todo servicio de nube cae en una de estas tres categorías, que se diferencian por cuánto control tienes y de cuánto mantenimiento te libras. La analogía clásica es la pizza, y funciona tan bien que la usan hasta en certificaciones:

- IaaS (Infraestructura como Servicio) es como si te rentaran la cocina equipada: tú llevas tus ingredientes y cocinas. El proveedor te da máquinas virtuales, redes y almacenamiento; tú instalas el sistema operativo, las aplicaciones y lo configuras todo. Ejemplo: rentar un servidor virtual en AWS EC2 o en DigitalOcean y ahí instalar lo que quieras.

- PaaS (Plataforma como Servicio) es como un restaurante donde pides los ingredientes preparados y tú armas tu platillo, o mejor: te dan la cocina con el horno ya caliente y los utensilios listos. El proveedor maneja el sistema operativo, el mantenimiento y la infraestructura; tú solo subes tu código y funciona. Ejemplos: Heroku, Google App Engine, o AWS Elastic Beanstalk. Tú te enfocas en programar, ellos se encargan de que haya servidores corriendo tu programa.
- SaaS (Software como Servicio) es pedir la pizza a domicilio: no cocinas nada, solo la comes. Usas el software directamente desde el navegador, sin instalar ni configurar nada. Ejemplos que usas todos los días: Gmail, Google Docs, Netflix, Dropbox, Microsoft 365. Tú eres el consumidor final del software terminado.

La diferencia clave está en la responsabilidad compartida. Imagina una pila de capas: el hierro físico, la virtualización, el sistema operativo, el entorno de ejecución, tu aplicación y tus datos. En IaaS, el proveedor cuida hasta la virtualización y tú todo lo demás. En PaaS, el proveedor sube hasta el entorno de ejecución y tú solo cuidas tu aplicación y tus datos. En SaaS, el proveedor cuida casi todo y tú solo cuidas tus datos y quién tiene acceso. Mientras más subes en la pirámide, menos control tienes pero menos trabajo te toca.

¿Cuándo conviene cada uno? IaaS conviene cuando necesitas control total: configuraciones especiales del sistema operativo, software raro, o cuando estás aprendiendo cómo funciona todo por dentro -que es tu caso ahora-. PaaS conviene cuando quieres lanzar rápido y no quieres administrar servidores: ideal para startups y proyectos personales. SaaS conviene cuando el problema ya está resuelto por alguien: no vas a programar tu propio correo electrónico si existe Gmail.

Error común de principiante: elegir IaaS para todo "porque así aprendo más", y terminar pasando más tiempo parchando el sistema operativo y peleando con configuraciones que construyendo su proyecto. Aprender con IaaS está bien al inicio, pero para proyectos reales considera PaaS: tu tiempo vale más que el orgullo de administrar servidores.

Buena práctica: empieza tus proyectos en PaaS o en servicios administrados, y baja a IaaS solo cuando tengas una razón concreta -una configuración que el PaaS no permite, un requisito de rendimiento especial-. La regla de oro: no administres lo que no te diferencia. Tu proyecto se distingue por lo que hace, no por cómo instalaste Linux.

Los grandes proveedores: AWS, Google Cloud y Azure

El mercado de la nube pública lo dominan tres gigantes. Conocerlos es obligatorio porque la mayoría de los empleos de nube piden al menos uno, y porque sus capas gratuitas son tu laboratorio sin costo.

AWS (Amazon Web Services) es el pionero y el líder del mercado. Nació en 2006, cuando Amazon decidió rentar la infraestructura que había construido para su tienda en línea. Tiene el catálogo de servicios más grande -más de 200- y la comunidad más extensa: si tienes un problema con AWS, alguien ya lo tuvo y lo documentó. Sus servicios estrella tienen nombres que vas a ver en todos lados: EC2 para máquinas virtuales, S3 para almacenamiento de archivos, RDS para bases de datos administradas, Lambda para cómputo sin servidor. AWS es como la ferretería gigante: encuentras de todo, pero al principio abruma la cantidad de opciones.

Google Cloud (GCP) es el segundo en innovación y el favorito para datos e inteligencia artificial, lo cual no sorprende viniendo de Google. Su red global es impresionante y sus servicios de datos -BigQuery para analizar cantidades masivas de información- son legendarios. Kubernetes, la herramienta estándar para manejar contenedores, nació en Google. Si tu proyecto involucra mucha IA o análisis de datos, Google Cloud merece una mirada seria.

Azure (Microsoft Azure) es el favorito de las empresas grandes que ya usan Windows, Office y todo el ecosistema Microsoft. Si una corporación tiene sus computadoras con Windows y su correo en Outlook, lo natural es que su nube sea Azure, porque todo se integra. Azure también pisa fuerte en nube híbrida: herramientas para conectar tus servidores locales con la nube.

Mención honorífica: hay proveedores más pequeños y simples que son excelentes para aprender y para proyectos personales: DigitalOcean, Linode (ahora parte de Akamai) y Hetzner. Tienen menos servicios, precios más simples y paneles menos intimidantes. Muchos desarrolladores despliegan su primer proyecto ahí antes de brincar a los gigantes.

Las capas gratuitas son tu mejor amiga como estudiante. Los tres grandes ofrecen free tiers generosos: AWS te da 12 meses de una máquina virtual pequeña, 5 GB de almacenamiento S3 y un millón de ejecuciones de Lambda al mes, entre otras cosas. Google Cloud te regala 300 dólares de crédito para gastar en 90 días más una capa gratuita permanente de varios servicios. Azure te da 200 dólares por 30 días más servicios gratuitos por 12 meses. Ojo: "gratis" tiene límites y condiciones; si te pasas, te cobran. Por eso más adelante hay una sección completa de costos. Pero con cuidado, puedes practicar meses sin pagar un peso.

Error común de principiante: abrir cuentas en los tres proveedores a la vez y no aprender bien ninguno. Elige uno para empezar -cualquiera de los tres sirve- y profundiza. Los conceptos se transfieren: si aprendes máquinas virtuales en AWS, entenderás las de Azure en una tarde.

Buena práctica: usa las calculadoras de precios de cada proveedor antes de crear recursos grandes, pon alertas de facturación desde el primer día -te enseñó cómo en la sección de costos- y apaga o borra todo lo que no estés usando. La nube cobra por minuto; una máquina olvidada encendida un mes entero es dinero tirado.

Tu primera máquina virtual, paso a paso

Una máquina virtual es una computadora simulada corriendo dentro de una computadora real. El proveedor toma un servidor físico gigante, lo parte en pedazos con un software llamado hipervisor, y te renta un pedazo: con su propio sistema operativo, su memoria, su disco y su dirección IP. Para ti se siente como una computadora normal a la que te conectas por internet; lo que pasa debajo es invisible.

Vamos a recorrer el proceso completo como lo harías en AWS con el servicio EC2. Los pasos son casi idénticos en Google Cloud -ahí se llaman Compute Engine- y en Azure -ahí se llaman Virtual Machines-; solo cambian los nombres de los botones.

Paso 1: elige la región. Los proveedores tienen centros de datos repartidos por el mundo, agrupados en regiones: us-east-1 está en Virginia, Estados Unidos; sa-east-1 en Sao Paulo, Brasil. Elige la región más cercana a tus usuarios para que la respuesta sea rápida. Para practicar, cualquiera sirve; muchos eligen us-east-1 porque suele ser la más barata.

Paso 2: elige el sistema operativo. Al crear la máquina escoges una imagen: Ubuntu Linux, Debian, Windows Server, Amazon Linux. Para aprender, Ubuntu es la opción más popular: gratis, con muchísima documentación y comunidad. No necesitas interfaz gráfica; los servidores se administran con comandos de terminal.

Paso 3: elige el tamaño. Los tamaños se miden en vCPU -procesadores virtuales- y memoria RAM. El más pequeño, como el t2.micro o t3.micro de AWS, tiene 1 vCPU y 1 GB de RAM: suficiente para practicar y entra en la capa gratuita. No pidas una máquina gigante "por si acaso": eso cuesta dinero real por hora.

Paso 4: configura la llave de acceso. En lugar de contraseña, la nube usa llaves criptográficas: descargas un archivo .pem que es tu llave privada. Guárdalo bien y no lo compartas: quien tenga ese archivo puede entrar a tu máquina. Es como la llave de tu casa en formato digital.

Paso 5: configura el firewall. Los proveedores llaman a esto "grupo de seguridad": una lista de reglas que dice qué tráfico puede entrar. Por defecto todo está bloqueado. Típicamente abres el puerto 22 para conectarte por SSH -el protocolo para administrar servidores por terminal-, el puerto 80 para páginas web y el 443 para páginas web seguras. Regla de oro: abre solo lo necesario.

Paso 6: lánzala y conéctate. Le das clic a "lanzar", esperas un minuto o dos, copias la dirección IP pública y te conectas desde tu terminal con un comando como: `ssh -i mi-llave.pem ubuntu@54.123.45.67`. Y listo: estás dentro de una computadora que está físicamente a miles de kilómetros, como si estuvieras sentado frente a ella.

Analogía cotidiana: rentar una máquina virtual es como rentar un departamento amueblado por noches. No compras el edificio, no pagas el mantenimiento del elevador, no te preocupas si se funde un foco del pasillo: llegas, usas el depa, y cuando te vas dejas de pagar. El hipervisor es el administrador del edificio que divide todo en unidades.

Error común de principiante: conectarse como usuario root -el superadministrador- para todo, o peor, abrir el puerto 22 al mundo entero sin restricciones. Los robots maliciosos escanean internet buscando servidores con SSH abierto e intentan adivinar contraseñas todo el día. Usa llaves en vez de contraseñas, no uses root para el trabajo diario y, si puedes, limita el acceso SSH a tu propia dirección IP.

Buena práctica: automatiza la creación de tus máquinas con scripts o herramientas de infraestructura como código -las verás en los ejemplos-. Crear servidores a mano con clics funciona una vez; cuando necesites diez iguales o quieras repetir el proceso sin errores, los clics no escalan y los scripts sí. Y etiqueta todo: ponle nombre y proyecto a cada recurso que crees; tu yo del futuro, revisando la factura, te lo va a agradecer.

Desplegar una página web estática y una API sencilla

"Desplegar" significa poner tu aplicación en un servidor para que el mundo la pueda usar. Hay dos tipos de despliegue que todo principiante debe dominar: la página estática y la API.

Una página web estática es un conjunto de archivos -HTML, CSS, JavaScript, imágenes- que no cambian según quién los pida: el servidor simplemente los entrega tal cual. Tu portafolio personal, la página de un negocio local, la documentación de un proyecto: todo eso puede ser estático. Y aquí viene lo bonito: alojar páginas estáticas en la nube es facilísimo y baratísimo, a veces gratis. El servicio S3 de AWS, por ejemplo, puede funcionar como un servidor web: subes tus archivos a un "bucket" -así se llaman las carpetas de S3-, activas la opción de "alojamiento de sitio web estático", y obtienes una dirección pública. Servicios como Netlify, Vercel o GitHub Pages lo hacen aún más simple: conectas tu repositorio de código y cada vez que subes cambios, tu página se actualiza sola. Para páginas estáticas, no necesitas una máquina virtual: es como usar un casillero en vez de rentar una bodega entera.

Una API -interfaz de programación de aplicaciones- es un programa que vive en un servidor y responde a peticiones de otros programas con datos, normalmente en formato JSON. Por ejemplo, una API del clima recibe "¿qué temperatura hay en Guadalajara?" y responde con el

dato. Desplegar una API ya requiere cómputo real: un programa corriendo y escuchando peticiones.

Tienes tres caminos principales para desplegar una API, de más simple a más control. El cómputo sin servidor -serverless-, con AWS Lambda como ejemplo famoso: subes tu función, el proveedor la ejecuta cada vez que llega una petición y te cobra por milisegundos de uso. No hay servidor que administrar, y si nadie usa tu API, pagas cero. Es ideal para proyectos pequeños o con tráfico irregular. El PaaS, como Heroku o App Engine: subes tu código con un comando y la plataforma se encarga de correrlo, de darle una dirección pública y hasta de agregar más capacidad si crece el tráfico. La máquina virtual: instalas todo a mano, máximo control, máximo trabajo.

Para elegir, piensa en tráfico y control. ¿Tu API la van a usar unas cuantas personas al día? Serverless o PaaS: simple y barato. ¿Necesitas configuraciones muy específicas o correr algo raro? Máquina virtual. ¿Estás aprendiendo cómo funciona todo por dentro? Máquina virtual, sin duda, porque te enseña lo que los otros caminos te ocultan.

Error común de principiante: desplegar la API con la base de datos, las contraseñas y las llaves secretas escritas directamente en el código, y luego subir ese código a un repositorio público. Los robots buscan llaves de AWS en GitHub en minutos y las usan para minar criptomonedas con tu cuenta. Las credenciales van en variables de entorno, nunca en el código.

Buena práctica: separa tu aplicación en ambientes: desarrollo en tu máquina, pruebas en un ambiente de nube barato, producción en el ambiente serio. Nunca pruebes cosas nuevas directamente en producción con usuarios reales. Y automatiza el despliegue: que subir una nueva versión sea un solo comando, no una tarde de copiar archivos a mano con miedo a romper algo.

Almacenamiento y bases de datos en la nube

Todo programa necesita guardar cosas, y la nube ofrece opciones para cada necesidad. La clave es elegir el tipo correcto de almacenamiento, porque usar el equivocado es como guardar sopa en una caja de zapatos: se puede, pero va a salir mal.

Hay tres tipos principales. El almacenamiento de objetos es para archivos sueltos: fotos, videos, documentos, respaldos. Funciona como una bodega infinita donde guardas cajas etiquetadas; cada archivo tiene una llave única y lo pides por internet. AWS S3, Google Cloud Storage y Azure Blob Storage son los ejemplos. Es baratísimo, duradero -los proveedores prometen que tus datos sobreviven aunque se queme un edificio entero, porque guardan copias en varios lugares- y perfecto para archivos que no cambian a cada rato.

El almacenamiento en bloque es el "disco duro" que le conectas a tu máquina virtual: rápido, pensado para que el sistema operativo lo use como su disco. Cuando tu máquina virtual necesita un disco, le asignas un volumen de este tipo. Si borras la máquina pero conservas el volumen, tus datos siguen ahí.

El almacenamiento de archivos compartido es para cuando varias máquinas necesitan leer los mismos archivos al mismo tiempo, como una carpeta de red. Menos común para quien empieza, pero existe.

Para datos estructurados están las bases de datos administradas: el proveedor se encarga de instalar, actualizar, respaldar y vigilar el servidor de base de datos; tú solo te conectas y guardas tus datos. AWS RDS te da MySQL, PostgreSQL y otros con unos clics; Google Cloud SQL y Azure Database hacen lo mismo. ¿Por qué pagar por esto en vez de instalar la base de datos tú mismo en tu máquina virtual? Porque los respaldos automáticos, la recuperación ante fallas y las actualizaciones de seguridad son trabajo tedioso y crítico: si tu base de datos se corrompe un domingo a las 3 de la mañana, prefieres que haya un equipo -o un servicio automatizado- cuidándola.

Además existen las bases de datos NoSQL administradas, como DynamoDB de AWS o Firestore de Google: guardan datos en formato flexible, tipo documentos JSON, y escalan automáticamente a millones de operaciones. Son ideales cuando tus datos no encajan bien en tablas rígidas o cuando el tráfico puede explotar de repente.

Analogía cotidiana: el almacenamiento de objetos es una bodega de auto-almacenamiento: barata, espaciosa, vas cuando necesitas algo. El almacenamiento en bloque es el cajón de tu escritorio: rápido, a la mano, para lo de todos los días. La base de datos administrada es contratar un contador: tú le das tus números y él se encarga de llevar los libros, hacer respaldos y no perder nada.

Error común de principiante: guardar archivos subidos por usuarios -fotos de perfil, documentos- dentro del disco de la máquina virtual. Cuando esa máquina se apague, se actualice o se reemplace, esos archivos se pierden. Los archivos de usuarios van en almacenamiento de objetos; el disco de la máquina es para el sistema y la aplicación, que deben poder reconstruirse en minutos.

Buena práctica: activa el control de versiones en tu almacenamiento de objetos para archivos importantes -así puedes recuperar una versión anterior si algo se sobrescribe por error-, cifra los datos sensibles, y define reglas de ciclo de vida: por ejemplo, que los respaldos viejos se muevan automáticamente a almacenamiento más barato o se borren después de cierto tiempo. Pagar por guardar basura digital que nadie va a usar es uno de los gastos hormiga más comunes en la nube.

Edge computing: el cómputo se acerca a los datos

Hasta aquí todo el cómputo pasaba en los centros de datos lejanos: tus datos viajaban por internet hasta la nube, se procesaban ahí y la respuesta viajaba de regreso. Para la mayoría de las aplicaciones eso está perfecto. Pero hay situaciones donde ese viaje de ida y vuelta es un problema, y ahí entra el edge computing: procesar los datos cerca de donde se generan, en "el borde" de la red.

La razón número uno es la latencia, o sea el tiempo que tarda un dato en ir y volver. La luz viaja rápido, pero no infinitamente rápido, y además tu paquete pasa por muchos equipos intermedios. Un viaje de ida y vuelta a un centro de datos en otro país puede tardar 100 o 200 milisegundos. Para cargar una página web no lo notas. Pero para un coche autónomo que debe frenar ante un peatón, 200 milisegundos a 60 kilómetros por hora son más de 3 metros de recorrido: la diferencia entre frenar a tiempo y no. Por eso el coche procesa las imágenes de sus cámaras en su propia computadora, ahí mismo, sin esperar a la nube.

La razón número dos es el internet de las cosas. Imagina una fábrica con 10,000 sensores reportando temperatura, vibración y presión cada segundo. Mandar todo ese torrente de datos crudos a la nube cuesta ancho de banda y dinero, y la mayoría de esos datos son aburridos: "todo normal, todo normal, todo normal". Con edge computing, una pequeña computadora en la fábrica analiza los datos localmente y solo manda a la nube lo importante: "el motor 7 está vibrando raro, mándame un técnico" o un resumen cada hora. Menos datos viajando, menos costo, respuesta inmediata.

La razón número tres es la privacidad y la regulación. Hay datos que por ley o por sentido común no deben salir de cierto lugar: historiales médicos, imágenes de cámaras de seguridad de una empresa, datos de un gobierno. Procesarlos localmente en el borde reduce el riesgo de exponerlos en el camino y ayuda a cumplir regulaciones que exigen que los datos se queden en el país o en las instalaciones.

¿Dónde vive el edge en la práctica? En una escala: primero, el dispositivo mismo -el sensor, la cámara, el teléfono- que cada vez tiene más capacidad de cómputo; a esto a veces se le llama "far edge" o cómputo en el dispositivo. Luego, las puertas de enlace y mini-servidores locales: una computadora en la sucursal, en la torre de telefonía celular, en la fábrica. Y finalmente la nube, para el análisis profundo, el almacenamiento de largo plazo y el entrenamiento de modelos de inteligencia artificial. El patrón típico es: el borde decide rápido con reglas simples o modelos ligeros, y la nube aprende lento con todos los datos agregados. Por ejemplo, el modelo que detecta fallas se entrena en la nube con datos de cien fábricas, y luego se envía la versión ligera a cada fábrica para que detecte fallas en tiempo real.

Ojo: edge no reemplaza a la nube, la complementa. La nube sigue siendo el cerebro central donde se guardan los historiales, se entrenan los modelos grandes y se coordinan las decisiones globales. El edge son los reflejos: rápidos, locales, automáticos. Un buen diseño usa cada uno para lo que hace mejor.

Analogía cotidiana: la nube es la oficina central de una empresa y el edge son las sucursales. La sucursal atiende al cliente que tiene enfrente sin llamar a la central para cada decisión pequeña; la central define la estrategia, guarda los archivos históricos y capacita al personal. Si la sucursal tuviera que pedir permiso a la central para cada cosa, todo sería lentísimo.

Error común de principiante: pensar que edge significa "sin nube" y diseñar sistemas donde cada dispositivo es una isla que nunca reporta nada. Sin la nube pierdes la visión global: no puedes comparar fábricas entre sí, ni mejorar tus modelos con datos nuevos, ni ver un tablero con todo tu negocio. El edge decide en local, pero casi siempre debe reportar resúmenes a la nube.

Buena práctica: define desde el diseño qué se procesa en el borde y qué en la nube, con criterios claros: ¿necesita respuesta en menos de 50 milisegundos? Borde. ¿Necesita ver datos de muchos lugares a la vez? Nube. ¿Los datos no pueden salir del edificio? Borde. Esa tabla de decisiones te ahorra discusiones infinitas más adelante.

Costos: cómo no llevarte sorpresas en la factura

Aquí va la lección que todo profesional de nube aprende, la mayoría por las malas: la nube es elástica y de pago por uso, lo que significa que también puedes gastar de más con la misma facilidad. Hay historias legendarias de estudiantes que dejaron una máquina potente encendida todo un mes y recibieron facturas de cientos de dólares, o de llaves filtradas que alguien usó para minar criptomonedas. No te va a pasar a ti, porque vas a aprender las reglas del dinero en la nube.

Regla 1: entiende que casi todo se cobra por separado. La máquina virtual cobra por hora encendida. El almacenamiento cobra por gigabyte al mes. La transferencia de datos hacia afuera de la nube -que tus usuarios descarguen cosas- cobra por gigabyte. Las peticiones a la base de datos, las ejecuciones de funciones, todo suma. La factura de nube parece el ticket de un supermercado con cien líneas: cada servicio es un producto distinto.

Regla 2: lo más caro suele ser lo que no ves. Los principiantes miran el precio de la máquina virtual y olvidan la transferencia de datos saliente, que en algunos proveedores es sorprendentemente cara. Sacar un terabyte de datos de la nube puede costar más que guardar ese terabyte todo un mes. Por eso los videos y archivos pesados se sirven desde redes de distribución de contenido -CDN-, que son más baratas para eso, y por eso el edge computing también ahorra dinero: menos datos viajando, menos factura.

Regla 3: apaga lo que no uses. Suena obvio, pero es la fuga de dinero número uno: máquinas de prueba encendidas, bases de datos de un experimento de hace tres meses, discos huérfanos que quedaron cuando se borró la máquina. La nube cobra por minuto, las 24 horas. Acostúmbrate al ritual: terminas de practicar, borras o apagas. Ponlo en tu lista de cierre como quien apaga las luces al salir.

Regla 4: usa las alarmas desde el día uno. Los tres grandes proveedores permiten crear alertas de facturación: "avísame por correo si mi gasto proyectado del mes pasa de 5 dólares". Configúrala hoy mismo, antes de crear tu primer recurso. Es gratis y es tu red de seguridad.

Regla 5: aprovecha los descuentos por compromiso solo cuando tengas historial. Los proveedores venden "instancias reservadas" o "planes de ahorro": pagas menos por hora a cambio de comprometerte a usar cierta capacidad por uno o tres años. Para una empresa estable es un gran ahorro; para ti, que vas empezando y todo cambia cada semana, es una trampa. Primero mide tu uso real durante meses, luego decides.

Regla 6: etiqueta y revisa. Ponle etiquetas -proyecto, ambiente, dueño- a cada recurso. Una vez al mes, abre el reporte de costos y pregunta de cada línea: "¿esto sigue siendo necesario?". Las empresas que no hacen esto terminan pagando por servidores que nadie recuerda para qué servían. Hay un término para eso: "zombis" de la nube.

Analogía cotidiana: la nube es como un taxi con taxímetro. Es comodísimo: lo pides cuando quieres, te lleva a donde quieras, no te preocupas por gasolina ni mantenimiento. Pero si te duermes en el asiento de atrás con el taxímetro corriendo, el susto al despertar es mayúsculo. Las alarmas de facturación son tu despertador.

Error común de principiante: probar un tutorial que crea recursos potentes -una base de datos grande, un clúster de Kubernetes- y cerrar la laptop sin borrarlos, pensando que "como cerré la página ya se apagó". No: los recursos de nube siguen corriendo aunque cierres el navegador. Borrar es una acción explícita que tienes que hacer en el panel o con comandos.

Buena práctica: para aprender, prefiere siempre el tamaño más pequeño que funcione y la capa gratuita; para proyectos reales, empieza pequeño y escala cuando las métricas lo pidan, no cuando la ansiedad lo pida. Y documenta cuánto cuesta tu arquitectura: un buen ingeniero

de nube sabe decir "este sistema cuesta aproximadamente X al mes y así se desglosa", igual que un buen cocinero sabe cuánto cuesta cada platillo.

Nube + IA + IoT: la combinación ganadora

Por separado, la nube, la inteligencia artificial y el internet de las cosas ya son poderosos. Juntos forman el trío que está detrás de los proyectos más interesantes de esta década, y entender cómo se combinan te pone por delante de la mayoría.

El internet de las cosas -IoT- son los dispositivos conectados: sensores, cámaras, wearables, maquinaria, electrodomésticos. Generan datos del mundo físico en cantidades brutales. La nube es donde esos datos se guardan, se procesan en grande y se convierten en decisiones. La inteligencia artificial es la que encuentra patrones en esos datos que ningún humano vería: predecir que una máquina va a fallar antes de que falle, detectar un fraude entre millones de transacciones, reconocer una enfermedad en una radiografía.

El ciclo virtuoso funciona así. Los dispositivos IoT capturan datos del mundo real y los mandan -directo o vía edge- a la nube. En la nube, esos datos alimentan modelos de IA que aprenden y mejoran: cada dato nuevo hace al modelo un poco más listo. Los modelos mejorados se despliegan de vuelta a los dispositivos o al edge, que ahora toman mejores decisiones en tiempo real. Mejores decisiones generan mejores datos, y el ciclo se repite. Es una espiral ascendente: mientras más se usa el sistema, más inteligente se vuelve.

Ejemplos reales que ya existen. Agricultura inteligente: sensores de humedad en el suelo, drones que fotografían los cultivos, modelos de IA que deciden exactamente cuánta agua y fertilizante necesita cada zona del terreno. Resultado: más cosecha con menos agua. Mantenimiento predictivo: sensores de vibración en motores de fábrica, IA entrenada en la nube que avisa "este rodamiento va a fallar en dos semanas", el técnico lo cambia en un mantenimiento programado en vez de sufrir un paro de emergencia que cuesta miles por hora. Ciudades inteligentes: cámaras y sensores de tráfico que ajustan los semáforos en tiempo real para reducir embotellamientos. Salud: relojes que detectan arritmias y avisan al médico, con los datos procesados parte en el dispositivo y parte en la nube.

¿Por qué la nube es indispensable en esta combinación? Porque entrenar modelos de IA serios requiere una potencia de cómputo brutal: cientos de procesadores gráficos trabajando en paralelo durante días o semanas. Ninguna empresa -y mucho menos una persona- quiere comprar ese fierro para usarlo de vez en cuando. La nube lo renta por horas: entrenas tu modelo, pagas el tiempo usado, sueltas las máquinas. Además, los proveedores ofrecen IA como servicio: no necesitas ser doctor en matemáticas para usar reconocimiento de imágenes, transcripción de voz o traducción automática; invocas una API y listo. Eso es PaaS y SaaS

aplicados a la inteligencia artificial, y es lo que permite que un estudiante como tú construya en un fin de semana algo que hace diez años requería un laboratorio.

Y el edge completa el cuadro donde la latencia o la conectividad mandan: el tractor en el campo sin buena señal de internet sigue tomando decisiones con el modelo que lleva a bordo; la cámara de la fábrica detecta el defecto en milisegundos sin subir el video completo. Nube para aprender en grande, edge para actuar en chico y rápido, IoT para sentir el mundo físico, IA para entenderlo.

Analogía cotidiana: IoT son los sentidos -ojos, oídos, piel- que perciben el mundo. La nube es el cerebro que recuerda todo y piensa en grande. La IA es la experiencia acumulada que te permite reconocer patrones. El edge son los reflejos: quitas la mano del comal caliente sin pensarlo. Un cuerpo necesita todo eso junto para funcionar bien; un sistema moderno, también.

Error común de principiante: querer meter IA a todo sin tener datos. La IA aprende de datos; si tus sensores no están generando datos de calidad, o si no tienes suficientes ejemplos, el modelo no va a servir por más nube que rentes. La regla es: primero instrumenta y recolecta, después analiza, al final predice. Muchos proyectos fracasan por querer empezar en el paso tres.

Buena práctica: empieza por el problema, no por la tecnología. "Quiero predecir fallas en mis motores" es un buen inicio; "quiero usar IoT con IA en la nube" es una solución buscando un problema. Define qué decisión quieres mejorar, qué datos necesitas para mejorarla, y de ahí diseña qué va en el dispositivo, qué en el edge y qué en la nube. La arquitectura sigue al problema, nunca al revés.

Ejemplos paso a paso

Ejemplo 1: crear recursos en la nube desde la terminal

Los profesionales rara vez crean recursos con clics en el navegador: usan la línea de comandos porque es repetible, documentable y automatizable. Cada proveedor tiene su herramienta: AWS CLI para Amazon, gcloud para Google Cloud, az para Azure. En este ejemplo usamos AWS CLI para crear una máquina virtual, porque los comandos ilustran los conceptos que ya viste: región, imagen, tamaño, llave y firewall.

Primero se configura la herramienta con tus credenciales -que obtienes al crear tu cuenta- y tu región favorita:

```
aws configure
# Te pide: AWS Access Key ID, AWS Secret Access Key, region (ej. us-east-1),
# formato de salida (json). La llave secreta nunca se comparte ni se sube a
internet.
```

Luego creas el grupo de seguridad, o sea el firewall, con reglas para SSH y web:

```
# Crear el grupo de seguridad
aws ec2 create-security-group \
  --group-name mi-primer-firewall \
  --description "Firewall para mi primera VM de practica"

# Abrir el puerto 22 (SSH) solo para tu IP. Cambia 203.0.113.10 por tu IP real.
aws ec2 authorize-security-group-ingress \
  --group-name mi-primer-firewall \
  --protocol tcp --port 22 --cidr 203.0.113.10/32

# Abrir el puerto 80 (web) para todo el mundo
aws ec2 authorize-security-group-ingress \
  --group-name mi-primer-firewall \
  --protocol tcp --port 80 --cidr 0.0.0.0/0
```

Después generas tu llave de acceso y lanzas la máquina:

```
# Crear la llave y guardarla en un archivo
aws ec2 create-key-pair --key-name mi-llave --query 'KeyMaterial' \
  --output text > mi-llave.pem
chmod 400 mi-llave.pem

# Lanzar una maquina t2.micro con Ubuntu (capa gratuita) en la region configurada
aws ec2 run-instances \
  --image-id ami-0c55b159cbfafa1f0 \
  --instance-type t2.micro \
  --key-name mi-llave \
  --security-groups mi-primer-firewall \
  --tag-specifications
  'ResourceType=instance,Tags=[{Key=Name,Value=mi-primer-vm},{Key=Proyecto,Value=curso-fusion}]'
```

El comando te devuelve un JSON con los datos de la máquina, incluyendo su ID. Con ese ID preguntas su dirección IP pública:


```
aws ec2 describe-instances \
  --instance-ids i-0abcd1234efgh5678 \
  --query 'Reservations[0].Instances[0].PublicIpAddress' \
  --output text
```

Y te conectas:

```
ssh -i mi-llave.pem ubuntu@54.123.45.67
```

¿Qué aprendiste aquí? Que cada clic del panel web tiene su equivalente en comando, y que una secuencia de comandos es un procedimiento que puedes guardar, compartir y repetir sin errores. Nota las etiquetas -Name y Proyecto-: son las que te salvan cuando revises la factura. Y nota que el ID de la imagen -ami-0c55b159cbfafa1f0- cambia por región y con el tiempo; en la práctica lo buscarías con un comando de consulta en vez de copiarlo de un tutorial. Cuando termines de practicar, no olvides apagarla: `aws ec2 terminate-instances --instance-ids i-0abcd1234efgh5678`.

Ejemplo 2: empaquetar tu API en un contenedor con Docker

Un contenedor es una cajita que lleva tu programa con todo lo que necesita: el lenguaje, las librerías, la configuración. La gracia es que corre igual en tu laptop, en la máquina virtual de un compañero y en la nube, porque la cajita aísla tu programa de las diferencias del ambiente. Docker es la herramienta estándar para crear y correr contenedores, y el archivo que describe cómo construir la cajita se llama Dockerfile.

Supón que tienes una API sencilla en Python con Flask, en un archivo `app.py`, y una lista de dependencias en `requirements.txt`. El Dockerfile se ve así:

```
# Imagen base: Python 3.11 liviano, ya trae el interprete instalado
FROM python:3.11-slim

# Carpeta de trabajo dentro del contenedor
WORKDIR /app

# Copiar primero las dependencias para aprovechar el cache de Docker
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

# Copiar el codigo de la aplicacion
COPY app.py .

# Documentar que la API escucha en el puerto 5000
EXPOSE 5000

# Comando que arranca la API cuando el contenedor inicia
CMD ["python", "app.py"]
```

El archivo requirements.txt es simple:

```
flask==3.0.3
gunicorn==21.2.0
```

Y para orquestar todo -la API más una base de datos PostgreSQL- usas un archivo YAML de Docker Compose, que describe los servicios como un plano:

```

version: "3.9"
services:
  api:
    build: .
    ports:
      - "5000:5000"
    environment:
      - DATABASE_URL=postgresql://usuario:secreto@db:5432/miapp
      - ENTORNO=desarrollo
    depends_on:
      - db
  db:
    image: postgres:16
    environment:
      - POSTGRES_USER=usuario
      - POSTGRES_PASSWORD=secreto
      - POSTGRES_DB=miapp
    volumes:
      - datos-db:/var/lib/postgresql/data
volumes:
  datos-db:

```

Fíjate en los detalles profesionales: las credenciales van en variables de entorno, no escritas en el código; la base de datos guarda sus datos en un volumen para que no se pierdan si el contenedor se reinicia; la API declara que depende de la base de datos. Con un solo comando levantas todo: `docker compose up`. Y esa misma cajita es la que luego subes a la nube: los servicios de contenedores de AWS, Google Cloud y Azure aceptan imágenes Docker tal cual. Aprende esto una vez y te sirve en los tres proveedores.

Ejemplo 3: script de despliegue de una página estática

Este ejemplo une varias piezas: una página estática sencilla, su subida a almacenamiento de objetos -S3 de AWS- y la automatización con un script de bash para que desplegar sea un solo comando. La página es un `index.html` mínimo:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Mi primera pagina en la nube</title>
</head>
<body>
<h1>Hola desde la nube</h1>
<p>Esta pagina vive en un bucket S3 de AWS.</p>
<p>Aprendiendo en Lucas EGA Academy: lucasegaacademy.com</p>
</body>
</html>
```

El script desplegar.sh hace todo el trabajo: crea el bucket si no existe, lo configura como sitio web, sube los archivos y muestra la dirección pública:

```
#!/bin/bash
# Script de despliegue de pagina estatica a AWS S3
# Uso: ./desplegar.sh

BUCKET="mi-pagina-fusion-12345"
REGION="us-east-1"

set -e # si algo falla, el script se detiene en vez de seguir a ciegas

echo "1. Creando el bucket $BUCKET..."
aws s3api create-bucket --bucket "$BUCKET" --region "$REGION" 2>/dev/null || echo "
  (el bucket ya existia, continuando)"

echo "2. Activando alojamiento de sitio web estatico..."
aws s3 website "s3://$BUCKET" --index-document index.html --error-document
index.html

echo "3. Quitando el bloqueo de acceso publico (solo lectura web)..."
aws s3api put-public-access-block --bucket "$BUCKET" \
                                --public-access-block-configuration
BlockPublicAcls=false,IgnorePublicAcls=false,BlockPublicPolicy=false,RestrictPublic
Buckets=false

echo "4. Permitiendo lectura publica de los archivos..."
aws s3api put-bucket-policy --bucket "$BUCKET" --policy '{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": "*",
    "Action": "s3:GetObject",
    "Resource": "arn:aws:s3:::'"$BUCKET"'/*"
  }]
}'

echo "5. Subiendo los archivos..."
aws s3 sync . "s3://$BUCKET" --exclude "desplegar.sh" --delete

echo "Listo. Tu pagina esta en:"
echo "http://$BUCKET.s3-website-$REGION.amazonaws.com"
```

Lo valioso de este ejemplo no es solo la página en línea: es el hábito. El script documenta el procedimiento completo, cualquiera puede repetirlo, y desplegar una actualización es correr un comando. Observa las buenas prácticas aplicadas: el script se detiene si algo falla -set -e-, el bucket tiene nombre único global -los nombres de S3 no se pueden repetir en todo el mundo-, y se excluye el propio script de la subida. En un proyecto real, este script viviría en tu

repositorio y un servicio de integración continua lo ejecutaría solo cada vez que subas cambios: eso es DevOps en miniatura, y ya lo estás haciendo.

Práctica

Ejercicio 1: clasifica estos servicios en IaaS, PaaS o SaaS

Clasifica cada uno y justifica en una línea: a) rentar una máquina virtual en AWS EC2 e instalarle Linux tú mismo; b) subir tu código a Heroku y que la plataforma lo ponga en línea; c) usar Gmail desde el navegador; d) guardar fotos en un bucket de S3; e) usar Google Docs para escribir un documento; f) contratar una base de datos PostgreSQL administrada en RDS.

Solución paso a paso. El criterio es: ¿quién administra qué? Si tú administras desde el sistema operativo hacia arriba, es IaaS. Si el proveedor administra la plataforma y tú solo subes tu aplicación o tus datos, es PaaS. Si usas un software terminado sin administrar nada técnico, es SaaS.

a) IaaS: el proveedor te da el fierro virtualizado y tú instalas y configuras el sistema operativo y todo lo demás. b) PaaS: tú entregas tu código y Heroku se encarga de servidores, sistema operativo y ejecución. c) SaaS: usas el correo terminado; no administras nada, solo tu cuenta y tus mensajes. d) IaaS en sentido de almacenamiento: es infraestructura cruda -espacio para archivos-, tú decides qué guardar y cómo organizarlo; algunos lo discuten, pero cae del lado de infraestructura. e) SaaS: igual que Gmail, software terminado en el navegador. f) PaaS -o más preciso, base de datos como servicio, DBaaS, que es una forma de PaaS-: tú usas la base de datos, el proveedor administra el servidor, los respaldos y las actualizaciones.

La lección: no memorices listas de productos, aplica la pregunta "¿qué me toca administrar a mí?". Esa pregunta clasifica cualquier servicio nuevo que aparezca en el futuro.

Ejercicio 2: diseña la arquitectura de nube para una taquería

La taquería "El Faro" quiere: una página web con su menú y ubicación, un sistema para recibir pedidos en línea, y guardar las fotos de sus platillos. Propón qué servicios de nube usarías para cada cosa y de qué modelo -IaaS, PaaS o SaaS- es cada pieza. Supón que el dueño no sabe nada de tecnología y tú le vas a administrar todo.

Solución paso a paso. Paso 1: separa las necesidades. La página del menú es contenido que casi no cambia: candidata perfecta para página estática. Los pedidos en línea necesitan un programa que reciba datos y los guarde: una API más una base de datos. Las fotos son archivos: almacenamiento de objetos.

Paso 2: elige servicios simples y baratos. Para la página estática: alojamiento de sitio estático en S3 o Netlify -costo casi cero, sin servidores que cuidar-. Para los pedidos: una API en un PaaS como Heroku, App Engine o una función serverless como AWS Lambda -no quieres administrar servidores para una taquería-. Para la base de datos de pedidos: una base de datos administrada pequeña o un servicio como Firestore -respaldos automáticos incluidos-. Para las fotos: un bucket de almacenamiento de objetos.

Paso 3: justifica el modelo. Página estática: entre PaaS y SaaS según el servicio -Netlify se siente como PaaS porque subes tu código, pero casi no administras nada-. API: PaaS o serverless. Base de datos administrada: PaaS. Fotos: IaaS de almacenamiento. Nada de máquinas virtuales: sería como comprar un camión para repartir un taco.

Paso 4: piensa en costos y crecimiento. Con esta arquitectura, si la taquería tiene 10 pedidos al día o 1,000, la nube escala sola y el costo crece proporcional al uso. Si hubieras rentado un servidor fijo, pagarías lo mismo con 10 o con 1,000 pedidos, y con 1,000 se caería.

La lección: la mejor arquitectura no es la más impresionante, es la más simple que resuelve el problema. Empieza con servicios administrados y solo baja a IaaS si algo concreto lo exige.

Ejercicio 3: escribe los comandos para apagar todo al terminar de practicar

Escribe la secuencia de comandos de AWS CLI para: a) listar tus máquinas virtuales encendidas con su nombre y estado; b) detener -no borrar- una máquina específica; c) borrar definitivamente una máquina que ya no necesitas; d) listar tus buckets de S3. Explica la diferencia entre detener y borrar.

Solución paso a paso. Comando a: para ver qué tienes encendido antes de tocar nada:

```
aws ec2 describe-instances \
    --query 'Reservations[*].Instances[*].[Tags[?Key==`Name`].Value | [0],
State.Name, InstanceId]' \
    --output table
```

Esto muestra nombre, estado -running, stopped- e ID de cada máquina en una tabla legible. Siempre mira antes de actuar.

Comando b: detener la máquina la apaga pero la conserva, con su disco y su configuración. Deja de cobrarte el cómputo, aunque el disco sigue generando un costo pequeño:

```
aws ec2 stop-instances --instance-ids i-0abcd1234efgh5678
```

Comando c: borrarla -terminar- la elimina para siempre. Úsalo cuando la práctica terminó y no la vas a necesitar más:

```
aws ec2 terminate-instances --instance-ids i-0abcd1234efgh5678
```

Comando d: listar tus buckets para verificar que no dejaste archivos olvidados que sigan costando:

```
aws s3 ls
```

La diferencia clave: detener es pausar -la máquina existe, apagada, lista para encenderse después-; borrar es tirar a la basura -desaparece y no hay vuelta atrás-. Para practicar, lo habitual es detener entre sesiones y borrar al terminar el tema. Y la lección de dinero: una máquina detenida casi no cuesta, una máquina encendida olvidada cuesta todos los días. Este ejercicio vale dinero real: conviértelo en ritual de cierre.

Ejercicio 4: decide dónde procesar: ¿nube o edge?

Para cada caso, decide si el procesamiento principal conviene en la nube, en el edge o en una combinación, y justifica con latencia, ancho de banda o privacidad: a) app que cuenta cuántas personas entran a una tienda usando la cámara de la entrada; b) tablero global de ventas de una cadena con 200 sucursales; c) pulsera médica que detecta caídas en personas mayores y pide ayuda; d) entrenar un modelo que reconoce defectos en botellas de una embotelladora; e) sistema que abre la puerta de un edificio reconociendo caras.

Solución paso a paso. Aplica los tres criterios: ¿necesita responder en milisegundos? ¿genera demasiados datos para mandarlos todos? ¿los datos no deben salir del lugar?

a) Combinación con énfasis en edge: contar personas necesita analizar video en tiempo real -muchos datos por segundo que no conviene subir completos-. Una pequeña computadora en la tienda procesa el video localmente y manda a la nube solo el conteo: "entraron 42 personas esta hora". Barato en ancho de banda y respeta la privacidad de los clientes.

b) Nube: el tablero necesita ver datos de las 200 sucursales a la vez para compararlas. Ninguna sucursal tiene la foto completa; la nube agrega todo y sirve el tablero. La latencia no es crítica: un reporte que tarda dos segundos en cargar está bien.

c) Edge en el dispositivo más nube de apoyo: detectar la caída debe pasar en el momento, aunque no haya internet -la persona podría estar en el baño sin señal-. La pulsera detecta el patrón localmente y pide ayuda por la red disponible. La nube guarda el historial para el médico y mejora el algoritmo con datos de miles de usuarios.

d) Nube para entrenar, edge para detectar: entrenar el modelo requiere miles de fotos de botellas buenas y defectuosas y mucha potencia de cómputo -trabajo de nube-. Pero en la línea de producción, cada botella pasa en una fracción de segundo: la cámara con el modelo ya entrenado decide ahí mismo, en el edge, sin esperar.

e) Edge por privacidad y latencia: las caras de empleados y visitantes son datos sensibles que no deberían viajar a un servidor lejano si no es necesario, y la puerta debe abrirse en uno o dos segundos. El reconocimiento corre en un equipo local del edificio; la nube, si acaso, recibe solo el registro: "Juan entró a las 9:03".

La lección: no hay respuesta única, hay criterios. Memoriza la tabla: milisegundos críticos, datos masivos o privacidad local apuntan al edge; visión global, cómputo pesado y datos agregados apuntan a la nube. Casi todos los sistemas reales usan ambos.

Ejercicio 5: estima el costo mensual de un proyecto pequeño

Un proyecto personal va a usar: una máquina virtual pequeña encendida 24/7 -supón 0.0116 dólares por hora-, 50 GB de almacenamiento de objetos -0.023 dólares por GB al mes-, y 100 GB de transferencia de datos hacia afuera al mes -0.09 dólares por GB-. Calcula el costo mensual de cada parte y el total. Luego propón dos formas de bajar ese costo sin perder funcionalidad.

Solución paso a paso. Paso 1: la máquina virtual. Un mes tiene aproximadamente 730 horas -24 por 30.4 días-. Multiplica: $730 \text{ por } 0.0116 = 8.47$ dólares al mes. Paso 2: el almacenamiento. $50 \text{ por } 0.023 = 1.15$ dólares al mes. Paso 3: la transferencia. $100 \text{ por } 0.09 = 9.00$ dólares al mes. Fíjate: la transferencia cuesta más que la máquina. Paso 4: el total. $8.47 + 1.15 + 9.00 = 18.62$ dólares al mes, unos 320 pesos mexicanos aproximadamente.

Paso 5: formas de bajarlo. Opción uno: si la máquina solo se usa para practicar unas horas al día, apagarla el resto del tiempo. Usándola 4 horas diarias en vez de 24, el cómputo baja de 8.47 a unos 1.41 dólares. Opción dos: servir los archivos pesados -imágenes, videos- desde un CDN, que cobra menos por transferencia que la salida directa, o comprimir las imágenes para que pesen menos: menos gigabytes transferidos, menos dólares. Opción tres: revisar si el uso entra en la capa gratuita del proveedor, que cubriría la máquina pequeña y parte del almacenamiento el primer año.

La lección: haz este cálculo antes de construir, no después de recibir la factura. Un ingeniero que sabe estimar costos es un ingeniero en quien se puede confiar un presupuesto.

Ejercicio 6: escribe un archivo docker-compose para un blog con base de datos

Escribe un archivo YAML de Docker Compose que levante dos servicios: un blog -usa la imagen `wordpress:latest`- en el puerto 8080, y una base de datos MySQL -imagen `mysql:8`- con usuario, contraseña y base de datos para el blog. El blog debe conectarse a la base de datos usando el nombre del servicio, y los datos de MySQL deben sobrevivir si el contenedor se reinicia. Explica cada parte.

Solución paso a paso. El archivo quedaría así:

```
version: "3.9"
services:
  blog:
    image: wordpress:latest
    ports:
      - "8080:80"
    environment:
      - WORDPRESS_DB_HOST=db
      - WORDPRESS_DB_USER=bloguser
      - WORDPRESS_DB_PASSWORD=clave-segura-123
      - WORDPRESS_DB_NAME=blogdb
    depends_on:
      - db
  db:
    image: mysql:8
    environment:
      - MYSQL_DATABASE=blogdb
      - MYSQL_USER=bloguser
      - MYSQL_PASSWORD=clave-segura-123
      - MYSQL_ROOT_PASSWORD=clave-root-456
    volumes:
      - datos-mysql:/var/lib/mysql
volumes:
  datos-mysql:
```

Explicación por partes. `version` indica el formato del archivo. `services` define los dos contenedores. `blog` usa la imagen oficial de WordPress y mapea el puerto 8080 de tu máquina al 80 del contenedor, así abres `http://localhost:8080` en tu navegador. Las variables de entorno le dicen a WordPress dónde está la base de datos: `WORDPRESS_DB_HOST=db` usa el nombre del servicio como dirección, porque Docker Compose crea una red interna donde los

servicios se encuentran por su nombre. `depends_on` asegura que la base de datos arranque primero.

El servicio `db` usa MySQL 8 y sus variables crean la base de datos, el usuario y las contraseñas al primer arranque. El `volumes` monta un volumen persistente en la carpeta donde MySQL guarda sus datos: si borras y recreas el contenedor, el volumen sigue ahí con todo. La sección `volumes` de abajo declara ese volumen.

Para levantarlo: `docker compose up -d`. Para verlo: abre `http://localhost:8080`. Para apagarlo sin perder datos: `docker compose down -el volumen sobrevive-`. Para borrarlo todo incluyendo datos: `docker compose down -v`.

La lección: un archivo de 30 líneas describe un sistema completo que antes requería instalar y configurar dos programas a mano. Esa es la magia de los contenedores y la infraestructura como código: tu ambiente de desarrollo, el de pruebas y el de producción pueden ser idénticos, y "en mi máquina sí funciona" deja de ser un problema.

Resumen de la unidad

Llegaste al final de la unidad 4. Repasa los puntos clave, que son los que te van a servir en exámenes, entrevistas y proyectos reales.

La nube son computadoras rentadas a través de internet, con cinco características esenciales: autoservicio bajo demanda, acceso por red, agrupación de recursos, elasticidad rápida y servicio medido. Cambió las reglas del juego porque eliminó la necesidad de planear y comprar capacidad por adelantado: pides lo que necesitas, cuando lo necesitas, y pagas solo lo que usas.

Los tres modelos de servicio se distinguen por cuánto administras tú: en IaaS el proveedor da la infraestructura y tú pones desde el sistema operativo hacia arriba; en PaaS el proveedor maneja la plataforma y tú solo subes tu código y tus datos; en SaaS usas software terminado sin administrar nada. La analogía de la pizza te ayuda a recordarlos: rentar la cocina, que te den ingredientes preparados, o pedir la pizza a domicilio. La responsabilidad compartida significa que mientras más subes en la pirámide, menos control y menos trabajo.

Los tres grandes proveedores son AWS, el pionero con el catálogo más grande; Google Cloud, fuerte en datos e inteligencia artificial; y Azure, el favorito de las empresas del ecosistema Microsoft. Los tres ofrecen capas gratuitas generosas para aprender, y existen alternativas más simples como DigitalOcean para empezar. Elige un proveedor para profundizar; los conceptos se transfieren entre ellos.

Crear una máquina virtual sigue siempre los mismos pasos: elegir región, sistema operativo, tamaño, llave de acceso, firewall y lanzar. Conectarte por SSH te pone frente a una computadora a miles de kilómetros. Para páginas estáticas no necesitas máquinas virtuales: el almacenamiento de objetos configurado como sitio web, o servicios como Netlify, son más simples y baratos. Para APIs tienes tres caminos: serverless para tráfico irregular, PaaS para lanzar rápido, máquina virtual para control total.

El almacenamiento tiene tres sabores: objetos para archivos, bloque como disco de tus máquinas, y archivos compartidos para varias máquinas a la vez. Las bases de datos administradas te liberan de respaldos y mantenimiento a cambio de una cuota. Los archivos de usuarios van en almacenamiento de objetos, nunca en el disco de una máquina desechable.

Edge computing es procesar cerca de donde se generan los datos. Conviene cuando la latencia es crítica -coche autónomo, videojuegos-, cuando los dispositivos generan demasiados datos para mandarlos todos -fábrica con miles de sensores-, o cuando los datos no deben salir del lugar por privacidad o regulación. No reemplaza a la nube: el borde decide rápido en local y la nube aprende en grande con los datos agregados.

Los costos se controlan con seis reglas: entiende que todo se cobra por separado, cuida la transferencia de datos saliente que suele ser lo más caro, apaga lo que no uses, pon alarmas de facturación desde el día uno, no te comprometas a descuentos de largo plazo sin historial, y etiqueta y revisa tus recursos cada mes. La nube es un taxi con taxímetro: comodísimo, pero no te duermas con el taxímetro corriendo.

La combinación ganadora es nube más IA más IoT: los dispositivos sienten el mundo, la nube guarda y procesa en grande, la IA encuentra patrones, y el edge actúa rápido donde hace falta. El ciclo virtuoso -datos que mejoran modelos que mejoran decisiones que generan mejores datos- está detrás de la agricultura inteligente, el mantenimiento predictivo y las ciudades inteligentes. Y la regla de oro: primero el problema, después la tecnología; primero datos de calidad, después inteligencia artificial.

En la siguiente unidad vas a necesitar todo esto: desplegarás proyectos que viven en la nube, usarás APIs de inteligencia artificial como servicio, y conectarás dispositivos que reportan datos desde el borde. La nube ya no es un tema exótico de especialistas: es el piso sobre el que se construye casi todo el software moderno, y ahora tú sabes cómo funciona por dentro. Sigue practicando con la capa gratuita, rompe cosas en tu ambiente de pruebas -así se aprende de verdad-, y nos vemos en la unidad 5.

UNIDAD 5: Ciberseguridad moderna

Introducción

Vivimos en una época en la que tu casa entera puede caber en un teléfono: fotos de tu familia, tus cuentas del banco, tus conversaciones privadas, tus documentos de trabajo, tus contraseñas y hasta la llave digital de tu casa inteligente. Todo eso convive en un aparato del tamaño de tu mano que llevas a todos lados y que conectas a redes que ni siquiera controlas. La ciberseguridad moderna es el arte de proteger ese universo digital con hábitos sencillos, herramientas accesibles y criterio bien afilado. No necesitas ser un experto para estar seguro, necesitas ser una persona informada.

Por qué la ciberseguridad es asunto de todos

Hay una idea equivocada muy común: "yo no soy nadie importante, a mí nadie me va a hackear". La realidad es justo la contraria. La mayoría de los ataques no buscan celebridades ni empresas gigantes; buscan volumen. Los programas maliciosos se lanzan a millones de direcciones de correo al mismo tiempo, como redes de pesca que atrapan a quien sea que muerda el anzuelo. No importa si eres estudiante, ama de casa, comerciante o gerente: si tienes un correo electrónico, una cuenta de banco o una red social, ya eres un objetivo potencial.

Piensa en tu casa. No dejas la puerta abierta solo porque "no tienes nada valioso". La cierras por hábito, por sentido común. En internet pasa lo mismo: la seguridad digital no es paranoia, es el equivalente a cerrar la puerta con llave. Los atacantes prefieren a las víctimas descuidadas porque son las más fáciles. Si aprendes a cerrar bien tus puertas digitales, la mayoría de los delincuentes buscarán otra víctima más fácil. Esa es la primera gran lección de esta unidad: no necesitas ser impenetrable, necesitas ser más difícil de atacar que el promedio.

El panorama de amenazas actual

El mundo de las amenazas digitales cambió muchísimo en los últimos años. Antes, un virus llegaba en un disquete y dañaba tu computadora. Hoy existe toda una industria criminal organizada. El ransomware secuestra los archivos de una empresa o una persona y pide un rescate en criptomonedas para devolverlos. El phishing imita correos de bancos, tiendas y servicios conocidos para robar tus contraseñas. La ingeniería social no ataca tu computadora, te ataca a ti: te manipula psicológicamente para que tú mismo entregues la información.

Además, los ataques ya no solo llegan por la computadora. Llegan por WhatsApp, por mensajes de texto, por llamadas telefónicas falsas, por redes WiFi públicas tramposas y hasta por los dispositivos de tu casa inteligente: cámaras, bocinas, focos y cerraduras que se conectan a internet. Cada aparato conectado es una puerta potencial. Por eso esta unidad cubre el panorama completo: desde cómo crear contraseñas fuertes hasta cómo proteger tu hogar inteligente y qué carrera profesional puedes construir en este campo que cada día tiene más demanda.

Lo que vas a lograr al terminar esta unidad

Al final de esta unidad sabrás identificar las amenazas más comunes y entenderás cómo piensan los atacantes. Aprenderás a crear y manejar contraseñas fuertes sin tener que memorizarlas todas, activarás la autenticación de dos factores en tus cuentas importantes y sabrás detectar un correo falso paso a paso antes de hacer clic. También aprenderás a navegar seguro en redes WiFi públicas, a configurar copias de seguridad automáticas para no perder nada nunca, a proteger los dispositivos inteligentes de tu hogar y a tener un plan claro de qué hacer si algún día te hackean. Y si el tema te apasiona, cerrarás la unidad con una guía de cómo convertir este conocimiento en una carrera profesional.

Esta unidad está pensada para que la leas con calma y practiques cada cosa. La seguridad no se aprende de memoria: se aprende haciendo. Cada sección incluye ejemplos concretos y la sección de práctica te va a poner a prueba con ejercicios reales. Vamos paso a paso.

Un dato para motivarte: los estudios de la industria muestran que más del 80 por ciento de los hackeos exitosos empiezan con un error humano (una contraseña débil, un clic en un enlace falso) y no con tecnología sofisticada. Eso significa que el eslabón más débil también puede convertirse en el más fuerte con puro conocimiento. Tú estás a punto de dar ese salto.

Tu mentalidad de seguridad: piensa como defensor

La herramienta de seguridad más poderosa no se descarga: es tu criterio. Los profesionales de ciberseguridad hablan de "mentalidad de defensor", que es un conjunto de tres reflejos que puedes entrenar desde hoy.

El primer reflejo es la desconfianza sana: no la paranoia, sino la costumbre de verificar antes de confiar. Si algo te pide dinero, datos o acceso urgente, tu primera reacción debe ser "voy a comprobarlo por mi cuenta", no "voy a obedecer rápido". Este reflejo, por sí solo, bloquea la mayoría de los fraudes.

El segundo reflejo es pensar en capas. Ninguna medida es perfecta: las contraseñas se filtran, el 2FA puede fallar si pierdes el teléfono, el antivirus no lo ve todo. La seguridad real viene de combinar varias capas para que, si una falla, las demás te sostengan. Contraseña fuerte más 2FA más respaldo más criterio: eso es una fortaleza, no una puerta.

El tercer reflejo es el mantenimiento. La seguridad no es algo que configuras una vez un domingo y olvidas para siempre. Es como la salud: revisiones periódicas, hábitos diarios y atención a las señales de alerta. Agenda en tu calendario una "revisión de seguridad" cada tres meses: actualizar, revisar permisos, verificar respaldos. Treinta minutos cuatro veces al año te ahorran el peor susto de tu vida digital.

Con esta mentalidad, todo lo técnico de esta unidad encaja: cada herramienta que aprendas será una capa más de tu fortaleza personal.

Explicación detallada

Amenazas: malware, ransomware, phishing e ingeniería social

Empecemos por conocer al enemigo. El malware es un término general para cualquier programa malicioso: virus, troyanos, programas espía (spyware) y programas de publicidad no deseada (adware). Un virus necesita que lo ejecutes para activarse, como un archivo adjunto infectado. Un troyano se disfraza de programa legítimo: parece un juego o una utilidad, pero por dentro trae el daño. El spyware se instala en silencio y registra lo que haces, incluyendo tus contraseñas. Entra a tu equipo cuando descargas programas de sitios no oficiales, cuando conectas memorias USB desconocidas o cuando haces clic en enlaces sospechosos.

El ransomware es hoy la amenaza más rentable para los criminales. Cifra todos tus archivos (documentos, fotos, bases de datos) y te muestra un mensaje exigiendo un pago, normalmente en criptomonedas, a cambio de la clave para recuperarlos. Ataca a personas, hospitales, escuelas y empresas. La regla de oro es clara: nunca pagues el rescate, porque no hay garantía de que te devuelvan nada y porque pagar financia más ataques. La verdadera defensa contra el ransomware es tener copias de seguridad actualizadas: si tus archivos están respaldados, el ataque pierde todo su poder.

El phishing es el engaño más popular de internet. Recibes un correo que parece de tu banco, de una paquetería, de Netflix o de tu jefe, con un mensaje urgente: "tu cuenta será suspendida", "tienes un paquete detenido", "verifica tu identidad". El enlace te lleva a una página idéntica a la real donde te piden tu contraseña. En cuanto la escribes, cae en manos del atacante. Existen variantes: el smishing (por SMS) y el vishing (por llamadas de voz). En México son muy comunes los mensajes falsos de paqueterías y de supuestos ejecutivos bancarios que te llaman para "verificar" un cargo.

La ingeniería social es la técnica madre de todas las anteriores. No hackea computadoras: hackea personas. El atacante estudia a su víctima en redes sociales, descubre nombres de familiares, mascotas, fechas importantes, y usa esa información para ganarse su confianza. Puede hacerse pasar por soporte técnico, por un compañero de trabajo o por un familiar en apuros. Su arma favorita es la urgencia: te presiona para que actúes rápido sin pensar. La defensa es simple y poderosa: detente, verifica por otro canal (llama tú mismo al número oficial) y nunca entregues códigos, contraseñas ni datos personales a quien te contacta de la nada.

Contraseñas fuertes y gestores de contraseñas

Tu contraseña es la primera puerta de tu casa digital, y la mayoría de las personas usan puertas de cartón. Los errores clásicos: usar la misma contraseña en todos lados, usar fechas de nacimiento o nombres de familiares, usar secuencias como "123456" o "password", y compartir contraseñas por WhatsApp. Los atacantes lo saben y usan programas que prueban millones de combinaciones por segundo, además de listas de contraseñas filtradas de hackeos anteriores.

Una contraseña fuerte de hoy tiene tres características: es larga (mínimo 12 caracteres, ideal 16 o más), es única (una distinta para cada servicio) y es impredecible (no contiene datos personales). El truco más práctico es la frase de contraseña: toma cuatro o cinco palabras al azar y únelas con números y símbolos. Por ejemplo, una frase como "taco bicicleta luna 42 !" es larguísima, fácil de recordar para ti e imposible de adivinar para un programa. Evita frases de canciones famosas o refranes conocidos.

Pero aquí viene el problema real: nadie puede memorizar 50 contraseñas distintas. La solución son los gestores de contraseñas: programas como Bitwarden, 1Password o KeePass que guardan todas tus contraseñas cifradas y solo te piden recordar UNA contraseña maestra. El gestor genera contraseñas aleatorias larguísimas para cada sitio y las rellena automáticamente cuando inicias sesión. Así, aunque un sitio sea hackeado y se filtre tu contraseña de ahí, tus demás cuentas siguen seguras porque cada una tiene una contraseña diferente. Es uno de los cambios con mayor impacto en tu seguridad y toma menos de una hora configurarlo.

Autenticación de dos factores (2FA)

Imagina que tu casa tiene dos cerraduras con llaves distintas: aunque un ladrón consiga una llave, no puede entrar. Eso es la autenticación de dos factores. Después de escribir tu contraseña (primer factor), el servicio te pide una segunda prueba de identidad (segundo factor): un código de 6 dígitos que cambia cada 30 segundos en una app de tu teléfono, una confirmación en tu celular o una llave física de seguridad.

Activa el 2FA al menos en tu correo principal, tu banco, tus redes sociales y tu gestor de contraseñas. El correo es la cuenta más importante de todas: con acceso a tu correo, un atacante puede restablecer las contraseñas de casi todo lo demás. Hay tres tipos de segundo factor, de menos a más seguro: códigos por SMS (mejor que nada, pero se pueden interceptar), apps autenticadoras como Google Authenticator o Authy (muy recomendables), y llaves físicas como las YubiKey (el nivel más alto, ideales para cuentas críticas).

Un error común es pensar "ya tengo contraseña fuerte, no necesito 2FA". Las contraseñas se filtran, se adivinan y se roban con phishing; el 2FA es tu red de seguridad cuando la contraseña falla. Otro error es no guardar los códigos de recuperación que el servicio te da al activar el 2FA: guárdalos impresos en un lugar seguro, porque si pierdes tu teléfono los necesitarás para recuperar tu cuenta.

Cómo detectar correos y mensajes falsos paso a paso

Detectar un phishing es una habilidad que se entrena como un músculo. Sigue este protocolo de cinco pasos cada vez que recibas un mensaje que te pida hacer algo urgente.

Paso 1: frena y respira. El phishing vive de la urgencia. Si el mensaje dice "actúa ahora o pierdes tu cuenta", esa prisa es la primera señal de alerta. Los bancos y servicios serios nunca te presionan así. Date 30 segundos para pensar antes de tocar nada.

Paso 2: revisa el remitente con lupa. No te fijas solo en el nombre que se muestra ("Banco Nacional"), fíjate en la dirección real de correo. Un correo legítimo del banco vendrá de un dominio oficial como `avisos@tubanco.com`. Un correo falso vendrá de algo parecido pero tramposo: `avisos@tubanco-soporte.com`, `seguridad@tubanco.com.mx.falso.net` o `tubanco@gmail.com`. En mensajes de WhatsApp o SMS, desconfía de números desconocidos que se presentan como empresas.

Paso 3: pasa el cursor sobre los enlaces sin hacer clic (en el teléfono, mantén presionado el enlace para ver a dónde lleva). La dirección real aparece abajo o en una ventanita. Si el texto dice "`www.tubanco.com`" pero el enlace real apunta a "`www.tubanco-verificacion.ru`" o a una dirección IP numérica, es fraude. Nunca descargues archivos adjuntos inesperados, sobre todo si terminan en `.exe`, `.zip`, `.scr` o son documentos de Office que piden "habilitar macros".

Paso 4: analiza el contenido. Busca faltas de ortografía, saludos genéricos ("estimado cliente" en lugar de tu nombre), amenazas y premios increíbles. Las empresas reales te llaman por tu nombre y escriben bien. Si te ofrecen un premio que no esperabas o te anuncian un cargo que no reconoces, sospecha.

Paso 5: verifica por otro canal. Si el mensaje dice ser de tu banco, no uses el enlace del mensaje: abre tú mismo la app del banco o marca el número oficial de atrás de tu tarjeta. Si dice ser tu jefe pidiéndote una transferencia urgente, llámalo por teléfono. Este paso, por sí solo, desactiva el 99 por ciento de los intentos de fraude.

Error común de principiantes: creer que solo los correos mal escritos son falsos. Hoy los atacantes usan inteligencia artificial para redactar mensajes perfectos en español. La ortografía ya no es garantía. Lo que sí delata al fraude es la combinación de urgencia más enlace más petición de datos.

Seguridad en WiFi y redes públicas: el papel de la VPN

Las redes WiFi públicas (cafés, aeropuertos, plazas) son cómodas y peligrosas. En una red abierta, sin contraseña, tu información viaja como una postal: cualquiera con las herramientas adecuadas puede leerla. Existe un ataque clásico llamado "gemelo malvado" (evil twin): el atacante crea una red con el nombre del café ("`CafeCentral_WiFi_Gratis`"), te conectas a ella pensando que es la legítima, y todo tu tráfico pasa por su computadora.

Las buenas prácticas en redes públicas son: prefiere siempre tus datos móviles sobre el WiFi público cuando vayas a hacer algo sensible como entrar al banco; desactiva la conexión automática a redes WiFi en tu teléfono para que no se conecte solo a redes tramposas; verifica con el personal del lugar cuál es el nombre exacto de la red oficial; y usa una VPN.

Una VPN (red privada virtual) crea un túnel cifrado entre tu dispositivo e internet. Es como mandar tus postales dentro de un tubo blindado: aunque alguien intercepte el tráfico, solo verá ruido cifrado. Una VPN es especialmente útil en redes públicas, cuando viajas al extranjero y cuando quieres privacidad extra. Elige un proveedor de VPN de pago con buena reputación (las gratuitas suelen vender tus datos, que es justo lo que quieres evitar), actívala antes de conectarte a la red pública y mantenla encendida durante toda la sesión.

En casa, tu red WiFi también necesita atención: cambia la contraseña de fábrica del módem (esas que vienen impresas en la etiqueta son públicas y conocidas), usa cifrado WPA3 o al menos WPA2, desactiva la administración remota del módem y crea una red de invitados separada para visitas y dispositivos inteligentes. Así, si un dispositivo barato es comprometido, no tendrá acceso a tu computadora principal.

Copias de seguridad y plan de recuperación

Hay dos tipos de personas: las que ya perdieron información importante y las que la van a perder. Un disco duro se puede dañar, un teléfono se puede robar, un ransomware puede cifrarlo todo. La única defensa real es la copia de seguridad (backup): una copia de tus archivos importantes guardada en otro lugar.

La regla de oro es la 3-2-1: 3 copias de tus datos (la original más dos respaldos), en 2 tipos de medio distintos (por ejemplo, un disco externo y la nube), con 1 copia fuera de tu casa (la nube o el disco en casa de un familiar). Así sobrevives al robo, al incendio y al ransomware al mismo tiempo.

Automatiza tus respaldos porque lo que depende de tu memoria no se hace. En Windows usa el Historial de archivos o una herramienta de respaldo programado; en Mac activa Time Machine con un disco externo; en el teléfono activa el respaldo automático a la nube (Google Fotos, iCloud) y verifica que tus contactos y chats importantes estén incluidos. Para documentos de trabajo, servicios como Google Drive, OneDrive o Dropbox con sincronización automática son suficientes para la mayoría.

Pero un respaldo que nunca probaste es solo una esperanza. Una vez al trimestre, verifica que tus copias existan y que puedas abrir los archivos: restaura un documento de prueba. Además, ten un plan de recuperación escrito: una lista simple de qué hacer si pierdes tu equipo (paso 1: cambiar contraseñas desde otro dispositivo, paso 2: activar el borrado remoto, paso 3: restaurar desde el respaldo, paso 4: avisar a tus contactos si sospechas suplantación). Tener el plan escrito antes de la crisis es lo que separa un susto de un desastre.

Error común: guardar el respaldo conectado permanentemente a la computadora. Si un ransomware cifra tu disco, también cifrará el disco externo conectado. Desconecta el disco después del respaldo o usa respaldos en la nube con historial de versiones.

Seguridad en dispositivos IoT y hogar inteligente

Tu casa se está llenando de computadoras disfrazadas: focos, bocinas, cámaras, televisiones, cerraduras, refrigeradores y hasta cafeteras con WiFi. Estos dispositivos del Internet de las Cosas (IoT) suelen tener seguridad muy débil: contraseñas de fábrica como "admin/admin", software que nunca se actualiza y fabricantes que desaparecen al año.

El riesgo es real: cámaras hackeadas que espían a familias, bocinas usadas para escuchar conversaciones y miles de dispositivos secuestrados para formar botnets (ejércitos de aparatos zombies que atacan a otros). En 2016, el ataque Mirai usó cámaras y módems inseguros de todo el mundo para tumbar servicios gigantes de internet. Tu dispositivo barato puede ser parte del problema sin que lo sepas.

Protégelos con estas medidas: cambia de inmediato todas las contraseñas de fábrica; conecta los dispositivos IoT a una red WiFi de invitados separada de tu red principal (así no pueden "ver" tu computadora ni tu teléfono); desactiva las funciones que no uses (si tu tele no necesita micrófono, apágalo); actualiza el firmware cuando el fabricante lo ofrezca; y antes de comprar, investiga si la marca tiene historial de actualizaciones de seguridad. Para cámaras dentro de casa, considera taparlas o desconectarlas cuando no las uses: la mejor cámara segura es la apagada.

Buena práctica extra: haz un inventario de tus dispositivos conectados. Anota cada aparato, su contraseña y la fecha de su última actualización. Parece exagerado hasta que tienes quince dispositivos y no recuerdas cuál sigue con la clave de fábrica.

Qué hacer si te hackean: protocolo de respuesta

Que te hackeen no es el fin del mundo si actúas rápido y en orden. El pánico lleva a malas decisiones, como pagar rescates o borrar evidencia. Guarda este protocolo y síguelo paso a paso si algún día lo necesitas.

Fase 1: contención inmediata (primeros minutos). Desconecta el equipo afectado de internet pero no lo apagues del todo si sospechas de un ataque en curso en una empresa (los especialistas necesitan la evidencia); en casa, desconectar el WiFi y los datos es suficiente. Desde OTRO dispositivo limpio, cambia de inmediato las contraseñas de tu correo principal y de tus cuentas bancarias. Si aún tienes acceso, activa el 2FA en todo lo que no lo tenga. Si

perdiste el acceso a una cuenta, usa el proceso oficial de recuperación del servicio (no enlaces de mensajes).

Fase 2: evaluación del daño (primeras horas). Revisa qué cuentas muestran actividad extraña: sesiones iniciadas en lugares que no reconoces, correos enviados que tú no escribiste, compras que no hiciste. Revisa tus estados de cuenta bancarios y de tarjeta. Haz una lista de qué información pudo quedar expuesta: ¿solo una contraseña vieja o también tu INE, tu dirección y tus fotos? La gravedad de tu respuesta depende de esta lista.

Fase 3: limpieza y recuperación (primeros días). En tu equipo, ejecuta un análisis completo con un antivirus reconocido y actualizado. Si el equipo está muy comprometido, la opción más segura es formatear e instalar el sistema desde cero y luego restaurar tus archivos desde el respaldo limpio (nunca restaures programas, solo documentos, fotos y archivos personales). Cambia TODAS tus contraseñas importantes usando tu gestor de contraseñas, una por una, empezando por el correo. Revisa y cierra las sesiones activas en tus redes sociales y servicios.

Fase 4: notificación y prevención (primera semana). Avisa a tus contactos si el atacante pudo suplantarte para pedirles dinero o enviarles malware. Si se expusieron datos bancarios, llama a tu banco para bloquear y reponer tarjetas. En México puedes levantar una denuncia ante la policía cibernética de tu estado o la Guardia Nacional; guarda capturas de pantalla y correos como evidencia. Finalmente, haz una autopsia honesta: ¿cómo entraron? ¿Un enlace, una contraseña repetida, una app pirata? Corrige esa puerta para que no vuelva a pasar.

Analogía cotidiana: un hackeo es como un robo en casa. Primero aseguras la puerta (contención), luego ves qué se llevaron (evaluación), después cambias las chapas (limpieza) y al final avisas a los vecinos y a la policía (notificación). Hacerlo en desorden, como cambiar las chapas antes de ver qué falta, solo complica las cosas.

Error común de principiantes: creer que formatear el teléfono borra el problema de raíz sin cambiar contraseñas. El atacante no vive en tu teléfono, vive en tus cuentas en la nube. Si no cambias las contraseñas, entrará de nuevo al minuto siguiente.

Privacidad: huellas digitales y permisos de apps

Cada vez que usas internet dejas huellas digitales: rastros de quién eres, dónde estás y qué te interesa. Tu navegador revela tu sistema operativo, tu idioma, tu zona horaria y las fuentes instaladas; combinado, eso forma una "huella" casi única que te identifica aunque borres las cookies. Los anunciantes y las apps gratuitas viven de recolectar estos datos: si el servicio es gratis, normalmente tú eres el producto.

No puedes desaparecer por completo, pero sí puedes reducir tu exposición. Usa un navegador con protección contra rastreo o instala extensiones que bloqueen rastreadores. Revisa la configuración de privacidad de tus redes sociales: pon tu perfil en privado, limita quién ve tus publicaciones y desactiva la ubicación en fotos. En Google y otras plataformas, revisa el panel de actividad y borra el historial que no quieras conservar. Usa un buscador que no te rastree para consultas sensibles.

Los permisos de las apps merecen una auditoría trimestral. Entra a los ajustes de tu teléfono y revisa qué apps tienen acceso a tu ubicación, micrófono, cámara, contactos y archivos. Pregúntate: ¿una app de linterna necesita mi ubicación? ¿Un juego necesita mis contactos? Revoca todo lo que no tenga justificación. Desinstala las apps que no uses: cada app abandonada es una puerta olvidada. Y lee antes de aceptar: si una app gratuita te pide acceso total a tus fotos y contactos a cambio de un filtro bonito, el precio real eres tú.

Buena práctica: separa tus identidades. Usa un correo para cosas importantes (banco, trabajo) y otro para registros en sitios dudosos o promociones. Así, cuando ese segundo correo se llene de spam o se filtre en un hackeo, tu identidad principal sigue limpia. Es como tener un correo de "batalla" para el ruido de internet.

Carrera en ciberseguridad: un campo con hambre de talento

La ciberseguridad no solo te protege: puede ser tu profesión. Es uno de los campos con más demanda y menos profesionales disponibles en el mundo. Empresas, bancos, gobiernos y hospitales buscan gente que sepa defender sistemas, y los sueldos suelen estar por encima del promedio tecnológico. Lo mejor: puedes empezar desde cero, sin título universitario, si demuestras habilidades reales.

El mapa de la carrera tiene varias rutas. El analista de seguridad monitorea sistemas y responde a incidentes (es la puerta de entrada más común). El pentester o hacker ético ataca sistemas con permiso para encontrar fallas antes que los criminales. El especialista en respuesta a incidentes es el "bombero" que actúa cuando hay un ataque real. El ingeniero de seguridad diseña sistemas protegidos desde el inicio. Y el consultor ayuda a empresas a cumplir normas y auditorías. Todas las rutas comparten una base: redes, sistemas operativos (sobre todo Linux), programación básica y mentalidad de atacante.

Para empezar hoy mismo: 1) aprende Linux y redes (hay cursos gratuitos en lucasegaacademy.com); 2) practica en plataformas legales como TryHackMe, HackTheBox o picoCTF, que son como videojuegos para aprender hacking ético; 3) obtén certificaciones reconocidas cuando estés listo, como CompTIA Security+ para empezar y luego OSCP o CISSP para niveles avanzados; 4) documenta lo que aprendes en un blog o en GitHub, porque

un portafolio visible vale más que un currículum; 5) únete a comunidades: hay grupos, conferencias y competencias de CTF (Capture The Flag) en español donde conocerás gente del medio.

La regla de oro ética: el conocimiento de esta unidad es para defender, nunca para atacar sin permiso. El hacking sin autorización es delito en México y en casi todo el mundo, con penas de cárcel. Los hackers éticos ganan bien precisamente porque usan sus habilidades con permiso y para proteger. Esa línea no se cruza jamás.

Compras en línea y banca digital segura

Comprar por internet y manejar tu dinero desde el teléfono es comodísimo, y por eso mismo es un blanco favorito. Las reglas para hacerlo seguro son pocas pero firmes.

Primero, verifica el candado y el dominio: antes de escribir tu tarjeta, confirma que la dirección empiece con `https://` y que el dominio sea exactamente el de la tienda (cuidado con "amaz0n-ofertas.com" o "mercadolibre-descuentos.net"). Los sitios falsos copian el diseño al 100 por ciento; el dominio es lo que no pueden falsificar bien. Segundo, desconfía de las ofertas imposibles: un iPhone a mitad de precio en un sitio que conociste hoy por un anuncio es una estafa casi segura. Tercero, paga con tarjeta de crédito o con métodos que ofrezcan protección al comprador (como PayPal), nunca con transferencia directa o depósito a cuentas de desconocidos: la transferencia no tiene reversa.

En la banca digital, descarga la app solo desde la tienda oficial (Play Store o App Store), nunca desde enlaces de mensajes. Activa las notificaciones de movimientos para detectar cargos raros al instante y revisa tu estado de cuenta cada semana, aunque sea por encima. Nunca entres a tu banco desde una computadora pública o prestada, y si lo haces por emergencia, cambia tu contraseña después desde tu propio equipo. Y jamás, bajo ninguna circunstancia, un "ejecutivo del banco" te va a llamar para pedirte tu NIP, tu contraseña o los códigos que llegan a tu teléfono: esa llamada es fraude el 100 por ciento de las veces. El banco ya tiene tus datos; quien te los pide por teléfono es un delincuente.

Buena práctica avanzada: usa una tarjeta digital o virtual con límite bajo para compras en línea. Muchos bancos permiten generar tarjetas desechables desde la app: si los datos se filtran, la tarjeta ya no sirve para nada más. Es como pagar con un billete que se autodestruye después de usarlo.

Actualizaciones, antivirus y el sistema inmune de tus equipos

Actualizar tu sistema y tus apps no es un capricho de los fabricantes: es como vacunarte. Cada actualización de seguridad tapa agujeros que los atacantes ya están explotando. Los ataques masivos más dañinos de la historia (como WannaCry, que paralizó hospitales y empresas en 2017) usaron vulnerabilidades para las que ya existía parche: las víctimas eran equipos que nadie había actualizado.

Activa las actualizaciones automáticas en tu teléfono, tu computadora y tus apps, y no las pospongas semanas. Cuando el sistema te pida reiniciar para instalar una actualización de seguridad, hazlo ese mismo día. Lo mismo vale para el módem de tu casa y tus dispositivos inteligentes: revisa cada pocos meses si hay firmware nuevo.

Sobre el antivirus: los sistemas modernos ya traen defensas integradas (Windows Defender, las protecciones de macOS y Android), y para la mayoría de las personas son suficientes si se combinan con buenos hábitos. Si instalas un antivirus de terceros, que sea de una marca reconocida y descargado de su sitio oficial; los "antivirus gratis" de anuncios suelen ser malware disfrazado. Ningún antivirus te protege si tú mismo instalas programas piratas o das clic a todo: el antivirus es el cinturón de seguridad, no el conductor.

Error común de principiantes: creer que "a la Mac no le entran virus" o que "con antivirus ya estoy blindado". Todos los sistemas tienen vulnerabilidades y ningún programa de seguridad sustituye el criterio. La combinación ganadora es: sistema actualizado más defensas activas más hábitos sanos. Ninguno de los tres funciona solo.

Redes sociales: suplantación y exceso de información

Tus redes sociales son la mina de oro de la ingeniería social. Cada foto con tu uniforme del trabajo, cada check-in en tiempo real, cada cumpleaños con tu fecha completa y cada nombre de tu mascota es una pieza que un atacante puede usar para adivinar tus contraseñas, responder tus preguntas de seguridad o inventar una historia creíble para engañarte a ti o a tu familia ("soy amigo de tu hijo, está en problemas").

Haz una limpieza de oversharing: no publiques tu ubicación en tiempo real (sube las fotos del viaje cuando ya volviste), no muestres documentos, boletos de avión ni credenciales en fotos, y evita publicar fechas de nacimiento completas, direcciones y rutinas ("todos los martes voy al gym a las 7"). Configura tus perfiles para que solo tus contactos vean tus publicaciones y revisa quiénes son tus contactos: aceptaste a mucha gente que no conoces.

La suplantación es otro riesgo cotidiano: crean un perfil con tu nombre y tus fotos y les escriben a tus contactos pidiendo dinero prestado "por una emergencia". Para prevenirlo,

marca tu perfil real como privado o verificado donde se pueda, y avisa a tu familia de un código secreto: si algún día les escriben pidiendo dinero urgente haciéndose pasar por ti, que te pidan esa palabra clave por llamada. Si sospechas que te suplantarón, reporta el perfil falso en la plataforma y avisa públicamente a tus contactos desde tu cuenta real.

Buena práctica: una vez al año, búscate a ti mismo en Google con tu nombre completo. Lo que encuentres es lo que un atacante también puede encontrar. Si aparece información sensible (tu dirección, tu teléfono), solicita su eliminación en esos sitios. Tu privacidad también se defiende en ataque, no solo en defensa.

El factor humano: seguridad en familia y en el trabajo

La tecnología ya la cubrimos; ahora hablemos de las personas, que es donde la mayoría de los ataques realmente triunfan. De nada sirve tu fortaleza digital si tu hijo instala un juego pirata lleno de malware en la computadora familiar o si tu mamá le dicta por teléfono el código de verificación a un "ejecutivo del banco".

Empieza por tu familia con reglas simples y sin tecnicismos. Para los niños y adolescentes: explícales que no todo lo que brilla en internet es real, que nunca compartan fotos con uniforme escolar ni su ubicación, que no acepten solicitudes de desconocidos y que te avisen si alguien les pide fotos o secretos (esto también es protección contra el grooming, el acoso de adultos a menores). Instala controles parentales no como espionaje, sino como entrenamiento con red de seguridad: la meta es que aprendan criterio, no que vivan vigilados para siempre.

Para los adultos mayores de tu familia, que son el blanco favorito de las estafas telefónicas: enséñales la regla de oro con una frase fácil de recordar: "el banco nunca pide códigos por teléfono". Haz con ellos un simulacro: llámales un día haciéndote pasar por un "ejecutivo" y ve si caen; si caen, no los regañes, felicítalos por aprender en un simulacro y no con dinero real. Configúrales el teléfono con las apps que necesitan, activa las actualizaciones automáticas y déjales anotado en papel el número oficial del banco. La paciencia aquí vale más que cualquier software.

En el trabajo, la seguridad es responsabilidad compartida. Nunca conectes memorias USB encontradas o de origen dudoso a equipos de la empresa; no instales software sin autorización del área de sistemas; bloquea tu computadora cada vez que te levantes (tecla Windows + L en Windows, Control + Comando + Q en Mac); y reporta de inmediato cualquier correo sospechoso en lugar de borrarlo en silencio, porque si te llegó a ti, le llegó a tus compañeros. Si manejas datos de clientes, trátalos como si fueran los tuyos: la filtración de datos de una empresa destruye confianzas y genera multas.

Y un principio para todos los ámbitos: crea una cultura donde preguntar no dé vergüenza. La mayoría de las víctimas no reportan el fraude por pena, y ese silencio es el mejor aliado de los delincuentes. Si alguien de tu familia o tu equipo cae en un engaño, la respuesta correcta es "qué bueno que me lo dijiste, vamos a arreglarlo juntos", nunca el regaño. Un equipo que avisa a tiempo contiene el daño; un equipo que oculta por miedo lo multiplica.

Buena práctica final: designa en tu familia a una "persona de seguridad", alguien con un poco más de conocimiento que ayude a los demás con configuraciones y dudas. Puede ser tú después de terminar esta unidad. La seguridad, como la salud, funciona mejor en comunidad.

Navegación segura: tu día a día en internet

Pasas horas al día en el navegador, así que vale la pena blindar ese hábito. La navegación segura se resume en cuatro frentes: a dónde entras, qué descargas, qué instalas en el navegador y qué haces en sitios desconocidos.

A dónde entras: fíjate siempre en la barra de direcciones antes de escribir datos sensibles. Busca el candado y el `https://`, pero entiende sus límites: el candado solo significa que la conexión está cifrada, no que el sitio sea honesto. Los sitios de phishing también usan `https` hoy en día. Lo que realmente importa es el dominio exacto: "banco-seguro.com" no es tu banco aunque tenga candado. Escribe tú mismo las direcciones de tus sitios importantes o usa tus marcadores guardados; no llegues a tu banco desde un enlace de un correo o un resultado de búsqueda patrocinado.

Qué descargas: descarga software solo desde el sitio oficial del fabricante o las tiendas oficiales. Los programas "gratis" de sitios de descargas dudosas son la vía clásica de infección: el instalador trae de regalo un troyano o un minero de criptomonedas que alenta tu equipo. Lo mismo con películas, música y libros piratas: el archivo "gratis" suele costar tus contraseñas. Si algo es de paga y te lo ofrecen gratis en un sitio raro, el producto eres tú.

Qué instalas en el navegador: las extensiones son programas con acceso a todo lo que ves. Instala solo las necesarias, de desarrolladores conocidos y desde la tienda oficial del navegador. Revisa cada seis meses qué extensiones tienes y elimina las que no uses. Una extensión abandonada y vendida a terceros puede convertirse en spyware de la noche a la mañana sin que cambie su nombre.

Qué haces en sitios desconocidos: cuando un sitio te pide crear cuenta para algo trivial, pregúntate si vale la pena entregar tu correo. Muchas veces puedes usar tu correo de "batalla" o un servicio de correo temporal. Nunca ejecutes en tu computadora comandos o scripts que te dicte un desconocido por teléfono o chat ("abre la terminal y escribe esto para arreglar tu computadora"): esa es la técnica favorita del fraude de soporte técnico falso. Y si un sitio te

pide desactivar tu antivirus o tu bloqueador para "ver el contenido", ciérralo: ningún contenido legítimo necesita que bajes tus defensas.

Analogía cotidiana: navegar es como caminar por una ciudad grande. Hay avenidas principales seguras (sitios oficiales), callejones oscuros (descargas dudosas) y vendedores que te jalen del brazo (ventanas emergentes). No dejas de salir a la ciudad por eso: aprendes por dónde caminar, qué ignorar y a quién no seguir. Con estos cuatro frentes cubiertos, tu día a día en internet queda blindado sin que tengas que pensar en ello a cada clic.

Estafas modernas: deepfakes, voz clonada y códigos QR

La inteligencia artificial le dio superpoderes a los estafadores, y necesitas conocer sus trucos nuevos. El más alarmante es la clonación de voz: con unos segundos de audio de tu voz (tomados de un video de redes sociales o un mensaje de voz), un programa puede generar audios falsos que suenan idénticos a ti. Ya hay casos de "secuestros virtuales" donde los papás reciben la llamada llorosa de su "hijo" pidiendo un rescate, cuando en realidad el hijo está bien en la escuela. La defensa es el código secreto familiar que viste antes: ante cualquier llamada de emergencia pidiendo dinero, cuelga y marca tú mismo el número de tu familiar, y pide la palabra clave.

Los deepfakes en video van por el mismo camino: caras y voces falsas en videollamadas para suplantar jefes o familiares. Si en una videollamada algo se ve raro (parpadeos extraños, boca que no sincroniza, calidad que cambia), desconfía y verifica por otro canal. Otro truco en auge es el quishing: phishing con códigos QR. Te llega un QR por correo o lo ves pegado en un parquímetro o restaurante, lo escaneas y te lleva a un sitio falso que roba tus datos o instala malware. Regla simple: no escanees QR de correos no solicitados ni de pegatinas en la calle; si un restaurante te da un QR en su mesa, bien, pero si el QR llegó por mensaje de un desconocido, no.

También crecieron las estafas de empleo falso y de inversión: te contactan por WhatsApp o redes con una "oportunidad de trabajo desde casa" donde primero debes pagar una "cuota de registro" o "comprar un kit", o te invitan a invertir en criptomonedas con ganancias garantizadas del 20 por ciento mensual. Ningún empleo legítimo te cobra por contratarte y ninguna inversión sería garantiza ganancias. Si te piden dinero para darte trabajo, es estafa; si te prometen rendimientos imposibles, es estafa.

El patrón común de todas estas estafas modernas es el mismo de siempre: urgencia, secreto ("no le digas a nadie") y un pago o dato que debes entregar ya. La tecnología del engaño cambia; la psicología del engaño no. Tu defensa tampoco cambia: frena, verifica por otro canal y consulta con alguien de confianza antes de mover dinero. Ante la duda, la respuesta es no.

Ejemplos paso a paso

La teoría sin práctica se olvida en una semana. Los tres ejemplos siguientes son herramientas reales que puedes construir y usar hoy mismo: un auditor de contraseñas en Python, el endurecimiento de un servidor Linux y un sistema de respaldos automáticos. Léelos con la terminal a la mano.

Ejemplo 1: auditar tus contraseñas con un script en Python

Vamos a construir una herramienta real: un script que revisa una lista de tus contraseñas (en un archivo de prueba, nunca tus reales en texto plano junto al script) y te dice cuáles son débiles, cuáles se repiten y cuáles aparecen en listas de contraseñas filtradas comunes. Es el mismo tipo de análisis que hacen los auditores de seguridad.

Primero crea un archivo de prueba con contraseñas de ejemplo. En la vida real, este análisis lo haría tu gestor de contraseñas automáticamente, pero hacerlo a mano te enseña exactamente qué hace débil a una contraseña.

```
# auditor_contraseñas.py
# Revisa la fortaleza de contraseñas de un archivo de prueba.
# ADVERTENCIA: usa solo contraseñas de ejemplo, nunca tus reales.

import re

CONTRASENAS_COMUNES = {
    "123456", "password", "12345678", "qwerty", "abc123",
    "111111", "123123", "admin", "welcome", "password1",
}

def evaluar(contraseña):
    """Devuelve puntaje de 0 a 100 y lista de problemas."""
    problemas = []
    puntaje = 0
    largo = len(contraseña)

    # Longitud: el factor mas importante
    if largo >= 16:
        puntaje += 40
    elif largo >= 12:
        puntaje += 30
    elif largo >= 8:
        puntaje += 15
        problemas.append("muy corta: usa minimo 12 caracteres")
    else:
```

```

        problemas.append("peligrosamente corta")

# Variedad de caracteres
tipos = 0
if re.search(r"[a-z]", contrasena):
    tipos += 1
else:
    problemas.append("sin minusculas")
if re.search(r"[A-Z]", contrasena):
    tipos += 1
else:
    problemas.append("sin mayusculas")
if re.search(r"[0-9]", contrasena):
    tipos += 1
else:
    problemas.append("sin numeros")
if re.search(r"^[a-zA-Z0-9]", contrasena):
    tipos += 1
else:
    problemas.append("sin simbolos")
puntaje += tipos * 10

# Patrones peligrosos
if contrasena.lower() in CONTRASENAS_COMUNES:
    problemas.append("aparece en listas de filtradas: cambiala YA")
    puntaje = min(puntaje, 10)
if re.search(r"(\1){1,10}", contrasena):
    problemas.append("tiene caracteres repetidos seguidos")
    puntaje -= 5
if re.search(r"123|abc|qwe", contrasena.lower()):
    problemas.append("contiene secuencias predecibles")
    puntaje -= 5

return max(puntaje, 0), problemas

def auditar(ruta):
    vistas = {}
    with open(ruta, encoding="utf-8") as f:
        lineas = [l.strip() for l in f if l.strip()]
    for c in lineas:
        vistas[c] = vistas.get(c, 0) + 1

    print(f"Analizando {len(lineas)} contraseñas...\n")
    for c in lineas:

```

```

    puntaje, problemas = evaluar(c)
    estado = "FUERTE" if puntaje >= 80 else "MEDIA" if puntaje >= 50 else "DEBIL"
    print(f"[{estado}] puntaje {puntaje}/100")
    if vistas[c] > 1:
        print(f" ! REPETIDA {vistas[c]} veces: reutilizar es grave")
    for p in problemas:
        print(f" - {p}")
    print()

if __name__ == "__main__":
    auditar("contrasenas_prueba.txt")

```

Crea el archivo de prueba con este contenido:

```

contrasenas_prueba.txt:
-----
123456
Password1
taco bicicleta luna 42 !
qwerty
MiBanco2020
taco bicicleta luna 42 !
-----

```

Ejecútalo con ``python3 auditor_contrasenas.py`` y observa el reporte: marca las débiles, detecta la repetida y premia la frase larga. Este ejercicio te enseña el criterio que usan los profesionales: longitud primero, unicidad siempre, y listas de filtradas como referencia. En la práctica diaria, tu gestor de contraseñas hará esto por ti, pero ahora entiendes qué está evaluando por dentro.

Para cerrar el ejemplo, agreguemos un generador de contraseñas seguras al mismo script. Así tendrás la pareja completa: auditor y generador, como las herramientas reales.

```
# Agrega esto al final de auditor_contrasenas.py
import secrets
import string

def generar(largo=20):
    """Genera una contraseña aleatoria criptograficamente segura."""
    alfabeto = string.ascii_letters + string.digits + string.punctuation
    while True:
        pwd = "".join(secrets.choice(alfabeto) for _ in range(largo))
        # Exige al menos un caracter de cada tipo
        if (any(c.islower() for c in pwd)
            and any(c.isupper() for c in pwd)
            and any(c.isdigit() for c in pwd)
            and any(c in string.punctuation for c in pwd)):
            return pwd

if __name__ == "__main__":
    # Descomenta para probar el generador:
    # print("Nueva contraseña segura:", generar(20))
    auditar("contrasenas_prueba.txt")
```

Nota por qué usamos `secrets` y no `random`: el módulo `random` de Python es predecible y sirve para juegos, pero no para seguridad; `secrets` usa el generador aleatorio del sistema operativo, diseñado para criptografía. Es el mismo tipo de detalle que separa un script de juguete de una herramienta seria. Con el auditor y el generador ya tienes el flujo profesional completo: detectar las débiles, reemplazarlas por aleatorias y guardarlas en tu gestor.

Ejemplo 2: endurecer un servidor Linux con comandos

Endurecer (hardening) un servidor significa cerrarle todas las puertas que no necesita. Este ejemplo muestra la secuencia básica que un administrador ejecuta en un servidor Ubuntu o Debian recién instalado. Léelo como receta: cada bloque tiene su explicación. ADVERTENCIA: practica primero en una máquina virtual o un servidor de prueba, nunca directo en producción sin respaldo.

```
# 1. Actualizar el sistema: la mayoría de ataques usan
#    vulnerabilidades YA corregidas por los fabricantes.
sudo apt update && sudo apt upgrade -y

# 2. Crear un usuario administrador y desactivar el login
#    directo como root (el root es el objetivo favorito).
sudo adduser operador
sudo usermod -aG sudo operador
```

```

# 3. Configurar llaves SSH y desactivar el login con contraseña.
#   (Genera tu llave en tu computadora con: ssh-keygen -t ed25519)
mkdir -p ~/.ssh && chmod 700 ~/.ssh
# Copia tu llave publica al archivo authorized_keys:
# cat ~/.ssh/id_ed25519.pub >> ~/.ssh/authorized_keys
chmod 600 ~/.ssh/authorized_keys

# 4. Endurecer la configuracion de SSH editando
#   /etc/ssh/sshd_config (con sudo nano o tu editor):
#   PermitRootLogin no
#   PasswordAuthentication no
#   PermitEmptyPasswords no
#   X11Forwarding no
#   MaxAuthTries 3
#   Luego reinicia el servicio:
sudo systemctl restart ssh

# 5. Activar el firewall y abrir solo lo necesario.
#   Ejemplo: servidor web (puertos 80 y 443) + SSH (22).
sudo ufw default deny incoming
sudo ufw default allow outgoing
sudo ufw allow 22/tcp
sudo ufw allow 80/tcp
sudo ufw allow 443/tcp
sudo ufw enable
sudo ufw status verbose

# 6. Instalar fail2ban: bloquea IPs que intentan
#   adivinar contraseñas por fuerza bruta.
sudo apt install -y fail2ban
sudo systemctl enable --now fail2ban

# 7. Activar actualizaciones automaticas de seguridad.
sudo apt install -y unattended-upgrades
sudo dpkg-reconfigure -plow unattended-upgrades

# 8. Verificar que no haya servicios innecesarios escuchando.
sudo ss -tulpn
# Todo puerto abierto que no reconozcas es una puerta
# que hay que investigar y probablemente cerrar.

```

Qué aprendiste aquí: el principio de mínimo privilegio (cada usuario y servicio solo tiene los permisos que necesita), la defensa en capas (firewall + SSH endurecido + fail2ban + parches) y que la seguridad es un proceso, no un producto. Un servidor así configurado resiste la gran

mayoría de los ataques automatizados que barren internet buscando víctimas fáciles. Como buena práctica, repite el paso 8 cada mes: los servicios nuevos a veces abren puertos sin avisar.

Como complemento, aquí tienes un script de auditoría rápida que puedes correr cada semana para detectar configuraciones que se hayan debilitado con el tiempo. Guárdalo como `~/bin/auditoria.sh` y ejecútalo con sudo.`

```
#!/bin/bash
# Auditoria semanal de seguridad del servidor.
echo "=== AUDITORIA DE SEGURIDAD $(date) ==="

echo "[1] Usuarios con acceso sudo:"
getent group sudo

echo "[2] Cuentas con contraseña vacia (no debe haber ninguna):"
sudo awk -F: '($2 == "" ) {print $1}' /etc/shadow

echo "[3] Ultimos inicios de sesion:"
last -n 5

echo "[4] Intentos SSH bloqueados por fail2ban:"
sudo fail2ban-client status sshd 2>/dev/null | grep "Banned IP"

echo "[5] Puertos abiertos:"
sudo ss -tulpn | grep LISTEN

echo "[6] Actualizaciones de seguridad pendientes:"
apt list --upgradable 2>/dev/null | grep -i security | head -5

echo "=== FIN DE LA AUDITORIA ==="
```

Cada sección del reporte te dice algo concreto: usuarios con sudo de más, cuentas sin contraseña, inicios de sesión que no reconoces, cuántos atacantes bloqueó fail2ban esta semana y si hay parches pendientes. Si algo sale raro, investiga antes de seguir. Los profesionales no endurecen una vez: auditan siempre. Este script de 20 líneas es tu rondín de vigilancia semanal.

Ejemplo 3: configurar copias de seguridad automatizadas

La teoría del respaldo no sirve de nada si depende de tu memoria. Este ejemplo crea un sistema de respaldo automático en Linux usando rsync y cron: cada noche se copian tus documentos a un disco externo, y cada domingo se hace una copia completa. Adáptalo a tus rutas reales.

```

# 1. Identifica tu disco de respaldo (ejemplo: /media/respaldo).
#   Verifica que este montado:
ls /media/respaldo

# 2. Crea el script de respaldo en ~/bin/respaldo.sh:
mkdir -p ~/bin
cat > ~/bin/respaldo.sh << 'EOF'
#!/bin/bash
# Respaldo automatico de documentos personales.
# Guarda bitacora de cada ejecucion.

ORIGEN="$HOME/Documentos"
DESTINO="/media/respaldo/documentos"
BITACORA="$HOME/respaldo.log"
FECHA=$(date "+%Y-%m-%d %H:%M")

# -a conserva permisos y fechas, -v muestra progreso,
# --delete borra del respaldo lo que borraste del origen
# (cuidado: solo usalo si el respaldo anterior esta verificado).
echo "[FECHA] Iniciando respaldo..." >> "$BITACORA"
rsync -av --delete "$ORIGEN/" "$DESTINO/" >> "$BITACORA" 2>&1
echo "[FECHA] Respaldo terminado." >> "$BITACORA"
EOF
chmod +x ~/bin/respaldo.sh

# 3. Pruebalo a mano una vez antes de automatizarlo:
~/bin/respaldo.sh
cat ~/respaldo.log

# 4. Automatizalo con cron para que corra cada noche a las 2 am:
crontab -e
# Agrega esta linea al final del archivo cron:
# 0 2 * * * /home/tu_usuario/bin/respaldo.sh

# 5. Verifica que quedo programado:
crontab -l

# 6. Prueba de restauracion (una vez al trimestre):
#   Copia un archivo del respaldo a una carpeta temporal
#   y confirma que se abre bien:
mkdir -p /tmp/prueba_restaura
cp /media/respaldo/documentos/archivo_importante.pdf /tmp/prueba_restaura/

```

Para Windows, el equivalente es activar el Historial de archivos (Configuración > Actualización y seguridad > Copia de seguridad) con un disco externo conectado; para Mac, Time Machine hace lo mismo con dos clics. En el teléfono, activa el respaldo automático a la nube hoy mismo y verifica que incluya fotos, contactos y chats.

La lección clave: un respaldo automático y verificado convierte el ransomware, el robo y el disco dañado en simples inconvenientes de una tarde. Sin respaldo, el mismo evento es una catástrofe. La diferencia entre ambos escenarios es este script y quince minutos de configuración.

Para verificar que cumples la regla 3-2-1, usa esta lista de comprobación cada trimestre. Puedes convertirla en un pequeño script que te recuerde qué revisar:

```
#!/bin/bash
# Lista de verificacion trimestral de respaldos (regla 3-2-1).
echo "=== VERIFICACION DE RESPALDOS ==="
echo "[ ] 1. El respaldo automatico corrio esta semana (revisa ~/respaldo.log)"
echo "[ ] 2. Hay copia en disco externo (conecta y verifica la fecha)"
echo "[ ] 3. Hay copia en la nube o fuera de casa (abre el servicio y confirma)"
echo "[ ] 4. Prueba de restauracion: recupera 1 archivo y abrelo"
echo "[ ] 5. El disco externo NO quedo conectado permanentemente"
echo "[ ] 6. Las fotos del telefono estan en el respaldo de nube"
echo "Marca cada punto a mano. Si alguno falla, corrígelo hoy mismo."
```

Esta lista parece simple, y lo es: la seguridad efectiva casi siempre es simple y constante, no compleja y ocasional. El trimestre que te saltes la verificación será justo el trimestre que necesites el respaldo. Márcalo en tu calendario como una cita contigo mismo: "revisión de respaldos", 20 minutos, cuatro veces al año.

Práctica

Ejercicio 1: auditoría de tus contraseñas actuales

Sin escribir tus contraseñas reales en ningún lado, haz una auditoría honesta: abre tu navegador o tu gestor y cuenta cuántas cuentas importantes tienes (correo, banco, redes sociales, trabajo). Luego responde: ¿cuántas comparten la misma contraseña o una variante? ¿Cuántas tienen menos de 12 caracteres? ¿Cuántas usan datos personales?

Solución paso a paso: 1) Haz la lista de tus 10 cuentas más importantes en papel, sin anotar contraseñas, solo los nombres de los servicios. 2) Marca con una palomita las que sabes que

tienen contraseña única y larga; marca con una X las que repiten contraseña o son cortas. 3) Instala un gestor de contraseñas (Bitwarden es gratuito y de código abierto) y crea tu contraseña maestra con el método de frase de 5 palabras. 4) Empezando por tu correo principal, genera una contraseña aleatoria de 20 caracteres con el gestor y guárdala ahí. 5) Repite con tu banco y tus redes sociales, una por día si son muchas. 6) En una semana, todas tus X deberán ser palomitas. La meta no es la perfección inmediata, es no tener ninguna cuenta importante con contraseña repetida.

Ejercicio 2: activa el 2FA en tus tres cuentas más importantes

Elige tu correo principal, tu cuenta bancaria y tu red social más usada. Activa la autenticación de dos factores en las tres usando una app autenticadora (no SMS si puedes evitarlo).

Solución paso a paso: 1) Instala Authy o Google Authenticator en tu teléfono. 2) En tu correo (Gmail, Outlook, etc.), entra a Configuración > Seguridad > Verificación en dos pasos y sigue el asistente escaneando el código QR con la app. 3) Guarda los códigos de recuperación que te entrega el servicio: imprímelos o anótalos en papel y guárdalos donde guardas documentos importantes. 4) Repite el proceso en tu banco (casi todos lo ofrecen en su app o sitio web) y en tu red social. 5) Haz una prueba: cierra sesión y vuelve a entrar para confirmar que te pide el código y que funciona. 6) Si cambias de teléfono algún día, transfiere la app autenticadora antes de borrar el viejo o usa los códigos de recuperación. Con esto, aunque alguien robe tu contraseña, no podrá entrar sin tu teléfono.

Ejercicio 3: detecta el phishing en 5 mensajes de ejemplo

Lee estos 5 mensajes y decide cuáles son legítimos y cuáles son phishing, aplicando el protocolo de 5 pasos de esta unidad:

A) SMS: "PAQUET33: Tu paquete no pudo entregarse. Reagenda aquí: bit.ly/3xEntrega ya o será devuelto." B) Correo de tu banco (dirección real avisos@tubanco.com): "Estimado Carlos Méndez: tu estado de cuenta de septiembre ya está disponible en la app." C) Correo de "soporte@tubanco-seguridad.com": "Estimado cliente: detectamos un acceso extraño. Verifica tu identidad aquí o tu cuenta será bloqueada en 24 horas." D) WhatsApp de número desconocido: "Hola, soy del soporte de WhatsApp, tu cuenta será suspendida. Mándame el código que te llegó por SMS para verificarla." E) Correo de tu jefe (dirección real de la empresa): "¿Puedes pasarme el reporte de ventas de ayer cuando tengas chance? Gracias."

Solución paso a paso: A) PHISHING: urgencia ("ya o será devuelto"), enlace acortado que oculta el destino real, remitente genérico. B) LEGÍTIMO: dominio oficial, te llama por tu nombre, no pide datos ni incluye enlace sospechoso, tono informativo. C) PHISHING: dominio

tramposo (tubanco-seguridad.com no es el banco), saludo genérico, amenaza de bloqueo en 24 horas, pide "verificar identidad" en un enlace. D) PHISHING: WhatsApp jamás pide códigos por mensaje; el código SMS es justamente el segundo factor que el atacante quiere robarte. Nunca compartas códigos. E) LEGÍTIMO (con matiz): dirección real y petición normal; aun así, si pidiera una transferencia urgente o datos sensibles, lo verificarías por otro canal. Moraleja: 3 de 5 eran trampas, y las trampas modernas se ven profesionales. El protocolo de verificación por otro canal es tu mejor defensa.

Ejercicio 4: endurece tu red WiFi casera en 20 minutos

Vas a aplicar lo aprendido sobre redes a tu propio hogar. Necesitas acceso a la configuración de tu módem (normalmente abriendo 192.168.1.1 o 192.168.0.1 en el navegador; el usuario y clave iniciales suelen estar en la etiqueta del aparato).

Solución paso a paso: 1) Entra a la configuración del módem y cambia de inmediato la contraseña de administrador por una frase larga que solo tú conozcas; anótala en papel. 2) Cambia el nombre de tu red WiFi (SSID) por algo que no revele tu identidad ni tu dirección, y ponle una contraseña WPA2 o WPA3 de al menos 16 caracteres. 3) Revisa la lista de dispositivos conectados: si ves alguno que no reconoces, cámbiale la contraseña a la red y vuelve a conectar solo tus aparatos. 4) Activa la red de invitados con otra contraseña y conecta ahí tus focos, bocinas y cámaras inteligentes, dejando tu red principal solo para computadoras y teléfonos. 5) Desactiva la administración remota (acceso al módem desde internet) y el WPS, que es un método de conexión con un PIN vulnerable. 6) Anota en una hoja la fecha de hoy y prográmate revisar esta configuración cada 6 meses. Con estos 6 pasos tu red casera queda mejor protegida que la mayoría de las redes domésticas del país.

Ejercicio 5: simula un plan de respuesta ante hackeo

Imagina este escenario: recibes un aviso de que alguien inició sesión en tu correo desde otro país y ya no puedes entrar porque cambiaron la contraseña. Aplica el protocolo de respuesta de esta unidad y escribe tu plan de acción ordenado por fases, con tiempos estimados.

Solución paso a paso: Fase 1, contención (0 a 30 minutos): desde el teléfono de un familiar o tu computadora del trabajo, entra al proceso oficial de recuperación de tu correo (no a ningún enlace del aviso). Si tienes 2FA activado, úsalo para recuperar el acceso; si no, usa tu correo o teléfono de recuperación. En cuanto recuperes el acceso, cambia la contraseña por una nueva generada con tu gestor y revisa las reglas de reenvío del correo (los atacantes suelen crear reglas para espiarte en silencio). Fase 2, evaluación (30 minutos a 2 horas): revisa la actividad reciente de la cuenta, los correos enviados y las sesiones activas en tus redes sociales y

banco; cierra todas las sesiones desconocidas. Haz una lista de qué vio el atacante: ¿leyó correos del banco? ¿descargó documentos? Fase 3, limpieza (mismo día): ejecuta un antivirus completo en tus equipos por si el acceso vino de un malware; cambia las contraseñas de las cuentas vinculadas a ese correo, empezando por el banco. Fase 4, notificación y prevención (primera semana): avisa a tus contactos cercanos que ignoraron mensajes extraños "tuyos"; si hubo datos bancarios expuestos, llama a tu banco; activa el 2FA en todo lo que te faltaba y guarda los códigos de recuperación. La autopsia final: lo más probable es que entraran por phishing o por una contraseña repetida filtrada; corrige exactamente esa puerta. Si escribiste este plan antes de necesitarlo, ya ganaste la mitad de la batalla.

Ejercicio 6: auditoría de privacidad de tu teléfono

Vas a reducir tu huella digital con una auditoría completa de tu teléfono. Reserva 30 minutos sin interrupciones.

Solución paso a paso: 1) Abre Ajustes > Privacidad (o Ubicación, según tu sistema) y revisa app por app qué tiene acceso a tu ubicación: deja "solo mientras se usa" únicamente en mapas y apps de transporte; quítaselo a todo lo demás. 2) Haz lo mismo con micrófono, cámara, contactos y archivos: si una app no necesita ese permiso para su función principal, revócalo. 3) Desinstala todas las apps que no hayas abierto en los últimos 3 meses; cada una es una puerta olvidada. 4) Entra a tus redes sociales y pon tu perfil en privado, limita quién puede etiquetarte y desactiva la ubicación en tus publicaciones pasadas y futuras. 5) En tu cuenta de Google (o equivalente), abre el panel de actividad, revisa qué se está guardando y activa el borrado automático cada 3 meses. 6) Crea un segundo correo para registros en promociones y sitios dudosos, y empieza a usarlo desde hoy para todo lo que no sea importante. Repite esta auditoría cada 3 meses como mantenimiento. La privacidad no se consigue una vez: se mantiene como se mantiene limpia una casa.

Resumen de la unidad

- Las amenazas modernas son una industria: el malware infecta equipos, el ransomware secuestra archivos pidiendo rescate (nunca pagues), el phishing suplanta identidades para robarte datos y la ingeniería social te manipula a ti directamente. La defensa empieza por conocer cómo operan.
- Las contraseñas fuertes son largas (12 caracteres mínimo, ideal 16 o más), únicas para cada servicio e impredecibles. Usa frases de varias palabras y delega la memoria a un gestor de contraseñas: tú recuerdas una contraseña maestra, él recuerda las demás.

- La autenticación de dos factores es tu segunda cerradura: actívala al menos en tu correo, tu banco y tus redes sociales. Prefiere apps autenticadoras sobre SMS y guarda los códigos de recuperación en papel.
- Para detectar correos y mensajes falsos aplica siempre el protocolo de 5 pasos: frena ante la urgencia, revisa el remitente real, inspecciona los enlaces sin hacer clic, analiza el contenido y verifica por otro canal. La verificación independiente desactiva casi todos los fraudes.
- En redes WiFi públicas usa tus datos móviles para operaciones sensibles, desactiva la conexión automática y usa una VPN de pago confiable. En casa, cambia las claves de fábrica del módem, usa WPA2 o WPA3 y separa tus dispositivos inteligentes en una red de invitados.
- Aplica la regla 3-2-1 para respaldos: 3 copias, en 2 medios distintos, con 1 fuera de casa. Automatiza el respaldo, verifica la restauración cada trimestre y ten un plan de recuperación escrito antes de necesitarlo.
- Protege tu hogar inteligente cambiando contraseñas de fábrica, aislando los dispositivos IoT en su propia red, desactivando funciones que no uses y manteniendo un inventario con fechas de actualización.
- Si te hackean, sigue el protocolo en orden: contención inmediata, evaluación del daño, limpieza y recuperación, notificación y prevención. Cambiar contraseñas va antes que formatear, porque el atacante vive en tus cuentas, no solo en tu equipo.
- Reduce tu huella digital: bloquea rastreadores, ajusta la privacidad de tus redes, audita los permisos de tus apps cada 3 meses, desinstala lo que no uses y separa un correo de "batalla" para registros dudosos.
- La ciberseguridad también es una carrera con alta demanda: empieza con Linux y redes, practica en plataformas como TryHackMe o HackTheBox, certífyate (Security+, OSCP, CISSP) y construye un portafolio visible. Y recuerda la línea ética: este conocimiento es para defender, nunca para atacar sin permiso.
- En compras en línea y banca digital verifica el dominio exacto y el https, desconfía de ofertas imposibles, paga con tarjeta de crédito o métodos con protección al comprador, y recuerda: ningún ejecutivo bancario te pedirá NIPs ni códigos por teléfono. Jamás.
- Mantén todo actualizado como si fueran vacunas: activa las actualizaciones automáticas, no pospongas los reinicios de seguridad y combina sistema al día con defensas integradas y buenos hábitos. Ninguno funciona solo.

- Cuida tus redes sociales contra la suplantación y el exceso de información: no publiques ubicación en tiempo real ni documentos, configura la privacidad, limpia contactos desconocidos y pacta una palabra clave familiar contra secuestros virtuales.
- Navega con criterio: entra a tus sitios importantes desde marcadores, descarga solo de fuentes oficiales, audita tus extensiones del navegador y nunca ejecutes comandos que te dicte un desconocido ni desactives tu antivirus para "ver contenido".
- Frente a las estafas modernas (voz clonada, deepfakes, QR tramposos, empleos falsos), la tecnología del engaño cambia pero la psicología no: urgencia más secreto más pago inmediato. Frena, verifica por otro canal y ante la duda di que no.
- La seguridad es un asunto de familia y de equipo: enseña a niños y adultos mayores con reglas simples, haz simulacros sin regaños, reporta los correos sospechosos en el trabajo y crea una cultura donde avisar a tiempo no dé vergüenza.

Cierre de la unidad: la ciberseguridad moderna no es un tema para especialistas encerrados en un cuarto oscuro; es una habilidad de vida, tan básica como saber cruzar la calle. Nadie te pide que seas un experto en criptografía, solo que cierres tus puertas digitales, desconfíes de los desconocidos que tocan a ellas y tengas un plan por si algún día alguien intenta forzarlas. Empieza hoy con tres acciones: instala un gestor de contraseñas, activa el 2FA en tu correo y haz tu primer respaldo. Esas tres acciones, que te toman una tarde, te ponen por encima del 90 por ciento de los usuarios. El resto de esta unidad ya vive en tu cabeza: úsalo, compártelo con tu familia y mantén el hábito. Tu yo del futuro te lo va a agradecer.

Y una última idea para llevar: la seguridad perfecta no existe, y no la necesitas. Lo que necesitas es ser un blanco difícil y tener un plan. Los delincuentes digitales son oportunistas: buscan la puerta abierta, no la fortaleza. Cada capa que agregaste en esta unidad (contraseñas únicas, 2FA, criterio ante el phishing, VPN, respaldos, IoT aislado, protocolo de respuesta) te aleja del grupo de las víctimas fáciles. Comparte lo aprendido: cada persona de tu familia o tu equipo que aplique estas bases multiplica tu propia seguridad, porque los ataques llegan por el eslabón más débil. La ciberseguridad moderna, al final, no es tecnología: es cultura. Y ahora tú eres parte de ella.

UNIDAD 6: Impresión 3D y fabricación digital

Imagínate que rompes la patita de un carrito de juguete de tu sobrino, esa pieza pequeña de plástico que ya no venden en ningún lado porque el modelo se discontinuó hace diez años. Antes, tu única opción era buscar en tianguis, en bazares o en internet a ver si alguien vendía la pieza usada, y cruzar los dedos de que llegara bien. Hoy tienes otra opción: medir la pieza, dibujarla en tu computadora y "imprimirla" en plástico en tu escritorio, en una o dos horas. Eso es la impresión 3D, y es exactamente de eso de lo que trata esta unidad.

La impresión 3D es una tecnología de fabricación aditiva: en lugar de cortar, taladrar o desbastar un bloque de material (eso sería fabricación sustractiva, como esculpir), la impresora construye el objeto capa por capa, depositando material solo donde se necesita. Piensa en una impresora de tinta de las de oficina, pero en lugar de poner tinta sobre papel, pone plástico derretido sobre plástico ya endurecido, una y otra vez, hasta formar un objeto sólido con volumen real. De ahí viene el nombre de "3D": crea las tres dimensiones.

Lo interesante es que la impresión 3D no es una tecnología nueva. Se inventó en los años ochenta y durante décadas fue carísima: solo las grandes empresas podían pagar máquinas de decenas de miles de dólares. Lo que cambió todo fue la expiración de patentes clave alrededor de 2009 y el movimiento de hardware abierto, que permitió que surgieran impresoras de escritorio de menos de 300 dólares. De pronto, una fábrica en miniatura cabía en tu cuarto. Hoy hay millones de impresoras 3D en hogares, escuelas y talleres de todo el mundo, y la tecnología sigue bajando de precio mientras sube en calidad.

Esta unidad es una de las más prácticas de todo el libro, porque la impresión 3D toca todas las unidades anteriores: necesitas computadoras y software (Unidad 1 y 2), entiendes mejor los materiales si viste electrónica y sensores (Unidad 3 y 4), y el cierre natural es conectar tu impresora a internet y automatizarla con inteligencia artificial (Unidad 5 y lo que verás al final de esta unidad). La fabricación digital es, en pocas palabras, la fusión de bits y átomos: diseñas en el mundo digital y el objeto aparece en el mundo físico.

Al terminar esta unidad vas a saber qué tecnologías de impresión 3D existen y cuál te conviene, cómo está armada una impresora por dentro, qué materiales usar para cada tipo de pieza, cómo pasar de una idea a un objeto impreso paso a paso, cómo configurar el software que prepara tus modelos, cómo calibrar tu máquina y arreglar los errores más comunes, y cómo esta tecnología ya está cambiando la medicina, la industria y los negocios pequeños. Y lo mejor: vas a diseñar tus primeras piezas con código, como un profesional.

De la fábrica al escritorio: la revolución de fabricar en casa

Para entender por qué la impresión 3D es tan importante, piensa en cómo se fabricaban las cosas durante el siglo XX. Si querías producir un objeto de plástico, necesitabas un molde de acero, una máquina de inyección gigante, y producir miles de unidades para que el costo por pieza fuera razonable. Eso significaba que solo las empresas grandes podían fabricar productos. El ciudadano común era consumidor, no productor.

La impresión 3D rompe esa lógica porque no necesita moldes. El "molde" es un archivo digital, y ese archivo se puede copiar gratis por internet, modificar y volver a imprimir. Esto tiene tres consecuencias enormes. Primera: la personalización. Cada pieza impresa puede ser diferente sin costo extra; puedes imprimir una prótesis de mano a la medida exacta de una persona, algo imposible de hacer rentable con moldes. Segunda: la descentralización. Ya no necesitas que el objeto viaje en barco desde otro continente; lo fabricas donde lo necesitas, cuando lo necesitas. Tercera: la reparación y la longevidad. Esa pieza de plástico que se rompió y que condenaba a la basura a todo un aparato, ahora la puedes reponer.

Hay una analogía que me gusta mucho: la impresión 3D es para los objetos físicos lo que la imprenta fue para los libros. Antes de la imprenta, cada libro se copiaba a mano y eran carísimos; después, el conocimiento se pudo replicar barato. Antes de la impresión 3D, cada objeto físico requería una fábrica; ahora, el diseño se replica barato y el objeto se materializa localmente. Todavía no estamos en el punto de "imprimir todo en casa" como en las películas de ciencia ficción, pero la dirección es clara.

En esta unidad vas a ver también el lado realista: la impresión 3D es lenta (una pieza mediana tarda horas), los objetos impresos tienen líneas visibles de capas, y no todos los materiales sirven para todo. No es magia, es una herramienta con fortalezas y límites, y un buen fabricante digital sabe exactamente cuándo usarla y cuándo no.

Lo que vas a construir en esta unidad

Esta unidad está diseñada para que termines con habilidades reales, no solo teoría. El plan es el siguiente: primero vas a entender las tres tecnologías principales de impresión 3D (FDM, SLA y SLS) con una comparación honesta de costos, calidad y usos. Después vas a conocer tu máquina por dentro: la anatomía de una impresora FDM, que es la que probablemente tendrás en casa por ser la más barata.

Luego viene la parte de materiales, que es donde muchos principiantes se equivocan: vas a aprender las diferencias entre PLA, PETG, ABS y TPU, y una regla simple para decidir cuál usar en cada proyecto. Después recorrerás el flujo de trabajo completo, del modelo al objeto: diseñar, laminar e imprimir, conociendo el software de cada etapa.

La parte más divertida es el diseño paramétrico con OpenSCAD: vas a escribir código que genera piezas en 3D, como si programaras objetos físicos. Es programación aplicada al mundo real y conecta directamente con lo que aprendiste de lógica y código en unidades anteriores. También vas a dominar el laminador (el "slicer"), que es el programa que convierte tu modelo en instrucciones para la impresora, y vas a entender sus parámetros clave.

Finalmente, verás la calibración y la solución de fallas (porque tu impresora VA a fallar, y eso es normal), las aplicaciones reales que ya están cambiando el mundo, y el cierre futurista: tu impresora conectada a internet, vigilada por cámaras y optimizada con inteligencia artificial. Tu fábrica en casa, literalmente.

Cómo usar esta unidad si aún no tienes impresora

Una pregunta común es: "¿necesito comprar una impresora 3D para aprovechar esta unidad?". La respuesta honesta es no, aunque tener una ayuda muchísimo. Todo el software que verás aquí es gratuito: Tinkercad funciona en el navegador, OpenSCAD se descarga gratis, y los laminadores como Cura o PrusaSlicer también son gratis. Puedes diseñar piezas, laminarlas y simular todo el proceso sin gastar un peso.

Si quieres imprimir tus diseños sin comprar máquina, tienes opciones: muchos FabLabs, bibliotecas públicas y universidades ofrecen servicio de impresión 3D barato o gratuito; también hay servicios en línea donde subes tu archivo y te mandan la pieza impresa por paquetería. Y cuando decidas comprar tu primera impresora, esta unidad te va a servir como guía de compra: vas a saber qué buscar, qué evitar y cuánto deberías pagar.

Mi recomendación: lee la unidad completa primero para entender el panorama, haz los ejercicios de diseño con OpenSCAD en tu computadora (son gratis), y cuando tengas acceso a una impresora, vuelve a la sección de calibración y fallas con la máquina enfrente. Verás que todo tiene más sentido con la impresora zumbando a tu lado.

Explicación detallada

Qué es la impresión 3D y las tres tecnologías principales

La impresión 3D, técnicamente llamada manufactura aditiva, es un conjunto de tecnologías que construyen objetos tridimensionales agregando material capa sobre capa a partir de un modelo digital. El proceso siempre empieza igual: tienes un archivo con la geometría del objeto (normalmente en formato STL u OBJ), un software llamado laminador lo "rebana" en cientos o miles de capas horizontales delgadas, y la impresora reproduce cada capa una tras otra hasta

completar la pieza. Es como construir con bloques de LEGO, pero con capas de décimas de milímetro y sin que tengas que poner cada bloque a mano.

Existen docenas de tecnologías de impresión 3D, pero tres dominan el mercado y son las que tienes que conocer: FDM, SLA y SLS. Vamos una por una.

FDM significa Fused Deposition Modeling, o modelado por deposición fundida. Es la tecnología más común y barata. Funciona así: un filamento de plástico (un hilo de 1.75 mm de diámetro, como un espagueti grueso enrollado en un carrete) entra a un extrusor, que lo empuja hasta una boquilla caliente (el "hotend") a unos 200 grados C. El plástico se derrite y la boquilla lo deposita como si fuera una pistola de silicón, dibujando cada capa sobre la anterior. Cuando el plástico sale, se enfría y se endurece en segundos. La analogía más clara: es como decorar un pastel con una manga pastelera, solo que el betún es plástico derretido y el dibujo tiene cientos de capas superpuestas.

Las impresoras FDM cuestan entre 150 y 600 dólares las de escritorio, los materiales son baratos (un kilo de PLA ronda los 20 dólares) y son relativamente fáciles de mantener. Su desventaja es la resolución: las capas se ven a simple vista (esas líneas horizontales características) y los detalles muy finos salen toscos. La precisión típica anda en 0.1 a 0.3 mm por capa. Para piezas funcionales, prototipos, juguetes, repuestos y la gran mayoría de proyectos caseros, es más que suficiente.

SLA significa Stereolithography, o estereolitografía. En lugar de fundir plástico, usa resina líquida fotosensible que se endurece con luz ultravioleta. La impresora tiene un tanque con resina líquida y una pantalla o láser UV que ilumina cada capa con la forma exacta; donde la luz toca, la resina se solidifica. La plataforma sube poquito a poquito y la pieza "emerge" del líquido, como una escultura saliendo del agua. Es un espectáculo verlo.

La calidad del SLA es muy superior: capas de 0.025 a 0.1 mm, superficies casi lisas y detalles finísimos, ideal para joyería, figuras de colección, moldes dentales y prototipos que deben verse profesionales. El costo es mayor: las impresoras de escritorio andan entre 200 y 800 dólares, la resina cuesta más que el filamento, y el proceso es más engorroso porque las piezas salen "verdes" (a medio curar), hay que lavarlas en alcohol isopropílico y curarlas con más luz UV. Además la resina líquida es tóxica antes de curarse: necesitas guantes, ventilación y cuidado. No es la mejor primera impresora para un principiante, pero es la mejor opción cuando la calidad visual es lo más importante.

SLS significa Selective Laser Sintering, o sinterizado selectivo por láser. Aquí el material es un polvo finísimo (normalmente nylon) esparcido en una cama; un láser dibuja cada capa fundiendo los granos de polvo justo donde debe haber material sólido. El polvo sin fundir sirve de soporte natural, así que puedes imprimir geometrías complejísimas sin estructuras de soporte. Las piezas salen fuertes, funcionales y con buen acabado.

El problema del SLS es el precio y la complejidad: las máquinas industriales cuestan decenas de miles de dólares, aunque ya existen versiones "de escritorio" de unos 5,000 dólares. Para ti como lector, lo más probable es que uses SLS a través de servicios en línea: subes tu archivo, ellos lo imprimen en sus máquinas industriales y te mandan la pieza. Es la tecnología favorita para producción de series cortas (decenas o cientos de piezas) y para piezas de ingeniería que deben aguantar esfuerzo mecánico real.

Para compararlas rápido, piensa así: FDM es la camioneta de trabajo (barata, versátil, aguanta todo), SLA es el auto deportivo (hermoso y preciso, pero delicado y caro de mantener), y SLS es el camión de carga industrial (potente y capaz, pero no lo tienes en tu cochera). En esta unidad nos vamos a enfocar en FDM, porque es la tecnología que tendrás en casa y la que te da más libertad para experimentar.

Anatomía de una impresora FDM: conoce tu máquina por dentro

Una impresora FDM parece complicada, pero en realidad son pocos componentes haciendo un baile muy preciso. Conocer cada parte te va a servir para calibrar, diagnosticar fallas y hasta mejorar tu máquina. Vamos de arriba hacia abajo, como si la desarmáramos mentalmente.

El marco o estructura es el esqueleto: normalmente perfiles de aluminio en forma de pórtico. Su trabajo es mantener todo rígido; si el marco se flexiona, las capas salen chuecas. Las impresoras baratas a veces traen marcos endebles y la primera mejora que les hace la gente es rigidizarlos. Piensa en el marco como los cimientos de una casa: si están chuecos, todo lo demás sale mal.

Los ejes de movimiento son X, Y y Z. El eje X mueve el cabezal de izquierda a derecha, el eje Y mueve la cama de adelante hacia atrás (en el diseño más común, tipo "bed slinger"), y el eje Z sube el cabezal capa por capa. Cada eje tiene un motor paso a paso (stepper), que gira en incrementos pequeñísimos y precisos, conectado a una banda dentada o a un tornillo sin fin (husillo). La precisión de tu impresora depende directamente de estos motores: un motor paso a paso típico tiene 200 pasos por vuelta, y con microstepping la electrónica lo divide en pasos aún más finos.

El extrusor es el sistema que alimenta el filamento. Hay dos tipos: Bowden, donde el motor que empuja el filamento está fijo en el marco y lo empuja por un tubo de teflón hasta el cabezal (más ligero el cabezal, se mueve más rápido, pero batalla con filamentos flexibles); y de accionamiento directo, donde el motor va montado sobre el cabezal (más pesado, pero empuja mejor y maneja filamentos flexibles como el TPU sin problema). Si planeas imprimir TPU, busca accionamiento directo.

El hotend es el corazón caliente: un bloque de aluminio con una resistencia que calienta hasta la temperatura de impresión (190 a 260 grados C según el material), un termistor que mide la temperatura cientos de veces por segundo, y la boquilla de latón (típicamente de 0.4 mm de diámetro) por donde sale el plástico fundido. Arriba del bloque caliente hay un disipador con ventilador que mantiene fría la zona superior para que el filamento no se derrita antes de tiempo (eso causaría un atasco). La boquilla es un consumible: se desgasta con el uso, sobre todo con filamentos abrasivos, y se cambia con una llave pequeña.

La cama de impresión es la superficie donde nace tu pieza. Puede ser de vidrio, de acero con recubrimiento PEI, o magnética flexible. Casi siempre es calefactable (60 grados C para PLA, hasta 100 o más para ABS) porque el calor ayuda a que la primera capa se pegue bien. La nivelación de la cama, es decir, que esté a la misma distancia de la boquilla en todos los puntos, es EL ajuste más importante de toda la impresora; una cama mal nivelada es la causa número uno de impresiones fallidas. Muchas impresoras modernas traen nivelación automática con un sensor, pero aun así conviene entender el proceso manual.

La electrónica es el cerebro: una tarjeta controladora (con firmware como Marlin o Klipper) que interpreta las instrucciones, una fuente de poder, y la pantalla con la perilla o botones para operarla. El archivo que la impresora lee se llama G-code: un texto plano con miles de líneas de comandos como "muévete a X=10 Y=20" o "calienta a 200 grados". Cuando lamines tus modelos, vas a generar G-code, y entender aunque sea por encima cómo se ve te va a ayudar a diagnosticar problemas.

Un error común de principiantes es tratar la impresora como una impresora de papel: "le doy imprimir y me olvido". No funciona así. La impresora 3D es más parecida a un instrumento musical: requiere afinación (calibración), práctica y atención, sobre todo al principio. Pero una vez que la conoces, se vuelve una herramienta confiable.

Materiales: PLA, PETG, ABS y TPU, y cuándo usar cada uno

Elegir el material correcto es la mitad del éxito en una impresión. Cada plástico tiene su personalidad: temperatura, resistencia, flexibilidad y mañas propias. Los cuatro que tienes que dominar son PLA, PETG, ABS y TPU. Todos vienen en carretes de filamento de 1.75 mm (el estándar) y se imprimen en FDM.

El PLA (ácido poliláctico) es el rey de los principiantes y el material con el que debes empezar. Es un bioplástico derivado del maíz o la caña de azúcar, biodegradable en condiciones industriales. Se imprime fácil: funde entre 190 y 220 grados C, la cama puede ir a 60 grados o incluso fría, casi no se deforma al enfriarse (poco "warping") y no necesita impresora cerrada. Los voladizos y detalles finos salen muy bien. Su debilidad es el calor y la fragilidad: se ablanda a partir de los 60 grados C (déjalo en un carro al sol y se deforma) y es quebradizo ante impactos fuertes. Úsalo para: prototipos, figuras decorativas, juguetes, organizadores, piezas que no sufran calor ni golpes. No lo uses para: piezas de carro expuestas al sol, engranes con carga, nada que vaya al horno o al microondas. Regla de oro: si no sabes qué material usar, usa PLA.

El PETG (tereftalato de polietileno glicolado) es el todoterreno. Es el punto medio perfecto entre la facilidad del PLA y la resistencia del ABS. Se imprime a 220-250 grados C con cama a 70-85 grados, es más resistente a impactos que el PLA, aguanta mejor el calor (hasta unos 80 grados C) y resiste bien la intemperie y los rayos UV. Es el favorito para piezas funcionales de uso diario. Sus mañas: tiende a soltar "hilitos" (stringing) si la retracción no está bien ajustada, y se pega MUY fuerte a algunas camas (conviene usar laca o pegamento en barra como separador). Úsalo para: repuestos funcionales, piezas para exterior, soportes, macetas, protectores, piezas que se mojan. Es mi recomendación como segundo material después del PLA: cuando el PLA no aguante, el PETG casi siempre sí.

El ABS (acrilonitrilo butadieno estireno) es el plástico de los LEGOs y de muchas piezas de carro. Es fuerte, aguanta impactos y calor (hasta unos 100 grados C), y se puede lijar, taladrar y pegar con acetona (que literalmente "suelta" las capas entre sí). Pero es exigente: se imprime a 230-260 grados C, necesita cama muy caliente (90-110 grados) e idealmente impresora cerrada, porque se contrae al enfriarse y las piezas grandes se deforman o se despegan (warping). Además, al imprimirse libera vapores con olor fuerte que conviene ventilar. Úsalo para: piezas que aguantan calor (cerca de motores), carcasas, piezas que vas a post-procesar (lijar y pintar), prototipos que simulan inyección de plástico. No lo uses si no tienes impresora cerrada o ventilación: te vas a frustrar. Muchos principiantes saltan al ABS demasiado pronto y culpan a la impresora de lo que en realidad es falta de cámara cerrada.

El TPU (poliuretano termoplástico) es el flexible: goma imprimible. Viene en distintas durezas (la escala Shore; 95A es semirrígido como la suela de un tenis, 85A es más suave). Se imprime lento (20-30 mm/s) y a 210-230 grados C, y funciona mucho mejor con extrusor de

accionamiento directo porque el filamento flexible se dobla dentro del tubo Bowden. Úsalo para: fundas de celular, juntas y empaques, llantas de robots, correas, piezas que deben doblarse o absorber golpes. Su maña principal es la velocidad: si imprimes rápido, se atasca. Paciencia y sale perfecto.

Hay una tabla mental que te va a servir siempre. ¿Pieza decorativa o prototipo rápido? PLA. ¿Pieza funcional de uso diario? PETG. ¿Calor, impactos fuertes o post-procesado serio? ABS (con impresora cerrada). ¿Necesita doblarse? TPU. Y un consejo de almacenamiento que casi nadie te dice al inicio: el filamento absorbe humedad del ambiente y el filamento húmedo imprime horrible (crepita, deja burbujas y capas débiles). Guarda tus carretes en bolsas selladas con bolsitas de sílica, y si vives en clima húmedo, considera una caja seca o un deshidratador de filamento. El 30 por ciento de los "problemas de la impresora" en realidad son problemas de filamento húmedo.

Del modelo al objeto: el flujo de trabajo completo

Ya conoces la máquina y los materiales. Ahora veamos el camino completo desde que la idea está en tu cabeza hasta que el objeto está en tu mano. Son tres etapas: diseño, laminado e impresión. Cada etapa tiene su software y sus decisiones, y un error en una etapa temprana se arrastra hasta el final, así que vale la pena entenderlas bien.

La etapa de diseño es donde creas o consigues el modelo 3D. Tienes tres caminos. El primero es descargar modelos listos de repositorios como Printables, Thingiverse o MakerWorld: hay millones de diseños gratuitos, desde juguetes hasta repuestos y herramientas. Es perfecto para empezar a imprimir el primer día. El segundo es diseñar con software CAD: Tinkercad (gratis, en el navegador, ideal para principiantes y educación), Fusion 360 (profesional, gratis para uso personal y estudiantes, la herramienta seria para piezas de ingeniería) o FreeCAD (gratis y de código abierto). El tercero, y el más poderoso para ti que vienes de programar, es el diseño paramétrico con OpenSCAD: escribes código y el código genera la geometría. Lo veremos a fondo en los ejemplos.

El formato de salida del diseño suele ser STL, que describe la superficie del objeto como miles de triangulitos. Piensa en el STL como una malla que envuelve tu pieza. Un STL debe ser "manifold", es decir, una malla cerrada sin hoyos: si tu modelo tiene paredes con grosor cero o huecos, el laminador se confunde. La mayoría de software moderno te avisa, pero es un concepto que conviene conocer.

La etapa de laminado es donde el "slicer" (Cura, PrusaSlicer, OrcaSlicer, todos gratuitos) convierte tu modelo 3D en G-code, las instrucciones capa por capa para la impresora. Aquí tomas las decisiones técnicas más importantes: altura de capa (0.2 mm es el estándar

equilibrado; 0.12 mm para calidad fina; 0.28 mm para prototipos rápidos), porcentaje de relleno o "infill" (15-20 por ciento basta para la mayoría de piezas; más relleno no siempre significa más fuerza, el grosor de paredes importa más), número de paredes o perímetros (3 paredes dan piezas muy resistentes), soportes (estructuras temporales para voladizos mayores a 45-60 grados, que luego se quitan), adherencia a la cama (falda, borde o balsa según qué tan difícil sea que la pieza se pegue), y temperaturas y velocidades según tu material.

Una buena práctica: antes de imprimir una pieza de 8 horas, imprime una "torre de temperatura" o un cubo de calibración de 20 minutos para verificar que tus ajustes funcionan con ese filamento específico. Cada carrete, incluso del mismo material y marca, puede comportarse un poco distinto. Los profesionales no imprimen a ciegas: calibran primero, imprimen después.

La etapa de impresión es la ejecución. Preparas la cama (limpia, nivelada, a temperatura), cargas el filamento, mandas el G-code (por tarjeta SD, USB o WiFi según tu impresora) y observas las primeras capas como halcón: si la primera capa sale mal, todo lo demás saldrá mal. La primera capa debe verse como líneas ligeramente aplastadas y bien pegadas entre sí, sin huecos ni grumos. Una vez que las primeras 5-10 capas van bien, la impresión suele seguir bien sola, aunque en piezas largas conviene echar un ojo cada tanto o poner una cámara.

El post-procesado es el paso opcional final: quitar soportes con pinzas, lijar, y si quieres un acabado profesional, aplicar masilla, pintar o, en el caso del ABS, un baño de vapor de acetona que alisa la superficie. Muchas piezas funcionales no necesitan nada más que quitar los soportes.

El error más común de principiantes en el flujo completo es saltarse el laminado consciente: descargar un modelo, laminar con los ajustes por defecto y esperar lo mejor. Los ajustes por defecto son un punto de partida, no una receta universal. Cada pieza (su tamaño, su orientación, su uso final) pide ajustes distintos, y aprender a leer una pieza y decidir "esta necesita más paredes" o "esta necesita soportes" es la habilidad central del fabricante digital.

Software de diseño: Tinkercad, Fusion 360 y OpenSCAD programable

Diseñar en 3D suena intimidante, pero hay una herramienta para cada nivel, y la curva de aprendizaje es más amable de lo que parece. Lo importante es empezar con la adecuada para tu nivel y subir de nivel cuando la herramienta se te quede chica.

Tinkercad es el punto de partida ideal. Es gratis, funciona en el navegador sin instalar nada, y su lógica es de "primaria": arrastras formas básicas (cubos, cilindros, esferas), las combinas, les haces hoyos con otras formas, y exportas el STL. En una tarde puedes diseñar un llavero

con tu nombre, un soporte para celular o una cajita. Sus límites: no sirve para piezas de precisión ni diseños complejos, y todo es manual (si quieres 20 hoyos iguales, los pones uno por uno). Úsalo para aprender los conceptos (qué es extruir, qué es un hoyo, cómo se combinan sólidos) y para piezas simples. Muchos maestros lo usan con niños de primaria, así que no hay pretexto: tú puedes.

Fusion 360 es la herramienta profesional. Es un CAD paramétrico completo: dibujas bocetos en 2D con medidas exactas, los extruyes a 3D, redondeas bordes, haces roscas, ensamblas piezas y hasta simulas esfuerzos. Es gratis para uso personal, estudiantes y startups pequeñas, y es lo que usan miles de ingenieros y diseñadores en la industria. Su curva de aprendizaje es más empinada (dedícale un par de semanas de tutoriales), pero es la herramienta que te permite diseñar cualquier cosa: desde un engrane hasta la carcasa de un dron. Si tu meta es diseñar piezas funcionales serias o trabajar en esto profesionalmente, Fusion 360 es tu destino. Buena práctica: aprende a acotar tus bocetos (ponerle medidas a todo) desde el día uno; un diseño sin medidas es un dibujo, no un plano.

OpenSCAD es la joya para programadores: en lugar de dibujar con el ratón, escribes código y el código genera el modelo 3D. Es gratis y de código abierto. Suena raro al principio, pero tiene una ventaja brutal: el diseño paramétrico. Defines variables (ancho = 50, alto = 30, diametro_hoyo = 5) y toda la pieza se construye a partir de ellas; si cambias una variable, todo se actualiza solo. ¿Necesitas la misma caja en 5 tamaños? Cambias un número y listo, en vez de redibujar 5 veces. Además, tu diseño es un archivo de texto: lo puedes versionar con Git, compartir, y generar variantes automáticamente. En los ejemplos paso a paso de esta unidad vas a escribir tus primeros scripts de OpenSCAD: una caja paramétrica, un engrane y un soporte. Si ya programas, OpenSCAD te va a parecer natural y divertido; si no, es una excelente excusa para aprender lógica de programación con resultados tangibles.

¿Cómo elegir? Empieza en Tinkercad para entender el espacio 3D, pasa a OpenSCAD si te gusta programar (o en paralelo), y sube a Fusion 360 cuando necesites precisión de ingeniería. No son excluyentes: muchos diseñadores usan las tres según el proyecto. Y recuerda los repositorios (Printables, Thingiverse): no todo hay que diseñarlo desde cero; descargar, modificar y remezclar diseños es parte legítima y valiosa del flujo de trabajo, siempre respetando las licencias de cada modelo.

Calibración y solución de fallas comunes: tu impresora VA a fallar

Te voy a decir algo que te va a ahorrar muchas frustraciones: tu impresora 3D va a fallar, y eso es normal. Hasta las máquinas caras fallan. La diferencia entre un principiante frustrado y un fabricante competente no es tener una impresora que nunca falla, sino saber diagnosticar y arreglar rápido. Esta sección es tu botiquín de primeros auxilios.

La calibración básica tiene tres pilares. Primero, la nivelación de la cama: la boquilla debe estar a la misma distancia (el grosor de una hoja de papel, unos 0.1 mm) en toda la superficie. Si está muy pegada, el plástico no sale y se atasca; si está muy separada, la pieza no se pega y se convierte en "espagueti". Nivelas con la cama y la boquilla calientes (los metales se expanden con el calor), ajusta las cuatro esquinas y verifica el centro. Segundo, los pasos del extrusor (E-steps): la impresora debe extruir exactamente la cantidad de filamento que cree estar extruyendo; se calibra marcando 100 mm de filamento, pidiendo que extruya 100 mm y midiendo cuánto avanzó realmente. Tercero, la temperatura ideal de tu filamento: imprime una torre de temperatura (un modelo que cambia de temperatura cada cierto número de capas) y elige visualmente el rango donde las capas se ven mejor, sin hilos ni grumos.

Ahora las fallas más comunes y sus remedios. El "espagueti" (la pieza se despegas a media impresión y la boquilla sigue depositando plástico en el aire) casi siempre es mala adherencia de la primera capa: nivelas la cama, limpias la superficie con alcohol isopropílico (la grasa de los dedos es enemiga), subes un poco la temperatura de la cama o usas laca para el cabello o pegamento en barra. El warping (las esquinas de la pieza se levantan) es contracción del plástico al enfriarse: usa cama más caliente, evita corrientes de aire, y para ABS usa impresora cerrada. El stringing (hilitos finos como telaraña entre partes de la pieza) se arregla aumentando la retracción en el laminador (que el extrusor "chupe" un poco el filamento al moverse entre zonas) y a veces bajando un poco la temperatura.

La subextrusión (capas débiles, con huequitos, piezas que se rompen fácil) puede ser boquilla parcialmente tapada (haz un "cold pull": calientas, metes filamento, enfrías a 90 grados y jalas de golpe para sacar la suciedad), filamento húmedo, o temperatura muy baja. La sobreextrusión (grumos, superficie rugosa) es lo contrario: baja el flujo al 95 por ciento o calibra los E-steps. El atasco total (no sale nada) suele ser por retracción excesiva, ventilador del disipador apagado, o filamento de mala calidad con diámetro irregular.

Buenas prácticas que te van a ahorrar horas: guardas un "diario de impresión" (material, temperaturas, velocidad y resultado de cada impresión importante); cuando algo salga perfecto, anotas los ajustes para repetirlos. Limpias la boquilla por fuera con un cepillo de latón en caliente cada pocas impresiones. Revisas que las bandas estén tensas (deben sonar como cuerda de guitarra al pellizcarlas, no flojas). Y nunca, nunca dejas una impresora trabajando sin supervisión por muchas horas si no tienes detección de fallas: un incendio por falla eléctrica es rarísimo pero posible; un detector de humo cerca de la impresora es barato y sensato.

Analogía final: calibrar una impresora 3D es como afinar una guitarra. Al principio te tardas media hora y suena mal; con práctica lo haces en dos minutos de oído. No te desesperes en las primeras semanas: todos los que imprimen bien hoy pasaron por la etapa del espagueti.

Aplicaciones reales: prótesis, repuestos, prototipos y negocio

La impresión 3D ya no es un juguete de aficionados: está resolviendo problemas reales en medicina, industria y negocios. Conocer estos casos te va a dar ideas para tus propios proyectos y te va a mostrar el tamaño de la oportunidad.

En medicina, el caso más inspirador son las prótesis. Organizaciones como e-NABLE coordinan voluntarios de todo el mundo que imprimen manos y brazos prostéticos para niños, gratis, con diseños de código abierto. Una prótesis comercial puede costar miles de dólares y el niño la deja de usar en un año porque creció; una impresa en 3D cuesta unos 50 dólares en material y se reimprime a la medida nueva cuando hace falta. También se imprimen guías quirúrgicas personalizadas (plantillas que ayudan al cirujano a cortar exactamente donde debe), modelos anatómicos a partir de tomografías para planear operaciones complejas, y férulas a medida. No todo lo médico se puede imprimir en casa (los implantes internos requieren materiales y certificaciones especiales), pero el impacto en prótesis externas y herramientas ya es enorme.

En el hogar y el taller, la aplicación más rentable es la de repuestos. Esa perilla de la estufa que se rompió, el soporte del espejo retrovisor, la pieza del juguete del inicio de la unidad, el clip del tendedero: millones de objetos se tiran a la basura cada año por una piecita de plástico que cuesta centavos producir. Hay comunidades enteras dedicadas a modelar repuestos de aparatos descontinuados. Esto no solo ahorra dinero: es ecología práctica, porque alarga la vida útil de las cosas. Consejo: cuando compres un aparato, guarda sus piezas plásticas críticas; si algún día se rompen, ya tienes la referencia para modelarlas.

En la industria, el uso estrella es el prototipado rápido. Antes, probar una idea de producto requería mandar a hacer un molde o maquinar piezas, tardando semanas y costando miles; hoy un ingeniero diseña en la mañana, imprime en la tarde y prueba en la noche. Ese ciclo de iteración rapidísimo es lo que permite innovar más rápido. Además, la impresión 3D permite geometrías imposibles de fabricar de otra manera: estructuras internas tipo panal que son ultraligeras pero resistentes, canales de refrigeración con formas orgánicas dentro de moldes, piezas que antes eran ensambles de 20 componentes ahora impresas en una sola pieza. En la industria aeroespacial ya vuelan piezas impresas en metal; en la automotriz se imprimen herramientas y soportes de línea de producción.

Y en los negocios, la impresión 3D abrió la puerta al microemprendimiento de manufactura. Con una o dos impresoras puedes vender: piezas personalizadas (llaveros con nombres, soportes a medida), repuestos difíciles de conseguir, prototipos para inventores locales, miniaturas para juegos de mesa, o un servicio de impresión por encargo en tu ciudad. La clave del negocio no es la impresora (que cualquiera puede comprar), sino el diseño y el servicio:

resolver el problema específico de un cliente. Un error común es querer competir por precio con la producción en masa; no vas a ganarle a una fábrica china en costo por pieza. Gánale en personalización, velocidad de entrega local y lotes pequeños que a las fábricas no les interesan.

Impresión 3D + IoT + IA: tu fábrica en casa

Aquí es donde esta unidad se conecta con todo el libro y mira al futuro: una impresora 3D conectada a internet y supervisada por inteligencia artificial deja de ser "una máquina en tu escritorio" y se convierte en una microfábrica automatizada. Vamos por partes.

Primero, la conectividad (IoT). Con una minicomputadora barata como una Raspberry Pi y el software OctoPrint (gratis y de código abierto), tu impresora se vuelve inteligente: la controlas desde el celular, le mandas archivos por WiFi sin andar con la tarjeta SD, ves el avance en tiempo real y recibes alertas. Le agregas una cámara web barata y puedes vigilar tus impresiones desde donde estés. Hay complementos que detectan fallas automáticamente: analizan la imagen de la cámara y si ven el temido "espagueti", pausan la impresión y te avisan por Telegram antes de que desperdicies horas de filamento. Eso, que hace unos años era ciencia ficción casera, hoy lo montas en una tarde.

Segundo, la automatización del flujo. Puedes tener una "cola de impresión": una carpeta en tu computadora donde dejas modelos, y un script que los lamina automáticamente con tus perfiles y los manda a la impresora uno tras otro. Sensores de filamento detectan cuando el carrete se acaba y pausan la impresión para que cambies el rollo sin perder la pieza. Enchufes inteligentes apagan la impresora cuando termina (o cuando algo sale mal). Todo esto lo puedes armar con herramientas que ya conoces de la Unidad 4 (sensores y actuadores) y la Unidad 5 (scripts e IA).

Tercero, la inteligencia artificial aplicada. La IA ya está entrando a la fabricación digital por varias puertas: generadores de diseños que crean piezas optimizadas a partir de una descripción en texto ("diseña un soporte ligero para una cámara de 200 gramos"); laminadores con perfiles auto-ajustados que analizan tu modelo y sugieren la mejor orientación y los mejores parámetros; visión por computadora que no solo detecta fallas sino que las corrige en marcha ajustando el flujo; y mantenimiento predictivo que avisa "tu boquilla se está desgastando" antes de que falle una impresión importante. No necesitas programar todo esto desde cero: cada vez más viene integrado en el software que ya usas.

El concepto de fondo se llama manufactura distribuida: en lugar de unas pocas fábricas gigantes produciendo todo, miles de microfábricas (tu casa, un taller local, una escuela) produciendo lo que se necesita localmente, coordinadas por internet y optimizadas por IA. Los

diseños viajan como archivos, los objetos se materializan cerca del consumidor. Menos transporte, menos inventario, menos desperdicio, más personalización. Todavía falta camino (la impresión 3D es lenta y no sirve para todo), pero la dirección es clara y tú, al terminar esta unidad, ya vas a estar del lado de los que producen, no solo de los que consumen.

Un consejo final antes de los ejemplos: empieza simple, termina simple. Tu primera semana con una impresora debería ser imprimir modelos descargados y calibrar; tu primer mes, diseñar piezas simples en Tinkercad u OpenSCAD; y a partir de ahí, cada proyecto un poco más ambicioso. La fabricación digital es una habilidad que se construye por capas, igual que tus impresiones.

Diseño para impresión 3D: tolerancias, orientación y reglas de oro

Saber usar la impresora es la mitad del juego; la otra mitad es diseñar piezas que salgan bien a la primera. Existen reglas de "diseño para manufactura aditiva" que te ahorran reimpresiones, y la mayoría se resumen en una idea: diseña pensando en cómo la impresora construye, capa por capa, desde abajo hacia arriba.

La regla de los 45 grados es la más famosa: cualquier voladizo con un ángulo mayor a 45 grados respecto a la vertical necesita soportes, porque el plástico recién depositado no tiene nada debajo que lo sostenga. Un voladizo a 45 grados o menos se "auto-soporta" porque cada capa se apoya un poco sobre la anterior, como una escalera. En la práctica esto significa que debes evitar salientes horizontales en tu diseño, o usar chaflanes (bordes inclinados a 45 grados) en lugar de esquinas rectas donde sea posible. Los puentes (bridging) son la excepción: la impresora puede tender un hilo de plástico entre dos puntos separados hasta unos 5-10 cm sin soportes, porque el filamento se estira tenso como un cable. Si tu diseño tiene un techo plano sostenido en dos extremos, a veces sale bien sin soportes; pruébalo antes de asumir que los necesitas.

La orientación de la pieza en la cama lo cambia todo. Las capas son el punto débil: una pieza es mucho más resistente a lo largo de las capas que a través de ellas (las capas se pueden separar como las hojas de un libro si la fuerza las jala en perpendicular). Orienta tu pieza de modo que los esfuerzos principales vayan en el plano de las capas, no cruzándolas. Además, orientar bien reduce o elimina soportes: una pieza en forma de T impresa de pie necesita soportes, acostada no. Antes de laminar, gira el modelo y pregúntate: "¿cuál orientación minimiza voladizos y pone la fuerza en la dirección correcta?". Ese minuto de reflexión vale horas de impresión.

Las tolerancias son el ajuste entre piezas que encajan. La impresora deposita un poco más de material del teórico (los bordes se expanden ligeramente), así que dos piezas diseñadas con

medidas exactas no van a encajar: necesitas holgura. Como punto de partida usa 0.2 mm por lado para ajustes deslizantes (una tapa que entra y sale) y 0.4-0.5 mm para piezas que deben moverse libremente (un eje dentro de un hoyo). Para ajustes a presión (press-fit), diseña el hoyo 0.1-0.2 mm más chico que la pieza que entra y confía en la ligera elasticidad del plástico. Cada impresora y material varían, así que imprime una "prueba de tolerancias" (un modelito con hoyos y pines de varias holguras) una sola vez y anota cuál holgura funciona en TU máquina; ese dato te sirve para todos tus diseños futuros.

Otras reglas de oro rápidas: el grosor mínimo de pared imprimible con boquilla de 0.4 mm es de unos 0.8 mm (dos líneas); paredes más delgadas salen frágiles o ni se imprimen. Los hoyos verticales pequeños (menos de 3 mm) tienden a salir más chicos de lo diseñado o a taparse: diséñalos un poco más grandes o perfóralos después con un taladro. Los textos y detalles en relieve necesitan al menos 0.6 mm de altura para verse bien. Y evita las masas sólidas gigantes: una pieza maciza tarda una eternidad y gasta filamento a lo tonto; con 3-4 paredes y 20-30 por ciento de relleno tienes casi la misma resistencia en una fracción del tiempo.

El error clásico del principiante es diseñar como si la impresora fuera magia ("el programa lo resolverá") y luego culpar a la máquina. El fabricante competente diseña CON la máquina en mente: ángulos amables, orientación inteligente, tolerancias probadas. Cuando domines estas reglas, tus piezas van a salir bien a la primera con una frecuencia que va a sorprender a quienes te rodean.

Ejemplos paso a paso

Ejemplo 1: caja paramétrica con tapa en OpenSCAD

Nuestro primer diseño programado es una cajita con tapa, como las que se usan para guardar tornillos o componentes electrónicos. La gracia del diseño paramétrico es que defines las medidas una sola vez como variables, y si mañana necesitas la caja más grande, cambias dos números y todo se regenera. Copia este código en OpenSCAD, presiona F5 para previsualizar y F6 para renderizar antes de exportar el STL.


```

// Caja parametrica con tapa - Lucas EGA Academy
// Cambia estas variables y toda la pieza se ajusta sola

ancho = 60;          // medida interior en X (mm)
fondo = 40;          // medida interior en Y (mm)
alto = 30;           // altura interior en Z (mm)
pared = 2;           // grosor de las paredes (mm)
holgura = 0.4;       // espacio entre caja y tapa (mm)

module caja() {
    difference() {
        // cuerpo exterior solido
        cube([ancho + pared*2, fondo + pared*2, alto + pared]);
        // vaciado interior (se resta del solido)
        translate([pared, pared, pared])
            cube([ancho, fondo, alto + 1]);
    }
}

module tapa() {
    difference() {
        // tapa exterior
        cube([ancho + pared*2 + holgura*2,
            fondo + pared*2 + holgura*2,
            pared + 4]);
        // rebaje donde entra la caja
        translate([holgura, holgura, -1])
            cube([ancho + pared*2, fondo + pared*2, pared + 5]);
    }
}

// Descomenta lo que quieras ver o exportar:
caja();
// Para imprimir la tapa, comenta la linea de arriba,
// descomenta la de abajo y exporta otro STL:
// translate([0, 0, 0]) tapa();

```

Puntos clave del código: `cube()` crea un cubo sólido, `difference()` resta el segundo objeto del primero (así se hacen los vaciados), y `translate()` mueve objetos en el espacio. La variable `holgura` es el secreto de las piezas que encajan: sin esos 0.4 mm la tapa no entraría porque la impresora deposita un poco más de material del teórico. Buena práctica: imprime primero solo la caja, mide con un vernier cuánto mide realmente, y ajusta la holgura si tu impresora extruye de más o de menos.

Ejemplo 2: engrane paramétrico en OpenSCAD

Los engranes son piezas clásicas de la fabricación digital: se rompen en juguetes y aparatos, y son caros o imposibles de conseguir como repuesto. Este ejemplo usa trigonometría básica para colocar los dientes alrededor de un círculo. No te asustes por las matemáticas: el código las hace por ti.

```
// Engrane parametrico - Lucas EGA Academy

dientes = 12;          // numero de dientes
diametro = 30;         // diametro primitivo (mm)
grosor = 8;            // grosor del engrane (mm)
diametro_eje = 6;      // diametro del hoyo central (mm)

radio = diametro / 2;
paso angular = 360 / dientes;
ancho_diente = (2 * 3.14159 * radio / dientes) * 0.45;

module diente() {
    // cada diente es una cajita en el borde
    translate([radio, 0, 0])
        cube([ancho_diente, ancho_diente, grosor], center = true);
}

module engrane() {
    difference() {
        union() {
            // disco base
            cylinder(h = grosor, r = radio, center = true, $fn = 64);
            // dientes distribuidos en circulo
            for (i = [0 : dientes - 1])
                rotate([0, 0, i * paso angular])
                    diente();
        }
        // hoyo del eje
        cylinder(h = grosor + 2, r = diametro_eje/2, center = true, $fn = 32);
    }
}

engrane();
```

Cómo funciona: `cylinder()` con `\$fn = 64` dibuja un disco liso (más segmentos = más redondo), el `for` coloca cada diente rotado alrededor del centro, y `difference()` abre el hoyo del eje. Prueba cambiar `dientes` a 20 y `diametro` a 40 y observa cómo se regenera todo.

Para que dos engranes engranen entre sí, deben compartir el mismo "módulo" (diámetro dividido entre dientes): si el engrane A tiene 12 dientes y 30 mm de diámetro (módulo 2.5), el engrane B debe tener también módulo 2.5, por ejemplo 20 dientes y 50 mm. Imprímelo en PETG (más resistente al desgaste que el PLA) con 4 paredes y 40 por ciento de relleno.

Ejemplo 3: perfil de laminado como archivo de configuración

Los perfiles del laminador se pueden guardar, compartir y versionar como archivos de texto. Este es un perfil equilibrado para PLA en una impresora FDM típica con boquilla de 0.4 mm, en un formato de configuración sencillo que puedes adaptar a Cura, PrusaSlicer u OrcaSlicer (los nombres de parámetros cambian un poco entre programas, pero los conceptos son los mismos).

```

# Perfil: PLA Equilibrado v1.0 - Lucas EGA Academy
# Boquilla 0.4 mm - Piezas generales

[calidad]
altura_capa = 0.2          # equilibrio entre velocidad y detalle
ancho_linea = 0.42         # un poco mas que la boquilla: mejor adhesión
primera_capa_altura = 0.24 # primera capa mas gruesa: se pega mejor

[paredes]
perimetros = 3             # 3 paredes = piezas resistentes
relleno = 20               # 20% suficiente para la mayoría de piezas
patron_relleno = giroide   # fuerte en todas direcciones, rapido de imprimir
capas_superiores = 4
capas_inferiores = 3

[velocidad]
velocidad_impresion = 60   # mm/s, velocidad segura para PLA
velocidad_primera_capa = 20 # lento al inicio: la base es sagrada
velocidad_pared_externa = 30 # pared externa lenta: mejor acabado

[temperatura]
temp_boquilla = 205        # ajusta con tu torre de temperatura
temp_cama = 60
temp_primera_capa_cama = 65 # cama un poco mas caliente al inicio

[retraccion]
retraccion_distancia = 5   # Bowden: 4-6 mm; directo: 0.5-1.5 mm
retraccion_velocidad = 45
evitar_hilos = si

[soportes]
soportes = automatico      # solo donde el angulo pase de 50 grados
angulo_voladizo = 50
distancia_z_soporte = 0.2  # separación para quitarlos facil

[adherencia]
tipo_adherencia = falda    # cambia a borde si la pieza se despeg

```

Cómo usarlo: crea un perfil nuevo en tu laminador y copia estos valores como punto de partida; luego imprime un cubo de calibración y ajusta. Guarda una copia cada vez que encuentres una combinación ganadora, con nombre descriptivo ("PLA_MarcaX_Equilibrado_v2"). Con el tiempo tendrás una biblioteca de perfiles: uno rápido para prototipos (capa 0.28, relleno 10), uno fino para piezas de exhibición (capa 0.12, relleno

20) y uno resistente para piezas funcionales (capa 0.2, 4 paredes, relleno 40). Esa biblioteca vale oro.

Ejemplo 4: checklist de calibración como script

Este script en Python es tu lista de verificación interactiva: te guía por la calibración completa paso a paso, registra tus mediciones y te dice si cada prueba pasó o necesita ajuste. Guárdalo como `calibrar.py` y ejecútalo con tu impresora enfrente.

```
# Checklist de calibracion - Lucas EGA Academy
# Ejecutar con: python3 calibrar.py

def preguntar(texto):
    return input(texto + " [s/n]: ").strip().lower() == "s"

def medir(texto):
    return float(input(texto + " (mm): ").strip())

print("=== CALIBRACION DE IMPRESORA FDM ===")
reporte = []

# 1. Nivelacion de cama
print("\n[1] NIVELACION DE CAMA")
print("Calienta cama a 60 y boquilla a 200. Mueve la boquilla")
print("a cada esquina y ajusta hasta que una hoja de papel")
print("rose con ligera friccion.")
if preguntar("Cama nivelada en las 4 esquinas y el centro"):
    reporte.append("Nivelacion: OK")
else:
    reporte.append("Nivelacion: REPETIR - ajusta resortes/tornillos")

# 2. Pasos del extrusor
print("\n[2] PASOS DEL EXTRUSOR (E-steps)")
print("Marca 120 mm de filamento desde la entrada del extrusor.")
print("Pide extruir 100 mm y mide cuanto avanza realmente.")
avance = medir("Cuanto filamento avanza realmente")
if 99 <= avance <= 101:
    reporte.append("E-steps: OK (%.1f mm)" % avance)
else:
    pasos = 93 * 100 / avance # 93 = valor tipico inicial
    reporte.append("E-steps: AJUSTAR a %.1f pasos/mm" % pasos)

# 3. Primera capa
print("\n[3] PRIMERA CAPA")
print("Imprime un cuadrado de 60x60 mm de una sola capa.")
```

```

grosor = medir("Grosor medido de la capa (esperado: 0.2)")
if 0.18 <= grosor <= 0.24:
    reporte.append("Primera capa: OK (%.2f mm)" % grosor)
else:
    reporte.append("Primera capa: AJUSTAR altura Z o nivelacion")

# 4. Cubo de calibracion
print("\n[4] CUBO DE 20 mm")
print("Imprime el cubo de calibracion y midelo con vernier.")
for eje in ["X", "Y", "Z"]:
    m = medir("Medida del eje " + eje + " (esperado: 20.0)")
    estado = "OK" if 19.7 <= m <= 20.3 else "REVISAR"
    reporte.append("Cubo eje %s: %.2f mm -> %s" % (eje, m, estado))

# 5. Torre de temperatura
print("\n[5] TORRE DE TEMPERATURA")
if preguntar("La torre muestra capas limpias sin hilos ni grumos"):
    reporte.append("Temperatura: OK")
else:
    reporte.append("Temperatura: PROBAR otro rango de 5 grados")

print("\n=== REPORTE FINAL ===")
for linea in reporte:
    print("- " + linea)
print("\nGuarda este reporte con la fecha. Si todo dice OK,")
print("tu impresora esta lista para imprimir en serio.")

```

La idea del script es convertir la calibración en un ritual repetible en vez de un acto de fe. Ejecútalo cada vez que cambies de boquilla, de material o después de mover la impresora de lugar. Los valores que registra (E-steps, medidas del cubo) son la "huella digital" de tu máquina: si un día todo sale mal, comparas con tu último reporte bueno y sabes exactamente qué cambió.

Práctica

Ejercicio 1: tu primer llavero paramétrico en OpenSCAD

Diseña un llavero rectangular con tu inicial perforada (o un círculo para la argolla), usando variables para el tamaño. Debe imprimirse sin soportes y en menos de 30 minutos.

Solución paso a paso:

1. Abre OpenSCAD y define las variables: ``largo = 50; ancho = 25; grosor = 4; diametro_argolla = 6;``.
2. Crea el cuerpo con ``cube([largo, ancho, grosor])``. Para que no tenga esquinas filosas, puedes usar ``minkowski()`` con una esferita, pero no es obligatorio para este ejercicio.
3. Haz el hoyo de la argolla con ``difference()``: resta un cilindro (``cylinder(h = grosor + 2, r = diametro_argolla/2, center = true)``) trasladado cerca de un extremo, dejando al menos 4 mm de material alrededor del hoyo para que no se rompa.
4. Agrega tu inicial usando ``text()``: ``translate([10, 5, grosor - 1]) linear_extrude(height = 1.5) text("E", size = 12);`` unido al cuerpo con ``union()``. La letra queda en relieve sobre la superficie.
5. Previsualiza con F5, renderiza con F6, exporta a STL y láminalo: capa 0.2, relleno 20 por ciento, sin soportes, en PLA.
6. Verifica el tiempo estimado en el laminador: debe ser menor a 30 minutos. Si se pasa, reduce el relleno al 15 por ciento o la altura de capa a 0.24.

Lo que aprendiste: variables paramétricas, ``difference()`` para hoyos, ``text()`` y ``linear_extrude()`` para detalles en relieve, y a leer el tiempo estimado del laminador antes de imprimir.

Ejercicio 2: elige el material correcto (casos prácticos)

Para cada caso, elige entre PLA, PETG, ABS y TPU, y justifica tu elección: a) una perilla de repuesto para la estufa (cerca del calor), b) una funda para tu celular, c) un soporte para el celular que va en el tablero del carro bajo el sol, d) una figura decorativa de un dragón para tu escritorio.

Solución paso a paso:

1. Caso a) perilla de estufa: el calor es el factor crítico (la estufa genera más de 60 grados C cerca de las hornillas). El PLA se ablandaría. La respuesta es ABS, porque aguanta hasta unos 100 grados C. Requiere impresora cerrada y buena ventilación.
2. Caso b) funda de celular: necesita flexibilidad y absorber golpes. La respuesta es TPU (dureza 95A), impreso lento a 20-30 mm/s, idealmente con extrusor directo.
3. Caso c) soporte en el tablero del carro: sol directo y calor encerrado (un carro al sol supera los 70 grados C fácil). Ni PLA ni PETG aguantan eso de forma segura a largo plazo. La respuesta es ABS (o ASA, su primo resistente a UV). Impresora cerrada obligatoria.

4. Caso d) figura decorativa: no sufre esfuerzo ni calor, lo que importa es el detalle y que sea fácil de imprimir. La respuesta es PLA: imprime hermoso, barato y sin complicaciones.
5. Regla general que acabas de aplicar: primero identifica la condición más agresiva del caso (calor, impacto, flexibilidad, intemperie) y elige el material que la sobreviva; entre los que sobreviven, elige el más fácil de imprimir.

Lo que aprendiste: a decidir material por requisitos y no por costumbre, y a detectar el factor crítico de cada aplicación.

Ejercicio 3: lamina la caja del Ejemplo 1 para dos usos distintos

Toma el STL de la caja paramétrica del Ejemplo 1 de la sección anterior y crea dos perfiles: Perfil A "prototipo rápido" (terminar lo antes posible para verificar medidas) y Perfil B "pieza final resistente" (la caja se usará a diario para guardar tornillos pesados).

Solución paso a paso:

1. Perfil A (prototipo rápido): altura de capa 0.28 mm (menos capas = menos tiempo), relleno 10 por ciento en patrón de líneas, 2 perímetros, velocidad 80 mm/s, sin soportes (la caja no los necesita). La pieza saldrá con líneas visibles y menos fuerte, pero en la mitad de tiempo, perfecta para verificar que las medidas y la tapa encajan.
2. Perfil B (pieza final): altura de capa 0.2 mm (buen equilibrio), relleno 30 por ciento en patrón giroide, 4 perímetros (las paredes dan más resistencia que el relleno), 4 capas superiores e inferiores, velocidad 50 mm/s para mejor calidad. Tardará más del doble que el perfil A, pero aguantará el uso diario.
3. En ambos perfiles: primera capa a 20 mm/s y cama a 60 grados para PLA, porque la adherencia no se negocia.
4. Lamina la misma pieza con ambos perfiles y compara en el laminador: tiempo estimado, gramos de filamento y número de capas. Anota las diferencias: verás que el perfil B usa notablemente más material y tiempo.
5. Imprime primero el perfil A, verifica que la tapa encaja con la holgura correcta, y solo entonces imprime el perfil B. Así no desperdicias horas en una pieza final con medidas equivocadas.

Lo que aprendiste: que no existe "el perfil perfecto", sino el perfil adecuado para cada objetivo, y el hábito profesional de prototipar rápido antes de imprimir la versión final.

Ejercicio 4: diagnostica estas tres fallas con foto mental

Describe el diagnóstico y la solución para: a) a mitad de una impresión de 6 horas encuentras un nido de "espagueti" plástico y la pieza a medio hacer despegada de la cama; b) las esquinas de tu pieza de ABS se levantaron de la cama y la base quedó curva; c) entre las torres de tu pieza hay decenas de hilos finos como telaraña.

Solución paso a paso:

1. Caso a) espagueti por despegue: diagnóstico = falla de adherencia de la primera capa (cama mal nivelada, superficie sucia con grasa de dedos, o cama fría). Solución inmediata: cancela, limpia la cama con alcohol isopropílico, renivela con cama caliente, y reimprime usando borde (brim) o laca como ayuda de adherencia. Prevención: nunca toques la superficie de impresión con los dedos y observa siempre las primeras capas.
2. Caso b) warping en ABS: diagnóstico = contracción térmica; el ABS se encoge al enfriarse y las esquinas se despegan porque no hay cámara cerrada o la cama está fría. Solución: sube la cama a 100-110 grados, evita corrientes de aire (cierra puertas y ventanas), y lo ideal: imprime dentro de una impresora cerrada o una caja improvisada que mantenga el aire caliente. Alternativa pragmática: si no tienes impresora cerrada, cambia la pieza a PETG, que casi no sufre warping y aguanta bastante bien.
3. Caso c) stringing (hilos): diagnóstico = el filamento gotea durante los movimientos en vacío; típico de PETG o temperatura alta. Solución: en el laminador aumenta la distancia de retracción (prueba subir 1 mm a la vez), sube la velocidad de retracción, y baja la temperatura de la boquilla 5 grados. Verifica con una prueba pequeña (dos torrecitas separadas) antes de reimprimir la pieza grande.
4. Paso final común a los tres casos: anota en tu diario de impresión qué falló, qué cambiaste y si funcionó. La próxima vez que veas el mismo síntoma, tu yo del futuro te lo va a agradecer.

Lo que aprendiste: el método de diagnóstico (síntoma -> causa probable -> solución -> prevención) que usan los técnicos, aplicado a las tres fallas más comunes.

Ejercicio 5: diseña un soporte paramétrico para tu celular en OpenSCAD

Diseña un soporte de escritorio para celular con ángulo ajustable por variable. Debe sostenerse solo, sin pegamento, y el celular no debe resbalarse.

Solución paso a paso:

1. Define variables: ``ancho_cel = 75; grosor_cel = 10; angulo = 65; base_largo = 90;``. El ángulo de 65 grados respecto a la base es cómodo para ver la pantalla.
2. Crea la base: ``cube([base_largo, ancho_cel + 20, 4])``, una placa de 4 mm de grosor.
3. Crea el respaldo inclinado: un ``cube([4, ancho_cel + 20, 90])`` rotado con ``rotate([0, -angulo, 0])`` y posicionado con ``translate()`` de modo que su borde inferior toque la base a unos 25 mm del borde trasero. Aquí tendrás que jugar con los números y previsualizar: es normal ajustar 3 o 4 veces.
4. Agrega un tope frontal para que el celular no resbale: ``translate([5, 0, 0]) cube([6, ancho_cel + 20, 12])``, una cejita de 12 mm de alto cerca del borde delantero.
5. Agrega un hoyo o ranura para el cable del cargador en el tope: resta con ``difference()`` un ``cube([8, 15, 6])`` centrado en el tope. Detalle profesional que se agradece a diario.
6. Une todo con ``union()``, renderiza, exporta y lamina en PETG (más resistente que el PLA para algo que se manipula a diario), 3 paredes, 25 por ciento de relleno, sin soportes si orientas bien las piezas (imprime el respaldo acostado si el ángulo genera voladizos imposibles).
7. Prueba de fuego: cambia ``ancho_cel`` a 85 y ``angulo`` a 55, regenera y verifica que todo sigue encajando. Si sí, tu diseño es verdaderamente paramétrico.

Lo que aprendiste: ``rotate()`` para piezas inclinadas, a diseñar pensando en la orientación de impresión (evitar soportes), y a validar que un diseño paramétrico realmente responde a sus variables.

Ejercicio 6: monta tu microfábrica conectada (plan por etapas)

Elabora un plan por etapas para convertir una impresora FDM básica en una estación conectada con monitoreo remoto y detección de fallas, con presupuesto estimado en dólares y sin comprar una impresora nueva.

Solución paso a paso:

1. Etapa 1 - control por red (unos 45 dólares): compra una Raspberry Pi (o reutiliza una que tengas) e instala OctoPrint. Conecta la impresora por USB a la Pi y la Pi a tu WiFi. Resultado: mandas archivos sin tarjeta SD, controlas temperatura y avance desde el navegador o el celular, y recibes notificaciones.
2. Etapa 2 - vigilancia visual (unos 25 dólares): agrega una cámara web USB barata apuntando a la cama, configurada en OctoPrint. Instala un complemento de detección de espagueti (como OctoEverywhere o el detector de fallas de tu elección), que analiza la

imagen y pausa la impresión si detecta desastre. Resultado: puedes salir de casa tranquilo durante impresiones largas.

3. Etapa 3 - seguridad y automatización (unos 20 dólares): un enchufe inteligente WiFi para apagar la impresora a distancia o por horario, un sensor de filamento (muchas impresoras ya lo traen; si no, hay kits baratos) que pausa al acabarse el rollo, y un detector de humo en la habitación. Resultado: apagas todo desde el celular y reduces riesgos.
4. Etapa 4 - cola automática (gratis, solo tu tiempo): escribes un script que vigile una carpeta, lamine automáticamente los STL nuevos con tu perfil guardado usando el laminador en modo línea de comandos, y los ponga en la cola de OctoPrint. Resultado: dejas 5 piezas en la carpeta en la noche y despiertas con 5 piezas impresas.
5. Presupuesto total estimado: 90 dólares más una tarde de configuración por etapa. Documenta cada etapa (fotos, ajustes, contraseñas) igual que documentas tus perfiles de laminado.
6. Reflexión final: con esto ya tienes una microfábrica: diseño paramétrico (OpenSCAD) + laminado automatizado + impresión monitoreada por IA + notificaciones. Eso que hace diez años requería un departamento de ingeniería, hoy cabe en tu escritorio por menos de 100 dólares extra.

Lo que aprendiste: a planear mejoras por etapas con costo-beneficio, y a ver tu impresora como un sistema conectable (IoT) y no como un aparato aislado.

Resumen de la unidad

- La impresión 3D es manufactura aditiva: construye objetos capa por capa desde un modelo digital, sin moldes. Las tres tecnologías principales son FDM (plástico fundido, barata y versátil, la ideal para empezar), SLA (resina curada con luz UV, gran detalle pero más costosa y delicada) y SLS (polvo sinterizado con láser, piezas fuertes de grado industrial, normalmente vía servicios en línea).
- Una impresora FDM tiene marco, ejes X/Y/Z con motores paso a paso, extrusor (Bowden o directo), hotend con boquilla, cama calefactable, y electrónica con firmware que lee G-code. La nivelación de la cama es el ajuste más importante de toda la máquina.
- Materiales: PLA para empezar y piezas decorativas (fácil, pero no aguanta calor); PETG como todoterreno funcional (resistente, para exterior y piezas de uso diario); ABS para calor e impactos (fuerte, pero exige impresora cerrada y ventilación); TPU para piezas flexibles (imprimir lento y con extrusor directo). El filamento húmedo arruina impresiones: guárdalo sellado con sílica.

- El flujo de trabajo es diseño (descargar, Tinkercad, Fusion 360 u OpenSCAD) -> laminado (el slicer genera el G-code con altura de capa, relleno, paredes, soportes y temperaturas) -> impresión (vigila las primeras capas) -> post-procesado opcional (quitar soportes, lijar, pintar).
- El diseño paramétrico con OpenSCAD te permite programar piezas: variables, ``cube()``, ``cylinder()``, ``difference()``, ``union()``, ``translate()``, ``rotate()`` y ``for`` son todo lo que necesitas para empezar a crear engranes, cajas y soportes a medida.
- En el laminador no hay ajustes universales: 0.2 mm de capa, 3 paredes y 20 por ciento de relleno son un buen punto de partida; calibra con torre de temperatura y cubo de 20 mm antes de piezas largas, y guarda tus perfiles ganadores.
- Las fallas son normales: espagueti por mala adherencia (nivela y limpia la cama), warping por contracción (cama caliente, sin corrientes, impresora cerrada para ABS), stringing por goteo (más retracción, menos temperatura), atascos por filamento húmedo o boquilla sucia. Lleva un diario de impresión.
- Aplicaciones reales: prótesis accesibles, repuestos que salvan aparatos de la basura, prototipado rápido industrial y microemprendimientos de manufactura local donde la ventaja es la personalización, no el precio.
- El futuro es la microfábrica conectada: OctoPrint + Raspberry Pi + cámara con detección de fallas por IA + automatización, todo por menos de 100 dólares extra. Diseños que viajan por internet y objetos que se materializan localmente: manufactura distribuida.
- Regla de oro de la unidad: empieza simple (imprime modelos descargados y calibra), sube de nivel por etapas (diseña tus piezas, domina el laminador) y documenta todo (perfiles, reportes de calibración, diario). La fabricación digital se construye por capas, igual que tus impresiones.

UNIDAD 7: Energías renovables y tecnología

Imagina que cada mañana, sin que muevas un dedo, el sol recarga tu teléfono, enciende tu refrigerador y mantiene fría tu casa sin que pagues un solo peso por esa electricidad. Suena como ciencia ficción, pero es exactamente lo que ya hacen miles de hogares en México y en todo el mundo con energía solar. La energía renovable dejó de ser un lujo de científicos o de millonarios: hoy es una tecnología accesible, rentable y, sobre todo, necesaria.

En esta unidad vas a aprender todo lo que necesitas para entender y aprovechar las energías renovables en tu casa, tu negocio o tus proyectos. No vamos a quedarnos en la teoría abstracta: vas a calcular el consumo eléctrico de tu hogar paso a paso, vas a dimensionar un sistema solar básico como lo haría un técnico profesional, vas a programar en Python un calculador de dimensionamiento y hasta vas a conectar un ESP32 con MicroPython para monitorear tu propio panel solar en tiempo real. Al final tendrás las bases técnicas y prácticas para tomar decisiones reales: qué comprar, cuánto te va a costar, cuándo recuperas tu inversión y cómo mantener tu sistema funcionando por décadas.

Introducción

Por qué las energías renovables son el futuro (y el presente)

Piensa en esto: cada hora, el sol envía a la Tierra más energía que toda la electricidad que la humanidad consume en un año completo. Más energía en sesenta minutos que la que usamos en doce meses. Esa cifra, que parece imposible, es real y está verificada por agencias internacionales de energía. El recurso está ahí, es gratis, no se agota y no contamina. La única pregunta interesante es: por qué tardamos tanto en aprovecharlo.

La respuesta tiene historia. Durante más de un siglo, el mundo construyó su economía sobre combustibles fósiles: petróleo, carbón y gas natural. Eran densos en energía, fáciles de transportar y baratos de extraer. Pero traían tres problemas gigantescos que hoy ya no podemos ignorar. Primero, se agotan: cada litro de gasolina que quemas es un litro menos que existirá mañana. Segundo, contaminan: la quema de fósiles es la principal causa del cambio climático, con tormentas más fuertes, sequías más largas y calores extremos que ya se sienten en las ciudades mexicanas. Tercero, el precio: la gasolina y la electricidad de origen fósil suben y bajan al ritmo de guerras, crisis y especulación internacional. Depender de los fósiles es depender de algo que no controlas.

Las energías renovables resuelven los tres problemas de raíz. El sol, el viento, el agua y el calor de la tierra son recursos que se renuevan solos todos los días. No hay que importarlos, no hay que quemarlos, y nadie puede cortarte el suministro desde otro continente. Además, y esto es lo que más te va a interesar en esta unidad, el precio de la tecnología solar se desplomó en la última década: un panel solar hoy cuesta una fracción de lo que costaba en 2010, mientras que su eficiencia se duplicó. Instalar energía solar en casa pasó de ser un experimento de ricos a una inversión que se paga sola en pocos años.

En México esto tiene un atractivo especial. Nuestro país es uno de los más soleados del planeta: la mayor parte del territorio recibe entre 5 y 7 horas pico de sol al día, un nivel que muchos países europeos ni sueñan. Eso significa que un sistema solar en México produce más electricidad con menos paneles que el mismo sistema instalado en Alemania o Inglaterra. Y aun así, Alemania es líder mundial en solar. Imagina el potencial que tenemos aquí si lo aprovechamos.

Pero ojo: esta unidad no es un folleto de ventas de paneles solares. Vas a aprender la ingeniería detrás de la tecnología, con números reales, cálculos paso a paso y herramientas de programación que te permitan verificar por ti mismo si una instalación vale la pena, cuánto necesitas y cómo sacarle el máximo provecho. Porque la energía renovable no es magia: es física, electrónica y datos. Y los datos, cuando sabes leerlos, te dan poder.

Qué vas a construir en esta unidad

Olvídate de memorizar definiciones. Esta unidad está diseñada para que termines con tres cosas concretas en las manos: conocimiento, herramientas y proyectos.

Primero, el conocimiento. Vas a entender cómo funciona un panel solar por dentro (el efecto fotovoltaico explicado sin fórmulas intimidantes), qué hace cada componente de un sistema (paneles, inversor, baterías, controlador de carga), cómo se mide la electricidad que consumes y generas, y qué significan términos como kilovatio-hora, eficiencia y retorno de inversión. Todo con analogías de la vida cotidiana: tuberías de agua para la corriente, cubetas para las baterías, y peajes de carretera para los inversores.

Segundo, las herramientas. Vas a aprender a hacer una hoja de cálculo de consumo eléctrico de tu casa, paso a paso, como la hacen los instaladores profesionales antes de cotizar. Vas a escribir un script en Python que dimensione automáticamente un sistema solar: le das tu consumo mensual y te dice cuántos paneles, qué inversor y qué baterías necesitas. Y vas a programar un ESP32 con MicroPython para medir voltaje y corriente de un panel en tiempo real y subir los datos a tu teléfono o computadora. Esas herramientas no son ejercicios de juguete: son las que usan los técnicos de verdad, simplificadas para que las domines desde cero.

Tercero, los proyectos. Al final de la unidad encontrarás proyectos guiados que puedes armar con componentes baratos: un cargador solar USB portátil, un sistema de respaldo para emergencias (para cuando se va la luz en plena tormenta) y un huerto con riego automatizado alimentado por energía solar. Cada proyecto está pensado para que lo puedas hacer en un fin de semana con presupuesto de estudiante, y cada uno te enseña un principio distinto: conversión de energía, almacenamiento y automatización.

Cómo está organizada esta unidad

La unidad tiene cuatro secciones grandes, y conviene que las recorras en orden porque cada una construye sobre la anterior.

En "Explicación detallada" vamos a desmenuzar los conceptos: el efecto fotovoltaico, los componentes del sistema solar, la energía eólica a pequeña escala, el monitoreo con IoT, la eficiencia energética con domótica, y los números que mandan: costos, retorno de inversión y mantenimiento. Aquí también vas a encontrar las analogías cotidianas, los errores típicos de principiantes (para que no los cometas tú) y las buenas prácticas que separan una instalación decente de una excelente.

En "Ejemplos paso a paso" pasamos a la acción: construirás la hoja de cálculo de consumo de tu casa como una tabla de texto que podrás replicar en papel o en Excel, escribirás el script Python de dimensionamiento solar línea por línea con explicación, y programarás el código MicroPython para que tu ESP32 mida un panel solar real.

En "Práctica" tienes seis ejercicios numerados con su solución explicada paso a paso. No son preguntas de opción múltiple: son problemas reales de cálculo y diseño que te obligan a usar lo aprendido. Haz el intento serio antes de leer la solución; la diferencia entre leer y resolver es la diferencia entre saber y poder.

Finalmente, en el "Resumen de la unidad" encontrarás los puntos clave condensados: tu chuleta de referencia rápida para cuando necesites recordar un concepto o un número sin releer todo.

Una nota sobre las unidades de medida: en esta unidad vas a ver mucho "W" (vatios, potencia), "Wh" y "kWh" (vatios-hora y kilovatios-hora, energía). La distinción entre potencia y energía es el concepto más importante de toda la unidad, así que lo repetiremos hasta que salga natural: la potencia es la velocidad a la que usas energía, y la energía es la cantidad total. Es como la diferencia entre la velocidad de tu carro (km/h) y la distancia recorrida (km). Guárdate esa analogía porque la vamos a usar muchísimo.

También verás "V" (voltios), "A" (amperios) y "Ah" (amperios-hora, capacidad de baterías). Nada de esto requiere matemáticas avanzadas: con sumas, multiplicaciones y divisiones tienes más que suficiente. La física la ponemos nosotros; tú pones las ganas.

Empecemos por lo fundamental: cómo un pedazo de vidrio oscuro convierte la luz del sol en la electricidad que carga tu celular.

Explicación detallada

Cómo funciona un panel solar: el efecto fotovoltaico sin miedo

Un panel solar parece una ventana oscura y misteriosa, pero por dentro es sorprendentemente simple. Está hecho de celdas de silicio, el mismo material de los chips de tu computadora, tratado de una forma especial. Para entender qué pasa ahí dentro, usa esta analogía: imagina una cancha de fútbol llena de pelotas (los electrones) quietas. De pronto cae una lluvia de pelotas de tenis lanzadas desde las gradas (los fotones, las partículas de luz del sol). Cada pelota de tenis golpea una pelota de fútbol y la pone en movimiento. Si la cancha está inclinada de forma inteligente, todas las pelotas de fútbol ruedan hacia la misma portería: ese flujo ordenado de pelotas es la corriente eléctrica.

Eso, en esencia, es el efecto fotovoltaico: la luz golpea el silicio, libera electrones, y el diseño de la celda los canaliza en una sola dirección para formar corriente continua. No hay partes móviles, no hay combustión, no hay ruido. Solo luz entrando y electrones saliendo en fila.

Ahora, un detalle técnico que sí vale la pena entender. El silicio de las celdas viene en dos "sabores": tipo N (con electrones de sobra) y tipo P (con huecos donde caben electrones). Cuando los pegas forman una unión que crea un campo eléctrico interno, como una rampa invisible. Los fotones liberan electrones, la rampa los empuja en una dirección, y listo: tienes corriente continua (DC). Una sola celda produce alrededor de 0.5 voltios, así que los paneles conectan 60 o 72 celdas en serie para llegar a los 30-40 voltios que entrega un panel comercial típico.

Hay tres tecnologías principales de paneles que vas a encontrar en el mercado. Los monocristalinos (los de color negro uniforme) son los más eficientes, entre 19 y 23 por ciento de la luz la convierten en electricidad, y también los más caros. Los policristalinos (azul con textura de mosaico) son un poco menos eficientes, 15 a 18 por ciento, pero más baratos. Los de película delgada son flexibles y ligeros, ideales para mochilas solares o techos curvos, pero su eficiencia ronda el 10-12 por ciento y necesitas más superficie para la misma potencia. Para un techo de casa, los monocristalinos son hoy la mejor relación calidad-precio.

La eficiencia de un panel se mide en condiciones estándar de laboratorio: 1000 W/m² de irradiación y 25 grados C de temperatura de celda. En la vida real, tu panel se calienta al sol (una celda puede llegar a 60 o 70 grados C en verano) y cada grado por encima de 25 le resta un poco de potencia: típicamente pierdes entre 0.3 y 0.4 por ciento por cada grado extra. Por eso los paneles rinden un poco menos al mediodía de agosto que en una mañana fresca de octubre, aunque haya más sol. Es contraintuitivo pero real: el calor es enemigo del panel solar.

Error común de principiante: creer que un panel de "550 W" produce 550 vatios todo el día. Esa cifra es la potencia pico, el máximo en condiciones ideales. Lo que importa para tus cálculos es la energía diaria, que depende de las horas pico de sol de tu zona. Un panel de 550 W en una zona con 5.5 horas pico genera unos 3 kWh al día ($550 \times 5.5 / 1000 = 3.025$ kWh, menos pérdidas). Más adelante haremos este cálculo con lupa.

Buena práctica: cuando compares paneles, no mires solo el precio por panel, mira el precio por vatio pico y la garantía. Un panel decente trae 25 años de garantía de producción y 10-12 años de garantía de producto. Si el vendedor no te da ficha técnica con esos datos, busca otro vendedor.

Los cuatro componentes de un sistema solar: el equipo completo

Un panel solo no hace un sistema. Para llevar la electricidad del sol a tus enchufes necesitas cuatro componentes trabajando en equipo. Piensa en ellos como una cadena de producción: cada uno transforma o guarda la energía para el siguiente.

1. Los paneles solares son los generadores. Captan la luz y producen corriente continua (DC). Se conectan en serie (para sumar voltaje) o en paralelo (para sumar corriente) formando "arreglos". La analogía: son como los obreros que cosechan la energía.
2. El controlador de carga es el vigilante de las baterías. Regula el voltaje y la corriente que llega a las baterías para no sobrecargarlas ni dañarlas. Hay dos tipos: PWM (más barato, menos eficiente) y MPPT (más caro, hasta 30 por ciento más eficiente porque busca siempre el punto de máxima potencia del panel). La analogía: es como el guardia de un estacionamiento que decide cuántos carros entran y a qué velocidad para que nadie

choque. Buena práctica: si tu sistema es de más de 200 W, usa MPPT; la diferencia de precio se paga sola con la energía extra que cosechas.

3. Las baterías son el almacén. Guardan la energía del día para usarla de noche o cuando está nublado. Se miden en amperios-hora (Ah) y en voltaje (12, 24 o 48 V típicamente). Hay tres químicas principales: plomo-ácido (baratas, pesadas, duran 3-5 años, solo puedes usar el 50 por ciento de su capacidad sin dañarlas), AGM/gel (versión mejorada del plomo, sin mantenimiento) y litio LiFePO₄ (caras al inicio, ligeras, duran 10+ años, puedes usar el 80-90 por ciento de su capacidad). La analogía: son cubetas donde guardas agua (energía) para cuando no llueve (no hay sol). Error común: comprar baterías de carro para un sistema solar. Las baterías automotrices están diseñadas para dar un golpe fuerte de corriente al arrancar y luego recargarse; un sistema solar las descarga lento y profundo todos los días, y las mata en meses. Usa baterías de ciclo profundo, diseñadas para eso.
4. El inversor es el traductor. Convierte la corriente continua (DC) de paneles y baterías en corriente alterna (AC) de 127 V, que es la que usan tus aparatos y la que sale de los enchufes de tu casa. Hay inversores de onda modificada (baratos, pero pueden dañar aparatos sensibles como computadoras o equipos de audio) y de onda pura o senoidal (más caros, compatibles con todo). La analogía: es como un intérprete que traduce del idioma de los paneles al idioma de tu casa. Buena práctica: compra inversor de onda pura siempre que vayas a conectar electrónica; el ahorro del inversor barato te puede costar una laptop.

Además de los cuatro grandes, un sistema real lleva protecciones: fusibles o breakers DC entre componentes, un desconectador, cableado del calibre correcto (cables delgados se calientan y pierden energía, e incluso pueden ser riesgo de incendio) y una estructura de montaje que incline los paneles al ángulo óptimo. En México, el ángulo ideal suele ser igual a tu latitud: si vives a 20 grados de latitud norte, inclina los paneles unos 20 grados hacia el sur. Esa inclinación también ayuda a que la lluvia los limpie sola.

Potencia vs energía: el concepto que lo ordena todo

Si solo te quedas con una idea técnica de esta unidad, que sea esta: la potencia (W) es velocidad, la energía (Wh o kWh) es distancia. Un foco de 60 W consume energía a una "velocidad" de 60 vatios; si lo dejas encendido 5 horas, recorrió una "distancia" de 300 Wh, o sea 0.3 kWh. La fórmula es simple: Energía = Potencia x Tiempo.

Tu recibo de luz no te cobra por la potencia de tus aparatos, te cobra por la energía total que consumiste en el bimestre, medida en kWh. Por eso un aparato muy potente que usas poco tiempo puede costarte menos que uno modesto que nunca apagas. Ejemplo: una plancha de

1000 W usada 1 hora al día consume 1 kWh diario. Un módem de internet de 10 W encendido las 24 horas consume 0.24 kWh diario. La plancha "corre" cien veces más rápido, pero el módem nunca se detiene. Haz la cuenta del mes: plancha 30 kWh, módem 7.2 kWh. ¿Ves cómo el tiempo manda?

Esta distinción es crítica para dimensionar tu sistema solar. Los paneles se compran por potencia (W pico), pero tu necesidad se mide en energía diaria (kWh/día). El puente entre ambas es el número de horas pico de sol (HSP) de tu zona: la cantidad de horas al día en que el sol brilla con intensidad equivalente a 1000 W/m². En la mayor parte de México, las HSP están entre 4.5 y 6.5 según la región y la época del año. La fórmula de dimensionamiento básica es: Potencia de paneles necesaria (W) = Energía diaria necesaria (Wh) / HSP / factor de pérdidas. El factor de pérdidas (típicamente 0.75 a 0.85) considera que los cables, el inversor, el polvo y el calor se comen una parte de la energía. Más adelante lo calcularemos con números reales y luego lo automatizaremos con Python.

Error común de principiante: sumar las potencias de todos tus aparatos y comprar paneles por esa suma. Eso dimensiona para el peor instante posible (todo encendido a la vez), no para tu consumo real, y terminas comprando el doble de paneles que necesitas. Lo correcto es sumar energías diarias (potencia x horas de uso de cada aparato), que es justo lo que hace la hoja de cálculo de consumo que construiremos en la sección de ejemplos.

Buena práctica: antes de comprar un solo panel, mide tu consumo real durante una semana con un medidor de enchufe (wattmeter, cuestan unos pocos cientos de pesos). Los datos reales siempre sorprenden: casi todos descubren un "vampiro" eléctrico, un aparato que consume mucho sin que lo noten. El culpable típico en México: el refrigerador viejo. Un refri de más de 15 años puede consumir 3 o 4 kWh al día él solito, más que todo lo demás junto. A veces la inversión más rentable no es un panel, sino cambiar el refri.

Cálculo de consumo eléctrico de tu casa, paso a paso

Dimensionar un sistema solar sin conocer tu consumo es como comprar zapatos sin saber tu talla. Este es el procedimiento profesional, simplificado en cinco pasos.

Paso 1: haz el inventario. Recorre tu casa cuarto por cuarto y anota cada aparato eléctrico: focos, refrigerador, televisor, ventiladores, lavadora, microondas, computadora, cargadores, bomba de agua, aire acondicionado, todo. No olvides los que están siempre conectados: módem, decodificadores, bocinas inteligentes.

Paso 2: anota la potencia de cada uno. Viene en una etiqueta o placa del aparato, en vatios (W). Si dice amperios (A), multiplícalos por 127 V para obtener vatios aproximados. Si no encuentras el dato, usa valores típicos: foco LED 9 W, televisor LED de 50 pulgadas 100 W,

ventilador 70 W, laptop 65 W, refrigerador moderno 150 W (pero ojo, no trabaja todo el tiempo).

Paso 3: estima las horas de uso diario de cada aparato. Sé honesto: el televisor no está 8 horas encendido aunque esté 8 horas "disponible". Pregunta en casa, observa una semana normal.

Paso 4: calcula la energía diaria de cada aparato: potencia (W) x horas = Wh. Súmalos todos y divide entre 1000 para pasar a kWh. Ese total es tu consumo diario promedio.

Paso 5: agrega un margen de seguridad del 20 al 25 por ciento. Los consumos reales siempre superan las estimaciones (visitas, días de calor, olvidos), y dimensionar justo te deja sin margen en días nublados.

Ejemplo rápido con una casa modesta: 8 focos LED de 9 W x 5 h = 360 Wh; refrigerador 150 W x 8 h efectivas = 1200 Wh (el compresor no trabaja continuo); televisor 100 W x 4 h = 400 Wh; 2 ventiladores 70 W x 6 h = 840 Wh; laptop 65 W x 5 h = 325 Wh; lavadora 500 W x 1 h = 500 Wh; módem 10 W x 24 h = 240 Wh; cargadores y varios 200 Wh. Total: 4065 Wh, o sea 4.07 kWh/día. Con margen del 25 por ciento: 5.08 kWh/día. Ese número, 5 kWh al día, es tu "talla de zapato": todo el dimensionamiento parte de ahí.

Analogía cotidiana: este inventario es como hacer la lista del súper antes de ir a la tienda. Puedes ir sin lista y comprar al tanteo, pero regresas con cosas de más, cosas de menos, y gastas de más. La hoja de cálculo es tu lista.

Error común: olvidar los picos de arranque. Motores como los del refrigerador, la bomba de agua o el aire acondicionado consumen al arrancar entre 3 y 7 veces su potencia nominal durante unos segundos. Tu inversor debe soportar ese pico aunque tu cálculo de energía diaria no lo refleje. Un refri de 150 W puede pedir 600 W al arrancar; si tu inversor es de 500 W, se protegerá y apagará justo cuando más lo necesitas. Buena práctica: suma los picos de arranque de tus motores y elige un inversor con al menos 25 por ciento de holgura sobre ese pico.

Dimensionar un sistema solar básico: del consumo a la lista de compras

Ya tienes tu consumo diario con margen, digamos 5 kWh/día. Ahora convertimos ese número en una lista de compras. Vamos paso a paso con la zona centro de México como referencia (5.5 HSP) y factor de pérdidas de 0.8.

Paso 1: potencia de paneles. Fórmula: Wh diarios / HSP / 0.8. Con 5000 Wh: $5000 / 5.5 / 0.8 = 1136$ W. Redondeas hacia arriba al tamaño comercial: 3 paneles de 450 W (1350 W) o 2 de 550 W (1100 W, un poco justo) o 3 de 550 W (1650 W, con holgura para crecer).

Recomendación práctica: 3 paneles de 550 W te dan margen para días nublados y para conectar más cosas después. Recuerda: es más barato comprar un panel de más hoy que agregar uno después con otra estructura.

Paso 2: baterías. Decide tu autonomía: cuántas noches o días nublados quieres aguantar sin sol. Para un sistema de respaldo básico, 1 día de autonomía es lo normal. Energía a almacenar: 5000 Wh. Si usas baterías de litio de 12.8 V y 100 Ah, cada una guarda 1280 Wh (12.8×100), y puedes usar el 90 por ciento: 1152 Wh útiles. Necesitas $5000 / 1152 = 4.3$, o sea 5 baterías. Con plomo-ácido de 12 V y 200 Ah (2400 Wh nominales, 50 por ciento utilizable = 1200 Wh útiles) necesitarías $5000 / 1200 = 4.2$, o sea 5 baterías también, pero más pesadas y con menos vida útil. Por eso, aunque el litio cuesta más al inicio, suele salir más barato por kWh almacenado a lo largo de su vida.

Paso 3: controlador de carga. Debe soportar la corriente que entregan los paneles. Con 3 paneles de 550 W en un sistema de 48 V: corriente = $1650 \text{ W} / 48 \text{ V} = 34 \text{ A}$. Eliges un MPPT de 40 A o más (siempre redondea hacia arriba y deja holgura). Si tu sistema es de 12 V, la corriente sería mucho mayor (137 A), lo que exige cables gruesísimos y controladores caros: por eso los sistemas medianos y grandes se arman a 24 o 48 V. Buena práctica: a partir de 1 kW de paneles, trabaja a 48 V.

Paso 4: inversor. Suma las potencias de los aparatos que pueden estar encendidos al mismo tiempo (no todos los de la casa, los simultáneos realistas) y agrega el pico de arranque del motor más grande. Ejemplo: refri 150 W + TV 100 W + ventiladores 140 W + focos 72 W + laptop 65 W = 527 W simultáneos; pico de arranque del refri 600 W. Un inversor de onda pura de 1500 W a 48 V te cubre con holgura y deja espacio para crecer. Nunca compres el inversor "justo": operarlo siempre al límite lo calienta, lo desgasta y dispara sus protecciones.

Paso 5: cableado y protecciones. Cable del calibre adecuado según corriente y distancia (hay tablas AWG; para 30 A a pocos metros, calibre 10 AWG es un punto de partida común), fusibles DC en cada tramo, y puesta a tierra. Este paso no se improvisa: un cable delgado con mucha corriente se calienta, pierde energía y es riesgo de incendio.

Lista de compras resultante para nuestro ejemplo: 3 paneles monocristalinos de 550 W, 1 controlador MPPT de 40 A (48 V), 5 baterías LiFePO4 de 12.8 V/100 Ah (o su equivalente en un banco de 48 V), 1 inversor de onda pura de 1500 W a 48 V, estructura de montaje, cableado, fusibles y conectores MC4. Con esa lista ya puedes pedir cotizaciones comparables a tres proveedores y detectar al que te quiere vender de más.

Error común de principiante: dimensionar con el consumo del recibo de luz sin desglosar. El recibo te da el promedio bimestral, pero no te dice a qué horas consumes ni qué aparatos son los culpables. Dos casas con el mismo recibo pueden necesitar sistemas muy distintos si una

consume de día (ideal para solar directo) y otra consume de noche (necesita más baterías). El inventario aparato por aparato siempre gana.

Energía eólica a pequeña escala: el complemento del sol

El sol no brilla de noche y a veces se esconde días enteros detrás de nubes. Ahí es donde el viento puede ayudar: un aerogenerador pequeño produce de día, de noche y con nublado, justo cuando el solar descansa. Los sistemas híbridos solar-eólico son comunes en zonas costeras y del norte de México, donde el viento sopla fuerte y constante.

Un aerogenerador casero funciona con el mismo principio que un ventilador al revés: en lugar de usar electricidad para mover aire, usa el aire en movimiento para generar electricidad. El viento gira las aspas, las aspas giran un generador, y el generador produce corriente continua que va al mismo banco de baterías del sistema solar (a través de su propio controlador). Los modelos pequeños para casa van de 400 W a 2 kW de potencia nominal.

Pero aquí viene la advertencia honesta: la energía eólica a pequeña escala es mucho más exigente que la solar. La potencia del viento crece con el cubo de su velocidad: si el viento sopla al doble de velocidad, la potencia se multiplica por ocho. Eso suena genial hasta que entiendes el reverso: si el viento sopla a la mitad, la potencia cae a la octava parte. Un aerogenerador de 1000 W nominales en un sitio con viento flojo puede producir menos de 100 W la mayor parte del tiempo. Por eso el paso número uno, antes de comprar nada, es medir el viento de tu ubicación durante semanas o meses con un anemómetro barato, o consultar los mapas de recurso eólico de tu región.

Reglas prácticas: necesitas un promedio anual de al menos 4 a 5 m/s (metros por segundo) para que valga la pena. El aerogenerador debe instalarse en una torre de 9 a 15 metros de altura, lejos de árboles y construcciones que generen turbulencia: el viento "sucio" (turbulento) produce poco y desgasta mucho las aspas y los rodamientos. Y considera el ruido: un aerogenerador pequeño zumba, y a tus vecinos podría no gustarles. En ciudad, la eólica casera casi nunca es viable por el viento turbulento y las restricciones; en ranchos, costas y zonas altas abiertas, puede ser excelente.

Analogía: el panel solar es como un empleado puntual que trabaja solo de día pero sin falta; el aerogenerador es como un empleado nocturno talentoso pero impredecible: cuando sopla fuerte produce muchísimo, cuando no, nada. Juntos hacen un buen equipo si el sitio tiene ambos recursos.

Buena práctica: si tu zona tiene buen viento, empieza con solar (más predecible y barato por vatio) y agrega eólica después como expansión, usando el mismo banco de baterías. Así divides el riesgo y aprendes con la tecnología más noble primero.

Monitoreo inteligente: mide tu generación y tu consumo con IoT

Hay un dicho entre instaladores solares: "lo que no se mide, no se mejora". Un sistema solar sin monitoreo es como manejar un carro sin tablero: no sabes a qué velocidad vas, cuánta gasolina queda ni si el motor se está calentando. El monitoreo inteligente usa sensores baratos y microcontroladores como el ESP32 para medir en tiempo real cuánto generan tus paneles, cuánto consumes y cuánta carga tienen tus baterías, y mostrarlo en tu teléfono o en una gráfica en tu computadora.

Qué se mide y con qué. Tres magnitudes mandan: voltaje (V), corriente (A) y, derivada de ambas, potencia ($W = V \times A$). Para medir el voltaje de un panel o una batería se usa un divisor de voltaje (dos resistencias) que reduce el voltaje a un nivel seguro para el ESP32, que solo acepta hasta 3.3 V en sus entradas analógicas. Para medir corriente se usa un sensor como el ACS712 o el INA219: el INA219 es mejor porque mide voltaje y corriente a la vez, se comunica por I2C y no necesita calibración complicada. Con esas dos lecturas, el ESP32 calcula la potencia instantánea y acumula la energía (Wh) con el tiempo.

Dónde se colocan los sensores. Un sistema bien monitoreado mide en tres puntos: la salida de los paneles (generación), la entrada/salida de las baterías (carga y descarga) y la salida del inversor hacia la casa (consumo). Con esos tres números puedes responder las preguntas importantes: estoy generando más de lo que consumo (las baterías se están cargando), estoy consumiendo más de lo que genero (las baterías se están descargando), o están equilibrados. En la sección de ejemplos programaremos el código MicroPython para el primer punto: medir un panel con ESP32 e INA219.

Qué haces con los datos. El ESP32 puede enviar las lecturas por WiFi a un servidor sencillo, a una hoja de cálculo en la nube o a un tablero como los que se arman con herramientas gratuitas. Incluso sin nube, puedes guardar los datos en una tarjeta microSD y analizarlos después en tu computadora. Los patrones que aparecen son oro puro: descubrirás que tu panel genera el 70 por ciento de su energía entre las 10 de la mañana y las 3 de la tarde (esa es tu ventana para usar la lavadora), que tu consumo tiene un pico a las 8 de la noche (cuando ya no hay sol), o que un día nublado específico produjo apenas el 15 por ciento de lo normal (dato clave para dimensionar tu autonomía).

Analogía: el monitoreo es como la app de tu banco. Antes de tenerla, gastabas sin saber en qué; después de tenerla, ves cada cargo y empiezas a gastar mejor sin que nadie te regañe. Los datos cambian el comportamiento solos.

Error común de principiante: medir solo el voltaje de la batería y creer que eso dice cuánta carga tiene. El voltaje de una batería baja cuando le sacas corriente y sube cuando la cargas, aunque su carga real no haya cambiado: es como juzgar cuánta agua tiene un tinaco mirando

cuánto se mueve la superficie cuando lo agitas. Para saber el estado de carga real hay que medir corriente entrante y saliente a lo largo del tiempo (conteo de coulombios) o usar un monitor de batería dedicado. El voltaje sirve como referencia aproximada, no como verdad absoluta.

Buena práctica: empieza monitoreando un solo punto (la salida de tus paneles) antes de instrumentar todo el sistema. Aprenderás a calibrar, a filtrar ruido en las lecturas (promediando varias muestras) y a interpretar gráficas. Cuando eso funcione sólido, agrega los otros puntos. En IoT, como en solar, se crece por módulos.

Eficiencia energética y domótica: el vatio más barato es el que no usas

Aquí va una verdad incómoda para los vendedores de paneles: antes de generar tu propia energía, lo más rentable es dejar de desperdiciarla. Cada vatio que ahorras con eficiencia es un vatio de panel que no tienes que comprar, una fracción de batería que no necesitas y un inversor más pequeño que te sirve. La eficiencia energética no es sacrificio: es inteligencia.

Las medidas de eficiencia con mejor retorno, en orden. Primero, iluminación LED: cambiar focos incandescentes o ahorradores viejos por LED reduce el consumo de iluminación entre 70 y 80 por ciento, y los focos cuestan poco. Segundo, el refrigerador: como vimos, un refri viejo puede ser el mayor consumidor de la casa; uno moderno con etiqueta de eficiencia alta se paga solo con el ahorro. Tercero, elimina los consumos fantasma: aparatos en "standby" (televisores, decodificadores, consolas, cargadores conectados) pueden representar del 5 al 10 por ciento de tu recibo; una regleta con interruptor que apague todo de un golpe cuesta poco y ahorra siempre. Cuarto, ventilación y sombra: en clima cálido, pintar el techo de blanco o instalar un techo verde puede bajar varios grados la temperatura interior y reducir el uso de ventiladores o aire acondicionado.

Y aquí entra la domótica: la automatización del hogar aplicada al ahorro. Con sensores y actuadores baratos (los mismos ESP32 de esta unidad, o focos y enchufes inteligentes comerciales) puedes hacer que tu casa gestione sola su energía. Ejemplos concretos: un sensor de presencia que apague las luces de cuartos vacíos; un temporizador que apague el calentador eléctrico a cierta hora; enchufes inteligentes que corten la corriente de la TV y la consola cuando no hay nadie; un sensor de apertura que apague el minisplit si la ventana está abierta (enfriar la calle es caro). Cada automatización es pequeña, pero juntas recortan el consumo entre 15 y 30 por ciento en un hogar típico.

La domótica también se lleva de maravilla con el solar: puedes programar que la lavadora, la bomba de agua o el calentador se enciendan automáticamente en tu ventana de máxima generación (10 am a 3 pm), cuando la energía es gratis y abundante, en lugar de usarlas de

noche sacándole jugo a las baterías. Eso se llama "desplazar la carga" y es una de las estrategias más poderosas: no reduces tu consumo, lo mueves a cuando el sol lo paga.

Analogía: la eficiencia es como tapar las fugas de una cubeta antes de ir por más agua. Puedes ir cien veces al pozo (comprar más paneles) o tapar los hoyos (ahorrar) y descubrir que con la mitad de viajes te alcanza. La domótica es contratar a un ayudante que tape los hoyos solo, sin que se lo pidas cada vez.

Error común: automatizar antes de medir. Comprar diez focos inteligentes sin saber cuánto consume tu iluminación es gastar a ciegas. Primero el inventario de consumo, luego la eficiencia básica (LED, regletas), y al final la domótica donde los datos digan que hay ahorro real. El orden importa porque cada peso mal gastado en automatización innecesaria es un peso que no fue a un panel.

Buena práctica: calcula el retorno de cada medida de eficiencia igual que el de los paneles (lo veremos en la siguiente subsección). Un foco LED que cuesta 50 pesos y ahorra 200 pesos al año tiene un retorno de 3 meses: imbatible. Esa es la vara con la que se mide todo lo demás.

Costos, retorno de inversión y mantenimiento: los números que mandan

Llegó la hora de hablar de dinero sin rodeos. Cuánto cuesta, cuándo lo recuperas y qué cuidados necesita tu sistema. Sin estos números, todo lo anterior es un pasatiempo caro.

Costos realistas en México (cifras aproximadas, varían por región y proveedor). Un panel monocristalino de 550 W ronda los 3,500 a 5,000 pesos. Un controlador MPPT de 40 A, de 2,500 a 5,000 pesos. Una batería LiFePO4 de 12.8 V/100 Ah, de 4,000 a 7,000 pesos. Un inversor de onda pura de 1500 W, de 4,000 a 8,000 pesos. Estructura, cableado, protecciones y conectores para un sistema pequeño: 3,000 a 6,000 pesos. Para el sistema de ejemplo de esta unidad (3 paneles de 550 W, 5 baterías de litio, MPPT de 40 A, inversor de 1500 W), la inversión total en materiales anda entre 45,000 y 70,000 pesos si lo instalas tú mismo. La instalación profesional agrega típicamente entre 20 y 40 por ciento más, pero incluye garantía de instalación y trámites.

El retorno de inversión (ROI) se calcula así: divides la inversión total entre el ahorro anual que genera el sistema. Si tu sistema produce 5 kWh al día, son unos 150 kWh al mes. Si tu tarifa eléctrica es de 3 pesos por kWh (tarifa doméstica de alto consumo; las tarifas básicas son más baratas y las DAC más caras), ahorras 450 pesos al mes, 5,400 al año. Con una inversión de 55,000 pesos, el retorno simple es de unos 10 años. Parece mucho, hasta que consideras tres cosas: uno, las tarifas eléctricas suben cada año (históricamente entre 4 y 8 por ciento anual en México), así que tu ahorro crece con el tiempo y el retorno real se acorta; dos, los paneles duran 25+ años, así que después del retorno tienes 15 años de electricidad casi gratis; tres, si

estás en tarifa DAC (la tarifa cara para alto consumo), el ahorro por kWh puede ser el doble o triple y el retorno cae a 4-6 años.

Analogía del ROI: comprar un sistema solar es como comprar la vaca en lugar de comprar la leche todos los días. Al inicio duele el desembolso, pero cada día que pasa la vaca te da leche "gratis" (bueno, hay que alimentarla: ese es el mantenimiento). La pregunta no es si la vaca es cara, sino en cuántos años la leche que te ahorras paga la vaca, y cuántos años de leche gratis te quedan después.

Mantenimiento: la buena noticia es que un sistema solar bien instalado pide muy poco. Limpieza de paneles: el polvo puede robarte del 5 al 15 por ciento de la generación; en zonas urbanas límpialos cada 1-2 meses con agua y un trapo suave (nunca abrasivos), de preferencia temprano en la mañana cuando están fríos. Revisión de conexiones: una vez al año aprieta terminales y busca cables pelados o corroídos, sobre todo cerca de la costa donde la sal acelera la corrosión. Baterías: las de litio casi no piden nada; las de plomo-ácido necesitan revisar niveles de electrolito y mantenerlas cargadas (nunca las dejes descargadas por semanas: se sulfatan y mueren). Monitoreo: revisa tus gráficas cada mes; una caída gradual de generación sin explicación (misma época del año, mismo clima) suele ser suciedad o una conexión floja, y detectarla a tiempo evita pérdidas silenciosas.

Vida útil esperada: paneles 25-30 años (pierden alrededor de 0.5 por ciento de potencia por año; a los 25 años siguen dando ~87 por ciento), inversor 10-15 años (es el componente que más probablemente reemplazarás una vez), controlador 10-15 años, baterías de litio 10+ años o 3000-5000 ciclos, baterías de plomo 3-5 años. Cuando calcules el ROI a 25 años, incluye un reemplazo de inversor y, según la química, uno o dos cambios de baterías. Aun así, los números salen a favor del solar en la mayoría de los casos mexicanos con buen sol.

Error común de principiante: calcular el ROI solo con el precio de hoy de la luz e ignorar que sube. Eso subestima el ahorro futuro y hace ver peor la inversión de lo que es. Buena práctica: haz el cálculo en tres escenarios (tarifa sube 3, 5 y 8 por ciento anual) y verás el rango realista de tu retorno. Y el error más caro de todos: comprar el sistema más barato sin ficha técnica ni garantías para "ahorrar". Un panel sin garantía que muere a los 6 años convierte tu inversión en pérdida; uno bueno con 25 años de garantía la convierte en patrimonio.

Con los conceptos claros, es hora de ensuciarnos las manos: vamos a construir la hoja de cálculo, el programa de dimensionamiento y el monitor con ESP32.

Ejemplos paso a paso

Ejemplo 1: hoja de cálculo de consumo eléctrico de tu casa

Vamos a construir el inventario de consumo como una tabla de texto que puedes replicar en papel, en Excel o en Google Sheets. Usaremos la casa de ejemplo de la sección anterior: una vivienda modesta con los aparatos típicos. La tabla tiene cinco columnas: aparato, cantidad, potencia unitaria (W), horas de uso al día y energía diaria (Wh). La última columna es potencia x cantidad x horas.

Sigue estos pasos para llenarla. Primero, recorre tu casa y lista los aparatos. Segundo, busca la potencia en la etiqueta de cada uno (si no la encuentras, usa el valor típico que anotamos). Tercero, estima honestamente las horas de uso. Cuarto, multiplica y suma. Aquí está la tabla completa del ejemplo:

+-----+-----+-----+-----+-----				
-+				
APARATO	CANTIDAD	POTENCIA W	HORAS	ENERGIA Wh
+-----+-----+-----+-----+-----				
-+				
Foco LED	8	9	5	360
Refrigerador (moderno)	1	150	8	1200
Televisor LED 50"	1	100	4	400
Ventilador de techo	2	70	6	840
Laptop	1	65	5	325
Lavadora	1	500	1	500
Modem de internet	1	10	24	240
Cargadores de celular	3	10	3	90
Bomba de agua 1/2 HP	1	370	0.5	185
Licuadora	1	400	0.25	100
Foco exterior LED	2	12	10	240
Bocina inteligente	1	15	24	360
+-----+-----+-----+-----+-----				
-+				
TOTAL DIARIO				4840
+-----+-----+-----+-----+-----				
-+				

Interpretación de la tabla. El total es 4840 Wh, o sea 4.84 kWh al día. Fíjate en quiénes son los culpables: el refrigerador solo representa casi el 25 por ciento del total (1200 de 4840). Los dos ventiladores juntos, 840 Wh, son el segundo consumidor. La bocina inteligente encendida todo el día gasta 360 Wh, lo mismo que los 8 focos de la casa juntos: ese es un vampiro clásico que la tabla deja al descubierto. La licuadora, aunque es muy potente (400 W), apenas suma 100 Wh porque se usa 15 minutos: potencia alta no significa consumo alto si el tiempo es corto.

Ahora aplica el margen de seguridad del 25 por ciento: $4840 \times 1.25 = 6050$ Wh, redondeamos a 6.05 kWh/día. Este es el número que alimenta el dimensionamiento: tu sistema debe generar 6.05 kWh cada día promedio.

Cómo adaptar la tabla a tu casa. Agrega o quita filas según tus aparatos reales. Si tienes aire acondicionado, prepárate: un minisplit de 1 tonelada consume unos 1100 W, y 6 horas de uso son 6600 Wh él solo, más que toda la tabla anterior junta. El aire acondicionado es el aparato que más cambia el tamaño (y el precio) de un sistema solar; mucha gente decide que el solar cubrirá todo menos el clima, y usa el clima solo con la red eléctrica. Esa es una decisión válida y común: no todo tiene que ser solar desde el día uno.

Verificación cruzada con tu recibo. Si tu recibo dice que consumes 300 kWh al bimestre, eso es un promedio de 5 kWh al día ($300 / 60$ días). Si tu tabla dice 4.84 kWh/día, están en el mismo orden: tu inventario es confiable. Si tu tabla dice 12 kWh/día y el recibo dice 5, algo está mal en tus estimaciones de horas (casi siempre son las horas: la gente sobreestima cuánto usa cada cosa). Ajusta hasta que tabla y recibo se parezcan; esa calibración es lo que separa un cálculo serio de un número inventado.

Ejemplo 2: script Python para dimensionar un sistema solar

Ahora automatizamos todo el dimensionamiento en Python. El programa recibe tu consumo diario (el número que sacaste de la tabla), las horas pico de sol de tu zona, los días de autonomía que quieres y el voltaje del sistema, y te devuelve la lista de compras: cuántos paneles, qué controlador, cuántas baterías y qué inversor. Léelo con calma: cada bloque está comentado y corresponde exactamente a los pasos que hicimos a mano en la explicación.

```
# dimensionador_solar.py
# Calcula los componentes de un sistema solar basico a partir
# de tu consumo diario. Solo usa ASCII en este archivo.

import math

def dimensionar(consumo_kwh_dia, hsp, dias_autonomia=1,
                voltaje_sistema=48, potencia_panel=550):
    # --- Datos de diseno (ajustables a tu caso) ---
    factor_perdidas = 0.80          # cables, inversor, polvo, calor
    margen_seguridad = 1.25         # 25 por ciento extra
    dod_litio = 0.90                # profundidad de descarga util LiFeP04
    cap_bateria_ah = 100            # bateria tipica de litio 12.8 V / 100 Ah
    voltaje_bateria = 12.8

    # --- Paso 1: energia diaria con margen ---
```

```

wh_dia = consumo_kwh_dia * 1000 * margen_seguridad

# --- Paso 2: potencia de paneles necesaria ---
# Potencia (W) = energia diaria (Wh) / horas pico de sol / perdidas
potencia_necesaria = wh_dia / hsp / factor_perdidas
num_paneles = math.ceil(potencia_necesaria / potencia_panel)
potencia_total = num_paneles * potencia_panel

# --- Paso 3: baterias para los dias de autonomia ---
wh_almacenar = wh_dia * dias_autonomia
wh_util_por_bateria = voltaje_bateria * cap_bateria_ah * dod_litio
num_baterias = math.ceil(wh_almacenar / wh_util_por_bateria)

# --- Paso 4: controlador MPPT ---
# Corriente = potencia de paneles / voltaje del sistema
corriente = potencia_total / voltaje_sistema
# Se elige el tamaño comercial siguiente con holgura
for tamaño in [20, 30, 40, 60, 80, 100]:
    if tamaño >= corriente * 1.25:
        mppt = tamaño
        break

# --- Paso 5: inversor sugerido ---
# Regla practica: 1.5 veces la potencia de paneles, minimo 1000 W
inversor_w = max(1000, math.ceil(potencia_total * 1.5 / 500) * 500)

return {
    "energia_diaria_wh": round(wh_dia),
    "potencia_paneles_w": round(potencia_necesaria),
    "num_paneles": num_paneles,
    "potencia_total_w": potencia_total,
    "num_baterias": num_baterias,
    "controlador_mppt_a": mppt,
    "voltaje_sistema_v": voltaje_sistema,
    "inversor_w": inversor_w,
}

def mostrar(resultado):
    print("=== DIMENSIONAMIENTO SOLAR ===")
    print("Energia diaria con margen :", resultado["energia_diaria_wh"], "Wh")
    print("Potencia de paneles      :", resultado["num_paneles"],
          "paneles de 550 W =", resultado["potencia_total_w"], "W")
    print("Baterias LiFePO4 12.8V/100Ah:", resultado["num_baterias"], "pzas")
    print("Controlador MPPT          :", resultado["controlador_mppt_a"],
          "A a", resultado["voltaje_sistema_v"], "V")

```

```

    print("Inversor onda pura      :", resultado["inversor_w"], "W")
    print("=====")

if __name__ == "__main__":
    # Ejemplo: casa de 4.84 kWh/día, 5.5 HSP (centro de Mexico),
    # 1 día de autonomía, sistema de 48 V
    r = dimensionar(consumo_kwh_dia=4.84, hsp=5.5,
                    dias_autonomia=1, voltaje_sistema=48)
    mostrar(r)

    print()
    print("--- Comparativa: 2 días de autonomía ---")
    r2 = dimensionar(consumo_kwh_dia=4.84, hsp=5.5,
                     dias_autonomia=2, voltaje_sistema=48)
    mostrar(r2)

```

Qué hace cada parte. La función `dimensionar` recibe tus datos y aplica las fórmulas paso a paso: primero agrega el margen del 25 por ciento a tu consumo; luego calcula la potencia de paneles dividiendo entre las horas pico de sol y el factor de pérdidas (0.80); con `math.ceil` redondea hacia arriba el número de paneles porque no puedes comprar medio panel. Para las baterías, multiplica tu consumo por los días de autonomía y lo divide entre la energía útil de cada batería (voltaje x amperios-hora x 90 por ciento de descarga permitida en litio). El controlador se elige buscando el primer tamaño comercial que supere la corriente calculada con 25 por ciento de holgura. El inversor se sugiere como 1.5 veces la potencia de paneles, redondeado a múltiplos de 500 W.

Si corres el programa con los datos del ejemplo, obtienes: energía diaria con margen 6050 Wh, 3 paneles de 550 W (1650 W), 6 baterías de litio, controlador MPPT de 60 A a 48 V e inversor de 2500 W. Con 2 días de autonomía, las baterías suben a 11 y lo demás se mantiene: fíjate cómo la autonomía pega directo en el costo de baterías, que es el componente más caro del sistema.

Cómo personalizarlo. Cambia `hsp` por el valor de tu zona (consulta mapas de irradiación o usa 5.0 como valor conservador para México). Cambia `potencia\_panel` si encuentras paneles de 450 W más baratos. Cambia `voltaje\_sistema` a 24 si tu sistema es pequeño (menos de 1 kW). Y si prefieres baterías de plomo-ácido, cambia `dod\_litio` a 0.50 y ajusta la capacidad. El programa es una herramienta de prediseño, no sustituye la revisión de un técnico para la instalación final, pero te pone en la mesa con números sólidos cuando pidas cotizaciones.

Ejemplo 3: monitorear un panel solar con ESP32 y MicroPython

Este es el proyecto estrella de la unidad: un ESP32 que mide el voltaje y la corriente de tu panel solar en tiempo real y calcula la potencia. Usaremos el sensor INA219 (mide voltaje y corriente por I2C, hasta 26 V y 3.2 A en su versión común) conectado entre el panel y la carga. El código está en MicroPython, el Python recortado que corre en microcontroladores.

Conexiones físicas (léelas dos veces antes de conectar nada). El INA219 se conecta al ESP32 por I2C: VCC del sensor a 3V3 del ESP32, GND a GND, SDA al pin 21, SCL al pin 22. El panel solar se conecta a los bornes de entrada del INA219 (VIN+ y VIN-), y la carga (por ejemplo, una resistencia de potencia o un foco de 12 V, o la entrada de tu controlador) a los bornes de salida. IMPORTANTE: el INA219 común soporta máximo 26 V; un panel de 550 W en circuito abierto puede dar más de 40 V y lo quemaría. Para paneles grandes usa la versión de alto voltaje del INA219 o un divisor de voltaje externo para el voltaje y un ACS712 para la corriente. Para este ejemplo asumimos un panel pequeño de prueba de 12 V nominal (típico de 20-50 W), perfecto para aprender sin riesgos.

El código:

```
# monitor_solar.py - MicroPython para ESP32 + INA219
# Mide voltaje, corriente y potencia de un panel solar.
# Solo ASCII en este archivo.

from machine import Pin, I2C
import time

# --- Direccion I2C del INA219 (0x40 por defecto) ---
INA219_ADDR = 0x40
REG_CONFIG = 0x00
REG_SHUNTV = 0x01
REG_BUSV = 0x02
REG_POWER = 0x03
REG_CURRENT = 0x04
REG_CALIB = 0x05

# Configuración: rango 32V, ganancia /8, ADC 12 bits, continuo
CONFIG_VAL = 0x399F
CALIB_VAL = 4096 # calibración para shunt de 0.1 ohm

def ina219_write(reg, val):
    buf = bytearray(3)
    buf[0] = reg
    buf[1] = (val >> 8) & 0xFF
    buf[2] = val & 0xFF
```



```

i2c.writeto(INA219_ADDR, buf)

def ina219_read(reg):
    i2c.writeto(INA219_ADDR, bytes([reg]))
    data = i2c.readfrom(INA219_ADDR, 2)
    val = (data[0] << 8) | data[1]
    if val > 32767:
        val -= 65536
    return val

def leer_medicion(num_muestras=10):
    # Promedia varias muestras para filtrar el ruido electrico
    v_total = 0.0
    i_total = 0.0
    for _ in range(num_muestras):
        bus_raw = ina219_read(REG_BUSV)
        voltaje = (bus_raw >> 3) * 0.004 # LSB = 4 mV
        corriente_raw = ina219_read(REG_CURRENT)
        corriente = corriente_raw * 0.001 # 1 mA por LSB con esta calibracion
        v_total += voltaje
        i_total += corriente
        time.sleep_ms(50)
    v = v_total / num_muestras
    i = i_total / num_muestras
    p = v * i
    return v, i, p

# --- Inicializacion ---
i2c = I2C(0, scl=Pin(22), sda=Pin(21), freq=400000)
ina219_write(REG_CONFIG, CONFIG_VAL)
ina219_write(REG_CALIB, CALIB_VAL)
print("Monitor solar iniciado. Midiendo cada 5 segundos...")
print("Formato: voltaje(V) | corriente(A) | potencia(W)")

energia_wh = 0.0
ultimo_t = time.ticks_ms()

while True:
    v, i, p = leer_medicion()
    ahora = time.ticks_ms()
    dt_horas = time.ticks_diff(ahora, ultimo_t) / 3600000.0
    ultimo_t = ahora
    energia_wh += p * dt_horas # acumula energia generada
    print("V={:.2f} I={:.3f} P={:.2f} | Energia acumulada: {:.2f} Wh".format(
        v, i, p, energia_wh))

```

```
time.sleep(5)
```

Cómo funciona el código. Primero configura el bus I2C y programa el INA219 con los registros de configuración y calibración (esos números hexadecimales le dicen al sensor su rango y su resistencia shunt). La función `leer\_medicion` toma 10 muestras y las promedia: esto es clave porque las lecturas analógicas siempre tienen ruido eléctrico, y promediar es el filtro más simple y efectivo. Cada lectura convierte los valores crudos del sensor a voltios y amperios usando los factores del datasheet (4 mV por LSB en voltaje). En el ciclo principal, además de mostrar voltaje, corriente y potencia instantánea, acumula la energía en Wh multiplicando la potencia por el tiempo transcurrido: así obtienes el dato que de verdad importa, cuánta energía generó tu panel en el día.

Qué vas a ver en la consola. Algo como: `V=17.42 I=1.834 P=31.95 | Energía acumulada: 12.40 Wh`. A pleno sol del mediodía verás la potencia cerca del máximo del panel; si pasas la mano sobre el panel verás la potencia desplomarse en tiempo real. Esa es la mejor clase de física que existe: tu sombra convertida en números.

Mejoras que puedes agregar después. Una: guardar las lecturas en un archivo en la memoria del ESP32 cada minuto, para descargarlas luego y graficarlas. Dos: conectar el ESP32 a tu WiFi y enviar los datos a un tablero gratuito en la nube para verlos en tu teléfono. Tres: agregar un segundo INA219 (cambiando su dirección I2C a 0x41) para medir el consumo de la casa al mismo tiempo y comparar generación contra consumo en vivo. Cada mejora es un proyecto de fin de semana y te acerca a un sistema de monitoreo profesional hecho por ti.

Seguridad, en serio. Trabaja siempre con el panel tapado o de noche cuando hagas conexiones: un panel al sol genera voltaje en cuanto lo destapas, no tiene interruptor. Usa fusible en serie con el panel. Y nunca midas directamente con el ESP32 un voltaje mayor a 3.3 V en sus pines: para eso está el sensor. La electricidad solar es de bajo voltaje y segura si respetas estas reglas; imprudente si no.

Práctica

Ejercicio 1: calcula el consumo diario de tu recámara

Tu recámara tiene: 2 focos LED de 9 W encendidos 6 horas al día, 1 ventilador de 70 W encendido 8 horas, 1 laptop de 65 W usada 4 horas, 1 cargador de celular de 10 W conectado 3 horas y 1 televisor de 80 W encendido 2 horas. Calcula la energía diaria total en Wh y en kWh. Después identifica qué aparato consume más y propón una medida concreta para reducir su consumo.

Solución paso a paso. Paso 1: calcula cada aparato con la fórmula energía = potencia x tiempo. Focos: $2 \times 9 \text{ W} \times 6 \text{ h} = 108 \text{ Wh}$. Ventilador: $70 \times 8 = 560 \text{ Wh}$. Laptop: $65 \times 4 = 260 \text{ Wh}$. Cargador: $10 \times 3 = 30 \text{ Wh}$. Televisor: $80 \times 2 = 160 \text{ Wh}$. Paso 2: suma todo: $108 + 560 + 260 + 30 + 160 = 1118 \text{ Wh}$. Paso 3: convierte a kWh dividiendo entre 1000: 1.118 kWh al día. Paso 4: identifica al culpable: el ventilador con 560 Wh representa el 50 por ciento del consumo de la recámara. Paso 5: medida concreta: usar el ventilador solo 6 horas en lugar de 8 (ahorra 140 Wh diarios) o cambiarlo por un ventilador moderno de bajo consumo de 45 W (ahorra 200 Wh diarios con las mismas 8 horas). Moraleja del ejercicio: el aparato más potente no siempre es el que más gasta; el que más horas trabaja, sí.

Ejercicio 2: dimensiona los paneles para una casa de 6 kWh al día

Una familia consume 6 kWh al día (ya incluye el margen de seguridad). Viven en una zona con 5 horas pico de sol (HSP = 5) y el factor de pérdidas del sistema es 0.8. a) Calcula la potencia de paneles necesaria en vatios. b) Si los paneles disponibles son de 450 W, cuántos se necesitan. c) Calcula la energía real que generará ese arreglo en un día promedio.

Solución paso a paso. Paso 1: convierte el consumo a Wh: $6 \text{ kWh} = 6000 \text{ Wh}$. Paso 2: aplica la fórmula potencia = energía / HSP / factor de pérdidas: $6000 / 5 / 0.8 = 1500 \text{ W}$. Esa es la respuesta a). Paso 3: divide entre la potencia de cada panel: $1500 / 450 = 3.33$; como no puedes comprar un tercio de panel, redondeas hacia arriba a 4 paneles. Esa es la respuesta b). Paso 4: la potencia instalada real es $4 \times 450 = 1800 \text{ W}$. Paso 5: la energía diaria real = potencia instalada x HSP x factor de pérdidas = $1800 \times 5 \times 0.8 = 7200 \text{ Wh} = 7.2 \text{ kWh}$. Esa es la respuesta c). Observa que generas 7.2 kWh para un consumo de 6 kWh: ese excedente del 20 por ciento es tu colchón para días nublados y para el crecimiento futuro del consumo. Redondear hacia arriba siempre te regala ese margen gratis.

Ejercicio 3: cuántas baterías necesitas para 2 días sin sol

Siguiendo el ejercicio anterior, la familia quiere 2 días de autonomía (aguantar 2 días nublados seguidos). Van a usar baterías LiFePO₄ de 12.8 V y 100 Ah, de las que se puede aprovechar el 90 por ciento de su capacidad. a) Calcula la energía total a almacenar. b) Calcula la energía útil de cada batería. c) Determina cuántas baterías se necesitan.

Solución paso a paso. Paso 1: la energía diaria con margen es 6000 Wh (del ejercicio 2). Para 2 días: $6000 \times 2 = 12000 \text{ Wh}$ a almacenar. Esa es la respuesta a). Paso 2: la capacidad nominal de cada batería es voltaje x amperios-hora: $12.8 \text{ V} \times 100 \text{ Ah} = 1280 \text{ Wh}$. La energía útil es el 90 por ciento: $1280 \times 0.90 = 1152 \text{ Wh}$ por batería. Esa es la respuesta b). Paso 3: divide la energía a almacenar entre la energía útil por batería: $12000 / 1152 = 10.42$; redondea

hacia arriba a 11 baterías. Esa es la respuesta c). Reflexión importante: 11 baterías de litio son el componente más caro del sistema, y por eso la autonomía es la decisión económica más fuerte del diseño. Cada día extra de autonomía cuesta miles de pesos. Pregunta honesta que debes hacerte: en tu zona, cuántos días nublados seguidos hay realmente al año. Si son 3 o 4, quizá te conviene 1 día de autonomía más un plan B (como reducir consumo esos días) en lugar de pagar por 3 días de baterías que casi nunca usarás.

Ejercicio 4: elige el controlador y el inversor correctos

Con los datos de los ejercicios 2 y 3 (4 paneles de 450 W = 1800 W, sistema a 48 V), responde: a) Qué corriente debe manejar el controlador MPPT y qué tamaño comercial eliges (opciones: 20, 30, 40, 60, 80 A). b) La familia tiene estos aparatos que pueden estar encendidos a la vez: refrigerador 150 W, televisor 100 W, 6 focos LED 54 W, laptop 65 W, ventilador 70 W; el refrigerador tiene un pico de arranque de 600 W. Qué potencia mínima de inversor de onda pura recomiendas.

Solución paso a paso. Paso 1: la corriente del controlador es potencia de paneles / voltaje del sistema: $1800 \text{ W} / 48 \text{ V} = 37.5 \text{ A}$. Paso 2: agrega 25 por ciento de holgura: $37.5 \times 1.25 = 46.9 \text{ A}$. Paso 3: elige el primer tamaño comercial que iguale o supere ese valor: 60 A. Esa es la respuesta a). Un MPPT de 40 A se quedaría corto y se protegería (o se dañaría) en las horas de máximo sol. Paso 4: suma las potencias simultáneas: $150 + 100 + 54 + 65 + 70 = 439 \text{ W}$ de consumo continuo. Paso 5: considera el pico de arranque: cuando el refrigerador arranca pide 600 W en lugar de 150 W, así que el pico total momentáneo es $600 + 100 + 54 + 65 + 70 = 889 \text{ W}$. Paso 6: el inversor debe cubrir el pico con holgura: un inversor de onda pura de 1500 W a 48 V es la recomendación (cubre los 889 W del pico con casi 70 por ciento de margen y deja espacio para agregar aparatos). Esa es la respuesta b). Nota: muchos inversores soportan el doble de su potencia nominal durante unos segundos para picos de arranque, pero dimensionar con holgura alarga su vida y evita apagones justo cuando el refrigerador más lo necesita.

Ejercicio 5: calcula el retorno de inversión de un sistema real

Un sistema solar completo (materiales + instalación) cuesta 60,000 pesos. Genera en promedio 7 kWh al día. La tarifa eléctrica de la familia es de 3.50 pesos por kWh y se espera que suba 5 por ciento cada año. a) Calcula el ahorro del primer año. b) Calcula en cuántos años se recupera la inversión con la tarifa subiendo (haz la cuenta año por año acumulando el ahorro). c) Si los paneles duran 25 años, estima la ganancia total aproximada después del retorno.

Solución paso a paso. Paso 1: energía anual generada: $7 \text{ kWh/día} \times 365 \text{ días} = 2555 \text{ kWh}$ al año. Paso 2: ahorro del primer año: $2555 \times 3.50 = 8,942.50$ pesos. Esa es la respuesta a). Paso 3: acumula año por año con la tarifa subiendo 5 por ciento anual. Año 1: 8,942.50 (acumulado 8,942.50). Año 2: tarifa 3.675, ahorro 9,389.63 (acumulado 18,332.13). Año 3: tarifa 3.85875, ahorro 9,859.11 (acumulado 28,191.24). Año 4: tarifa 4.05169, ahorro 10,352.06 (acumulado 38,543.30). Año 5: tarifa 4.25427, ahorro 10,869.67 (acumulado 49,412.97). Año 6: tarifa 4.46699, ahorro 11,413.15 (acumulado 60,826.12). En el año 6 el acumulado supera los 60,000 pesos: el retorno se alcanza durante el sexto año. Esa es la respuesta b). Paso 4: para estimar la ganancia total, calcula el ahorro acumulado en 25 años. La tarifa en el año n es $3.50 \times 1.05^{(n-1)}$ y el ahorro anual es $2555 \times$ esa tarifa. La suma de los 25 años da aproximadamente 425,000 pesos (puedes verificarlo con una hoja de cálculo o un pequeño programa). Resta la inversión de 60,000: ganancia aproximada de 365,000 pesos en 25 años, sin contar un posible reemplazo del inversor alrededor del año 12 (unos 6,000 pesos). Esa es la respuesta c). La lección: el retorno de 6 años suena largo hasta que ves que los 19 años restantes son ganancia casi pura. Y si la tarifa sube más rápido (8 por ciento anual), el retorno cae a unos 5 años y la ganancia total se dispara.

Ejercicio 6: diseña un sistema de respaldo para emergencias

Quieres un sistema pequeño e independiente (no conectado a la red) que en un apagón mantenga funcionando durante 8 horas: un refrigerador pequeño (100 W, trabaja 6 de esas 8 horas), 4 focos LED (9 W cada uno, 5 horas), un ventilador (70 W, 8 horas) y la carga de 2 celulares (10 W cada uno, 3 horas). La zona tiene 5 HSP. a) Calcula la energía que debe entregar el sistema en esas 8 horas. b) Dimensiona paneles (450 W, factor de pérdidas 0.8, agrega 25 por ciento de margen) y baterías de litio 12.8 V/100 Ah para 1 día de autonomía. c) Explica por qué este sistema es más barato que uno para toda la casa y en qué casos conviene.

Solución paso a paso. Paso 1: calcula la energía de cada carga. Refrigerador: $100 \text{ W} \times 6 \text{ h} = 600 \text{ Wh}$. Focos: $4 \times 9 \text{ W} \times 5 \text{ h} = 180 \text{ Wh}$. Ventilador: $70 \times 8 = 560 \text{ Wh}$. Celulares: $2 \times 10 \text{ W} \times 3 \text{ h} = 60 \text{ Wh}$. Total: $600 + 180 + 560 + 60 = 1400 \text{ Wh} = 1.4 \text{ kWh}$. Esa es la respuesta a). Paso 2: agrega el margen del 25 por ciento: $1400 \times 1.25 = 1750 \text{ Wh/día}$ a generar. Paso 3: potencia de paneles: $1750 / 5 / 0.8 = 437.5 \text{ W}$; con paneles de 450 W necesitas 1 solo panel ($437.5 / 450 = 0.97$, redondea a 1). Paso 4: baterías para 1 día de autonomía: $1750 \text{ Wh} / 1152 \text{ Wh útiles por batería} = 1.52$, redondea a 2 baterías. Paso 5: controlador: $450 \text{ W} / 12 \text{ V}$ (sistema pequeño a 12 V) = $37.5 \text{ A} \times 1.25 = 46.9 \text{ A}$, elige MPPT de 60 A... pero espera: un sistema tan pequeño a 12 V pide mucha corriente; alternativa más sensata: usa un panel de 200 W en lugar de 450 W ($200 / 5 / 0.8 = 320 \text{ W}$ necesarios... no alcanza). Recalcula con 2 paneles de 200 W = 400 W, que dan $400 \times 5 \times 0.8 = 1600 \text{ Wh}$, un poco justo; mejor 1 panel de 450 W y sistema a 24 V:

$450 / 24 = 18.75 \text{ A} \times 1.25 = 23.4 \text{ A}$, controlador MPPT de 30 A a 24 V. Inversor de onda pura de 1000 W a 24 V (pico del refri 400 W + resto 200 W = 600 W, con holgura). Esa es la respuesta b): 1 panel de 450 W, 2 baterías de litio, MPPT de 30 A a 24 V, inversor de 1000 W. Costo aproximado de materiales: 15,000 a 22,000 pesos. Paso 6: por qué es más barato: porque solo respalda lo esencial (1.4 kWh) en lugar de toda la casa (5-6 kWh), así que necesita menos de la tercera parte de paneles y baterías, que son lo más caro. Conviene cuando los apagones son frecuentes pero cortos, cuando el presupuesto no alcanza para el sistema completo, o como primer paso para aprender antes de escalar. Esa es la respuesta c). Este es exactamente el proyecto de "respaldo para emergencias" de la unidad: un sistema real, útil y alcanzable.

Proyecto 1: cargador solar USB portátil

El proyecto más simple y satisfactorio para empezar: un cargador que convierte luz solar en carga para tu celular. Necesitas: un panel solar pequeño de 10 W y 12 V (o dos de 5 V en serie), un módulo reductor USB (buck converter con salida USB de 5 V, cuestan poco y ya traen el puerto), un diodo Schottky para evitar que la batería del celular descargue de noche hacia el panel, cable y una cajita para montarlo. Conexión: panel -> diodo -> módulo USB -> tu cable de carga. A pleno sol, un panel de 10 W entrega unos 8 W reales; un celular cargando pide entre 5 y 10 W, así que cargará más lento que con el enchufe pero gratis y en cualquier lugar. Mejora opcional: agrega una batería pequeña de litio con su módulo de carga para guardar energía y cargar de noche. Lo que aprendes: conversión DC-DC, por qué el voltaje debe regularse antes de llegar a un aparato sensible, y a leer la etiqueta de un panel pequeño (V_{oc} , I_{sc} , V_{mp} , I_{mp}) sin miedo.

Proyecto 2: respaldo para emergencias

Es el sistema que diseñaste en el Ejercicio 6: 1 panel de 450 W, 2 baterías LiFePO4 de 12.8 V/100 Ah en serie (24 V), controlador MPPT de 30 A e inversor de onda pura de 1000 W a 24 V, más fusibles y cableado. Ármalo en una tabla o carrito para tenerlo como estación móvil: en un apagón conectas el refrigerador pequeño, los focos y el ventilador, y tienes 8 horas de tranquilidad. Agrega el monitor con ESP32 del Ejemplo 3 para ver en tu teléfono cuánta carga queda en las baterías: en una emergencia, ese dato vale oro porque te dice si puedes encender una cosa más o si debes ahorrar. Mantenimiento del proyecto: mantén las baterías cargadas aunque no haya apagones (una recarga completa al mes), limpia el panel cada mes y prueba el sistema cada tres meses simulando un apagón de una hora. Lo que aprendes: a cablear un sistema completo de verdad, a respetar polaridades y fusibles, y a dimensionar con autonomía en mente.

Proyecto 3: huerto con riego solar automatizado

El proyecto que une todo: energía solar + IoT + automatización. Un panel pequeño de 50 W carga una batería de 12 V a través de un controlador PWM barato; la batería alimenta una bomba de agua sumergible de 12 V y un ESP32. El ESP32 lee un sensor de humedad de suelo (capacitivo, no el resistivo barato que se corroe) y cuando la tierra está seca, activa la bomba unos minutos mediante un relevador o un MOSFET. El agua sale de un tinaco o cubeta elevada por gravedad o impulsada por la bomba hacia tus macetas o tu huerto. El código es una extensión natural del monitor solar: lees el sensor, comparas con un umbral, activas la bomba, y además reportas por WiFi la humedad y el estado de la batería. Riega de preferencia entre 10 am y 2 pm, cuando el panel genera al máximo y la evaporación aún no es brutal. Lo que aprendes: a combinar generación, almacenamiento y control automático en un solo sistema, que es exactamente lo que hacen las granjas solares inteligentes a gran escala. Y de paso, comes jitomates regados con el sol.

Resumen de la unidad

- La energía solar fotovoltaica convierte luz en electricidad mediante el efecto fotovoltaico en celdas de silicio, sin partes móviles ni ruido. Los paneles monocristalinos son hoy la mejor opción para techos por su eficiencia (19-23 por ciento) y su precio.
- Un sistema solar tiene cuatro componentes: paneles (generan DC), controlador de carga MPPT (protege y optimiza la carga de baterías), baterías (almacenan para la noche y días nublados) e inversor de onda pura (convierte DC a AC de 127 V para tus aparatos). Más protecciones, cableado del calibre correcto y estructura con la inclinación adecuada.
- Potencia (W) es velocidad, energía (Wh/kWh) es distancia: $\text{Energía} = \text{Potencia} \times \text{Tiempo}$. Tu recibo cobra kWh. Esta distinción ordena todos los cálculos de la unidad.
- El cálculo de consumo se hace aparato por aparato: inventario, potencia de la etiqueta, horas de uso honestas, multiplicación y suma, más 25 por ciento de margen. El refrigerador viejo y los consumos fantasma son los culpables típicos.
- El dimensionamiento sigue la fórmula: potencia de paneles (W) = energía diaria (Wh) / horas pico de sol / 0.8 de pérdidas. Las baterías se calculan por días de autonomía y energía útil ($\text{Ah} \times \text{V} \times \text{profundidad de descarga}$). El controlador por corriente (W/V con holgura) y el inversor por la carga simultánea más el pico de arranque de los motores.
- La energía eólica a pequeña escala complementa al solar donde hay viento constante (mínimo 4-5 m/s promedio), pero exige medir el viento antes de invertir y una torre alta lejos de turbulencias. En ciudad casi nunca conviene; en campo abierto sí.

- El monitoreo con IoT (ESP32 + sensor INA219) te muestra generación, consumo y energía acumulada en tiempo real. Medir cambia el comportamiento: los datos revelan tu ventana de generación (10 am a 3 pm) y tus picos de consumo.
- La eficiencia primero: LED, refrigerador moderno, eliminar standby y domótica (sensores, temporizadores, enchufes inteligentes) pueden recortar 15-30 por ciento del consumo. El vatio más barato es el que no usas, y cada vatio ahorrado es menos panel y menos batería que comprar.
- Los números mandan: calcula el ROI dividiendo la inversión entre el ahorro anual, considerando que la tarifa sube cada año. Un sistema bien dimensionado en México se paga en 5-10 años y los paneles duran 25+. El mantenimiento es mínimo: limpieza periódica, revisión anual de conexiones y baterías cuidadas.
- Errores que ya no vas a cometer: comprar paneles por potencia en lugar de por energía diaria, usar baterías de carro en solar, comprar inversor de onda modificada para electrónica, olvidar los picos de arranque de los motores, e instalar sin medir primero tu consumo real.
- Tus herramientas: la hoja de cálculo de consumo, el script Python de dimensionamiento y el monitor MicroPython con ESP32. Úsalas antes de cada compra y cada decisión.
- Proyectos para seguir: cargador solar USB (conversión), respaldo para emergencias (autonomía) y huerto con riego solar (automatización). Cada uno te deja una habilidad distinta y un sistema útil en casa.

El sol lleva miles de millones de años regalando energía; apenas estamos aprendiendo a recibirla bien. Ya sabes medirla, calcularla, dimensionarla, monitorearla y automatizarla. Lo que sigue es salir, medir tu casa y hacer tus números: la independencia energética empieza con una tabla, un programa y las ganas de construir. Nos vemos en la siguiente unidad.

UNIDAD 8: Automatización del hogar inteligente

Bienvenido a la unidad que va a cambiar la forma en que vives tu casa. Hasta ahora, en este libro, has aprendido cómo funciona la tecnología por dentro: el hardware, el software, las redes, la nube. Hoy todo eso se junta en un solo lugar: tu hogar. La automatización del hogar inteligente, también llamada domótica, es el arte y la técnica de hacer que tu casa trabaje por ti: que las luces se prendan solas cuando llegas, que el clima esté perfecto antes de que entres, que las persianas se cierren solas de noche y que tu casa te avise si algo raro pasa cuando no estás.

Suena a película de ciencia ficción, pero es tecnología que ya puedes comprar hoy en cualquier tienda en línea, que cueste menos de lo que crees y que puedes instalar tú mismo sin ser electricista ni programador. Esta unidad te va a enseñar todo desde cero: qué es la domótica, qué puedes automatizar, qué aparatos necesitas, cómo se comunican entre ellos, cómo instalar Home Assistant (la plataforma gratuita más poderosa del mundo para esto), y cómo crear rutinas y automatizaciones de verdad, con ejemplos que puedes copiar y adaptar.

Al terminar esta unidad vas a poder diseñar el plan de automatización de una casa completa, instalar y configurar un sistema central, crear escenas de iluminación, ahorrar energía con climatización inteligente, montar un sistema de seguridad casero y, algo muy importante, proteger tu privacidad en el proceso. Vamos paso a paso, como siempre: primero entiendes la idea, luego la técnica, luego la practicas.

Tu casa como un asistente silencioso

Piensa en la última vez que saliste de tu casa con prisa. ¿Apagaste todas las luces? ¿Cerraste bien la puerta? ¿Dejaste el clima prendido? ¿La plancha desconectada? La mayoría de nosotros hace un repaso mental mientras va en el tráfico, y a veces hasta nos regresamos a revisar. Ahora imagina que tu casa hiciera ese repaso por ti: que al salir detecte que ya no hay nadie y apague sola las luces, baje el clima, cierre la puerta con seguro y te mande un mensaje al celular diciendo "todo en orden, buen día".

Eso es la automatización del hogar: un conjunto de dispositivos conectados (focos, enchufes, sensores, cámaras, cerraduras, termostatos) que se comunican entre sí y con un sistema central, y que actúan solos según reglas que tú defines. La palabra domótica viene de "domus", que en latín significa casa, más el sufijo de automática. Y aunque suene muy técnico, la idea es muy humana: que la tecnología se encargue de las tareas repetitivas y aburridas para que tú vivas más cómodo y tranquilo.

Un buen sistema de hogar inteligente tiene tres características. Primero, es invisible: funciona en segundo plano sin que tengas que estar picándole a una aplicación a cada rato. Si tienes que abrir el celular para prender cada foco, eso no es automatización, eso es un control remoto caro. La automatización de verdad pasa sola. Segundo, es confiable: las cosas importantes (como la cerradura de la puerta o la alarma) tienen que funcionar siempre, incluso si se va el internet. Tercero, es tuyo: tú decides qué se automatiza, cuándo y cómo, y tus datos se quedan bajo tu control.

Hay una analogía que me gusta mucho para explicar esto. Piensa en tu casa como en una orquesta. Cada dispositivo (el foco, el sensor, la bocina) es un músico con su instrumento. Sin director, cada uno toca lo suyo y hay ruido. El sistema central, como Home Assistant, es el director de orquesta: les dice a todos cuándo entrar, a qué volumen tocar y cuándo callarse. Y las automatizaciones que tú creas son la partitura: las instrucciones escritas que el director sigue. Cuando todo está bien coordinado, lo que escuchas no es ruido, es música: tu casa funcionando en armonía.

Lo que puedes automatizar hoy (sin remodelar nada)

Una de las dudas más comunes es: "¿tengo que remodelar mi casa o hacer obra para esto?" La respuesta es no. La gran mayoría de los dispositivos inteligentes actuales están diseñados para instalarse sin taladrar paredes ni cambiar cableado. Un foco inteligente se enrosca igual que un foco normal. Un enchufe inteligente se conecta al enchufe que ya tienes en la pared. Un sensor de puerta se pega con cinta adhesiva. En una tarde puedes automatizar un departamento completo de renta sin pedirle permiso al casero.

Esto es lo que puedes automatizar hoy mismo, de lo más sencillo a lo más avanzado:

Iluminación: focos que cambian de color y de intensidad, tiras LED para ambientar cuartos, apagadores inteligentes que conservan el apagador físico de la pared. Puedes programar que las luces se prendan al atardecer, que se atenúen solas a la hora de dormir, o que simulen que hay gente en casa cuando sales de vacaciones.

Climatización: termostatos inteligentes que aprenden tus horarios, minisplits controlados por infrarrojo inteligente, sensores de temperatura en cada cuarto. El sistema puede apagar el clima cuando detecta que no hay nadie y prenderlo veinte minutos antes de que llegues, para que entres a una casa fresca sin haber gastado luz todo el día.

Seguridad: cámaras que te avisan al celular cuando detectan movimiento, sensores en puertas y ventanas que disparan una alarma, cerraduras que se abren con tu huella o con un código, timbres con video para ver quién toca sin levantarte del sillón.

Entretenimiento: bocinas que ponen música en varios cuartos a la vez, televisiones que se prenden con tu voz, rutinas de "noche de películas" que con una sola orden atenúan las luces, cierran las persianas y prenden la tele en tu servicio de streaming favorito.

Ahorro de energía y agua: enchufes que miden cuánta luz consume cada aparato, válvulas que cierran el agua si detectan una fuga, regadores de jardín que riegan solos según el clima. Hay gente que ha bajado su recibo de luz de forma notable solo con automatizar el clima y las luces.

Tareas cotidianas: cafeteras que empiezan a hacer café cuando suena tu alarma, persianas que se abren solas con el amanecer, aspiradoras robot que limpian mientras trabajas, recordatorios hablados por las bocinas ("ya son las diez, mañana hay junta temprano").

Y lo mejor: todo esto se puede combinar. Una sola rutina de "buenas noches" puede apagar todas las luces, cerrar las persianas, poner la alarma, bajar el clima y apagar la tele. Una sola orden y listo. Eso es vivir en una casa que trabaja para ti.

Lo que NO es la domótica (para no llevarte sorpresas)

Antes de seguir, aclaremos tres malentendidos muy comunes, porque te van a ahorrar dinero y frustración.

Primero: un hogar inteligente no es lo mismo que tener muchos aparatos con su propia aplicación. Si tienes el foco de una marca con su app, la cámara de otra marca con otra app y el clima de otra marca con una tercera app, lo que tienes es un rompecabezas, no un sistema. La domótica de verdad integra todo en un solo lugar para que los dispositivos trabajen juntos. Por eso en esta unidad le damos tanta importancia al sistema central: sin él, solo tienes gadgets sueltos.

Segundo: no necesitas comprar todo de una vez ni gastar una fortuna. El error clásico del principiante es querer automatizar toda la casa el primer fin de semana, comprar un montón de aparatos de marcas distintas y terminar con la mitad sin funcionar. La buena práctica es empezar chico: un par de focos y un enchufe inteligente, aprender cómo funciona el sistema, y luego ir creciendo cuarto por cuarto. Tu casa inteligente se construye por etapas, no en un día.

Tercero: la automatización no reemplaza el sentido común en seguridad. Una cerradura inteligente es comodísima, pero sigue siendo buena idea tener una chapa tradicional como respaldo. Una cámara te avisa de un intruso, pero no sustituye una buena puerta. La tecnología suma capas de comodidad y protección, pero la base sigue siendo la de siempre: buenas cerraduras, buenas puertas, vecinos atentos.

Con esto claro, ya tienes la mentalidad correcta. Ahora sí, vamos a la parte técnica: cómo se hablan entre sí todos estos aparatos, qué es un hub, qué protocolos existen y por qué importa elegir bien desde el principio.

Explicación detallada

El ecosistema: asistentes de voz, hubs y protocolos

Para que tu casa funcione como un equipo, sus aparatos necesitan tres cosas: una forma de escucharte (asistentes de voz), un cerebro que coordine (hub o sistema central) y un idioma común para hablar entre ellos (protocolos). Vamos a ver cada pieza.

Los asistentes de voz son la cara más conocida del hogar inteligente: bocinas como las de Amazon (Alexa), Google (Google Assistant) o Apple (Siri en el HomePod) que obedecen órdenes habladas como "prende la luz de la sala" o "pon música relajante". Son cómodos y baratos para empezar, pero tienen un límite importante: cada marca quiere que compres solo sus aparatos, y sus automatizaciones son básicas. Sirven perfecto como "micrófono" de tu casa, pero no como cerebro completo. La buena noticia es que puedes usarlos como complemento: le hablas a Alexa y ella le pasa la orden a tu sistema central, que es el que hace el trabajo pesado.

El hub (o sistema central) es el cerebro. Es el aparato o programa que conecta todos tus dispositivos, sin importar la marca, y ejecuta tus automatizaciones. Hay hubs comerciales cerrados (como los de Samsung SmartThings o Apple HomeKit) que son fáciles de usar pero te amarran a su ecosistema. Y está Home Assistant, que es software libre, gratuito y el más poderoso de todos: se instala en una mini computadora como una Raspberry Pi o en cualquier computadora vieja, y controla miles de dispositivos de cientos de marcas. En esta unidad vamos a trabajar con Home Assistant porque es gratis, no depende de ninguna empresa, funciona sin internet para lo importante y es lo que usan los entusiastas serios en todo el mundo. Aprenderlo te da una habilidad que sirve con cualquier marca.

Los protocolos son los idiomas que hablan los dispositivos entre sí. Los principales son cuatro y conviene conocerlos antes de comprar:

WiFi: es el que ya conoces, el de tu internet. Muchos aparatos baratos lo usan porque no necesitas nada extra: se conectan directo a tu red. La ventaja es el precio y la facilidad. La desventaja es que cada aparato WiFi ocupa un lugar en tu red, consume más batería y, si se cae tu internet o tu router se satura con 40 aparatos colgados, todo se pone lento. Para empezar está bien, pero no es lo ideal para una casa con muchos dispositivos.

Zigbee: es un protocolo inalámbrico creado justo para el hogar inteligente. Consume poquísima energía (un sensor Zigbee puede durar dos años con una pila de moneda), y los dispositivos forman una "malla": cada aparato conectado a la corriente ayuda a pasarle el mensaje a los demás, así que mientras más aparatos tengas, más fuerte es la red. Necesitas un coordinador Zigbee (un pequeño adaptador USB que cuesta poco) conectado a tu hub. Es el protocolo favorito de sensores, focos y apagadores.

Z-Wave: muy parecido a Zigbee en filosofía (bajo consumo, red en malla), con la ventaja de que opera en una frecuencia que tiene menos interferencia con el WiFi. Su catálogo es un poco más chico y suele ser más caro, pero es muy confiable, sobre todo en cerraduras y sensores de seguridad.

Matter: es el estándar nuevo, impulsado por las grandes empresas juntas (Apple, Google, Amazon, Samsung y otras). Su promesa es que cualquier aparato Matter funcione con cualquier sistema, sin importar la marca. Todavía está madurando, pero es el futuro: si vas a comprar aparatos nuevos hoy, busca el logotipo de Matter porque te garantiza compatibilidad a largo plazo.

Analogía cotidiana: piensa en los protocolos como idiomas en una vecindad. El WiFi es como gritar por la ventana: llega lejos pero hace ruido y gasta energía. Zigbee y Z-Wave son como una cadena de vecinos pasándose recados de puerta en puerta: eficiente y discreto. Matter es como si todos acordaran hablar el mismo idioma para no necesitar traductor. Y Home Assistant es el vecino bilingüe que habla todos los idiomas y coordina a todos.

Error común de principiante: comprar diez aparatos WiFi baratos de marcas distintas y descubrir que la red se satura, que cada uno necesita su propia app y que nada se habla con nada. Buena práctica: elige un protocolo principal para tu casa (Zigbee es una excelente opción para empezar), compra un coordinador compatible y crece sobre esa base. Los aparatos WiFi déjalos para casos puntuales.

Home Assistant: instalación y primeros pasos

Home Assistant (a veces abreviado HA) es el cerebro de tu casa inteligente. Es un programa de código abierto que instalas en un equipo que se queda prendido todo el tiempo en tu casa, y al que accedes desde el navegador o desde una app en tu celular. Desde ahí ves todos tus dispositivos, creas automatizaciones, revisas historiales de consumo y controlas todo.

¿Qué equipo necesitas? La opción más popular es una Raspberry Pi (una mini computadora del tamaño de una tarjeta de crédito), porque consume muy poca luz y está diseñada para estar prendida siempre. También puedes usar una computadora vieja, un mini PC o hasta un disco duro con el sistema en una máquina virtual. Para empezar, con una Raspberry Pi 4 o 5 y

una tarjeta microSD de buena calidad tienes suficiente. Consejo de buena práctica: usa una fuente de poder oficial y una tarjeta de memoria de marca reconocida; la mayoría de los fallos raros de Home Assistant vienen de una tarjeta barata o una fuente chafa.

La instalación es más sencilla de lo que parece. Descargas la imagen oficial de Home Assistant, la grabas en la tarjeta con un programa como Raspberry Pi Imager, metes la tarjeta en la Raspberry, la conectas a la red con cable y la prendes. Después de unos minutos, abres en tu computadora la dirección <http://homeassistant.local:8123> y sigues el asistente de bienvenida: creas tu usuario, pones tu ubicación (para que sepa a qué hora amanecer y atardece en tu ciudad) y listo. Todo el proceso toma menos de media hora.

Una vez dentro, lo primero que verás es el panel principal (dashboard) con tarjetas para cada dispositivo que el sistema detecte solo en tu red. Home Assistant tiene "integraciones": pequeños conectores para cada marca o protocolo. Muchas se configuran solas con un par de clics. Tus primeros pasos recomendados son: primero, cambia tu contraseña por una fuerte y activa la autenticación en dos pasos; segundo, haz una copia de seguridad (hay una opción de "backup" en los ajustes) y guárdala fuera del equipo; tercero, agrega tus primeros dispositivos con las integraciones automáticas; cuarto, instala la app de Home Assistant en tu celular para controlar todo desde ahí.

Un concepto clave que debes entender desde el inicio: en Home Assistant todo gira alrededor de "entidades". Una entidad es cualquier cosa que el sistema puede ver o controlar: un foco (que se puede prender y apagar), un sensor de temperatura (que solo reporta datos), un interruptor, una cámara. Cada entidad tiene un "estado" (prendido, apagado, 23 grados, etc.) y las automatizaciones no son más que reglas del tipo "cuando esta entidad cambie a tal estado, haz esto con aquella otra". Cuando entiendes las entidades, entiendes el ochenta por ciento de Home Assistant.

Error común de principiante: instalar Home Assistant, agregar veinte dispositivos el primer día y perderse entre nombres genéricos como "light.luz_2" o "sensor.sensor_3". Buena práctica: renombra cada entidad en cuanto la agregues ("luz de la sala", "sensor puerta principal") y organízalas por áreas (sala, cocina, recámara). Tu yo del futuro, cuando tenga cincuenta dispositivos, te lo va a agradecer.

Otro error típico: no hacer respaldos hasta que la tarjeta se corrompe y se pierde todo. Configura los respaldos automáticos desde la primera semana. Perder meses de automatizaciones por no tener un backup es una de las experiencias más dolorosas y más evitables de este hobby.

Iluminación inteligente: focos, tiras LED y escenas

La iluminación es la puerta de entrada perfecta a la domótica: es barata, se instala en minutos, el efecto se nota de inmediato y le enseña a toda la familia la magia de la automatización. Hay tres tipos de productos que debes conocer.

Los focos inteligentes reemplazan tus focos normales: se enroscan igual (la rosca E27 es la estándar en México) y se controlan por app, voz o automatización. Los hay blancos regulables (cambian de luz cálida a luz fría) y RGB (millones de colores). Un detalle técnico importante: el foco inteligente necesita tener corriente siempre para estar "despierto" y recibir órdenes. Si alguien apaga el apagador físico de la pared, el foco se queda sin luz eléctrica y ya no responde hasta que lo vuelven a prender. Por eso la buena práctica es: o dejas el apagador siempre prendido y controlas todo por voz/app/automatización, o instalas un apagador inteligente que mantiene la corriente y manda la orden por software. Muchas familias resuelven esto con un apagador inteligente al lado de la puerta para los invitados y el control total por automatización para el día a día.

Las tiras LED son cintas flexibles con lucecitas que se pegan detrás del televisor, debajo de los muebles de la cocina, alrededor de un espejo o en el techo. Son el truco más barato para que un cuarto se vea espectacular: una tira LED bien colocada cambia por completo el ambiente. Se cortan cada ciertos centímetros por marcas indicadas, se pegan con su adhesivo y se conectan a un controlador. Ideales para luz ambiental de noche, para cuartos de videojuegos o para pasillos.

Las escenas son combinaciones guardadas de varios dispositivos: por ejemplo, la escena "noche de películas" atenúa la luz de la sala al 20 por ciento en tono cálido, apaga la luz de la cocina y prende la tira LED detrás de la tele en azul tenue. La escena "despertar" sube gradualmente la luz de la recámara de 0 a 100 por ciento en quince minutos simulando un amanecer. Las escenas son lo que convierte focos sueltos en experiencias: en lugar de controlar cinco focos uno por uno, activas una escena con una palabra o con una automatización.

Hablemos de temperatura de color, porque es el secreto de la iluminación que se siente bien. Se mide en kelvin (K): la luz cálida (2700K, amarillenta como la de un atardecer) relaja y es ideal para la noche y la sala; la luz neutra (4000K) sirve para cocinar o trabajar; la luz fría (6500K, azulada como el mediodía) ayuda a concentrarse pero en la noche altera el sueño. Una automatización clásica y muy saludable es la "iluminación circadiana": que tus focos sigan al sol automáticamente, fríos y brillantes en la mañana, cálidos y tenues en la noche. Home Assistant puede hacerlo solo con la hora y tu ubicación.

Analogía cotidiana: manejar la luz de tu casa sin escenas es como tener que afinar cada instrumento antes de cada canción; con escenas es como tener botones de "modo fiesta", "modo lectura", "modo dormir" que afinan todo de un golpe.

Error común de principiante: comprar focos RGB para toda la casa y descubrir que la familia los odia porque nadie se acuerda de qué color poner para cenar. Buena práctica: los colores divertidos van en tiras LED y cuartos de juego; en sala, cocina y recámaras usa focos blancos regulables con temperatura de color, que son los que se usan todos los días sin pensar.

Otro error: poner focos inteligentes en lámparas que se controlan con apagador de pared que la familia apaga por costumbre, y luego quejarse de que "no jalan". Decide desde el inicio cómo se va a controlar cada punto de luz y edúcalo con la familia: un cartelito pegado al lado del apagador los primeros días no es exageración, es estrategia.

Climatización y ahorro de energía

El clima (aire acondicionado o calefacción) es el aparato que más luz consume en la mayoría de los hogares mexicanos. Por eso automatizarlo es donde más dinero puedes ahorrar. La lógica es simple: el clima solo debe trabajar fuerte cuando hay alguien en casa y cuando hace falta; el resto del tiempo puede descansar.

Hay tres formas de volver inteligente tu climatización. La primera es un termostato inteligente (como los de Ecobee o similares compatibles) que reemplaza al termostato de pared de tu sistema central. La segunda, muy popular en México, es un controlador infrarrojo inteligente: un aparatito que aprende los comandos del control remoto de tu minisplit y los repite por ti, así conviertes en inteligente cualquier minisplit viejo sin cambiarlo. La tercera es un enchufe inteligente con medición de consumo para climas portátiles o ventiladores.

La automatización estrella aquí combina tres ingredientes: un sensor de presencia (para saber si hay alguien), un sensor de temperatura (para saber si hace falta) y tu ubicación del celular (para saber si vienes en camino). Ejemplo real: "si no hay nadie en la casa desde hace 30 minutos, sube el clima a modo ahorro; cuando mi celular esté a 15 minutos de la casa, regresa a la temperatura cómoda". Así entras a una casa fresca sin haber gastado luz enfriando una casa vacía todo el día. Mucha gente reporta ahorros notables en el recibo con solo esta rutina.

Los enchufes inteligentes con medición de consumo merecen mención aparte: te dicen cuántos watts está gastando cada aparato en tiempo real. Conectas uno a la tele, otro al refrigerador viejo, otro a la computadora, y descubres a los "vampiros energéticos": aparatos que consumen luz aunque estén apagados (televisiones, consolas, cargadores). Una automatización típica es cortarles la corriente de noche a los vampiros y devolvérsela en la mañana. Es ahorro automático sin cambiar ningún hábito.

Buena práctica de oro: mide antes de automatizar. Deja los sensores de consumo una o dos semanas registrando datos antes de crear reglas de ahorro. Los datos te dicen dónde está el gasto real; sin datos, estás adivinando. Home Assistant guarda historiales y gráficas de consumo muy útiles para esto.

Error común de principiante: programar el clima con horarios fijos muy rígidos ("apagar a las 8, prender a las 18") y luego sufrir porque un día llegaste temprano y la casa era un horno, u otro día saliste tarde y se apagó contigo adentro. Buena práctica: combina horarios con sensores de presencia y ubicación. Los horarios son el plan, los sensores son la realidad; la automatización buena escucha a la realidad.

Otro error: poner el sensor de temperatura al lado de una ventana soleada o encima del refrigerador y luego preguntarse por qué el clima se vuelve loco. Los sensores van en lugares representativos del cuarto, lejos de fuentes de calor, del sol directo y de corrientes de aire.

Seguridad del hogar: cámaras, sensores y cerraduras

La seguridad es el área donde la automatización más tranquilidad da, pero también donde más cuidado debes tener, porque un fallo aquí sí importa. Vamos por capas, como la seguridad de verdad: disuasión, detección y respuesta.

La disuasión es hacer que tu casa se vea ocupada y vigilada: luces que se prenden solas al atardecer y se apagan a distintas horas (simulando que alguien anda por la casa), una luz exterior con sensor de movimiento que se enciende cuando alguien se acerca, el timbre con video visible. La mayoría de los ladrones buscan casas fáciles y oscuras; una casa que "reacciona" suele ser suficiente para que sigan de largo.

La detección son los ojos y oídos: sensores de apertura en puertas y ventanas (un imán chiquito que avisa cuando se separan), sensores de movimiento en pasillos y patios, cámaras en puntos estratégicos, sensores de humo y de fuga de agua o gas conectados al sistema. La clave está en las notificaciones inteligentes: no quieres que tu celular vibre cada que el gato pasa por la sala. Configura zonas y horarios: de día, las cámaras solo graban; de noche o cuando no hay nadie, cualquier movimiento te manda alerta con foto.

Las cámaras merecen un apartado especial. Para un sistema casero serio, busca cámaras que permitan grabar en local (en tu propia red, no solo en la nube del fabricante), con buena visión nocturna y detección de personas (para no alertar por cada perro o rama que se mueva). Un timbre con video en la puerta principal es una de las mejores compras de seguridad: ves y hablas con quien toca desde donde estés. Y recuerda la lección de privacidad que veremos más adelante: las cámaras van en exteriores y áreas comunes, nunca en recámaras ni baños, ni siquiera "apagadas".

Las cerraduras inteligentes son comodidad pura: abrir con huella, con código, con el celular o automáticamente cuando llegas. Son especialmente útiles para familias con niños (adiós a las llaves perdidas) y para rentas vacacionales (códigos temporales por huésped). Pero ojo con la buena práctica: elige cerraduras con llave física de respaldo y con baterías que avisen antes de agotarse, y configura que se cierren solas después de unos minutos. Y nunca, nunca automatices la APERTURA de la puerta principal solo con presencia o ubicación sin una confirmación extra: que se abra sola cuando llegas suena cómodo hasta que piensas en quién más podría aprovecharlo.

La respuesta es lo que pasa cuando algo se detecta: que suene una sirena, que se prendan todas las luces de la casa, que te llegue una alerta al celular con video, que se mande un mensaje a un familiar. En Home Assistant puedes armar un "modo alarma" con niveles: "en casa de noche" (solo sensores perimetrales) y "fuera de casa" (todo activo). Prácticalo con la familia para que todos sepan qué hacer cuando suene.

Error común de principiante: poner la cámara apuntando al vecino o a la calle grabando todo el día sin avisar, lo que además de ser mala onda puede meterte en problemas legales según tu localidad. Buena práctica: apunta tus cámaras a TUS accesos, avisa con un letrero pequeño que hay videovigilancia y revisa la normativa de tu ciudad.

Otro error grave: depender al cien por ciento de la nube del fabricante para la alarma. Si se cae tu internet, tu "sistema de seguridad" se vuelve decoración. Buena práctica: lo crítico (sensores, sirena, cerraduras) debe funcionar en red local aunque no haya internet. Home Assistant brilla justo aquí.

Rutinas y automatizaciones: la partitura de tu casa

Ya tienes los músicos (dispositivos) y al director (Home Assistant). Ahora necesitas la partitura: las automatizaciones. Una automatización tiene siempre tres partes, y si te aprendes esta estructura ya sabes crear cualquiera:

El disparador (trigger) es el "cuando": cuando se detecta movimiento, cuando son las 7 de la mañana, cuando el sol se pone, cuando llegas a casa. La condición (condition) es el "solo si": solo si es de noche, solo si no hay nadie en casa, solo si la temperatura pasa de 28 grados. La acción (action) es el "entonces haz": prende la luz, manda una notificación, cierra las persianas.

Con esas tres piezas puedes armar rutinas completas. Las rutinas son conjuntos de acciones que se ejecutan juntas, normalmente activadas por una frase de voz, un botón o un horario. Las clásicas que toda casa inteligente debería tener son:

"Buenos días": a la hora que suene tu alarma, las persianas se abren poco a poco, la luz de la recámara sube gradualmente, la cafetera empieza a trabajar y la bocina te dice el clima de hoy y tus pendientes del calendario.

"Me voy": al salir (lo detecta por tu ubicación o porque activas la rutina), se apagan todas las luces, el clima entra en modo ahorro, la puerta se cierra con seguro, las cámaras pasan a modo alerta y te llega un resumen al celular.

"Buenas noches": con una frase antes de dormir, se apagan las luces de toda la casa excepto una tenue del pasillo, se cierran las persianas, se activa la alarma en modo nocturno, el clima se ajusta para dormir y la tele se apaga.

"Noche de películas": atenúa luces, cierra persianas, prende la tele y la tira LED en modo cine, y pone el celular en no molestar.

"Simulacro de presencia": cuando andas de vacaciones, prende y apaga luces a horarios variados, abre y cierra persianas y hasta prende la tele un rato en la noche, para que la casa se vea habitada.

Analogía cotidiana: una automatización es como dejarle instrucciones a una persona de confianza que cuida tu casa: "si ves que ya son las siete y está oscuro (disparador + condición), prende las luces de afuera (acción)". La diferencia es que tu ayudante digital nunca se duerme, nunca se le olvida y trabaja gratis.

Error común de principiante: crear automatizaciones que pelean entre sí. Ejemplo clásico: una que apaga la luz de la sala a las 23:00 y otra que la prende cuando detecta movimiento, y a las 23:05 alguien pasa y la luz se prende sola "misteriosamente". Buena práctica: antes de crear una automatización nueva, revisa las que ya afectan a los mismos dispositivos, y usa condiciones de horario o de modo (por ejemplo, que la de movimiento no funcione después de las 23:00 si el modo nocturno está activo).

Otro error: automatizar de más al inicio y volver loca a la familia ("¿por qué se apagó la tele sola?"). Buena práctica: implementa una rutina, vívela una semana, ajústala con la retroalimentación de todos y luego sigue con la siguiente. La casa inteligente se adopta por consenso familiar, no por decreto.

Voz + IA en casa: hablarle a tu hogar

Controlar tu casa con la voz es la parte más futurista y la más divertida: "Alexa, buenas noches", "Oye Google, pon música para cocinar", "Siri, cierra las persianas". Pero detrás de esa magia hay dos tecnologías distintas que conviene no confundir: el reconocimiento de voz (convertir tu habla en texto y en una orden) y la inteligencia artificial conversacional (entender lo que quieres aunque lo digas con tus palabras).

Los asistentes comerciales (Alexa, Google, Siri) hacen el reconocimiento de voz en sus servidores: tu bocina graba tu frase, la manda a la nube, la nube la entiende y devuelve la orden. Funciona muy bien, pero implica que tus conversaciones pasan por servidores de una empresa. Home Assistant tiene su propio asistente de voz local (Assist) que puede funcionar sin mandar tu voz a internet, y que además entiende órdenes en español bastante naturales: "prende la luz de la cocina", "pon el clima a 24 grados", "activa la escena de películas".

Aquí es donde la IA moderna cambia el juego. Con un modelo de lenguaje conectado a Home Assistant (hay integraciones que lo permiten, corriendo en local o en la nube según tu decisión de privacidad), puedes hablarle a tu casa como a una persona: "hace calor y voy a llegar en veinte minutos, deja la casa cómoda", y el sistema interpreta: revisa tu ubicación, calcula el tiempo, ajusta el clima y te confirma. O "mañana tengo junta a las ocho, despiértame con tiempo": el sistema revisa tu calendario, calcula a qué hora levantarte, programa la rutina de despertar y te avisa. Esto ya no es programar reglas rígidas, es delegar intenciones.

Pero ojo: la IA en casa necesita barandales. Buena práctica: dale a la IA permiso para informar y sugerir, pero exige confirmación para acciones importantes (abrir puertas, desactivar alarmas, hacer compras). Y define qué datos puede ver: no necesita acceso a tus cámaras para ajustar el clima. El principio es el de siempre en seguridad: mínimo privilegio, solo lo necesario.

Error común de principiante: conectar un asistente de IA con acceso total "para probar" y descubrir que interpretó mal una frase y apagó algo importante. Empieza con permisos de solo lectura y acciones inofensivas (luces, música), y amplía poco a poco cuando confíes en cómo interpreta tus órdenes.

Privacidad en el hogar inteligente: tu casa no debe espiarte

Este es el tema que muchos tutoriales se saltan y que en este libro nos tomamos en serio: una casa llena de micrófonos, cámaras y sensores sabe muchísimo de ti: a qué hora te levantas, cuándo no hay nadie, qué ves en la tele, hasta cuándo discutes (por el volumen de la voz). Esa información vale oro para las empresas de publicidad, y es un riesgo si cae en manos equivocadas. La buena noticia es que TÚ decides cuánta privacidad quieres, con decisiones concretas.

Primera decisión: local contra nube. Muchos aparatos baratos funcionan mandando todo a los servidores del fabricante en otro país: cada vez que prendes un foco, alguien lo registra. Home Assistant te permite operar en local: tus dispositivos hablan con tu Raspberry dentro de tu casa y nada sale a internet a menos que tú lo permitas. Regla de oro: para cámaras, micrófonos y cerraduras, prefiere siempre dispositivos y configuraciones locales. Para un foco de la cocina, la nube es menos crítica. Clasifica tus dispositivos por sensibilidad y actúa en consecuencia.

Segunda decisión: qué micrófonos hay en tu casa y quién los escucha. Las bocinas inteligentes escuchan siempre esperando su palabra de activación ("Alexa", "Oye Google"). Eso es normal en su diseño, pero revisa periódicamente el historial de grabaciones en la app del fabricante, borra lo que no necesites y considera usar el asistente local de Home Assistant en las áreas privadas (recámaras) y dejar las bocinas comerciales solo en áreas comunes.

Tercera decisión: las cámaras. Ya lo dijimos pero lo repetimos porque es lo más importante: cámaras en exteriores y entradas, jamás en recámaras o baños. Si una cámara apunta a un área donde la familia espera privacidad, mejor que no exista. Además, graba en local siempre que puedas y cifra los respaldos.

Cuarta decisión: tu red. Una buena práctica de seguridad es separar tu red en dos: una red para tus aparatos personales (celular, computadora) y otra para los dispositivos inteligentes (la "red IoT"). Así, si algún foco barato trae un problema de seguridad, no puede brincar a tu computadora. La mayoría de los routers modernos permiten crear una red de invitados o una VLAN para esto en diez minutos.

Quinta decisión: actualizaciones y contraseñas. Cambia las contraseñas de fábrica de TODOS tus dispositivos (muchos ataques usan las contraseñas default que vienen en el manual), mantén actualizado el firmware y desactiva las funciones que no uses (si tu tele inteligente no necesita micrófono, apágalo).

Analogía cotidiana: tu casa inteligente es como tener empleados domésticos digitales: son utilísimos, pero no les darías las llaves de la caja fuerte ni les contarías tus secretos. Dales acceso solo a lo que necesitan para su trabajo, revisa qué hacen de vez en cuando y mantén lo más íntimo fuera de su alcance.

Error común de principiante: pensar "no tengo nada que ocultar, no me importa la privacidad". El problema no es lo que ocultas hoy, sino quién puede usar tus patrones de vida mañana: saber cuándo tu casa está vacía es información que un ladrón pagaría por tener. La privacidad no es paranoia, es higiene digital, como lavarte las manos.

Errores comunes y buenas prácticas: el resumen del veterano

Para cerrar la teoría, aquí va la lista que todo principiante debería pegar en el refrigerador:

Empieza chico y crece por cuartos. Dos focos y un enchufe te enseñan más que veinte aparatos sin configurar.

Un protocolo principal. Elige Zigbee o Matter como base y evita el zoológico de marcas incompatibles.

Nombra todo desde el día uno. "Foco sala 1" hoy te ahorra una hora de adivinanzas dentro de seis meses.

Respalda antes de experimentar. Un backup te cuesta un minuto; reconstruir todo te cuesta un fin de semana.

Lo crítico funciona sin internet. Alarmas, cerraduras y sensores importantes deben operar en red local.

Automatiza con la familia, no contra ella. Si a tu pareja le molesta, no es automatización, es un problema nuevo.

Mide antes de ahorrar. Los sensores de consumo primero, las reglas de ahorro después.

Seguridad por capas. Ningún aparato solo te protege; la combinación de disuasión, detección y respuesta sí.

Privacidad por diseño. Local primero, contraseñas cambiadas, cámaras solo donde corresponde.

Documenta tus automatizaciones. Un cuaderno (o una nota en el celular) con qué hace cada rutina te salva cuando algo se comporta raro dentro de un año.

Con la teoría dominada, es hora de ensuciarnos las manos: en la siguiente sección vas a ver automatizaciones reales, código real, que puedes copiar, entender y adaptar a tu casa.

Ejemplos paso a paso

Ejemplo 1: Automatización en YAML - "Luz del pasillo con sensor de movimiento (solo de noche)"

Este es el "hola mundo" de la domótica: entras al pasillo de noche y la luz se prende sola; sales y, después de dos minutos sin movimiento, se apaga sola. En Home Assistant las automatizaciones se pueden crear con el editor visual, pero por debajo se guardan en formato YAML, un lenguaje de configuración muy legible. Entender el YAML te permite copiar automatizaciones de internet, compartir las tuyas y hacer cosas que el editor visual no deja.

Supongamos que ya tienes dos entidades: "binary_sensor.movimiento_pasillo" (el sensor) y "light.luz_pasillo" (el foco). La automatización completa se ve así:

```
alias: "Pasillo - luz nocturna con movimiento"
description: "Prende la luz del pasillo si hay movimiento de noche; la apaga 2 min despues"
trigger:
  - platform: state
    entity_id: binary_sensor.movimiento_pasillo
    to: "on"
condition:
  - condition: sun
    after: sunset
    after_offset: "-00:30:00"
  - condition: sun
    before: sunrise
    before_offset: "00:30:00"
action:
  - service: light.turn_on
    target:
      entity_id: light.luz_pasillo
    data:
      brightness_pct: 40
  - wait_for_trigger:
      - platform: state
        entity_id: binary_sensor.movimiento_pasillo
        to: "off"
        for: "00:02:00"
  - service: light.turn_off
    target:
      entity_id: light.luz_pasillo
mode: restart
```

Vamos línea por línea para que no sea magia negra:

"alias" y "description" son el nombre y la descripción: ponles nombres claros porque dentro de un año tendrás docenas.

El "trigger" (disparador) usa la plataforma "state": se activa cuando el sensor de movimiento cambia su estado a "on" (detectó algo).

Las "condition" (condiciones) usan la posición del sol: solo funciona entre media hora antes del atardecer y media hora después del amanecer. Home Assistant sabe a qué hora pasa eso en tu ciudad gracias a la ubicación que configuraste al instalarlo. Así la luz no se prende a mediodía cuando no hace falta.

La "action" (acción) tiene tres pasos: primero prende la luz al 40 por ciento de brillo (suficiente para ver, sin deslumbrar a las 3 de la mañana); luego espera ("wait_for_trigger") a que el sensor lleve 2 minutos en "off" (sin movimiento); y entonces apaga la luz.

El "mode: restart" al final es un detalle importante: si mientras la automatización está esperando los dos minutos alguien vuelve a pasar por el pasillo, la automatización se reinicia y el conteo empieza de nuevo. Sin esto, la luz se apagaría con gente adentro.

Para usarla: en Home Assistant ve a Ajustes, Automatizaciones, Crear automatización, cambia al modo YAML (los tres puntitos), pega este código, cambia los nombres de tus entidades por los reales y guárdala. Pruébala caminando por el pasillo de noche. Si algo no jala, revisa la pestaña de "trazas" de la automatización: te muestra paso a paso qué se ejecutó y dónde se detuvo, es tu mejor herramienta de depuración.

Ejemplo 2: Script en Python - "Reporte nocturno de la casa por Telegram"

A veces quieres que tu casa te informe, no solo que actúe. Este script en Python se conecta a Home Assistant, revisa el estado de tus dispositivos y te manda un resumen a Telegram cada noche a las 22:00: cuántas luces quedaron prendidas, si alguna puerta quedó abierta, la temperatura de la casa y el consumo del día. Es un ejemplo real de cómo programar se combina con la domótica.

Primero necesitas dos cosas: un "token de acceso" de Home Assistant (lo creas en tu perfil de usuario, es como una contraseña solo para programas) y un bot de Telegram (lo creas gratis hablando con @BotFather en Telegram, te da un token). Guarda ambos en variables de ambiente, nunca escritos directo en el código.

```
import os
import requests
from datetime import datetime
```



```

HA_URL = "http://homeassistant.local:8123"
HA_TOKEN = os.environ["HA_TOKEN"]
TG_TOKEN = os.environ["TG_TOKEN"]
TG_CHAT_ID = os.environ["TG_CHAT_ID"]

headers = {
    "Authorization": f"Bearer {HA_TOKEN}",
    "Content-Type": "application/json",
}

def get_states():
    r = requests.get(f"{HA_URL}/api/states", headers=headers, timeout=10)
    r.raise_for_status()
    return r.json()

def resumen_nocturno(states):
    luces_on = [s for s in states
                 if s["entity_id"].startswith("light.")
                 and s["state"] == "on"]
    puertas_abiertas = [s for s in states
                        if s["entity_id"].startswith("binary_sensor.puerta")
                        and s["state"] == "on"]
    temps = [s for s in states
             if s["entity_id"].startswith("sensor.temperatura")]
    lineas = [f"Reporte nocturno - {datetime.now():%d/%m %H:%M}", ""]
    if luces_on:
        nombres = ", ".join(s["entity_id"] for s in luces_on)
        lineas.append(f"Luces prendidas ({len(luces_on)}): {nombres}")
    else:
        lineas.append("Luces prendidas: ninguna. Bien.")
    if puertas_abiertas:
        nombres = ", ".join(s["entity_id"] for s in puertas_abiertas)
        lineas.append(f"OJO - puertas abiertas: {nombres}")
    else:
        lineas.append("Puertas: todas cerradas.")
    for t in temps:
        nombre = t["entity_id"].replace("sensor.temperatura_", "")
        lineas.append(f"Temp {nombre}: {t['state']} C")
    return "\n".join(lineas)

def enviar_telegram(texto):
    url = f"https://api.telegram.org/bot{TG_TOKEN}/sendMessage"
    requests.post(url, json={"chat_id": TG_CHAT_ID, "text": texto},
                  timeout=10)

```

```
if __name__ == "__main__":
    estados = get_states()
    enviar_telegram(resumen_nocturno(estados))
    print("Reporte enviado.")
```

Cómo funciona, por partes:

Los "tokens" se leen de variables de ambiente para no dejar contraseñas en el código. En Linux las defines con "export HA_TOKEN=tu_token" antes de correr el script.

"get_states()" llama a la API REST de Home Assistant (la dirección /api/states) y trae la lista completa de entidades con sus estados actuales. Es la misma información que ves en el panel, pero en formato que un programa puede leer.

"resumen_nocturno()" filtra esa lista con comprensión de listas: busca entidades que empiecen con "light." y estén en "on", sensores de puerta abiertos y sensores de temperatura. Arma un texto legible con los resultados. Fíjate que el código está escrito para adaptarse a TUS nombres: si tus puertas se llaman "binary_sensor.puerta_principal", el filtro las encuentra solo.

"enviar_telegram()" usa la API de Telegram para mandarte el mensaje. Telegram es ideal para esto porque es gratis, llega al instante y no necesitas abrir puertos en tu router.

Para que corra solo cada noche, lo programas con cron en la Raspberry: "0 22 * * * /usr/bin/python3 /home/pi/reporte_nocturno.py". Así cada noche a las 10 recibes tu resumen sin mover un dedo.

Variaciones que puedes intentar después: que también te avise el consumo eléctrico del día sumando los sensores de energía, o que si detecta una puerta abierta te mande el mensaje de inmediato en lugar de esperar a las 10 (eso se hace mejor como automatización nativa de Home Assistant, pero el script te sirve para reportes complejos).

Ejemplo 3: Escenas en JSON - "Noche de películas" y "Despertar suave"

Las escenas guardan el estado deseado de varios dispositivos para activarlos de un golpe. Home Assistant las maneja internamente, pero entender su estructura en JSON te ayuda a crearlas, duplicarlas y compartirlas. Una escena es básicamente una lista que dice: "cuando me activen, deja cada aparato así".

Escena "Noche de películas": atenúa la sala, apaga la cocina, prende la tira LED en modo cine y ajusta el clima:

```

{
  "name": "Noche de peliculas",
  "entities": {
    "light.luz_sala_principal": {
      "state": "on",
      "brightness_pct": 20,
      "color_temp_kelvin": 2700
    },
    "light.luz_sala_lampara": {
      "state": "on",
      "brightness_pct": 10,
      "color_temp_kelvin": 2700
    },
    "light.luz_cocina": {
      "state": "off"
    },
    "light.tira_led_tv": {
      "state": "on",
      "brightness_pct": 60,
      "hs_color": [210, 80]
    },
    "climate.minisplit_sala": {
      "state": "on",
      "temperature": 24,
      "hvac_mode": "cool"
    },
    "media_player.television_sala": {
      "state": "on"
    }
  }
}

```

Escena "Despertar suave": simula un amanecer subiendo la luz poco a poco y abre las persianas:

```
{
  "name": "Despertar suave",
  "entities": {
    "light.luz_recamara": {
      "state": "on",
      "brightness_pct": 5,
      "color_temp_kelvin": 2200
    },
    "cover.persiana_recamara": {
      "state": "open",
      "position": 30
    },
    "media_player.bocina_recamara": {
      "state": "on",
      "volume_level": 0.25
    }
  }
}
```

Qué significa cada campo:

"name" es el nombre con el que la vas a llamar por voz ("activa la escena noche de películas") o desde una automatización.

"entities" es el diccionario de aparatos: cada clave es el ID de la entidad y su valor describe cómo debe quedar. "state" es prendido/apagado/abierto, "brightness_pct" es el brillo en porcentaje, "color_temp_kelvin" es la temperatura de color (2700 es cálida acogedora), "hs_color" es color en tono y saturación (210 es azul), "position" en persianas es qué tan abierta (0 cerrada, 100 abierta total).

Fíjate en el detalle de "Despertar suave": la luz empieza en 5 por ciento muy cálida (2200K, casi como velas) y la persiana solo al 30 por ciento. La idea es que una automatización dispare esta escena a las 6:30 y otra escena ("Despertar completo") a las 6:45 con más luz y la persiana al 100. Así el despertar es gradual y no te despierta de golpe la luz en la cara.

Cómo llevar esto a Home Assistant: ve a Ajustes, Automatizaciones y escenas, pestaña Escenas, Crear escena, agrega tus dispositivos y ajusta cada uno como quieres que quede, y guárdala con el nombre que elegiste. El JSON de arriba es lo que el sistema guarda por dentro; si algún día necesitas editarlo a mano está en el archivo scenes.yaml de tu configuración.

Combínalo todo: crea una automatización cuyo disparador sea la frase "noche de películas" en tu asistente de voz y cuya acción sea activar la escena. O programa "Despertar suave" de

lunes a viernes a las 6:30. Las escenas son los bloques de construcción y las automatizaciones son las que deciden cuándo usar cada bloque.

Práctica

Ejercicio 1: Diseña el plano de automatización de tu casa

Toma papel y lápiz (o una app de notas) y dibuja el plano de tu casa cuarto por cuarto. Para cada cuarto anota: qué luces hay, qué aparatos consumen más energía, qué te gustaría automatizar y por qué. Luego clasifica cada idea en tres niveles: nivel 1 (hazlo este mes: barato y fácil, como focos y enchufes), nivel 2 (próximos tres meses: sensores y cámaras) y nivel 3 (cuando el sistema ya funcione: climatización avanzada, persianas). Al final escribe tu lista de compras del nivel 1 con precios aproximados.

Solución paso a paso: primero, recorre tu casa físicamente con la libreta en la mano; es fácil olvidar el foco del pasillo o el enchufe detrás del sillón si lo haces de memoria. Segundo, para cada cuarto pregúntate "¿qué hago aquí todos los días que una máquina podría hacer por mí?": prender la luz al entrar, apagar todo al salir, ajustar el clima. Tercero, marca con una estrella las ideas que ahorran energía o dan seguridad, porque esas son las que se pagan solas. Cuarto, arma la lista de compras del nivel 1 siguiendo esta receta probada: dos focos inteligentes blancos regulables para el cuarto donde más estás, un enchufe inteligente con medición de consumo para el aparato más gastalón, y un sensor de movimiento para el pasillo o el baño. Quinto, verifica compatibilidad antes de comprar: que todo hable el mismo protocolo (idealmente Zigbee o Matter) y que tenga integración con Home Assistant (revísalo en la página de integraciones). Un ejemplo de resultado: sala (2 focos + tira LED, nivel 1), recámara (foco + sensor de movimiento, nivel 1), cocina (enchufe medidor para el refrigerador viejo, nivel 1), entrada (cámara con video timbre, nivel 2), toda la casa (controlador infrarrojo para los minisplits, nivel 3). Si tu lista del nivel 1 cuesta más de lo que pensabas, recorta a un solo cuarto: es mejor un cuarto perfectamente automatizado que cinco a medias.

Ejercicio 2: Instala Home Assistant y agrega tu primer dispositivo

Instala Home Assistant en una Raspberry Pi (o en una computadora vieja) siguiendo el proceso de instalación, crea tu usuario, configura tu ubicación y haz tu primer respaldo. Luego agrega tu primer dispositivo real: puede ser un foco inteligente, un enchufe o incluso una integración de software (como el clima de tu ciudad o tu calendario). Renómbralo con un nombre claro y asígnalo a su área.

Solución paso a paso: primero, descarga Raspberry Pi Imager en tu computadora y la imagen oficial de Home Assistant para tu modelo de Raspberry; grábala en una tarjeta microSD de buena calidad. Segundo, conecta la Raspberry a tu router con cable de red (para la instalación inicial el cable evita dolores de cabeza), inserta la tarjeta y conéctala a la corriente. Tercero, espera unos 10 a 20 minutos y abre <http://homeassistant.local:8123> en tu navegador; si no abre, busca la IP de la Raspberry en tu router. Cuarto, sigue el asistente: crea tu usuario con contraseña fuerte, pon tu ubicación exacta (esto alimenta las automatizaciones con el sol) y elige tu unidad de temperatura. Quinto, ve a Ajustes, Sistema, Respaldos y crea tu primer respaldo manual; descárgalo a tu computadora. Sexto, agrega tu primer dispositivo: ve a Ajustes, Dispositivos, Agregar integración, busca la marca de tu aparato y sigue los pasos (normalmente es ponerlo en modo de emparejamiento y darle a "descubrir"). Séptimo, en cuanto aparezca, entra a su configuración, cámbiale el nombre a algo humano ("Foco sala principal" en lugar de "light.luz_3") y asígnalo al área "Sala". Octavo, Pruébalo desde el panel: préndelo, apágalo, cambia el brillo. Si algo falla, revisa que el aparato y la Raspberry estén en la misma red y que el aparato tenga corriente (recuerda: si el apagador físico está apagado, el foco inteligente está muerto). Repite el proceso con un segundo dispositivo para agarrar práctica; a partir del tercero ya lo harás dormido.

Ejercicio 3: Crea la automatización "Bienvenida al llegar a casa"

Crea una automatización que detecte cuando llegas a casa (usando la ubicación de tu celular) y haga tres cosas: prenda la luz de la entrada, ajuste el clima a temperatura cómoda y te mande una notificación de "bienvenido". Debe funcionar solo si está oscuro afuera y no debe dispararse si solo pasas caminando por la banqueta.

Solución paso a paso: primero, instala la app oficial de Home Assistant en tu celular e inicia sesión con tu usuario; la app reporta tu ubicación al sistema (verifica en los ajustes de la app que el sensor de ubicación esté activado y con permiso "siempre"). Segundo, define tu "zona casa": en Ajustes, Personas, Zonas, ajusta el radio de la zona "Home" a unos 100 o 150 metros; un radio muy chico hace que la detección falle, uno muy grande la dispara desde la tienda de la esquina. Tercero, crea la automatización: disparador de tipo "zona", cuando tu persona "entre" a la zona "home". Cuarto, agrega una condición de "sol": solo después del atardecer (o usa un sensor de luminosidad si tienes uno, que es más preciso en días nublados). Quinto, agrega las acciones: prender "light.luz_entrada" al 80 por ciento, poner "climate.minisplit_sala" en 24 grados modo cool, y enviar una notificación a tu celular con el servicio "notify.mobile_app_tu_celular". Sexto, para evitar que se dispare al pasar por la banqueta, agrega una condición extra: que tu celular haya estado "fuera" de la zona por lo menos 10 minutos antes (hay una condición de "estado" con duración para esto), o combina con que se conecte a tu WiFi de casa. Séptimo, pruébala: sal a dar la vuelta a la manzana de

noche y regresa; revisa la traza de la automatización para confirmar que cada paso se ejecutó. Si la luz se prende pero el clima no, el problema casi siempre es el nombre de la entidad del clima: revísalo en Herramientas de desarrollo, Estados. Ajusta el radio de la zona según tu experiencia de una semana: la detección por GPS consume batería, así que no pongas el radio diminuto "para que sea exacta"; 100 metros es el punto dulce entre precisión y batería.

Ejercicio 4: Monta el "modo vacaciones" (simulacro de presencia)

Crea un sistema que, cuando actives el modo vacaciones, haga que tu casa parezca habitada: luces que se prenden y apagan a horarios variados cada noche, persianas que se mueven una vez al día y una notificación diaria confirmando que todo está en orden. Debe incluir una forma fácil de activarlo y desactivarlo.

Solución paso a paso: primero, crea un interruptor virtual que represente el modo: en Ajustes, Dispositivos, Ayudantes, crea un "interruptor" (input_boolean) llamado "Modo vacaciones". Este será tu botón maestro. Segundo, crea tres automatizaciones separadas, todas con la condición de que el modo vacaciones esté prendido: la primera, "luces de noche aleatorias", con disparador de hora a las 19:00, que prenda la luz de la sala y la del pasillo, y con una segunda automatización a las 22:30 que las apague; para que no sea mecánico, usa un pequeño truco: en lugar de hora fija usa la puesta de sol con un desplazamiento aleatorio ("after_offset" con minutos distintos cada día no es aleatorio de verdad, así que mejor crea dos o tres automatizaciones con horarios ligeramente distintos y actívalas por día de la semana). Tercero, la automatización de persianas: disparador a las 9:00, condición de modo vacaciones prendido, acción de abrir las persianas al 50 por ciento, y otra a las 18:00 que las cierre. Cuarto, la notificación diaria: disparador a las 12:00, condición de modo vacaciones, acción que te mande un mensaje con el estado de puertas y cámaras (puedes reutilizar la lógica del script de Python del Ejemplo 2 o hacer una automatización que revise los sensores). Quinto, crea un botón en tu panel principal ligado al interruptor "Modo vacaciones" para prenderlo y apagarlo con un toque, y comparte el acceso con alguien de confianza por si hay que desactivarlo a distancia. Sexto, haz una prueba de un día completo antes de irte: activa el modo un sábado, sal de casa y pide a un familiar que pase en la noche a verificar que las luces se prendieron. Séptimo, regla de seguridad importante: el modo vacaciones NUNCA debe desactivar tus cámaras ni tus sensores de alarma; solo simula presencia con luces y persianas, la seguridad sigue al cien. Cuando regreses, apaga el modo y revisa el historial para confirmar que todo funcionó como debía.

Ejercicio 5: Auditoría de privacidad de tu hogar inteligente

Haz una revisión completa de privacidad: lista todos tus dispositivos inteligentes, clasifícalos por nivel de sensibilidad (micrófonos y cámaras = alta, sensores = media, focos = baja), verifica cuáles mandan datos a la nube, cambia todas las contraseñas de fábrica, revisa los historiales de grabación de voz y configura una red separada para tus dispositivos IoT.

Solución paso a paso: primero, haz el inventario: abre Home Assistant y exporta la lista de integraciones y dispositivos, y agrégale los aparatos que no están integrados (bocinas, teles). Anota marca, modelo y cómo se conecta cada uno. Segundo, clasifica: marca en rojo los que tienen micrófono o cámara, en amarillo los sensores que revelan patrones de vida (presencia, consumo) y en verde los actuadores simples (focos, enchufes sin medición). Tercero, investiga la nube: para cada aparato en rojo o amarillo, busca en su app o manual si funciona sin internet (desconecta el cable del router cinco minutos y prueba: lo que siga funcionando es local, lo que muera depende de la nube). Anota el resultado. Cuarto, cambia contraseñas: entra a la configuración de cada dispositivo y cambia la contraseña default por una única y fuerte; guarda todo en un gestor de contraseñas. Quinto, limpia historiales: en las apps de Alexa, Google o Siri revisa el historial de grabaciones de voz, escucha algunas por curiosidad (te vas a sorprender de lo que hay) y borra el historial completo; activa el borrado automático cada 3 meses si la app lo permite. Sexto, cámaras: verifica que ninguna apunte a áreas privadas ni a la casa del vecino, que graben en local y que el acceso remoto tenga autenticación en dos pasos. Séptimo, la red separada: entra a tu router, crea una red WiFi de invitados o una VLAN llamada "IoT", mueve ahí todos los dispositivos inteligentes y deja tus celulares y computadoras en la red principal; prueba que Home Assistant siga viendo los dispositivos (si tu Raspberry está por cable en la red principal, puede que necesites una regla en el router para que las redes se hablen solo en lo necesario). Octavo, escribe tus hallazgos en una hoja: qué aparatos dependen de la nube, cuáles vas a reemplazar por versiones locales cuando puedas y qué fecha le pones a la próxima auditoría (cada seis meses es buena práctica). Este ejercicio no se hace una vez: la privacidad se mantiene, no se instala.

Ejercicio 6: Calcula el ahorro real de automatizar tu clima

Usa datos reales para decidir si vale la pena automatizar tu climatización: mide el consumo de tu minisplit o clima durante una semana normal con un enchufe medidor (o con los datos de tu medidor de luz), calcula cuánto gastas al mes, diseña la automatización de ahorro (apagar sin presencia + pre-enfriar antes de llegar) y estima el ahorro mensual y el tiempo en que se paga la inversión.

Solución paso a paso: primero, instala un enchufe inteligente con medición en tu clima (si es de más de 15 amperes usa un medidor de riel o un sensor de gancho en el centro de carga; no

sobrecargues un enchufe casero). Segundo, mide una semana NORMAL sin cambiar tus hábitos: anota el consumo diario en kWh que reporta Home Assistant en la pestaña de Energía. Tercero, calcula tu costo: multiplica los kWh semanales por 4.3 para estimar el mes, y multiplícalo por el precio de tu tarifa eléctrica (revisa tu recibo: en México la tarifa DAC es mucho más cara que la básica, y el aire acondicionado es lo que más empuja a la tarifa cara). Ejemplo: si tu minisplit consume 35 kWh a la semana, son unos 150 kWh al mes; a 3 pesos el kWh son 450 pesos mensuales solo del clima. Cuarto, estima el desperdicio: revisa en el historial cuántas horas el clima trabajó sin nadie en casa (cruza el consumo con el sensor de presencia o con tus horarios); si de 10 horas diarias, 4 fueron con casa vacía, el 40 por ciento es desperdicio potencial. Quinto, diseña la automatización: "si no hay presencia en 30 minutos, sube 3 grados el setpoint; cuando mi ubicación esté a 15 minutos, regresa a la temperatura cómoda". Sexto, calcula el ahorro esperado: si eliminas la mitad del desperdicio, en el ejemplo ahorrarías unos 180 pesos al mes; un controlador infrarrojo inteligente cuesta alrededor de 600 a 900 pesos, así que se paga solo en 4 o 5 meses, y después es ganancia. Séptimo, implementa y mide de nuevo durante dos semanas comparando con la semana base; ajusta los tiempos (quizá 30 minutos sin presencia es mucho o poco para tu rutina). Octavo, documenta: guarda una tabla simple de "antes y después" porque estos números son tu mejor argumento cuando alguien en casa dude de si "esas cosas inteligentes sirven para algo". Los datos convencen más que las promesas.

Resumen de la unidad

Llegaste al final de la unidad 8 y tu casa ya no debería verse igual que al principio: ahora la ves como un sistema que puedes diseñar, no como un conjunto de aparatos sueltos. Vamos a repasar los puntos clave para que se queden grabados.

La domótica es hacer que tu casa trabaje por ti mediante dispositivos conectados que actúan solos según reglas que tú defines. No necesitas remodelar ni gastar una fortuna: un foco inteligente se enrosca como uno normal y un sensor se pega con cinta. La automatización de verdad es invisible (pasa sola sin que le piques a nada), confiable (lo importante funciona sin internet) y tuya (tú decides las reglas y controlas tus datos).

Todo sistema de hogar inteligente tiene tres piezas: los asistentes de voz para hablarle a tu casa, el hub o sistema central que coordina todo, y los protocolos que son el idioma común. Los cuatro protocolos que debes conocer son WiFi (fácil y barato, pero satura tu red), Zigbee (bajo consumo, red en malla, ideal para sensores y focos), Z-Wave (confiable y con poca interferencia) y Matter (el estándar universal del futuro). La lección de compra más importante de la unidad: elige un protocolo principal y evita el zoológico de marcas incompatibles.

Home Assistant es el cerebro recomendado: software libre y gratuito que instalas en una Raspberry Pi o computadora dedicada, que integra miles de dispositivos de cientos de marcas y que funciona en tu red local sin depender de la nube. Sus conceptos base son las entidades (todo lo que puede ver o controlar), las áreas (organización por cuartos) y las automatizaciones. Tus primeros pasos siempre: contraseña fuerte, respaldo inmediato, nombres claros para cada entidad y crecer poco a poco.

La iluminación es la mejor puerta de entrada: focos inteligentes, tiras LED y escenas. Recuerda el detalle del apagador físico (si lo apagan, el foco inteligente muere), la temperatura de color (cálida para relajarse, fría para concentrarse) y la iluminación circadiana que sigue al sol. Las escenas convierten aparatos sueltos en experiencias de un toque: "noche de películas", "despertar suave".

La climatización es donde más dinero se ahorra porque el clima es lo que más luz consume. La fórmula ganadora combina sensor de presencia, sensor de temperatura y tu ubicación: ahorrar cuando no hay nadie y pre-enfriar antes de que llegues. Los enchufes con medición revelan a los vampiros energéticos, y la regla de oro es medir antes de automatizar.

La seguridad se construye por capas: disuasión (luces que simulan presencia), detección (sensores, cámaras, timbres con video) y respuesta (sirenas, alertas, luces). Las cámaras van en exteriores y accesos, nunca en áreas privadas; las cerraduras inteligentes llevan llave física de respaldo; y lo crítico debe funcionar sin internet, en red local.

Las automatizaciones tienen tres partes: disparador (cuándo), condición (solo si) y acción (entonces haz). Con eso armas las rutinas clásicas: buenos días, me voy, buenas noches, noche de películas y modo vacaciones. Cuida que tus automatizaciones no peleen entre sí e impleméntalas de una en una con el consenso de tu familia.

La voz más la IA llevan tu casa al siguiente nivel: de órdenes rígidas a intenciones en lenguaje natural. Pero la IA necesita barandales: permisos mínimos, confirmación para acciones importantes y empezar con acciones inofensivas.

Y la privacidad no es opcional: prefiere dispositivos locales sobre los que mandan todo a la nube, cambia las contraseñas de fábrica, revisa y borra historiales de voz, separa tu red IoT de tu red personal y audita tu sistema cada seis meses. Tu casa no debe espiarte.

Los errores de principiante que ya sabes evitar: comprar todo de golpe, mezclar protocolos sin plan, no renombrar entidades, no hacer respaldos, depender de la nube para la seguridad y automatizar sin consultar a la familia. Las buenas prácticas que te llevas: empezar chico, un protocolo base, medir antes de ahorrar, seguridad por capas, privacidad por diseño y documentar tus rutinas.

En la siguiente unidad seguiremos construyendo sobre esta base. Por ahora, tu misión es clara: elige tu primer cuarto, compra tu primer par de focos y enciende tu primera automatización. Bienvenido a tu casa inteligente.

Un último consejo antes de cerrar: guarda este archivo y tu cuaderno de automatizaciones como tu bitácora de domótica. Cada vez que agregues un dispositivo nuevo, anota la fecha, el modelo y por qué lo compraste; cada vez que una automatización te falle, anota qué pasó y cómo la arreglaste. En seis meses esa bitácora va a valer oro, porque vas a tener el mapa completo de tu casa inteligente y vas a poder ayudar a alguien más a empezar. La domótica se aprende haciendo, se domina documentando y se disfruta compartiendo. Nos vemos en la siguiente unidad.

UNIDAD 9: Robótica y drones

Introducción

Hay una escena que se repite en miles de casas: un robot aspiradora recorre la sala esquivando las patas de las sillas, un dron despegua en el parque para tomar una foto desde 50 metros de altura, y en la cocina un brazo robótico de juguete mueve piezas de un lado a otro siguiendo las órdenes que tú mismo programaste. Nada de esto es ciencia ficción. Es tu vida cotidiana en 2026, y lo más interesante es que todo esto lo puedes entender, armar y programar tú mismo con piezas que cuestan menos que unos tenis.

Esta unidad es el puente entre el mundo del software que ya dominas y el mundo físico. Hasta ahora has aprendido a programar: variables, funciones, lógica, datos. Eso es poderosísimo, pero vivía dentro de la pantalla. La robótica es lo que pasa cuando tu código sale de la pantalla y empieza a mover cosas reales: ruedas que giran, motores que zumban, sensores que "sienten" el mundo y drones que desafían la gravedad. Es el momento en que dejas de ser solo programador y te conviertes en creador de máquinas.

Y ojo, no necesitas un laboratorio millonario ni un título de ingeniero para empezar. Con una tarjeta Arduino que cabe en la palma de tu mano, dos motores de corriente directa, un par de sensores que cuestan lo de una torta y un chasis de acrílico, puedes construir tu primer robot seguidor de línea en una tarde. Con un dron de juguete de menos de mil pesos puedes aprender los principios del vuelo que usan los drones profesionales de miles de dólares. Esta unidad te va a llevar de cero a ese punto, paso a paso, sin saltarte nada.

Los robots ya viven entre nosotros (pero no como en las películas)

Olvídate por un momento de los robots de las películas: esos humanoides metálicos que hablan perfecto español y a veces quieren conquistar el mundo. Los robots reales son mucho menos dramáticos y muchísimo más útiles. Un robot, en términos simples, es una máquina que percibe su entorno con sensores, toma decisiones con un "cerebro" (que puede ser un programa que tú escribiste) y actúa sobre el mundo con motores o mecanismos. Eso es todo. Con esa definición, tu lavadora es medio robot, tu carro moderno está lleno de pequeños robots y el brazo que solda puertas en una fábrica de autos es un robot de tiempo completo.

Los robots ya están en todas partes, aunque no los veas. En las fábricas de todo el mundo hay más de 4 millones de robots industriales armando carros, soldando piezas y empacando cajas a una velocidad y precisión que ningún humano puede igualar. En los hospitales hay robots

que ayudan a los cirujanos a operar con una precisión de fracciones de milímetro, y otros que llevan medicinas por los pasillos. En tu casa, la aspiradora robot mapea tu sala y la limpia sola. En el campo, tractores guiados por GPS siembran con precisión centimétrica. Y en el cielo, los drones hacen repartos, inspeccionan puentes y toman las fotos aéreas que antes solo se podían conseguir rentando un helicóptero.

¿Por qué te cuento todo esto? Porque la robótica ya no es una carrera del futuro: es una habilidad del presente. Las empresas buscan gente que sepa programar máquinas físicas, y la buena noticia es que la barrera de entrada nunca había estado tan baja. Las piezas son baratas, los tutoriales abundan y las comunidades en línea te ayudan con cualquier duda. Si ya sabes programar, tienes el 50 por ciento del camino recorrido. Esta unidad te da el otro 50 por ciento: el hardware, los sensores, los motores y la lógica para que el mundo físico obedezca a tu código.

Por qué aprender robótica te cambia la forma de ver la tecnología

Hay algo mágico que pasa la primera vez que tu código mueve algo real. Escribes unas líneas, las subes a una tarjetita, y de pronto un carrito empieza a seguir una línea negra dibujada en el piso, solo, sin que lo toques. En ese momento tu cerebro hace clic: entiendes que programar no es mover píxeles, es darle instrucciones a la materia. Esa sensación no la da ningún programa de computadora, por bueno que sea.

Además, la robótica te obliga a pensar de una forma que el software puro no exige. En software, si tu programa falla, lo corriges y lo vuelves a correr. En robótica, si tu programa falla, el robot se estrella contra la pared. Tienes que pensar en el mundo real: la fricción, la batería que se acaba, el sensor que se confunde con la luz del sol, el cable que se suelta con la vibración. Aprendes a probar por partes, a poner "redes de seguridad" en tu código (¿qué hace el robot si pierde la línea? ¿se detiene o sigue de frente al vacío?) y a depurar con todos tus sentidos: escuchas el motor, ves el LED, tocas si algo se calienta. Esa disciplina te vuelve mejor programador en todo lo demás.

Y hay una razón práctica y directa: la robótica combina todo lo que has aprendido en este curso. Usa programación (Unidad de Python y de JavaScript), usa lógica y matemáticas, usa redes e Internet cuando conectas tu robot a la nube (IoT), usa inteligencia artificial cuando le pones visión por computadora, y usa datos cuando analizas lo que registran sus sensores. Es la unidad donde todo se junta. Por eso en Lucas EGA Academy la pusimos aquí, en la recta final: porque ya tienes las bases para entenderla de verdad, no solo para copiar proyectos sin saber qué hacen.

Qué vas a construir y aprender en esta unidad

Vamos al grano, esto es lo que te espera. Primero vas a entender qué es un robot de verdad y los tipos que existen: industriales, de servicio, educativos y humanoides, con ejemplos de cada uno para que sepas en qué mundo vive cada tipo. Después vas a desarmar mentalmente un robot para conocer su anatomía: los motores que son sus músculos, los sensores que son sus sentidos, los controladores que son su cerebro y el chasis que es su esqueleto. Vas a ver que un robot es, en el fondo, un cuerpo con sentidos y músculos obedeciendo a un programa.

Luego viene lo bueno: vas a programar tu primer robot con Arduino, la tarjeta favorita de los makers de todo el mundo. Vas a construir (aunque sea en simulador, si todavía no tienes las piezas) los dos proyectos clásicos con los que todo roboticista empieza: el robot seguidor de línea, que persigue una línea negra como si fuera un tren sobre rieles, y el robot evasor de obstáculos, que usa un sensor ultrasónico como "murciélago" para detectar paredes y esquivarlas. Te voy a dar el código completo, explicado línea por línea, y también los errores típicos para que no pierdas horas atorado donde todos nos atoramos la primera vez.

Después despegamos, literalmente: vas a entrar al mundo de los drones. Qué tipos existen (de juguete, de fotografía, de carreras FPV, profesionales), de qué partes están hechos, cómo demonios se mantienen en el aire y cómo se controlan. También vas a aprender la parte seria: la normativa básica y el vuelo responsable, porque un dron mal usado puede causar un accidente de verdad y en México (como en casi todo el mundo) hay reglas que debes conocer antes de despegar.

También vas a descubrir para qué se usan los drones en el mundo real: fotografía y video, agricultura de precisión, inspección de infraestructura (puentes, torres, paneles solares) y reparto de paquetes. Y para cerrar con broche de oro, dos temas que son el futuro: la visión por computadora (cómo un robot "ve" con una cámara y detecta objetos, con un ejemplo real en Python y OpenCV que vas a poder correr en tu computadora) y la combinación ganadora de robótica más inteligencia artificial más Internet de las cosas: el robot conectado que no solo se mueve, sino que aprende y se comunica.

Al terminar esta unidad no solo vas a saber qué es un robot: vas a haber programado comportamientos robóticos reales, vas a entender cómo vuela un dron y vas a tener las bases para seguir construyendo por tu cuenta.

Explicación detallada

Qué es un robot, de verdad y sin películas

Vamos a ponernos formales por un minuto, porque la definición importa: un robot es una máquina programable capaz de percibir su entorno a través de sensores, procesar esa información para tomar decisiones y actuar sobre el mundo físico mediante actuadores (motores, mecanismos, luces, sonidos). Tres palabras lo resumen todo: percibir, decidir, actuar. Si una máquina hace esas tres cosas en un ciclo continuo, es un robot. Si solo hace una o dos, es otra cosa: una lavadora que sigue un programa fijo sin "ver" nada es más bien un autómatas; un carro a control remoto sin sensores es un juguete teledirigido, no un robot, porque quien decide eres tú, no él.

La analogía que nunca falla es la del cuerpo humano, porque un robot es básicamente un cuerpo artificial simplificado. Tus ojos y tu piel son los sensores: captan luz, temperatura, presión. Tu cerebro es el controlador: recibe esas señales, decide y manda órdenes. Tus músculos son los actuadores: ejecutan las órdenes y mueven tu cuerpo. Tu esqueleto es el chasis: le da forma y sostiene todo. Cuando entiendas esta correspondencia, cada robot que veas se vuelve legible: "ah, ese es su sensor, ese es su motor, ahí va su cerebro".

Ahora, los tipos de robots. No todos los robots se parecen ni sirven para lo mismo, y conocer las familias te ayuda a ubicar en qué liga juega cada proyecto que veas.

Los robots industriales son los obreros incansables de las fábricas. Brazos articulados gigantes que sueldan, pintan, atornillan y empacan con una precisión de décimas de milímetro, 24 horas al día, sin cansarse ni distraerse. Los ves en las plantas armadoras de autos: esos brazos naranjas o amarillos que se mueven con una coreografía perfecta. Son caros (cuestan lo de varias casas), potentes y peligrosos: trabajan encerrados en jaulas de seguridad porque un golpe de un brazo industrial puede matar. Su programación es repetitiva y exacta: hacen lo mismo millones de veces. La lección que te dejan es la precisión y la repetibilidad.

Los robots de servicio trabajan cerca de los humanos y para los humanos. La aspiradora robot que mapea tu casa, el robot que lleva bandejas en algunos restaurantes, los robots que desinfectan hospitales con luz ultravioleta, los que cortan el pasto solos. Son más baratos, más seguros y más "amables" que los industriales, porque están diseñados para convivir con gente. Su reto principal es navegar en entornos desordenados y cambiantes: tu sala no es una fábrica, hay juguetes tirados, mascotas que se cruzan y muebles que cambias de lugar. Por eso usan muchos sensores y algoritmos de mapeo.

Los robots educativos son los que te interesan directamente ahorita. Kits como los basados en Arduino, los sets de robótica para escuelas, los carritos seguidores de línea: están diseñados para aprender. Son baratos, se arman con las manos, se programan con código sencillo y están hechos para que te equivoques sin consecuencias graves (un carrito de juguete no te va

a romper un dedo). No te dejes engañar por su apariencia modesta: los principios que aprendes con ellos (leer sensores, controlar motores, tomar decisiones con lógica) son exactamente los mismos que usan los robots industriales y los drones profesionales. Es como aprender a manejar en un vocho: el principio es el mismo que en un tráiler.

Los robots humanoides son los que parecen personas: cabeza, torso, dos brazos, dos piernas. Son los más difíciles de construir porque caminar en dos piernas es un problema de equilibrio brutalmente complejo (tu cerebro lo resuelve sin que te des cuenta, pero programarlo es otra historia). Se usan en investigación, en atención al público y como plataforma para probar inteligencia artificial. Son caros y todavía torpes comparados con un humano, pero avanzan rapidísimo.

Errores comunes de principiante en este tema: primero, creer que "robot" significa "humanoide que habla". La mayoría de los robots no tienen forma humana ni la necesitan: un dron es un robot, una aspiradora robot es un robot, un brazo industrial es un robot. Segundo, pensar que necesitas piezas caras para empezar. No: un kit básico de Arduino con dos motores te enseña el 80 por ciento de los fundamentos. Tercero, querer construir el robot definitivo en el primer intento. Los profesionales construyen prototipos feos y simples primero, y los mejoran por versiones. Tu primer robot va a estar chueco y se va a atorar: eso no es fracaso, es la versión 1.0.

Buenas prácticas: antes de armar nada, dibuja tu robot en papel y escribe qué debe hacer en una lista de comportamientos ("si ve línea, avanzar; si pierde la línea, girar buscando"). Esa lista es tu programa antes de ser código. Y documenta todo: tómale fotos a tu armado, guarda cada versión del código. Cuando algo falle (y va a fallar), esas fotos y versiones son tu máquina del tiempo para saber qué cambió.

Anatomía de un robot: músculos, sentidos, cerebro y esqueleto

Ya tienes la analogía del cuerpo humano; ahora vamos a abrir el cuerpo del robot pieza por pieza. Todo robot, desde el carrito de juguete hasta el brazo industrial, está hecho de cuatro sistemas: actuadores, sensores, controlador y chasis (más la energía que los alimenta, que es como la comida).

Los actuadores son los músculos: convierten energía eléctrica en movimiento. El más común en robótica educativa es el motor de corriente directa (motor DC): le das voltaje y gira; le das más voltaje y gira más rápido; inviertes los cables y gira al revés. Simple y barato. Su primo más refinado es el servomotor: no gira libremente, sino que se mueve a un ángulo exacto que tú le ordenas (0 a 180 grados, por ejemplo). Los servos son los que mueven los brazos de los robots de juguete y las pinzas que agarran cosas: les dices "muévete a 90 grados" y ahí se

quedan, con fuerza. Y están los motores a pasos (stepper), que giran en pasos discretos muy precisos: son los que mueven las impresoras 3D y los robots que necesitan posicionamiento exacto. Piensa en los actuadores como los músculos: los DC son como tus piernas corriendo (potencia y velocidad), los servos son como tus dedos (posición precisa).

Los sensores son los sentidos: convierten el mundo físico en números que tu programa puede leer. El sensor ultrasónico es el más famoso para empezar: manda un "grito" de ultrasonido y mide cuánto tarda el eco en regresar, como un murciélago. Con ese tiempo calcula la distancia a los obstáculos. El sensor infrarrojo de línea detecta si hay una línea negra bajo el robot: emite luz infrarroja y mide cuánta se refleja (el blanco refleja mucho, el negro casi nada). Hay sensores de luz, de temperatura, de humedad, acelerómetros (sienten el movimiento y la inclinación, como tu oído interno), giroscopios, sensores de distancia láser y cámaras. Cada sensor responde una pregunta del mundo: "¿qué tan lejos está la pared?", "¿estoy sobre la línea?", "¿me estoy inclinando?". Sin sensores, el robot es ciego y sordo: solo puede repetir movimientos a ciegas, como caminar con los ojos cerrados.

El controlador es el cerebro: una pequeña computadora que lee los sensores, ejecuta tu programa y manda órdenes a los actuadores. En robótica educativa el rey es Arduino: una tarjeta barata, con un microcontrolador, pines para conectar sensores y motores, y un lenguaje de programación sencillo basado en C++. Arduino no es una computadora completa: no tiene pantalla ni sistema operativo; es como un cerebro muy enfocado que hace una sola cosa muy bien: correr tu programa en un ciclo infinito, rapidísimo. Su primo más potente es la Raspberry Pi, que sí es una mini computadora con Linux: sirve cuando tu robot necesita más poder, como procesar video de una cámara. Regla práctica: Arduino para controlar motores y leer sensores; Raspberry Pi (o una computadora) para visión por computadora e inteligencia artificial.

El chasis es el esqueleto: la estructura que sostiene todo. Puede ser acrílico cortado con láser, madera, piezas impresas en 3D o hasta cartón en tu primer prototipo. No lo subestimes: un chasis chueco o flojo arruina el mejor programa del mundo, porque las ruedas no tocan el piso parejo o los sensores apuntan a donde no deben. Y la energía es la comida: casi siempre baterías recargables. Aquí hay una lección de física que todo principiante aprende a la mala: los motores comen mucha corriente, y si alimentas los motores desde la misma fuente débil que el controlador, cada vez que los motores arrancan el voltaje se cae y el cerebro se reinicia. Por eso los kits usan un driver de motores (como el famoso L298N o el TB6612): es el intermediario que maneja la potencia bruta de los motores mientras el Arduino solo le dice "gira para adelante a tal velocidad".

Errores comunes de principiante en anatomía: conectar los motores directo a los pines del Arduino (los pines no dan suficiente corriente y puedes quemar la tarjeta; siempre usa un driver). No poner tierra común entre la batería de los motores y el Arduino (sin tierra

compartida, las señales no tienen referencia y nada funciona: todos los "negativos" deben unirse). Elegir sensores sin leer su rango (un ultrasónico que mide de 2 cm a 400 cm no sirve para detectar algo a 1 cm). Y el clásico: armar todo el robot completo antes de probar nada, para luego no saber qué parte falla.

Buenas prácticas de anatomía: prueba cada pieza por separado antes de armar el robot completo (primero haz girar un motor, luego lee un sensor, luego combina). Usa cables de colores con un código fijo (rojo para positivo, negro para tierra) y no lo cambies nunca: te va a ahorrar horas de confusión. Fija bien los sensores con su orientación correcta (un ultrasónico chueco mide la pared de al lado, no la de enfrente). Y calcula tu consumo de energía: suma cuánta corriente piden tus motores y elige una batería que la aguante con margen; un robot que se apaga a los cinco minutos no sirve para probar nada.

Tu primer robot programable: Arduino y los dos proyectos clásicos

Aquí es donde la teoría se convierte en movimiento. Arduino es una tarjeta con un microcontrolador que ejecuta tu programa una y otra vez en un ciclo infinito, miles de veces por segundo. Su programa tiene una estructura fija de dos funciones: `setup()`, que se ejecuta una sola vez al inicio (ahí configuras pines y enciendes cosas), y `loop()`, que se repite para siempre (ahí va la lógica de tu robot: leer sensores, decidir, mover motores). Todo robot con Arduino es un loop eterno de percibir, decidir y actuar. Grábatelo: `setup` una vez, `loop` para siempre.

Antes de tocar código, entiende la idea central con una analogía: el robot es como un chofer novato que solo puede hacer tres preguntas a sus copilotos. En el seguidor de línea, el robot lleva dos sensores infrarrojos abajo, apuntando al piso: uno a la izquierda y uno a la derecha. Cada sensor responde "veo blanco" o "veo negro". La pista es una línea negra sobre fondo blanco. La lógica es de una simpleza hermosa: si los dos sensores ven blanco, el robot está centrado sobre la línea (porque la línea negra está justo en medio, entre los dos sensores)... espera, vamos a pensarlo mejor, porque aquí es donde todos se confunden al inicio.

Imagínalo desde arriba. La línea negra va al centro, el robot la sigue como un tren sus rieles. Los dos sensores están uno a cada lado de la línea, sobre el fondo blanco. Caso 1: los dos ven blanco, todo bien, avanza derecho. Caso 2: el sensor izquierdo ve negro, significa que la línea se movió hacia la izquierda (o el robot se cargó a la derecha), entonces hay que girar a la izquierda para volver a centrarla: frenas o reversas la rueda izquierda y avanzas la derecha. Caso 3: el sensor derecho ve negro, giras a la derecha. Caso 4: los dos ven negro, estás en un cruce o te saliste por completo: lo sensato es detenerse o girar buscando la línea. Esa tablita de cuatro casos ES el programa. Todo lo demás es traducción a código.

La analogía cotidiana: es como caminar por la banqueta de noche guiándote por la línea amarilla del centro de la calle con los pies: si tu pie izquierdo pisa la línea, te cargas a la derecha; si la pisa el derecho, te cargas a la izquierda. Tus pies son los sensores, tu cerebro es el loop de Arduino.

El segundo proyecto clásico es el evasor de obstáculos, y su analogía es el murciélago. El robot lleva un sensor ultrasónico al frente que mide la distancia a lo que tenga enfrente. La lógica: avanza; si la distancia es menor a cierto umbral (digamos 20 centímetros), hay una pared: detente, gira un poco a un lado, mide de nuevo, y si ya está libre, sigue avanzando. Algunos le agregan un servomotor que mueve el sensor como una cabeza que mira a izquierda y derecha para elegir el lado con más espacio, igual que tú volteas a los dos lados antes de cruzar la calle. Este comportamiento de "avanza hasta que algo estorbe, esquívalo y sigue" es la base de las aspiradoras robot: tu Roomba hace exactamente esto, solo que con más sensores y un mapa en la memoria.

Ahora, hablemos de cómo se controlan los motores, porque aquí hay un concepto que debes dominar: el PWM. Las ruedas no solo necesitan girar o detenerse; necesitan velocidades intermedias para girar suavemente. Arduino no puede dar "medio voltaje" con un pin digital normal, pero puede hacer trampa: prende y apaga el pin cientos de veces por segundo. Si está prendido el 50 por ciento del tiempo, el motor "siente" la mitad de la potencia y gira a media velocidad. Esa técnica se llama PWM (modulación por ancho de pulso) y en Arduino se usa con la instrucción `analogWrite(pin, valor)`, donde el valor va de 0 (apagado) a 255 (toda velocidad). Es como controlar la velocidad de un ventilador prendiéndolo y apagándolo rapidísimo: a más tiempo prendido, más aire. Con PWM puedes hacer que un robot gire suavemente en lugar de dar volantazos.

Errores comunes de principiante programando robots: el más famoso es la lógica invertida de los sensores ("mi robot huye de la línea en lugar de seguirla"): casi siempre es porque el sensor devuelve 0 donde esperabas 1 o al revés; la solución es imprimir los valores del sensor en el monitor serial y calibrar con la realidad, no con lo que crees. Otro clásico: el robot tiembla o se mueve a sacudidas sobre la línea; eso pasa por girar a toda velocidad en lugar de usar PWM suave, o por leer los sensores muy lento por culpa de `delay()` largos en el código (un delay de 500 milisegundos es una eternidad para un robot que se mueve: en ese tiempo ya se salió de la pista). También es típico que el ultrasónico dé lecturas locas (un cero o un 400 de la nada): ponle un pequeño filtro en el programa, como ignorar lecturas fuera de rango o promediar tres lecturas seguidas. Y el error más frustrante: todo el código está bien pero el robot no se mueve porque las baterías están bajas; los motores son los primeros en resentir la batería débil.

Buenas prácticas que te van a ahorrar noches de frustración: primero, calibra siempre. Los sensores infrarrojos se comportan distinto con luz de día que con luz artificial, y sobre cartón blanco que sobre piso blanco; haz una rutina de calibración o al menos ajusta tus umbrales probando en la pista real. Segundo, empieza lento: ponle velocidad baja al robot mientras pruebas la lógica; un robot lento y correcto es mejor que uno rápido y descontrolado; ya lo acelerarás cuando funcione. Tercero, usa el monitor serial como tu ventana al cerebro del robot: imprime qué ven los sensores y qué decisión toma; depurar un robot sin ver sus lecturas es como arreglar un carro con los ojos vendados. Cuarto, programa comportamientos de seguridad: ¿qué hace tu robot si pierde la línea por completo? ¿si el ultrasónico deja de responder? Define un "modo seguro" (detenerse) antes de que lo necesites. Y quinto, prueba en el mundo real, no solo en tu cabeza: la simulación y la lógica en papel son el inicio, pero la fricción del piso, la luz del cuarto y el peso de las baterías solo aparecen cuando el robot se mueve de verdad.

Un consejo de oro para cuando no tengas las piezas todavía: usa Tinkercad Circuits, el simulador gratuito en línea. Te deja armar el Arduino, los sensores y los motores en pantalla y probar tu código sin gastar un peso. No es igual que el robot físico (el simulador no tiene fricción ni baterías bajas), pero te enseña la lógica y el código al cien. Muchos roboticistas prueban primero en simulador y luego pasan al hardware con la confianza de que la lógica ya jala.

Drones: tipos, componentes y cómo demonios vuelan

Si el robot con ruedas fue tu primer amor, el dron es el que te hace soñar. Un dron es un robot volador: percibe (con giroscopios, acelerómetros, barómetro, GPS y cámara), decide (su controladora de vuelo ajusta los motores miles de veces por segundo) y actúa (variando la velocidad de sus hélices). La mayoría de los drones que vas a manejar son multirrotores: cuadricópteros (4 hélices), hexacópteros (6) o octocópteros (8). Olvídate de los aviones de juguete con un solo motor: el multirrotor es el diseño que democratizó el vuelo porque es mecánicamente simple (no tiene partes móviles complicadas como un helicóptero real) y electrónicamente inteligente (una computadora hace el trabajo de estabilización que en un helicóptero hace el piloto con años de práctica).

¿Cómo se mantiene en el aire una cosa con cuatro hélices? Con pura física y matemáticas rápidas. Cada hélice genera empuje hacia arriba al girar. Si las cuatro generan el mismo empuje, el dron sube o se mantiene quieto. Para moverse, el dron desbalancea a propósito: acelera las hélices de atrás y el dron se inclina hacia adelante, y al inclinarse parte del empuje apunta hacia adelante y avanza. Para girar sobre su propio eje, acelera dos hélices diagonales y frena las otras dos, creando un par de rotación. Todo esto lo calcula la controladora de vuelo

(una tarjetita con sensores de movimiento) cientos de veces por segundo, corrigiendo el viento y tus órdenes del control. Tú con el control solo dices "sube", "avanza", "gira"; la controladora traduce eso a velocidad exacta de cada motor. Es como si el dron tuviera un piloto automático interno que nunca se distrae.

Los componentes de un dron, pieza por pieza: el frame o chasis (los brazos que sostienen todo, de fibra de carbono en los buenos), los motores brushless (sin escobillas: potentes, eficientes y duraderos, muy distintos a los motorcitos DC de tu carrito), las hélices (su tamaño y paso definen cuánta carga levantan), los ESC (controladores electrónicos de velocidad: son los que le dicen a cada motor qué tan rápido girar, recibiendo órdenes de la controladora), la controladora de vuelo (el cerebro: lleva giroscopio, acelerómetro y a veces barómetro y GPS), la batería LiPo (de polímero de litio: ligeras y potentes, pero delicadas, hay que cargarlas con cuidado y nunca perforarlas), el receptor de radio (recibe tus órdenes del control remoto) y, en los drones con cámara, el gimbal (un estabilizador de 3 ejes que mantiene la cámara nivelada aunque el dron se incline: es la magia detrás de esos videos aéreos suaves).

Tipos de drones según su misión: los drones de juguete (baratos, ligeros, para aprender a volar sin llorar si se estrellan; vuelan 5 a 10 minutos), los drones de fotografía (como los famosos de la marca líder: cámara 4K con gimbal, GPS, 20 a 30 minutos de vuelo, modos inteligentes que te siguen solo o orbitan un punto), los drones de carreras FPV (pequeños, rapidísimos, con cámara que transmite video en vivo a unos goggles que te pones en la cara: vuelas "dentro" del dron como un videojuego; son los más divertidos y los más difíciles de dominar) y los drones profesionales e industriales (grandes, caros, con cámaras térmicas, sensores multispectrales para agricultura o capacidad de cargar paquetes). Cada tipo es una herramienta distinta: no compras un dron de carreras para tomar fotos de bodas, ni un dron fotográfico para competir.

Errores comunes de principiante con drones: volar el dron caro el primer día (se estrella, garantizado; empieza con uno barato o en simulador). Volar con viento fuerte (un dron ligero con viento es una hoja al aire; si las ramas de los árboles se mueven mucho, mejor no despegar). No calibrar la brújula y el GPS antes del primer vuelo en un lugar nuevo (el dron se puede desorientar y hacer cosas raras). Volar hasta que la batería muera (cuando la batería baja mucho el dron puede caer; aterriza con al menos 20 por ciento de reserva). Y el clásico: despegar sin revisar las hélices (una hélice astillada vibra, gasta batería y puede romperse en vuelo).

Buenas prácticas: aprende primero en un simulador de vuelo (hay varios gratuitos) o con un dron de juguete barato; tus primeros 10 vuelos son para aprender orientación (cuando el dron viene hacia ti, los controles se sienten invertidos: izquierda es derecha). Despega siempre en un lugar abierto, lejos de gente, árboles y cables. Revisa batería, hélices y GPS antes de cada

vuelo, siempre, sin flojera. Y graba tus vuelos: ver el video después te enseña más que cien consejos.

Normativa básica y vuelo responsable: la parte seria

Vamos a hablar de lo que nadie quiere leer pero todos deben saber: las reglas. Un dron no es un juguete cualquiera: es una aeronave, y una aeronave mal operada puede lastimar gente, tirar cables, interrumpir un aeropuerto o invadir la privacidad de alguien. Por eso en México y en casi todo el mundo existen normas para volar drones, y volar sin conocerlas te puede costar el dron (las autoridades pueden asegurarlo), una multa o, peor, un accidente con consecuencias legales.

Te doy el panorama general, válido como principios en la mayoría de los países, con notas de México. Primero: registra e identifica tu dron si la norma lo pide. En México, la Agencia Federal de Aviación Civil (AFAC) es la autoridad, y su norma para sistemas de aeronaves pilotadas a distancia (RPAS) distingue por peso y uso: los drones recreativos pequeños tienen reglas más simples, pero los de mayor peso o uso comercial requieren registro, y en algunos casos licencia o autorización del piloto. La regla de oro: antes de volar un dron que no sea de juguete, revisa la norma vigente de tu país en la página oficial de su autoridad de aviación. Las normas cambian, y este libro no sustituye la ley.

Segundo: dónde puedes volar. Las reglas típicas en casi todo el mundo son: vuela solo de día y con buena visibilidad; mantén siempre el dron a la vista (sin binoculares ni pantallas como única referencia); no vuelas sobre multitudes ni sobre personas que no participen en la operación; mantente lejos de aeropuertos y helipuertos (típicamente varios kilómetros a la redonda, o solo con autorización); no vuelas cerca de instalaciones sensibles (cárceles, instalaciones militares, infraestructura crítica); y respeta la altura máxima permitida (en muchos países son 120 metros sobre el nivel del suelo, que es más que suficiente para casi todo). En México, además, hay zonas donde el vuelo está restringido o prohibido y conviene consultar los mapas oficiales antes de despegar.

Tercero: la privacidad y el sentido común. Que puedas volar tu dron con cámara sobre la casa del vecino no significa que debas: grabar patios ajenos, ventanas o personas sin permiso es invasión a la privacidad y en muchos lugares es delito. Vuela como te gustaría que volaran sobre ti. Y cuarto: el dron es tu responsabilidad de principio a fin. Si tu dron causa un daño, respondes tú. Por eso los pilotos serios usan listas de verificación pre-vuelo (te dejo una en la sección de ejemplos), mantienen sus baterías en buen estado y nunca vuelan si algo no se siente bien.

La analogía: manejar un dron es como manejar un carro. Nadie te suelta en periférico sin enseñarte el reglamento de tránsito, los límites de velocidad y la cortesía vial. El cielo tiene su propio reglamento, y ser un buen piloto es tan importante como ser un buen constructor. Un roboticista completo no solo arma y programa: opera con responsabilidad.

Errores comunes: creer que "como es chiquito no aplica la ley" (el tamaño no te exime; un dron de 250 gramos mal usado también puede hacer daño). Volar en parques llenos de gente para "probar" (el lugar de pruebas es un campo abierto y vacío). Publicar videos donde se ve que volaste en zona prohibida (la evidencia en redes también cuenta). Y no revisar actualizaciones de la norma: lo que era legal hace tres años puede no serlo hoy.

Buenas prácticas: antes de comprar un dron serio, dedica una tarde a leer la norma de tu país en fuente oficial. Únete a un club o comunidad local de pilotos: aprendes más en una mañana con pilotos experimentados que en un mes de videos. Considera un seguro de responsabilidad civil si vas a volar con frecuencia o de forma comercial: es barato comparado con un accidente. Y adopta la mentalidad de piloto: cada vuelo se planea, se verifica y se registra (fecha, lugar, duración, incidentes). Ese hábito te convierte en profesional.

Aplicaciones de drones: del hobby a la industria

Ya sabes volar con responsabilidad; ahora mira todo lo que puedes hacer con ese superpoder. Los drones dejaron de ser juguetes caros hace años: hoy son herramientas de trabajo en industrias completas, y entender sus aplicaciones te abre los ojos a oportunidades reales de negocio y empleo.

Fotografía y video aéreo es la aplicación más visible. Bienes raíces (casas y terrenos se venden mejor con tomas aéreas), bodas y eventos, turismo, cine y publicidad: un dron con buena cámara hace en minutos lo que antes requería un helicóptero rentado por miles de dólares. Es también la puerta de entrada más común al trabajo profesional con drones, pero ojo: es el mercado más competido. La diferencia la hacen la calidad del pilotaje, la edición y los permisos en regla.

Agricultura de precisión es una de las aplicaciones más transformadoras y menos conocidas. Drones con cámaras multispectrales vuelan sobre los cultivos y detectan qué plantas están estresadas, enfermas o con falta de agua antes de que el ojo humano lo note (las plantas enfermas reflejan la luz distinto, y el dron lo mide). Otros drones fumigan con precisión centimétrica, aplicando agroquímicos solo donde se necesitan, ahorrando producto y contaminando menos. En un país agrícola como México, esto es tecnología con impacto directo en la comida que llega a tu mesa.

Inspección de infraestructura: puentes, torres eléctricas, turbinas eólicas, paneles solares, techos industriales, antenas. Antes, inspeccionar una torre de alta tensión requería que un técnico subiera con arnés, con riesgo y costo enormes. Hoy un dron con cámara de alta resolución o cámara térmica hace el recorrido en minutos, y el técnico revisa las fotos desde el suelo, seguro. Las cámaras térmicas detectan puntos calientes en paneles solares o conexiones eléctricas que anuncian fallas futuras. Es un trabajo bien pagado y con demanda creciente.

Reparto y logística: empresas en varios países ya reparten medicinas, muestras de laboratorio y paquetes pequeños con drones en zonas rurales o de difícil acceso. Todavía hay retos regulatorios y de autonomía de batería, pero la dirección es clara: el cielo se va a llenar de rutas de reparto. Y no olvides la respuesta a emergencias: drones que buscan personas extraviadas con cámaras térmicas de noche, que evalúan daños tras un sismo o que llevan un desfibrilador a donde una ambulancia tardaría demasiado.

La lección para ti: cada aplicación combina el dron con otra habilidad. Dron más fotografía es negocio de foto aérea. Dron más agronomía es agricultura de precisión. Dron más visión por computadora (lo veremos enseguida) es inspección automatizada. El dron solo es la plataforma; el valor está en lo que le pones encima y en el problema que resuelves. Piensa qué problemas hay a tu alrededor que se verían mejor desde 50 metros de altura.

Errores comunes al soñar con aplicaciones: querer abarcar todo ("mi empresa de drones hará foto, agro, inspección y reparto"): empieza con UNA aplicación, domínala y luego crece. Subestimar la parte comercial: volar bien es el 30 por ciento; conseguir clientes, cotizar, entregar a tiempo y facturar es el 70. Y olvidar los permisos: el trabajo comercial con drones casi siempre exige registro, y a veces certificaciones del piloto; opera en regla desde el día uno.

Buenas prácticas: elige una aplicación, estudia a quienes ya la hacen bien y ofrece tus primeros trabajos a precio de aprendizaje para construir portafolio. Documenta todo con fotos y videos de calidad: tu portafolio es tu mejor vendedor. Y mantén tu equipo impecable: un profesional se reconoce por el estado de sus baterías, sus hélices y sus listas de verificación, no solo por cómo vuela.

Visión por computadora: dándole ojos al robot

Hasta ahora tus robots "sienten" el mundo con sensores sencillos: distancia, luz, contacto. Pero hay un sentido que cambia las reglas del juego: la vista. Una cámara le da al robot millones de datos por segundo, y la visión por computadora es el arte de convertir esos píxeles en decisiones: "ahí hay una pelota roja", "esa es una cara", "la línea del camino va hacia la

izquierda". Es lo que permite que un dron siga a un ciclista solo, que un robot clasifique piezas por color en una fábrica o que un carro autónomo lea las señales de tránsito.

¿Cómo "ve" una computadora? Aquí viene la idea clave que desmitifica todo: para la computadora no hay imágenes, hay matrices de números. Una foto a color es una cuadrícula donde cada cuadrito (píxel) tiene tres números: cuánto rojo, cuánto verde y cuánto azul hay en ese punto (el famoso RGB, de 0 a 255 cada uno). "Ver" es hacer matemáticas con esos números. Por ejemplo, detectar una pelota roja es buscar los píxeles donde el número de rojo es mucho mayor que el de verde y azul. Eso es todo el truco inicial: convertir colores en rangos de números y buscar regiones que cumplan la condición.

La herramienta favorita para esto en el mundo real es OpenCV, una biblioteca gratuita de visión por computadora que funciona con Python (el lenguaje que ya conoces de este curso). Con unas 30 líneas de Python puedes abrir tu webcam, buscar el color rojo en cada cuadro de video, dibujar un rectángulo alrededor de lo que encuentres y decirle al robot hacia dónde moverse. En la sección de ejemplos te dejo ese script completo y explicado: vas a poder correrlo en tu computadora hoy mismo, sin comprar nada.

Pero detectar colores es solo el primer escalón. La escalera completa de la visión por computadora se ve así. Escalón 1: detección de color y formas básicas (encuentra lo rojo, encuentra círculos). Escalón 2: detección de movimiento (compara un cuadro de video con el anterior y marca lo que cambió: así funcionan muchas cámaras de seguridad). Escalón 3: reconocimiento de patrones con plantillas (busca en la imagen algo que se parezca a una foto de referencia). Escalón 4: aprendizaje automático y redes neuronales (el sistema aprende de miles de fotos qué es un gato, una cara o una señal de alto, y luego lo reconoce en fotos nuevas). Los escalones 1 a 3 los puedes programar tú con lógica directa; el escalón 4 usa modelos de inteligencia artificial ya entrenados, que hoy puedes descargar gratis y usar sin ser experto en IA.

La analogía para entender la diferencia: detectar color es como buscar a alguien en una fiesta porque trae playera roja (simple, rápido, a veces te confundes con otra playera roja). Usar una red neuronal entrenada es como reconocer a tu amigo por su cara entre cien personas (mucho más robusto, pero necesitas haberlo "conocido" antes, es decir, entrenado). Cada nivel tiene su lugar: si tu robot solo necesita seguir una pelota roja en tu cuarto, no necesitas IA; la detección de color basta y es rapidísima.

Ahora conecta esto con la robótica: un robot con cámara y visión es un robot que toma decisiones ricas. El seguidor de línea con sensores infrarrojos solo sabe "línea a la izquierda o a la derecha"; el mismo robot con cámara puede ver la curva que viene, leer marcas en el piso, esquivar obstáculos que reconoce por forma y hasta seguir a una persona. Un dron con visión

puede mantener su posición sin GPS (mirando el suelo y corrigiendo la deriva), seguirte mientras corres y esquivar ramas solo. La cámara es el sensor más versátil que existe, porque casi toda la información del mundo entra por la vista.

Errores comunes de principiante en visión: primero, la luz. El enemigo número uno de la detección de color es el cambio de iluminación: el rojo de tu pelota se ve distinto con luz de día que con foco amarillo, y tu programa calibrado en la mañana falla en la tarde. La solución profesional es trabajar en el espacio de color HSV (tono, saturación, valor) en lugar de RGB, porque el tono del color cambia menos con la luz que los valores RGB crudos; en el ejemplo de código verás cómo se hace. Segundo, el ruido: la cámara siempre tiene píxeles "sucios" que tu programa confunde con detecciones; se corrige con filtros que eliminan regiones muy pequeñas. Tercero, querer reconocer objetos complejos con pura lógica de color: si tu meta es "detectar cualquier carro", no lo hagas con rangos de color; usa un modelo de IA preentrenado. Y cuarto, olvidar el rendimiento: procesar video es pesado; si tu programa va lento, reduce la resolución de la imagen o procesa menos cuadros por segundo; un robot que "ve" a 5 cuadros por segundo pero decide bien le gana a uno que se congela intentando ver perfecto.

Buenas prácticas: empieza con la webcam de tu laptop y buena luz natural; es el laboratorio más barato del mundo. Calibra tus rangos de color con la herramienta interactiva (en el ejemplo verás cómo mostrar los valores mientras apuntas la cámara al objeto). Guarda fotos de prueba de tu entorno real y prueba tu programa con ellas antes de salir al mundo. Y separa la visión de la acción: primero logra que tu programa detecte bien y dibuje el rectángulo correcto; después conecta esa detección a los motores. Si mezclas las dos cosas desde el inicio, no sabrás si el fallo está en los ojos o en las piernas.

Robótica más IA más IoT: el robot conectado

Llegamos a la cima de la unidad, donde todo lo que has aprendido se fusiona. Ya tienes robots que se mueven (robótica), que ven (visión por computadora) y que aprenden patrones (inteligencia artificial). Falta la tercera pata: el Internet de las cosas (IoT), que es conectar tus dispositivos a Internet para que se comuniquen, se monitoreen y se controlen a distancia. Un robot con IA y conectado a Internet ya no es una máquina aislada: es un nodo inteligente de una red.

Piensa en el ejemplo completo: un robot vigilante para tu casa. Lleva una cámara y un modelo de IA que distingue entre tu familia, tu perro y un desconocido (visión más IA). Cuando detecta a un desconocido, te manda una alerta con foto a tu celular por Internet (IoT), y tú desde tu teléfono le ordenas que siga a la persona con la cámara o que haga sonar una alarma (control remoto). Tres tecnologías, un solo sistema, un problema real resuelto. Ninguna de las tres por separado lograría lo mismo.

La analogía: un robot sin conexión es como un empleado sin teléfono: hace su trabajo, pero si pasa algo no te avisa y no le puedes dar nuevas órdenes. Un robot conectado es como tenerlo en un grupo de WhatsApp contigo: te reporta, te pide instrucciones y comparte lo que ve. Y la IA es como si ese empleado además aprendiera de la experiencia: cada semana distingue mejor entre tu gato y un intruso.

¿Cómo se conecta un robot a Internet en la práctica? Con placas como el ESP32, primo del Arduino pero con WiFi y Bluetooth integrados y más potencia: cuesta poco más que un Arduino y te permite mandar datos a la nube, recibir órdenes desde una app y hasta hacer videollamada con tu robot. Los protocolos típicos son MQTT (un sistema de mensajería ligero, ideal para que dispositivos se manden mensajitos como "temperatura: 25 grados" o "obstáculo detectado") y HTTP (el mismo de las páginas web, útil para mandar fotos o consultar servicios). Y del lado de la nube hay plataformas gratuitas o baratas donde guardas los datos de tus sensores, los graficas y programas alertas. En el ejemplo de código de esta unidad verás el patrón básico: el robot lee un sensor y publica el dato, y tu programa lo recibe donde estés.

¿Y dónde entra la IA en el robot conectado? En tres niveles, de menor a mayor ambición. Nivel 1: IA en la nube. El robot manda la foto a un servicio de IA en Internet, que la analiza y le regresa la respuesta ("sí, es una persona"). Fácil de programar, pero necesita Internet y tiene un poco de retardo. Nivel 2: IA en el borde (edge AI). El modelo de IA corre dentro del propio robot, en una placa potente como una Raspberry Pi o una Jetson: responde rapidísimo y funciona sin Internet, pero el hardware cuesta más. Nivel 3: aprendizaje continuo. El robot no solo usa un modelo, sino que mejora con los datos que recoge (con supervisión tuya). Este nivel ya es territorio profesional, pero es hacia donde va todo.

Un caso que te va a sonar: las aspiradoras robot modernas. Mapean tu casa (robótica), reconocen obstáculos como cables y zapatos con IA (visión más IA) y te mandan el mapa y el reporte de limpieza a tu celular, desde donde las programas (IoT). Hace diez años eso era ciencia ficción; hoy lo compras en una tienda. Tú, con lo que aprendiste en esta unidad, entiendes cada una de esas piezas. Esa es la diferencia entre usar tecnología y comprenderla.

Errores comunes al soñar con el robot conectado: querer conectar todo a Internet desde el día uno (primero haz que el robot funcione solo; la conexión es el adorno, no el cimiento). Mandar video crudo a la nube sin pensar en el ancho de banda (un video HD saturan tu WiFi y tu cartera; procesa local y manda solo resultados: "persona detectada", no el video completo). Ignorar la seguridad: un robot conectado sin contraseña es una puerta abierta a tu red; ponle claves fuertes y no expongas servicios a Internet sin saber lo que haces. Y depender totalmente de la nube: si tu robot solo funciona con Internet y se cae la red, tienes un pisapapeles caro; diseña un modo local que funcione sin conexión.

Buenas prácticas: diseña por capas (movimiento, percepción, conexión) y prueba cada capa sola. Usa MQTT para mensajes cortos y frecuentes, HTTP para cosas pesadas y esporádicas. Guarda un registro (log) de lo que hace tu robot: cuando algo falle a las 3 de la mañana, el log es tu testigo. Y piensa en la energía desde el diseño: el WiFi gasta batería; si tu robot es móvil, mide cuántas horas reales dura con todo encendido, no solo con los motores.

Cierra los ojos e imagina a dónde lleva todo esto: fábricas donde robots, drones y sensores se coordinan solos por la red; campos donde drones fumigan guiados por IA que leyó las fotos del día anterior; ciudades donde drones de reparto conversan con semáforos inteligentes. Ese mundo lo van a construir personas que entienden las tres piezas juntas. Después de esta unidad, tú eres una de esas personas en formación. Bienvenido al futuro: ahora ve y constrúyelo.

Ejemplos paso a paso

La teoría sin práctica es como receta sin cocina: se antoja pero no alimenta. Aquí van cuatro ejemplos completos, listos para que los pruebes: dos programas de Arduino (seguidor de línea y evasor de obstáculos), un script de Python con OpenCV que detecta colores con tu webcam, y una lista de verificación pre-vuelo para dron escrita como un archivo de configuración que puedes adaptar. Todo el código está explicado línea por línea y usa solo componentes comunes y baratos.

Ejemplo 1: robot seguidor de línea con Arduino (C++)

Qué necesitas: un Arduino Uno, un driver de motores L298N, dos motores DC con ruedas, dos sensores infrarrojos de línea (tipo TCRT5000), un chasis, una batería para los motores y cables. Los sensores van abajo del robot, apuntando al piso, uno a cada lado de donde pasará la línea negra. La pista: una línea negra de unos 2 cm de ancho sobre fondo blanco (cartulina blanca con cinta de aislar negra funciona perfecto).

Cómo funciona el programa: en cada vuelta del loop lee los dos sensores y aplica la tablita de cuatro casos que vimos en la explicación. Usa PWM (analogWrite) para que los giros sean suaves y define una velocidad base que puedes ajustar. Los pines del ejemplo son una sugerencia: ajústalos a tu armado real.

```
// Robot seguidor de línea - Lucas EGA Academy
// Lee dos sensores IR y ajusta los motores con PWM para seguir la línea negra.
// NOTA: ajusta los numeros de pin segun tu armado.

// Pines del driver de motores L298N
```

```

const int ENA = 5;    // velocidad motor izquierdo (pin PWM)
const int IN1 = 6;    // direccion motor izquierdo
const int IN2 = 7;    // direccion motor izquierdo
const int ENB = 10;   // velocidad motor derecho (pin PWM)
const int IN3 = 8;    // direccion motor derecho
const int IN4 = 9;    // direccion motor derecho

// Pines de los sensores infrarrojos (digitales: HIGH = blanco, LOW = negro)
const int SENSOR_IZQ = 2;
const int SENSOR_DER = 3;

// Velocidades (0 a 255). Empieza lento mientras pruebas.
const int VEL_BASE = 150; // velocidad de avance
const int VEL_GIRO = 110; // velocidad al corregir

void setup() {
  pinMode(ENA, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(ENB, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  pinMode(SENSOR_IZQ, INPUT);
  pinMode(SENSOR_DER, INPUT);
  Serial.begin(9600); // para depurar: mira que ven los sensores
}

void loop() {
  int izq = digitalRead(SENSOR_IZQ); // HIGH = blanco, LOW = negro
  int der = digitalRead(SENSOR_DER);

  // Muestra las lecturas para calibrar (abre el monitor serial)
  Serial.print("IZQ=");
  Serial.print(izq);
  Serial.print(" DER=");
  Serial.println(der);

  if (izq == HIGH && der == HIGH) {
    // Los dos ven blanco: centrado, avanza derecho
    avanzar();
  } else if (izq == LOW && der == HIGH) {
    // El izquierdo ve negro: la linea esta a la izquierda, gira a la izquierda
    girarIzquierda();
  } else if (izq == HIGH && der == LOW) {
    // El derecho ve negro: gira a la derecha

```

```

        girarDerecha();
    } else {
        // Los dos ven negro: cruce o linea perdida, detenerse es lo seguro
        detener();
    }
    delay(20); // pequena pausa para estabilizar lecturas
}

// --- Funciones de movimiento ---

void avanzar() {
    digitalWrite(IN1, HIGH); digitalWrite(IN2, LOW);
    digitalWrite(IN3, HIGH); digitalWrite(IN4, LOW);
    analogWrite(ENA, VEL_BASE);
    analogWrite(ENB, VEL_BASE);
}

void girarIzquierda() {
    // Frena la rueda izquierda y avanza la derecha: gira a la izquierda
    digitalWrite(IN1, LOW); digitalWrite(IN2, LOW);
    digitalWrite(IN3, HIGH); digitalWrite(IN4, LOW);
    analogWrite(ENA, 0);
    analogWrite(ENB, VEL_GIRO);
}

void girarDerecha() {
    digitalWrite(IN1, HIGH); digitalWrite(IN2, LOW);
    digitalWrite(IN3, LOW); digitalWrite(IN4, LOW);
    analogWrite(ENA, VEL_GIRO);
    analogWrite(ENB, 0);
}

void detener() {
    digitalWrite(IN1, LOW); digitalWrite(IN2, LOW);
    digitalWrite(IN3, LOW); digitalWrite(IN4, LOW);
    analogWrite(ENA, 0);
    analogWrite(ENB, 0);
}

```

Paso a paso para probarlo: 1) Arma solo la parte de motores primero y verifica que avanzar() mueve el robot hacia adelante (si va hacia atrás, invierte IN1/IN2 o IN3/IN4 en el código, no los cables). 2) Conecta un sensor, abre el monitor serial y pasa un papel blanco y uno negro debajo: confirma que blanco da HIGH y negro da LOW; si sale al revés, invierte la lógica en los if. 3) Coloca el robot sobre la pista con la línea entre los dos sensores y enciéndelo con velocidad baja. 4) Si se sale en las curvas, baja VEL_BASE o acerca más los sensores a la línea. Si tiembla, sube un poco el delay o suaviza con velocidades menores en los giros.

Ejemplo 2: robot evasor de obstáculos con Arduino y sensor ultrasónico (C++)

Qué necesitas: lo mismo del ejemplo 1 (Arduino, driver L298N, dos motores) más un sensor ultrasónico HC-SR04 al frente y, opcional, un servomotor para mover el sensor como "cabeza". El HC-SR04 tiene cuatro pines: VCC, GND, TRIG (dispara el pulso) y ECHO (recibe el eco). La distancia se calcula midiendo cuánto tarda el eco: distancia en cm = tiempo en microsegundos dividido entre 58.

Cómo funciona: el robot avanza; en cada vuelta mide la distancia al frente; si hay algo a menos de 20 cm, se detiene, retrocede un poco, gira hacia un lado y vuelve a medir. Es el comportamiento de "murciélago" que vimos antes.

```
// Robot evasor de obstaculos - Lucas EGA Academy
// Usa un HC-SR04 para detectar paredes y esquivarlas.
// NOTA: ajusta los numeros de pin segun tu armado.

const int ENA = 5;
const int IN1 = 6;
const int IN2 = 7;
const int ENB = 10;
const int IN3 = 8;
const int IN4 = 9;

const int TRIG = 12; // pin de disparo del ultrasonico
const int ECHO = 11; // pin de eco del ultrasonico

const int UMBRAL_CM = 20; // distancia minima antes de esquivar
const int VEL = 160;

void setup() {
  pinMode(ENA, OUTPUT); pinMode(IN1, OUTPUT); pinMode(IN2, OUTPUT);
  pinMode(ENB, OUTPUT); pinMode(IN3, OUTPUT); pinMode(IN4, OUTPUT);
  pinMode(TRIG, OUTPUT); pinMode(ECHO, INPUT);
  Serial.begin(9600);
}
```

```

void loop() {
    long dist = medirDistancia();
    Serial.print("Distancia: ");
    Serial.print(dist);
    Serial.println(" cm");

    if (dist > UMBRAL_CM) {
        avanzar(); // camino libre
    } else {
        // Obstaculo cerca: detente, retrocede y gira a la derecha
        detener();
        delay(200);
        retroceder();
        delay(400);
        girarDerecha();
        delay(500);
    }
    delay(50);
}

// Mide la distancia con el HC-SR04 y filtra lecturas invalidas
long medirDistancia() {
    digitalWrite(TRIG, LOW);
    delayMicroseconds(2);
    digitalWrite(TRIG, HIGH); // pulso de 10 microsegundos
    delayMicroseconds(10);
    digitalWrite(TRIG, LOW);

    long tiempo = pulseIn(ECHO, HIGH, 30000); // espera el eco (timeout 30 ms)
    long cm = tiempo / 58; // conversion a centimetros

    // Filtro: ignora lecturas fuera del rango util del sensor (2 a 400 cm)
    if (cm < 2 || cm > 400) {
        return 400; // sin lectura valida = supone camino libre lejos
    }
    return cm;
}

void avanzar() {
    digitalWrite(IN1, HIGH); digitalWrite(IN2, LOW);
    digitalWrite(IN3, HIGH); digitalWrite(IN4, LOW);
    analogWrite(ENA, VEL); analogWrite(ENB, VEL);
}

```



```

void retroceder() {
    digitalWrite(IN1, LOW); digitalWrite(IN2, HIGH);
    digitalWrite(IN3, LOW); digitalWrite(IN4, HIGH);
    analogWrite(ENA, VEL); analogWrite(ENB, VEL);
}

void girarDerecha() {
    digitalWrite(IN1, HIGH); digitalWrite(IN2, LOW);
    digitalWrite(IN3, LOW); digitalWrite(IN4, HIGH);
    analogWrite(ENA, VEL); analogWrite(ENB, VEL);
}

void detener() {
    digitalWrite(IN1, LOW); digitalWrite(IN2, LOW);
    digitalWrite(IN3, LOW); digitalWrite(IN4, LOW);
    analogWrite(ENA, 0); analogWrite(ENB, 0);
}

```

Paso a paso para probarlo: 1) Prueba solo el sensor primero: abre el monitor serial, pon tu mano enfrente y verifica que la distancia sube y baja de forma coherente. Si siempre dice 0, revisa que TRIG y ECHO no estén invertidos. 2) Pon el robot en el piso con una caja a 50 cm: debe avanzar hasta los 20 cm, detenerse, retroceder y girar. 3) Si gira siempre hacia el mismo lado y choca con la pared de ese lado, cambia girarDerecha() por una versión que gire a la izquierda, o mejor: agrega el servomotor con el sensor para "mirar" a ambos lados y elegir el lado con más espacio (ese es tu siguiente reto). 4) Si el robot no detecta obstáculos delgados como patas de silla, es normal: el ultrasonido se escapa entre objetos angostos; para eso existen los sensores láser, pero cuestan más.

Ejemplo 3: detección de color con Python y OpenCV (los ojos del robot)

Qué necesitas: una computadora con webcam, Python 3 y las bibliotecas opencv-python y numpy (se instalan con pip install opencv-python numpy). No necesitas ningún robot: tu laptop es el laboratorio. El script abre la cámara, convierte cada cuadro a HSV (el espacio de color resistente a cambios de luz), busca píxeles rojos, dibuja un rectángulo alrededor del objeto rojo más grande y te dice si está a la izquierda, al centro o a la derecha. Esa información ("izquierda/centro/derecha") es exactamente lo que un robot necesitaría para seguir el objeto.

```

# Detector de color rojo con OpenCV - Lucas EGA Academy
# Requiere: pip install opencv-python numpy
# Muestra un rectangulo sobre el objeto rojo mas grande y dice su posicion.

import cv2
import numpy as np

```

```

# Rango HSV para el color rojo (el rojo aparece en dos zonas del tono: 0-10 y 170-180)
ROJO_BAJO_1 = np.array([0, 120, 70])
ROJO_ALTO_1 = np.array([10, 255, 255])
ROJO_BAJO_2 = np.array([170, 120, 70])
ROJO_ALTO_2 = np.array([180, 255, 255])
AREA_MINIMA = 500 # ignora manchas mas pequenas (ruido)

camara = cv2.VideoCapture(0) # 0 = webcam principal

while True:
    ok, cuadro = camara.read()
    if not ok:
        print("No se pudo leer la camara")
        break

    # 1. Convertir a HSV (mas robusto ante cambios de luz que RGB)
    hsv = cv2.cvtColor(cuadro, cv2.COLOR_BGR2HSV)

    # 2. Crear mascara de pixeles rojos (une los dos rangos del rojo)
    mascara1 = cv2.inRange(hsv, ROJO_BAJO_1, ROJO_ALTO_1)
    mascara2 = cv2.inRange(hsv, ROJO_BAJO_2, ROJO_ALTO_2)
    mascara = mascara1 | mascara2

    # 3. Limpiar ruido: quita puntitos pequenos
    kernel = np.ones((5, 5), np.uint8)
    mascara = cv2.morphologyEx(mascara, cv2.MORPH_OPEN, kernel)

    # 4. Buscar contornos (regiones) en la mascara
    contornos, _ = cv2.findContours(mascara, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

    if contornos:
        # Quedarse con la region mas grande que supere el area minima
        mejor = max(contornos, key=cv2.contourArea)
        if cv2.contourArea(mejor) > AREA_MINIMA:
            x, y, w, h = cv2.boundingRect(mejor)
            cx = x + w // 2 # centro horizontal del objeto
            ancho = cuadro.shape[1]

            # 5. Decidir posicion: izquierda, centro o derecha
            if cx < ancho // 3:
                posicion = "IZQUIERDA"
            elif cx > 2 * ancho // 3:
                posicion = "DERECHA"

```

```

        else:
            posicion = "CENTRO"

            cv2.rectangle(cuadro, (x, y), (x + w, y + h), (0, 255, 0), 2)
            cv2.putText(cuadro, posicion, (x, y - 10),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 255, 0), 2)
            print("Objeto rojo en:", posicion)

    cv2.imshow("Detector - presiona Q para salir", cuadro)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

camara.release()
cv2.destroyAllWindows()

```

Paso a paso para probarlo: 1) Instala las bibliotecas y corre el script; autoriza el acceso a la cámara si tu sistema lo pide. 2) Sostén un objeto rojo (una tapa, un juguete) frente a la cámara con buena luz natural: debe aparecer el rectángulo verde con la posición. 3) Si no detecta nada, el problema casi siempre es la luz: prueba con más luz o ajusta los valores de saturación (el 120 de los rangos) a la baja. 4) Mueve el objeto a izquierda, centro y derecha y verifica que el texto cambie correctamente. 5) Reto: cambia los rangos HSV para detectar azul o verde (pista: el azul vive alrededor del tono 110, el verde alrededor del 60). 6) Conexión con robótica: imagina que en lugar de print, ese "IZQUIERDA/DERECHA" se manda por el puerto serial al Arduino del ejemplo 1: tendrías un robot que sigue una pelota roja con la cámara. Ese puente serial es un proyecto excelente para tu siguiente nivel.

Ejemplo 4: checklist pre-vuelo de dron como archivo de configuración

Los pilotos profesionales no despegan "a ver qué pasa": siguen una lista de verificación, siempre, sin excepciones. Aquí te dejo una en formato de configuración que puedes copiar, imprimir o adaptar a tu dron. La idea es que la leas en voz alta antes de cada vuelo y no despegues hasta que todo diga OK. Guárdala como prevuelo.txt junto a tu equipo.

```

# CHECKLIST PRE-VUELO DE DRON - Lucas EGA Academy
# Leelo en voz alta antes de CADA vuelo. No despegues con un NO sin resolver.
# Marca OK / NO en cada punto.

[DOCUMENTOS_Y_LEGAL]
zona_permitida = OK          # Verifique que no es zona restringida ni cerca de
aeropuerto
norma_vigente_revisada = OK # Lei la norma actual de mi pais (AFAC en Mexico)
personas_informadas = OK    # Acompañantes saben que voy a volar

[CLIMA]
viento = OK                  # Ramas apenas se mueven; sin rafagas fuertes
visibilidad = OK             # Puedo ver el dron claramente a 100 m
lluvia = OK                  # Sin lluvia ni amenaza cercana

[EQUIPO]
bateria_dron_cargada = OK    # 100% al despegar; reserva de 20% para aterrizar
baterias_extra = OK          # Llevo repuestos cargados si el plan lo requiere
helices = OK                 # Sin grietas, astillas ni flojedad; bien apretadas
motores = OK                 # Giran libres, sin ruidos raros al probarlos
tren_o_patas = OK           # Firmes, sin piezas sueltas
camara_y_gimbal = OK        # Gimbal se calibra solo al encender; lente limpio
memoria = OK                 # Tarjeta con espacio libre suficiente

[CONTROL_Y_SENAL]
control_cargado = OK         # Bateria del control remoto al 100%
enlace_radio = OK            # Control conectado al dron, sin alertas
gps = OK                     # Suficientes satelites fijos antes de despegar
brujula_calibrada = OK       # Calibrada si cambie de zona de vuelo
retorno_a_casa = OK          # Punto de retorno (RTH) configurado y verificado

[SITIO_DE_VUELO]
area_despejada = OK          # Sin gente, animales, arboles ni cables cerca
punto_despegue = OK          # Superficie plana y libre a 5 m a la redonda
plan_de_vuelo = OK           # Se que voy a hacer: ruta, altura maxima, duracion
aterrizaje_alterno = OK      # Tengo un punto B por si algo sale mal

[DESPEGUE]
anuncio = OK                 # Aviso en voz alta: "despegando"
prueba_corta = OK            # Subo a 2 m, verifico que responde a los 4 mandos
listo = OK                   # Todo lo anterior en OK -> puedo volar

```

Cómo usarla en la práctica: imprímela y plastifícala, o guárdala en tu celular. Antes de despegar, recórrela de arriba abajo sin saltarte nada; si un punto dice NO, lo resuelves o no

vuelas. Después del vuelo, anota en una libreta fecha, lugar, duración y cualquier incidente: ese registro te convierte en piloto serio y te sirve si algún día necesitas demostrar que operas con responsabilidad. Y adapta la lista a tu dron: un dron de juguete sin GPS no necesita el punto de satélites, pero sí necesita el de hélices y baterías.

Práctica

Llegó la hora de ensuciarte las manos. Estos seis ejercicios van de menor a mayor dificultad y están diseñados para que los hagas con lo que tengas: algunos solo necesitan papel y lógica, otros un simulador gratuito y otros piezas reales si ya las tienes. Cada uno trae su solución explicada paso a paso, pero intenta resolverlo tú primero antes de leerla: la robótica se aprende atorando y desatorando, no leyendo soluciones.

Ejercicio 1. Clasifica estas máquinas: ¿robot o no robot?

Decide cuáles de las siguientes máquinas son robots según la definición de esta unidad (percibir, decidir y actuar en un ciclo) y justifica tu respuesta: a) un ventilador con temporizador, b) una aspiradora robot, c) un carro a control remoto sin sensores, d) un semáforo con sensor de tráfico que alarga el verde si hay fila de carros, e) una impresora 3D.

Solución paso a paso: Paso 1: repasa la definición: necesita sensores (percibir), un programa que tome decisiones con esa información (decidir) y actuadores que modifiquen el mundo (actuar), todo en ciclo continuo. Paso 2: analiza el ventilador con temporizador: no percibe nada del entorno (el temporizador es tiempo fijo, no un sensor del mundo), así que no es robot: es un autómata. Paso 3: la aspiradora robot: percibe con sensores de distancia y choque, decide su ruta y actúa moviendo sus ruedas y succionando: sí es robot. Paso 4: el carro a control remoto sin sensores: percibe (tú eres sus ojos), pero quien decide eres tú, no él: no es un robot autónomo, es una máquina teledirigida. Paso 5: el semáforo con sensor de tráfico: percibe la fila de carros, decide alargar el verde y actúa cambiando las luces: sí es un robot, aunque esté clavado al poste y no se mueva (los robots no necesitan patas). Paso 6: la impresora 3D: percibe (sensores de temperatura, de fin de carrera), decide (ajusta el flujo según el programa y las lecturas) y actúa (mueve los ejes y extruye plástico): sí es un robot. Conclusión: son robots la aspiradora, el semáforo inteligente y la impresora 3D. La lección: robot no significa "con forma humana", significa "máquina que siente, piensa y actúa".

Ejercicio 2. Dibuja la anatomía de tu robot ideal

En papel, diseña un robot que riegue tus plantas cuando la tierra está seca. Dibuja sus cuatro sistemas (actuadores, sensores, controlador, chasis) más su energía, nombra cada pieza concreta que usarías y escribe en una frase qué pregunta del mundo responde cada sensor.

Solución paso a paso: Paso 1: define el comportamiento en una frase: "si la tierra está seca, abre la válvula de agua 10 segundos; si está húmeda, no hagas nada; repite cada hora". Esa frase es tu programa antes de ser código. Paso 2: elige el sensor: un sensor de humedad de suelo (responde "¿qué tan húmeda está la tierra?"); opcionalmente un sensor de nivel en el tinaco de agua (responde "¿queda agua?"). Paso 3: elige el actuador: una mini bomba de agua de 5V o una electroválvula (responde a la orden "deja pasar agua"), más un LED que indique que está regando. Paso 4: elige el controlador: un Arduino Nano o un ESP32 si quieres que además te avise por WiFi a tu celular (responde ejecutando tu programa en loop). Paso 5: el chasis y la energía: una cajita plástica impermeable para la electrónica y una batería o un eliminador de corriente (la bomba gasta más que el Arduino: aliméntala por separado con tierra común, como aprendiste en la anatomía). Paso 6: verifica tu diseño contra la definición: percibe (humedad), decide (comparar contra un umbral en el programa), actúa (bomba): es un robot completo. Si tu dibujo tiene las cinco piezas nombradas y cada sensor con su pregunta, lo hiciste bien.

Ejercicio 3. Depura un seguidor de línea que huye de la línea

Copiaste el programa del Ejemplo 1, pero tu robot hace lo contrario: cuando el sensor izquierdo ve la línea negra, el robot gira a la derecha y se sale de la pista. Tienes el monitor serial abierto y ves que sobre blanco los sensores marcan LOW y sobre negro marcan HIGH (al revés del ejemplo). Explica la causa y escribe la corrección exacta.

Solución paso a paso: Paso 1: interpreta las lecturas: tu sensor entrega LOW en blanco y HIGH en negro, es decir, su lógica está invertida respecto a la que supone el programa (algunos módulos IR vienen así según su circuito). Paso 2: ubica el problema en el código: los cuatro if comparan contra HIGH esperando blanco y LOW esperando negro; con tu sensor, esas comparaciones deciden al revés. Paso 3: la corrección más limpia no es reescribir los cuatro if, sino normalizar la lectura al inicio del loop, justo después de digitalRead: agrega dos líneas que inviertan el valor, por ejemplo "izq = !izq;" y "der = !der;" (el operador ! invierte HIGH a LOW y viceversa), o usa variables nuevas como "int izqN = (izq == HIGH) ? LOW : HIGH;". Paso 4: con la lectura normalizada, el resto del programa funciona sin tocarlo: ahora blanco es HIGH y negro es LOW como espera la lógica. Paso 5: verifica en el monitor serial que al pasar negro bajo el sensor izquierdo la lectura normalizada sea LOW y el robot gire a la izquierda. Paso 6: lección general: nunca supongas la lógica de un sensor, mírela con el monitor serial y

calibra el programa contra la realidad; este error ("lógica invertida") es el número uno entre principiantes y se resuelve en dos líneas cuando sabes dónde mirar.

Ejercicio 4. Mejora el evasor de obstáculos: que elija el mejor lado

El programa del Ejemplo 2 siempre gira a la derecha al encontrar un obstáculo, así que a veces gira directo hacia otra pared. Mejora el comportamiento: al detectar un obstáculo, el robot debe detenerse, "mirar" a la izquierda y a la derecha (girando su cuerpo un poco a cada lado y midiendo), y luego girar hacia el lado con más espacio libre. Describe el algoritmo paso a paso y escribe el fragmento de código que reemplaza al bloque de evasión.

Solución paso a paso: Paso 1: define el nuevo comportamiento en palabras: "si hay obstáculo a menos del umbral: detente; gira el cuerpo 45 grados a la izquierda, mide y guarda la distancia; regresa al centro; gira 45 grados a la derecha, mide y guarda; gira hacia el lado con mayor distancia; avanza". Paso 2: traduce "girar el cuerpo un poco" a funciones que ya tienes: puedes reusar girarDerecha() y girarIzquierda() con delays cortos calibrados (por ejemplo 250 ms para unos 45 grados; el valor exacto lo calibras probando). Necesitas agregar la función girarIzquierda() espejo de girarDerecha(). Paso 3: escribe el nuevo bloque que reemplaza al else del loop:

```

} else {
    // Obstaculo al frente: detenerse y explorar ambos lados
    detener();
    delay(200);

    // Mirar a la izquierda
    girarIzquierda();
    delay(250);          // calibrar: aprox 45 grados
    detener();
    delay(200);
    long distIzq = medirDistancia();

    // Regresar al centro y mirar a la derecha
    girarDerecha();
    delay(250);
    detener();
    delay(200);
    girarDerecha();
    delay(250);
    detener();
    delay(200);
    long distDer = medirDistancia();

    // Elegir el lado con mas espacio y avanzar hacia alla
    if (distIzq > distDer) {
        girarIzquierda();
        delay(500);      // giro completo hacia la izquierda
    } else {
        girarDerecha();
        delay(500);      // giro completo hacia la derecha
    }
    detener();
}
}

```

Paso 4: explica por qué funciona: el robot convierte su propio cuerpo en la "cabeza" que mira, usando el mismo sensor; la decisión usa datos reales de ambos lados en lugar de una dirección fija. Paso 5: calibra los delays con el robot real, porque 250 ms no son 45 grados exactos en todos los robots: prueba y ajusta hasta que el giro de exploración sea simétrico. Paso 6: mejora opcional de nivel avanzado: en lugar de girar el cuerpo, monta el sensor en un servomotor que barra de izquierda a derecha sin mover el robot (más rápido y elegante); el algoritmo de comparar distancias es el mismo.

Ejercicio 5. Plan de vuelo responsable para tu primer dron

Vas a volar tu primer dron (uno de juguete con cámara) en el parque de tu colonia un sábado por la mañana. Escribe tu plan de vuelo en cinco puntos: verificación legal, verificación del sitio, verificación del equipo, plan de vuelo y plan de emergencia. Usa lo aprendido en normativa y en el checklist del Ejemplo 4.

Solución paso a paso: Paso 1: verificación legal: confirma que el parque no esté en zona restringida ni cerca de un aeropuerto (revisa el mapa oficial), que volarás de día y mantendrás el dron a la vista; si tu dron supera el peso que la norma marca para registro, regístralo antes de volar. Paso 2: verificación del sitio: llega temprano, camina el área y elige una zona abierta lejos de la gente que hace ejercicio, de los árboles grandes, de los cables de luz y de los juegos infantiles; si el parque está lleno, cambia de lugar o de horario: nunca vuelas sobre multitudes. Paso 3: verificación del equipo: recorre el checklist (baterías al 100 por ciento, hélices sin daños, control cargado, cámara con memoria libre); haz una prueba de 2 metros verificando los cuatro mandos antes de alejarlo. Paso 4: plan de vuelo: define qué vas a hacer antes de despegar, por ejemplo "vuelos de 5 minutos a máximo 20 metros de altura, practicando despegue, vuelo estacionario, giros suaves y aterrizaje, siempre dentro de la cancha vacía"; pon un temporizador para aterrizar con 20 por ciento de batería. Paso 5: plan de emergencia: si pierdes la orientación, suelta los mandos de dirección y baja despacio en vertical; si el viento aumenta, aterriza de inmediato; si el dron se acerca a una persona, corta motores solo si ya está a centímetros del suelo (nunca apagues motores en el aire sobre gente); ten a la mano el número de emergencias. Si tus cinco puntos cubren legal, sitio, equipo, plan y emergencia, tienes un plan de piloto responsable, no de improvisado.

Ejercicio 6. Diseña tu robot conectado: del problema al sistema

Elige un problema real de tu casa o tu colonia y diseña un sistema completo de "robótica más IA más IoT" que lo resuelva. Describe: el problema, los sensores y actuadores, dónde corre la IA (nube o borde) y por qué, qué datos viajan por Internet y qué se queda local, y un modo de seguridad por si falla la conexión.

Solución paso a paso (ejemplo guía, el tuyo puede ser distinto): Paso 1: el problema: "se meten gatos ajenos en la noche y tiran la basura del patio". Paso 2: sensores y actuadores: una cámara con visión nocturna (sensor principal), un sensor de movimiento PIR para despertar al sistema solo cuando algo se mueve (ahorra energía), un ESP32-CAM o Raspberry Pi como controlador, y como actuador una luz potente más un aspersor de agua (los gatos odian el agua; no se lastima a nadie). Paso 3: dónde corre la IA: en el borde, dentro de la Raspberry Pi con un modelo ligero preentrenado que distingue gato de persona y de perro; ¿por qué en el borde y no en la nube? porque debe reaccionar en segundos, funcionar

aunque se caiga el Internet y no mandar video de tu patio a servidores ajenos toda la noche. Paso 4: qué viaja por Internet y qué se queda local: local se queda el video crudo y la decisión en tiempo real; a la nube (por MQTT o una app) solo viajan eventos cortos: "gato detectado 02:14, foto adjunta", más una foto cuando hay detección; así no saturas tu WiFi. Paso 5: control remoto: desde tu celular puedes ver el historial de eventos, desactivar el aspensor si vas a salir al patio de noche y ajustar la sensibilidad. Paso 6: modo de seguridad sin conexión: si se cae el WiFi, el sistema sigue funcionando local (detecta y activa el aspensor) y guarda los eventos para enviarlos cuando vuelva la red; si falla la cámara, el PIR solo activa la luz como respaldo. Si tu diseño responde las seis preguntas (problema, piezas, dónde va la IA y por qué, qué viaja y qué no, control remoto y modo sin conexión), acabas de diseñar como un ingeniero de sistemas, no como un aficionado que pega piezas. Ese es el nivel al que apunta esta unidad.

Resumen de la unidad

Cierra el libro un momento y piensa en todo lo que ahora sabes que hace unas páginas no sabías. Un robot es una máquina que percibe con sensores, decide con un programa y actúa con motores, en un ciclo continuo; no necesita forma humana: un dron, una aspiradora robot y una impresora 3D son robots tan legítimos como un humanoide. Conoces las cuatro familias: industriales (precisión incansable en fábricas), de servicio (conviven con humanos), educativos (baratos, para aprender los mismos principios que usan los grandes) y humanoides (los más difíciles, todavía en desarrollo).

Conoces la anatomía completa: los actuadores son los músculos (motores DC para velocidad, servos para posición exacta, steppers para precisión), los sensores son los sentidos (ultrasónicos como murciélagos, infrarrojos para líneas, cámaras para ver el mundo), el controlador es el cerebro (Arduino para control en tiempo real, Raspberry Pi cuando necesitas más poder como video) y el chasis es el esqueleto que sostiene todo, alimentado por baterías y drivers que manejan la potencia bruta. Y te quedaron grabadas las lecciones de taller: tierra común siempre, driver para los motores, calibrar sensores en el entorno real y probar pieza por pieza antes de armar el todo.

Programaste (o simulaste) los dos comportamientos clásicos: el seguidor de línea, que con dos sensores y una tablita de cuatro casos persigue una línea como tren sobre rieles, y el evasor de obstáculos, que mide distancias con ultrasonido y esquivo como murciélago. Dominas el PWM para controlar velocidades con analogWrite, sabes usar el monitor serial para ver lo que "piensa" tu robot y conoces los errores típicos (lógica invertida, delays eternos, baterías bajas) con sus soluciones.

Despegaste al mundo de los drones: sabes que un multirrotor vuela porque su controladora ajusta cada hélice cientos de veces por segundo, conoces sus componentes (frame, brushless, ESC, controladora, LiPo, gimbal) y sus tipos (juguete, fotografía, FPV de carreras, profesionales). Y lo más importante: conoces la parte seria, la normativa y el vuelo responsable: revisar la norma vigente (AFAC en México), volar de día y a la vista, lejos de aeropuertos y multitudes, respetar la privacidad y usar checklist pre-vuelo siempre. Un buen roboticista también es un buen piloto.

Viste las aplicaciones que ya mueven industrias: fotografía aérea, agricultura de precisión, inspección de infraestructura con cámaras térmicas, reparto y emergencias; y entendiste que el valor está en combinar el dron con otra habilidad y resolver un problema real. Le diste ojos a tus robots con la visión por computadora: las imágenes son matrices de números, el color se detecta con rangos HSV, OpenCV y Python bastan para empezar, y la IA preentrenada te lleva al reconocimiento de objetos reales. Y cerraste con la fusión total: robótica más IA más IoT, el robot conectado que percibe, aprende y se comunica, con la IA en la nube o en el borde según el caso, y siempre con un modo local por si falla la red.

Tu siguiente paso es físico: consigue un kit básico de Arduino o abre Tinkercad Circuits hoy mismo y haz que algo se mueva con tu código. Después, un dron de juguete para aprender a volar sin miedo. Cada curso de esta academia te dio una pieza; esta unidad te enseñó a ensamblarlas en máquinas que tocan el mundo real. En lucasegaacademy.com te esperan más cursos para seguir creciendo, pero el taller más importante ya lo tienes: tu curiosidad, tus manos y las ganas de construir. Nos vemos en la siguiente unidad, futuro roboticista.

UNIDAD 10: Proyectos que combinan varias tecnologías

Llegaste a la última unidad del Curso Completo de Tecnología Fusión, y el tema no es casualidad. En las unidades anteriores aprendiste cada tecnología por separado: sensores y actuadores, microcontroladores como el ESP32, computadoras pequeñas como la Raspberry Pi, programación en Python y MicroPython, energía solar, inteligencia artificial, impresión 3D, nube, tableros de control y redes con MQTT. Cada una de esas piezas es poderosa por sí sola. Pero el verdadero poder aparece cuando las juntas.

Un proyecto fusión es exactamente eso: un proyecto que combina dos o más tecnologías para resolver un problema real de una forma que ninguna tecnología podría lograr por sí sola. Un sensor de humedad solo mide. Una bomba de agua solo riega. Una app en tu celular solo muestra datos. Pero júntalos -sensor que mide, microcontrolador que decide, bomba que riega, panel solar que alimenta todo y app que te avisa- y de pronto tienes un invernadero inteligente que cuida tus plantas mientras duermes. Eso ya no es un experimento: es un sistema.

Esta unidad es el cierre del curso porque aquí todo se conecta. Vas a aprender la metodología para planear un proyecto fusión de principio a fin: cómo nace la idea, cómo defines el alcance para no morder más de lo que puedes masticar, cómo calculas el presupuesto sin sustos, cómo construyes un prototipo rápido y cómo lo mejoras por iteraciones. Después vas a construir tres proyectos completos paso a paso, con código real que puedes copiar y adaptar: un invernadero inteligente, una estación meteorológica conectada y un asistente de voz casero. Y en la práctica vas a diseñar otros dos -un sistema de seguridad con inteligencia artificial y un monitor de energía del hogar- con ejercicios guiados y soluciones explicadas.

Al final, además, vas a aprender algo que casi ningún curso técnico enseña: cómo documentar tus proyectos para que otros los entiendan, cómo armar un portafolio que te abra puertas y cómo convertir un proyecto en un pequeño negocio si te animas. Porque de nada sirve construir algo increíble si nadie sabe que existe.

De las piezas sueltas al sistema completo

Piensa en todo lo que aprendiste como piezas de un rompecabezas. La unidad de sensores te dio los ojos del sistema. Los microcontroladores te dieron el cerebro pequeño y barato. Python te dio el lenguaje para pensar. La energía solar te dio independencia del enchufe. La inteligencia artificial te dio la capacidad de decidir con datos. La impresión 3D te dio el cuerpo

físico: carcasas, soportes, piezas a la medida. Y la nube con MQTT te dio la voz para que tu proyecto hable con el mundo.

Un proyecto fusión arma ese rompecabezas. Y aquí hay una idea clave que te va a servir toda la vida: todo sistema fusión, por complejo que parezca, se puede descomponer en seis capas. Uno: la capa física, que son los sensores y actuadores tocando el mundo real. Dos: la capa de control, el microcontrolador o la computadora que toma decisiones. Tres: la capa de conectividad, normalmente Wi-Fi y MQTT, que lleva los datos de un lado a otro. Cuatro: la capa de datos, donde la información se guarda y se analiza, en la nube o en local. Cinco: la capa de presentación, que es lo que tú ves: la app, el tablero, las notificaciones. Seis: la capa de energía, que alimenta todo lo anterior, con baterías o paneles solares.

Mira cómo se ve esto con un ejemplo que ya conoces: un termostato inteligente. La capa física es un sensor de temperatura y un relé que prende el ventilador. La capa de control es un ESP32 que compara la temperatura con el valor que quieres. La conectividad es el Wi-Fi de tu casa publicando por MQTT. Los datos se guardan para graficar cómo cambia la temperatura durante el día. La presentación es una app donde ajustas la temperatura desde el sillón. Y la energía es un simple eliminador USB... o un panel solar si quieres ponerlo en el jardín. Seis capas, un solo sistema, cero magia.

Cuando entiendes las seis capas, ningún proyecto te asusta, porque siempre puedes preguntarte: ¿qué va en cada capa? Esa pregunta convierte un proyecto gigante en seis problemas pequeños. Y seis problemas pequeños sí se pueden resolver uno por uno. Esa es la mentalidad de esta unidad: divide, conquista e integra.

Hay otra razón por la que los proyectos fusión son especiales: te obligan a diseñar para el mundo real, no para el escritorio. En tu escritorio el Wi-Fi nunca se cae, la luz nunca se va y los sensores siempre están limpios. En el mundo real, el Wi-Fi se cae a las 3 de la mañana, llueve sobre tu caja de conexiones y un cable se afloja justo cuando no estás. Un buen proyecto fusión anticipa eso: se reconecta solo, guarda los datos aunque no haya internet, y te avisa cuando algo anda mal en lugar de quedarse callado. En esta unidad vas a aprender esos trucos de robustez que separan un juguete de un sistema de verdad.

Por qué los proyectos fusión valen oro en tu carrera

Hablemos de dinero y de trabajo, sin rodeos. Si algún día quieres conseguir un empleo en tecnología, o vender tus servicios como independiente, hay una verdad incómoda: a nadie le importa cuántos cursos tomaste. Lo que importa es qué construiste. Un certificado dice "tomé un curso". Un proyecto funcionando dice "resuelvo problemas".

Imagina dos candidatos. Uno dice: "Sé Python, sé IoT y sé algo de inteligencia artificial". El otro abre su celular y muestra: "Mira, este es mi invernadero: el sensor mide la humedad, el sistema riega solo, aquí está el tablero con los datos del mes y esta es la app que me avisa si algo falla. Lo documenté todo en mi portafolio". ¿A quién contratas? ¿A quién le pagas por automatizar su negocio? La respuesta es obvia. Los proyectos fusión son la mejor entrevista de trabajo que existe, porque hablan por ti.

Y no solo demuestran técnica. Un proyecto fusión completo demuestra que sabes planear (definiste un alcance), que sabes administrar dinero (hiciste un presupuesto), que sabes comunicar (lo documentaste) y que sabes terminar lo que empiezas (lo iteraste hasta que funcionó). Esas son exactamente las habilidades que separan a un técnico de un profesional. Por eso en esta unidad le damos tanta importancia a la metodología y a la documentación, no solo al código.

Incluso si no quieres dedicarte a esto profesionalmente, los proyectos fusión se pagan solos. Un sistema de riego automático comercial cuesta varios miles de pesos; el tuyo cuesta una fracción y además aprendiste en el proceso. Un monitor de energía te ayuda a detectar qué aparato está disparando tu recibo de luz. Una estación meteorológica te da datos de tu propia colonia, más precisos para ti que los de toda la ciudad. La tecnología fusión no es un gasto: es una inversión que regresa en ahorro, en conocimiento y, si quieres, en ingresos.

Hoja de ruta: los cinco proyectos de esta unidad

Vamos a lo concreto. En la sección de ejemplos vas a construir tres proyectos completos, paso a paso y con código real que puedes copiar, adaptar y mejorar:

- PROYECTO 1: Invernadero inteligente. Combina IoT, automatización, energía solar y una app de monitoreo. El ESP32 lee la humedad de la tierra y la temperatura, decide cuándo regar, activa la bomba y te reporta todo al celular. Es el proyecto ideal para empezar porque las plantas son pacientes: perdonan tus primeros errores.
- PROYECTO 2: Estación meteorológica conectada. Combina sensores, nube, tablero de control y alertas. Una Raspberry Pi mide temperatura, humedad y presión, sube los datos a la nube, los muestra en un tablero bonito y te manda un mensaje si va a helar o si el viento está muy fuerte.
- PROYECTO 3: Asistente de voz casero. Combina inteligencia artificial, IoT y hogar inteligente. Le hablas a tu Raspberry Pi, entiende lo que dices, prende y apaga luces, te dice el clima que mide tu propia estación meteorológica y hasta responde preguntas básicas.

Y en la sección de práctica vas a diseñar otros dos, con ejercicios guiados y soluciones explicadas paso a paso:

- PROYECTO 4: Sistema de seguridad con IA. Cámaras más visión por computadora: detecta personas, te manda foto al celular cuando hay movimiento y distingue entre tu perro y un extraño.
- PROYECTO 5: Monitor de energía del hogar. Mide cuánta luz consume tu casa en tiempo real, sube los datos a la nube y usa una carcasa impresa en 3D hecha a la medida.

El orden está pensado a propósito: cada proyecto reutiliza algo del anterior. La estación meteorológica usa los mismos sensores y la misma nube que el invernadero. El asistente de voz puede leer los datos de tu estación. El sistema de seguridad usa las mismas notificaciones. Al final no vas a tener cinco proyectos sueltos: vas a tener un ecosistema, igual que las unidades de este curso que se conectan entre sí.

En cuanto a herramientas generales, con lo que ya usaste en el curso basta: protoboard, cables dupont, un multímetro, cautín si quieres soldar las versiones finales, y tu computadora. Cada proyecto trae su propia lista de materiales con costos estimados en pesos mexicanos, para que sepas exactamente en qué te metes antes de comprar nada.

Una advertencia honesta antes de empezar: no necesitas comprar todos los componentes hoy. Lee primero la unidad completa, entiende la arquitectura de cada proyecto y compra por etapas, empezando por el proyecto 1. En la sección de explicación detallada te enseño a armar un presupuesto realista para que no haya sustos. Y un consejo de seguridad que va en serio: donde haya agua y electricidad juntas (como en el invernadero), las conexiones eléctricas van siempre en cajas selladas y elevadas del suelo. La tecnología es divertida; los cortos no.

Te recomiendo también llevar una bitácora del proyecto desde el día uno. Puede ser un cuaderno o un archivo de texto. Anota qué intentaste, qué funcionó, qué falló y qué cambiaste. Esa bitácora es oro puro: cuando algo deje de funcionar dentro de seis meses, vas a agradecer tenerla; y cuando documentes el proyecto para tu portafolio, la mitad del trabajo ya estará hecha.

¿Listo? Vamos primero con la metodología: el mapa que te lleva de la idea al proyecto terminado sin perderte en el camino.

Explicación detallada

Fase 1 y 2: la idea y el alcance (el mapa antes del territorio)

Todo proyecto fusión nace dos veces: primero en tu cabeza y después en tu mesa de trabajo. La primera vez que nace es la más peligrosa, porque las ideas son traicioneras: se sienten completas cuando en realidad están a la mitad. Por eso la metodología empieza con dos fases de pensamiento antes de tocar un solo cable.

La fase 1 es la idea, y tiene una regla de oro: define el problema, no la tecnología. "Quiero usar un ESP32 con MQTT" no es una idea, es una lista de compras. "Mis plantas se mueren cuando salgo de viaje porque nadie las riega" sí es una idea, porque describe un problema real con un doliente real (tus plantas... y tú). Hazte estas tres preguntas y escríbelas: ¿qué problema resuelvo?, ¿para quién lo resuelvo?, ¿cómo voy a saber que funcionó? Esa tercera pregunta es el criterio de éxito, y tiene que ser medible: "la tierra nunca baja del 30 por ciento de humedad durante 30 días" es un criterio; "que se vea padre" no lo es.

La fase 2 es el alcance, y su trabajo es protegerte de ti mismo. Cuando la idea te emociona, tu cerebro empieza a agregar cosas: "ya que estoy, le pongo cámara... y que también mida el pH... y una app con gráficas 3D...". Eso se llama alcance infinito y es la causa número uno de proyectos abandonados. La herramienta contra él es el MVP: el producto mínimo viable. Pregúntate: ¿cuál es la versión más simple que resuelve el problema? Para el invernadero, el MVP es: un sensor de humedad, una bomba que riega cuando la tierra está seca, y un mensaje que te avisa. Nada de cámaras, nada de app con gráficas. Si el MVP funciona durante dos semanas, entonces sí agregas lo demás, una cosa a la vez.

Escribe tu alcance en dos listas: "esta versión SÍ incluye" y "esta versión NO incluye (todavía)". La segunda lista es la más importante. Cada vez que se te ocurra una mejora, no la hagas: anótala en la lista del "todavía". Tu yo del futuro te lo va a agradecer.

Fase 3: el presupuesto sin sustos

Hablar de dinero antes de comprar es lo que separa un hobby sostenible de un cajón lleno de componentes sin usar. Un presupuesto de proyecto fusión tiene cuatro partes: componentes, herramientas que no tengas, envíos y un colchón para imprevistos.

Empieza listando cada componente con su precio real, no el que recuerdas. Búscalos en dos o tres tiendas y anota el precio con envío. Aquí tienes un ejemplo realista para el invernadero inteligente, con precios aproximados en pesos mexicanos:

- ESP32 DevKit v1: \$180
- Sensor DHT22 (temperatura y humedad del aire): \$120

- Sensor de humedad de suelo capacitivo (no compres el resistivo, se corroe): \$90
- Módulo relé de 5V para la bomba: \$60
- Mini bomba de agua sumergible 5V: \$150
- Panel solar de 10W con controlador de carga: \$450
- Batería 18650 con portapilas y protección: \$120
- Caja estanca IP65 para la electrónica: \$180
- Manguera, goteros y conexiones: \$150
- Cables dupont, protoboard y tornillería: \$100
- Subtotal: \$1,600

Ahora agrega el colchón: entre el 15 y el 20 por ciento para imprevistos. Algo siempre sale mal: un sensor llega defectuoso, necesitas un cable más largo, se te quema un componente aprendiendo. Con el 20 por ciento, el presupuesto real queda en \$1,920. Si al final sobra, mejor: ese sobrante financia tu siguiente proyecto.

Hay costos ocultos que casi nadie presupuesta y aquí te los digo de frente: los envíos (que pueden duplicar el precio de componentes baratos), las herramientas que no tienes (un buen caudín cuesta \$300), la impresión 3D si no tienes impresora (cotízala por pieza), y tu tiempo. Tu tiempo vale: si un componente genérico te ahorra tres horas de trabajo por \$50 extra, cómpralo.

Un truco profesional: compra por etapas. Etapa 1: lo mínimo para el prototipo en protoboard (microcontrolador, sensores, cables). Etapa 2: lo necesario para la versión robusta (caja estanca, fuente definitiva, conectores). Etapa 3: los lujos (pantalla, carcasa impresa). Si el proyecto muere en la etapa 1 -y algunos mueren, es normal-, perdiste poco dinero y ganaste aprendizaje.

Fase 4: prototipado inteligente (empieza feo, termina sólido)

El prototipo es la versión fea y honesta de tu proyecto: cables por todos lados, protoboard, cinta adhesiva. Y está bien que sea feo, porque su único trabajo es responder una pregunta: ¿la idea funciona? La regla de oro del prototipado es: nunca soldes nada el primer día, y nunca metas la electrónica en su caja final antes de que funcione en la mesa.

Trabaja por capas, de adentro hacia afuera. Primero la capa de control y la física: conecta un sensor, lee sus valores en la consola, verifica que tengan sentido (si el sensor dice 90 grados en tu cuarto, algo anda mal y es mejor descubrirlo ahora). Después agrega un actuador y comprueba que responde. Luego la conectividad: publica un dato por MQTT y léelo desde tu

celular. Después la energía definitiva: mide cuánto consume tu sistema y verifica que la batería o el panel aguanten. Al final, la presentación: el tablero, las notificaciones.

Cada capa se prueba sola antes de integrarse con la siguiente. Esto suena lento, pero es lo más rápido que existe: cuando algo falla -y va a fallar-, sabes exactamente en qué capa está el problema. El que conecta todo de golpe y luego no funciona pasa tres días adivinando.

Ponle versión a todo, como el software: v0.1 es el prototipo en protoboard, v0.2 es la versión en caja pero todavía en tu mesa, v1.0 es la instalada en su lugar definitivo funcionando una semana. Guarda cada versión del código con su número. Cuando la v0.3 deje de funcionar y no sepas por qué, vas a querer regresar a la v0.2 que sí funcionaba.

Fase 5: iteración con datos (mejorar no es adivinar)

Iterar significa mejorar por ciclos, y cada ciclo tiene tres pasos: medir, cambiar una sola cosa, volver a medir. La parte de "una sola cosa" es la que más cuesta respetar. Si cambias el sensor, el código y la posición de la caja al mismo tiempo y algo mejora, no sabes qué fue. Cambia una cosa, anota el resultado en tu bitácora, sigue.

Los datos mandan. "Siento que riega mejor" no es un dato. "La humedad del suelo se mantuvo entre 30 y 45 por ciento durante 14 días, y antes variaba entre 10 y 60" sí es un dato. Por eso tu proyecto debe registrar lo que hace desde el día uno: cada riego, cada lectura, cada error. Cuando tengas un mes de datos, las mejoras se vuelven obvias: vas a ver a qué hora se seca más rápido la tierra, cuánta agua usa por semana, si el sensor se desfasa con el calor.

Define desde el inicio cada cuánto vas a iterar. Para proyectos caseros, un ciclo de una o dos semanas funciona bien: tiempo suficiente para juntar datos, corto para no aburrirte. Y define también cuándo parar: un proyecto está terminado cuando cumple su criterio de éxito de forma estable, no cuando es perfecto. La perfección es el enemigo de lo terminado.

Documentación que cuenta: tu proyecto también se lee

Un proyecto sin documentación es un secreto: solo tú sabes cómo funciona, y dentro de seis meses ni tú te vas a acordar. Documentar no es un trámite para el final; es parte del proyecto. La buena noticia es que si llevaste tu bitácora desde el día uno, ya tienes la mitad hecha.

Todo proyecto merece un README, un documento que explique el proyecto a alguien que nunca lo ha visto. Debe tener, en este orden: qué es y qué problema resuelve (dos párrafos), una foto del proyecto terminado, la lista de materiales con precios, el diagrama de conexiones (aunque sea un dibujo a mano fotografiado), el código con instrucciones para cargarlo, cómo replicarlo paso a paso, y una sección de "problemas conocidos" donde anotes

honestamente qué no funciona perfecto. Esa última sección es la que más respetan los que saben: demuestra que probaste de verdad.

Para el portafolio, cada proyecto es una historia en tres actos: el problema (qué dolía), la construcción (qué hiciste, con fotos del proceso: el prototipo feo también cuenta, y mucho), y el resultado (qué lograste, con datos: gráficas, números, fotos del sistema funcionando). Los datos del resultado son tu prueba: "riega solo desde hace 3 meses" vale más que mil adjetivos.

Guarda todo en un repositorio (GitHub es gratis) con el código versionado, y publica el portafolio en una página sencilla. No necesita ser elegante: necesita ser claro. Un reclutador dedica 30 segundos a cada proyecto; en esos 30 segundos debe entender qué es, ver que funciona y encontrar el código.

Errores comunes que rompen proyectos fusión

Los siguientes errores los comete casi todo el mundo al menos una vez. Te los adelanto para que los cometas menos veces:

Uno: empezar por la tecnología en vez del problema. Comprar primero y pensar después termina en cajones llenos de componentes. Primero el problema, luego el alcance, luego la lista de compras.

Dos: el alcance infinito. "Ya que estoy..." es la frase más cara del maker. Cada agregado duplica el tiempo y el riesgo. Disciplínate con tu lista de "todavía no".

Tres: ignorar la alimentación. La energía es la causa número uno de fallas raras: el ESP32 se reinicia solo, el sensor da lecturas locas, el Wi-Fi se desconecta. Casi siempre es una fuente insuficiente o cables delgados. Mide el consumo real y sobredimensiona la fuente un 30 por ciento.

Cuatro: Wi-Fi frágil. Tu código debe asumir que el Wi-Fi se va a caer y reconectarse solo, sin que tengas que ir a reiniciarlo. Todo el código de ejemplo de esta unidad incluye reconexión por esta razón.

Cinco: no calibrar los sensores. Un sensor de humedad de suelo sin calibrar te dice un número sin significado. La calibración es simple: mide en tierra seca, mide en tierra bien regada, y esos dos valores son tu 0 y tu 100. Cinco minutos que te ahorran semanas de datos inútiles.

Seis: la caja bonita antes de tiempo. Meter todo en su caja final cuando aún no funciona es como empacar la maleta antes de saber a dónde viajas. Primero funciona en la mesa, después se viste de gala.

Siete: sin protección eléctrica. Un fusible de unos pesos protege tu batería de un corto. Un diodo protege contra polaridad invertida. Barato, simple, y te salva del humo.

Ocho: documentar "al final". El final nunca llega con calma: o el proyecto ya funciona y no quieres detenerte, o no funciona y estás frustrado. Documenta un poco cada semana, junto con tu bitácora.

Buenas prácticas de oro

Cierra la metodología con las prácticas que usan los que llevan años construyendo sistemas que duran:

- Cambia una sola cosa a la vez cuando estés probando o iterando.
- Usa deep sleep en todo lo que funcione con batería: un ESP32 dormido consume casi nada y despierta a medir cada cierto tiempo.
- Activa el watchdog timer: es un vigilante dentro del microcontrolador que lo reinicia solo si el programa se traba.
- Organiza tus tópicos MQTT con estructura fija, por ejemplo casa/invernadero/datos: cuando tengas diez dispositivos lo vas a agradecer.
- Guarda una copia local de los datos importantes aunque uses la nube: si el internet se cae, tu sistema sigue registrando.
- Etiqueta cada cable en la versión final. Parece exagerado hasta el día que tienes que reparar algo a las 11 de la noche.
- Nunca expongas tu broker MQTT directo a internet sin contraseña y cifrado. Lo local primero; lo remoto, con cuidado.
- Respeta el agua y la electricidad: conexiones selladas, cajas elevadas, y si algo se calienta más de lo normal, apágalo e investiga.
- Versiona tu código y guarda las versiones que funcionaron. El "voy a mejorar este código" ha roto más proyectos que cualquier otra frase.
- Comparte lo que aprendas. Publicar tu proyecto documentado te devuelve más de lo que crees: preguntas, ideas y hasta ofertas.

Ejemplos paso a paso

PROYECTO 1: Invernadero inteligente (IoT + automatización + energía solar + app)

Qué vas a construir: un sistema que cuida tus plantas solo. Un ESP32 mide la humedad de la tierra y la temperatura del aire cada 30 minutos; si la tierra está seca, activa una bomba que riega por goteo; todo se alimenta con un panel solar; y los datos llegan a tu celular por MQTT para que los veas en un tablero y recibas alertas. Tecnologías que combina: IoT (sensores y ESP32), automatización (riego por decisión), energía solar (panel y batería) y app de monitoreo (tablero MQTT en el celular).

Paso 1: alcance del MVP. Esta versión SÍ incluye: medir humedad de suelo y temperatura, regar automáticamente cuando la humedad baje del 30 por ciento, publicar datos por MQTT, alerta si la temperatura pasa de 35 grados, funcionar con panel solar. Esta versión NO incluye (todavía): cámara, medición de pH, app propia programada desde cero, control manual desde internet. Criterio de éxito: la humedad del suelo se mantiene entre 30 y 50 por ciento durante 30 días sin que riegues a mano ni una vez.

Paso 2: materiales y costo. La lista completa con precios aproximados en pesos mexicanos: ESP32 DevKit v1 (\$180), sensor DHT22 (\$120), sensor capacitivo de humedad de suelo (\$90), módulo relé 5V (\$60), mini bomba sumergible 5V (\$150), panel solar 10W con controlador (\$450), batería 18650 con protección (\$120), caja estanca IP65 (\$180), manguera y goteros (\$150), cables y protoboard (\$100). Subtotal \$1,600 más 20 por ciento de colchón: \$1,920 de presupuesto total.

Paso 3: arquitectura en las seis capas. Física: DHT22, sensor de suelo, relé y bomba. Control: ESP32 con MicroPython. Conectividad: Wi-Fi de tu casa y MQTT hacia un broker (puede ser una Raspberry Pi con Mosquitto o un broker en la nube). Datos: el broker guarda el historial y el tablero lo grafica. Presentación: app MQTT en tu celular (como MQTT Dash) y una automatización que te avisa por Telegram. Energía: panel solar de 10W, controlador de carga y batería 18650; el ESP32 usa deep sleep para que la batería dure semanas sin sol.

Paso 4: conexiones. Conecta el DHT22: VCC a 3V3, GND a GND, datos al pin 15. El sensor de suelo capacitivo: VCC a 3V3, GND a GND, salida analógica al pin 34. El relé: VCC a 5V (o VIN), GND a GND, señal al pin 26; la bomba va en serie con el contacto del relé y su propia alimentación de 5V. Ojo: el relé separa la parte de potencia de la lógica; nunca alimentes la bomba desde los pines del ESP32. La bomba va dentro de un recipiente con agua y la manguera lleva el agua a las macetas con goteros.

Paso 5: calibración del sensor de suelo (haz esto antes de programar los umbrales). Mete el sensor en tierra completamente seca y anota el valor analógico que lee; luego riega muy bien esa tierra, espera 10 minutos y anota el valor. Esos dos números son tu SUELO_SECO y tu SUELO_HUMEDO en el código. Sin este paso, los porcentajes no significan nada.

Paso 6: el código. Este programa en MicroPython hace todo el trabajo: se conecta al Wi-Fi (y si no hay, se duerme y reintenta), lee los sensores, decide si riega, publica los datos por MQTT y se duerme 30 minutos para ahorrar batería. Cópialo como main.py en tu ESP32. Dentro del código solo se usa ASCII, así que los comentarios van sin acentos.

```
import machine
import time
import network
import dht
from umqtt.simple import MQTTClient

# --- Configuración: ajusta estos valores a tu red ---
WIFI_SSID = "MiCasa_WiFi"
WIFI_PASS = "mi_clave_segura"
MQTT_BROKER = "192.168.1.50"
CLIENT_ID = "invernadero_01"

PIN_DHT = 15
PIN_SUELO = 34      # ADC del sensor capacitivo de humedad
PIN_RELE = 26       # rele de la bomba (activo en LOW)
PIN_LED = 2

# --- Umbrales: salen de TU calibración ---
SUELO_SECO = 2800   # valor ADC en tierra seca
SUELO_HUMEDO = 1500 # valor ADC en tierra bien regada
TEMP_MAX = 35.0

TOPICO_DATOS = b"casa/invernadero/datos"

def conectar_wifi():
    wlan = network.WLAN(network.STA_IF)
    wlan.active(True)
    if not wlan.isconnected():
        wlan.connect(WIFI_SSID, WIFI_PASS)
        for _ in range(20):
            if wlan.isconnected():
                break
        time.sleep(1)
    return wlan.isconnected()
```

```

def leer_sensores():
    sensor = dht.DHT22(machine.Pin(PIN_DHT))
    adc = machine.ADC(machine.Pin(PIN_SUELO))
    adc.atten(machine.ADC.ATTN_11DB)
    time.sleep(2)
    sensor.measure()
    temp = sensor.temperature()
    hum_aire = sensor.humidity()
    crudo = adc.read()
    pct = (SUELO_SECO - crudo) * 100 // (SUELO_SECO - SUELO_HUMEDO)
    pct = max(0, min(100, pct))
    return temp, hum_aire, pct

def regar(segundos=20):
    rele = machine.Pin(PIN_RELE, machine.Pin.OUT, value=1)
    print("Regando", segundos, "segundos...")
    rele.value(0)
    time.sleep(segundos)
    rele.value(1)
    print("Riego terminado")

def main():
    led = machine.Pin(PIN_LED, machine.Pin.OUT)
    led.value(1)
    if not conectar_wifi():
        print("Sin Wi-Fi: dormir 10 min y reintentar")
        machine.deepsleep(600000)
    try:
        temp, hum_aire, suelo = leer_sensores()
        print("Temp: %.1f C  Humedad aire: %.1f  Suelo: %d por ciento" % (temp,
hum_aire, suelo))
        rego = False
        if suelo < 30:
            regar(20)
            rego = True
        cliente = MQTTClient(CLIENT_ID, MQTT_BROKER)
        cliente.connect()
        mensaje = '{"temperatura": %.1f, "humedad_aire": %.1f, "humedad_suelo": %d,
"riego": "%s"}'
        mensaje = mensaje % (temp, hum_aire, suelo, "si" if rego else "no")
        cliente.publish(TOPICO_DATOS, mensaje)
        print("Publicado:", mensaje)
        cliente.disconnect()
    except Exception as e:
        print("Error:", e)

```

```
led.value(0)
machine.deepsleep(30 * 60 * 1000)

main()
```

Fíjate en tres decisiones de diseño: el deep sleep al final (la batería dura semanas), la reconexión de Wi-Fi con límite de intentos (si no hay red, duerme y reintenta en vez de trabarse), y el bloque try/except que evita que un error de lectura mate el programa. Esos tres detalles son la diferencia entre un prototipo y un sistema.

Paso 7: ver los datos y la alerta. Para comprobar que publica, desde tu computadora suscríbete al tópico con este comando (cambia la IP por la de tu broker):

```
mosquitto_sub -h 192.168.1.50 -t "casa/invernadero/datos" -v
```

Vas a ver llegar un mensaje JSON cada 30 minutos. En tu celular, una app como MQTT Dash puede mostrar esos valores en un tablero. Y esta automatización en YAML (para Home Assistant) te manda un Telegram si el invernadero se calienta de más:

```
automation:
- alias: "Avisar si el invernadero esta muy caliente"
  trigger:
    - platform: mqtt
      topic: "casa/invernadero/datos"
  condition:
    - condition: template
      value_template: "{{ trigger.payload_json.temperatura > 35 }}"
  action:
    - service: notify.telegram_casa
      data:
        message: "El invernadero llego a {{ trigger.payload_json.temperatura }}
grados. Revisa la ventilacion."
```

Paso 8: archivo de configuración. Guarda los parámetros en un config.json junto a tu documentación, para que cualquiera (incluido tu yo del futuro) sepa cómo está ajustado el sistema:


```
{
  "proyecto": "invernadero_inteligente",
  "version": "1.0",
  "umbrales": {
    "humedad_suelo_minima_pct": 30,
    "temperatura_maxima_c": 35.0,
    "segundos_riego": 20
  },
  "mqtt": {
    "broker": "192.168.1.50",
    "topico_datos": "casa/invernadero/datos"
  },
  "energia": {
    "fuente": "panel_solar_10W",
    "modo": "deep_sleep_30min"
  }
}
```

Paso 9: pruebas y primera iteración. Déjalo funcionando una semana en la mesa antes de instalarlo afuera. Revisa cada día: ¿los riegos coinciden con tierra realmente seca? ¿Algún día no publicó (falla de Wi-Fi o de energía)? Ajusta SUELO_SECO y SUELO_HUMEDO con tus observaciones. Cuando pase una semana estable, mételo en la caja estanca, elévala del suelo y pon el panel solar donde le dé el sol directo. Tu criterio de éxito se cumple a los 30 días: documenta el resultado con una gráfica de humedad del mes. Esa gráfica es oro para tu portafolio.

PROYECTO 2: Estación meteorológica conectada (sensores + nube + dashboard + alertas)

Qué vas a construir: tu propia estación del clima en el techo o el jardín. Una Raspberry Pi mide temperatura, humedad y presión atmosférica cada 10 minutos, cuenta la lluvia con un pluviómetro casero, sube todo a la nube, lo muestra en un tablero web que puedes ver desde donde sea, y te manda una alerta a Telegram si detecta riesgo de helada o una caída brusca de presión (señal de tormenta). Tecnologías que combina: sensores ambientales, nube para guardar datos, dashboard web y sistema de alertas.

Paso 1: alcance del MVP. Esta versión SÍ incluye: medir temperatura, humedad, presión y lluvia; subir datos a la nube cada 10 minutos; tablero web con gráficas de 24 horas y 7 días; alerta de helada (temperatura bajo 4 grados) y alerta de tormenta (caída de presión mayor a 5 hPa en 3 horas). Esta versión NO incluye (todavía): anemómetro (velocidad del viento), sensor de rayos UV, predicción con machine learning. Criterio de éxito: 30 días de datos continuos con menos del 5 por ciento de huecos, y al menos una alerta real comprobada.

Paso 2: materiales y costo. Raspberry Pi 4 o 3B+ (\$1,200, o usa la que ya tengas), sensor BME280 de temperatura/humedad/presión (\$150), pluviómetro de balancín con reed switch (\$250) o uno casero con un embudo y un sensor de pulsos, caja estanca IP65 (\$200), mástil o soporte para el techo (\$150), eliminador de 5V 3A (\$180), cables y tornillería (\$100). Subtotal aproximado: \$2,230 más el colchón del 20 por ciento: \$2,676 de presupuesto. Nota: aquí no usamos panel solar porque la Raspberry Pi consume más; va conectada a la luz, con un UPS pequeño si quieres que sobreviva a los apagones.

Paso 3: arquitectura en las seis capas. Física: BME280 por I2C y pluviómetro por GPIO con interrupciones. Control: Raspberry Pi con Python. Conectividad: Wi-Fi hacia tu broker MQTT local y hacia internet. Datos: doble destino, MQTT local para el tablero de casa y la nube (ThingSpeak o similar) para el historial y acceso remoto. Presentación: dashboard web con gráficas y notificaciones por Telegram. Energía: eliminador de 5V; la Pi no duerme, así que necesita corriente continua.

Paso 4: instalación del software. En la Raspberry Pi, actualiza e instala las librerías. Activa I2C con raspi-config antes de conectar el BME280:

```
sudo apt update && sudo apt install -y python3-pip
pip3 install smbus2 bme280 paho-mqtt requests
sudo raspi-config
```

En raspi-config entra a "Interface Options", activa I2C y reinicia. Conecta el BME280: VCC a 3V3, GND a GND, SCL al pin 5, SDA al pin 3. El pluviómetro (reed switch) va entre el pin 12 (GPIO 18) y GND, usando la resistencia pull-up interna.

Paso 5: el código. Este script en Python hace el ciclo completo: lee el BME280, cuenta los pulsos de lluvia con interrupciones, publica por MQTT, sube a la nube y revisa las alertas. Guárdalo como estacion.py. Recuerda: dentro del código solo ASCII.

```
#!/usr/bin/env python3
# Estacion meteorologica conectada - Raspberry Pi
# Lee BME280 + pluviometro, publica MQTT, sube a la nube y alerta

import time
```

```

import smbus2
import bme280
import requests
import paho.mqtt.client as mqtt
from datetime import datetime

try:
    import RPi.GPIO as GPIO
    HAY_GPIO = True
except ImportError:
    HAY_GPIO = False

# --- Configuracion ---
MQTT_BROKER = "192.168.1.50"
TOPICO = "casa/clima/medicion"
THINGSPEAK_URL = "https://api.thingspeak.com/update"
THINGSPEAK_KEY = "TU_CLAVE_DE_THINGSPEAK"
TELEGRAM_TOKEN = "TU_TOKEN_DE_BOT"
TELEGRAM_CHAT = "TU_CHAT_ID"
PIN_LLUVIA = 18
MM_POR_PULSO = 0.3 # cada vuelco del balancin = 0.3 mm (calibrarlo)

pulsos_lluvia = 0
presion_anterior = None
hora_presion_anterior = None

def pulso_canal(canal):
    global pulsos_lluvia
    pulsos_lluvia += 1

def leer_bme280():
    bus = smbus2.SMBus(1)
    calib = bme280.load_calibration_params(bus, 0x76)
    datos = bme280.sample(bus, 0x76, calib)
    return round(datos.temperature, 1), round(datos.humidity, 1),
    round(datos.pressure, 1)

def publicar_mqtt(temp, hum, pres, lluvia):
    cliente = mqtt.Client(client_id="estacion_clima")
    cliente.connect(MQTT_BROKER, 1883, 60)
    mensaje = '{"fecha": "%s", "temperatura": %.1f, "humedad": %.1f, "presion": %.1f,
"lluvia_mm": %.1f}'
    mensaje = mensaje % (datetime.now().isoformat(timespec="minutes"), temp, hum,
pres, lluvia)
    cliente.publish(TOPICO, mensaje)

```

```

    cliente.disconnect()
    print("MQTT:", mensaje)

def subir_nube(temp, hum, pres, lluvia):
    params = {"api_key": THINGSPEAK_KEY, "field1": temp, "field2": hum,
              "field3": pres, "field4": lluvia}
    try:
        r = requests.get(THINGSPEAK_URL, params=params, timeout=15)
        print("Nube: codigo", r.status_code)
    except Exception as e:
        print("Nube: fallo, se reintentara:", e)

def avisar_telegram(texto):
    url = "https://api.telegram.org/bot%s/sendMessage" % TELEGRAM_TOKEN
    try:
        requests.post(url, data={"chat_id": TELEGRAM_CHAT, "text": texto}, timeout=15)
        print("Alerta enviada:", texto)
    except Exception as e:
        print("Telegram fallo:", e)

def revisar_alertas(temp, pres):
    global presion_anterior, hora_presion_anterior
    ahora = time.time()
    if temp < 4.0:
        avisar_telegram("Alerta de helada: la estacion marca %.1f grados. Protege tus
plantas." % temp)
    if presion_anterior is not None and hora_presion_anterior is not None:
        horas = (ahora - hora_presion_anterior) / 3600.0
        if horas >= 3:
            caida = presion_anterior - pres
            if caida > 5.0:
                avisar_telegram("Posible tormenta: la presion cayo %.1f hPa en 3
horas." % caida)
            presion_anterior, hora_presion_anterior = pres, ahora
        else:
            presion_anterior, hora_presion_anterior = pres, ahora

def main():
    global pulsos_lluvia
    if HAY_GPIO:
        GPIO.setmode(GPIO.BCM)
        GPIO.setup(PIN_LLUVIA, GPIO.IN, pull_up_down=GPIO.PUD_UP)
        GPIO.add_event_detect(PIN_LLUVIA, GPIO.FALLING, callback=pulso_canal,
bouncetime=300)
    while True:

```

```

    try:
        temp, hum, pres = leer_bme280()
        lluvia = pulsos_lluvia * MM_POR_PULSO
        pulsos_lluvia = 0
        publicar_mqtt(temp, hum, pres, lluvia)
        subir_nube(temp, hum, pres, lluvia)
        revisar_alertas(temp, pres)
    except Exception as e:
        print("Error en el ciclo:", e)
    time.sleep(600) # cada 10 minutos

if __name__ == "__main__":
    main()

```

Tres decisiones de diseño a notar: los datos van a dos destinos (MQTT local y nube) para que el sistema siga registrando aunque se caiga el internet; las alertas se calculan con tendencias (caída de presión en 3 horas), no con un solo dato, para evitar falsas alarmas; y el pluviómetro usa interrupciones por hardware para no perder ningún pulso aunque el programa esté ocupado.

Paso 6: que corra solo. Para que el script sobreviva a reinicios, créalo como servicio con systemd. Crea el archivo `/etc/systemd/system/estacion.service` con este contenido:

```

[Unit]
Description=Estacion meteorologica
After=network-online.target
Wants=network-online.target

[Service]
ExecStart=/usr/bin/python3 /home/pi/estacion.py
Restart=always
RestartSec=30
User=pi

[Install]
WantedBy=multi-user.target

```

Y actívalo con estos comandos:

```

sudo systemctl daemon-reload
sudo systemctl enable estacion
sudo systemctl start estacion
sudo systemctl status estacion

```

Paso 7: el dashboard. En ThingSpeak (gratis) cada field se grafica solo; configura cuatro gráficas: temperatura, humedad, presión y lluvia acumulada. Si prefieres tu tablero local, este JSON describe los paneles para un dashboard genérico y te sirve como documentación de qué muestra cada uno:

```
{
  "dashboard": "estacion_meteorologica",
  "fuente": "mqtt_topico_casa/clima/medicion",
  "paneles": [
    {
      "tipo": "grafica_linea", "titulo": "Temperatura ultimas 24 h", "campo":
      "temperatura", "unidad": "C"},
    {
      "tipo": "grafica_linea", "titulo": "Humedad ultimas 24 h", "campo": "humedad",
      "unidad": "%"},
    {
      "tipo": "grafica_linea", "titulo": "Presion atmosferica 7 dias", "campo":
      "presion", "unidad": "hPa"},
    {
      "tipo": "barras", "titulo": "Lluvia acumulada por dia", "campo": "lluvia_mm",
      "unidad": "mm"},
    {
      "tipo": "valor_actual", "titulo": "Ahora mismo", "campos": ["temperatura",
      "humedad", "presion"]}
  ]
}
```

Paso 8: calibración y pruebas. El pluviómetro casero se calibra echando con una jeringa una cantidad conocida de agua (por ejemplo 50 ml) y contando los pulsos; divides milímetros entre pulsos y ajustas MM_POR_PULSO. Compara tu temperatura con la de una estación oficial cercana durante una semana: si siempre marcas 2 grados de más, tu caja está recibiendo sol directo y necesita sombra ventilada. A los 30 días revisa los huecos de datos: si hay más del 5 por ciento, tu Wi-Fi del techo es débil y necesitas un repetidor o una antena mejor. Primera iteración natural: agregar el anemómetro que dejaste fuera del MVP.

PROYECTO 3: Asistente de voz casero (IA + IoT + hogar inteligente)

Qué vas a construir: un asistente al que le hablas en español y te obedece. Con una Raspberry Pi, un micrófono USB y una bocina, dices "prende la luz de la sala" y la luz se prende; dices "qué clima hace" y te responde con los datos de tu propia estación meteorológica; dices "activa modo cine" y baja las luces y te dice qué película te recomienda. Tecnologías que combina: inteligencia artificial (reconocimiento de voz y síntesis de habla), IoT (MQTT para controlar dispositivos) y hogar inteligente (rutinas y automatizaciones).

Paso 1: alcance del MVP. Esta versión SÍ incluye: reconocer 10 comandos en español, controlar 3 dispositivos por MQTT (dos luces y un enchufe), responder con voz el clima actual

de tu estación, y una rutina de "buenas noches" que apaga todo. Esta versión NO incluye (todavía): conversación libre, reconocer diferentes voces, funcionar sin internet (el reconocimiento usa la nube). Criterio de éxito: 9 de cada 10 comandos entendidos correctamente durante una semana de uso diario.

Paso 2: materiales y costo. Raspberry Pi 3B+ o 4 (\$1,200, reutiliza la de la estación si quieres, pero mejor dedicada), micrófono USB (\$250), bocina pequeña USB o con cable (\$200), 2 focos inteligentes o 2 módulos relé Wi-Fi (\$400), enchufe inteligente (\$250). Subtotal: \$2,300 más el colchón del 20 por ciento: \$2,760. Si ya tienes bocina y micrófono, el proyecto baja a menos de \$1,500.

Paso 3: arquitectura en las seis capas. Física: micrófono y bocina. Control: Raspberry Pi con Python. Conectividad: Wi-Fi y MQTT hacia los dispositivos. Datos: los "intents" (intenciones) definidos en un JSON local. Presentación: la voz misma es la interfaz, más un registro de comandos. Energía: eliminador de 5V. La IA vive en dos lugares: el reconocimiento de voz (nube de Google, gratis para este uso) y la interpretación de comandos (reglas locales en tu JSON de intents).

Paso 4: instalación. El reconocimiento de voz necesita algunas dependencias del sistema:

```
sudo apt update && sudo apt install -y portaudio19-dev flac espeak  
pip3 install SpeechRecognition pytttsx3 paho-mqtt
```

Prueba tu micrófono antes de programar nada: graba 5 segundos y escúchalos. Si el micrófono no se oye bien, ningún código lo va a arreglar.

Paso 5: los intents. Este JSON es el cerebro del asistente: cada intención tiene ejemplos de lo que el usuario puede decir y la acción MQTT correspondiente. Guárdalo como intents.json:

```
{
  "idioma": "es-MX",
  "intents": [
    {
      "nombre": "prender_luz_sala",
      "ejemplos": ["prende la luz de la sala", "enciende la luz"],
      "accion": {"topico": "casa/sala/luz", "mensaje": "ON"},
      "respuesta": "Listo, luz de la sala encendida."
    },
    {
      "nombre": "apagar_luz_sala",
      "ejemplos": ["apaga la luz de la sala", "apaga la luz"],
      "accion": {"topico": "casa/sala/luz", "mensaje": "OFF"},
      "respuesta": "Luz de la sala apagada."
    },
    {
      "nombre": "preguntar_clima",
      "ejemplos": ["que clima hace", "como esta el clima", "hace frio"],
      "accion": {"topico": null, "mensaje": null},
      "respuesta": "clima_actual"
    },
    {
      "nombre": "buenas_noches",
      "ejemplos": ["buenas noches", "me voy a dormir", "apaga todo"],
      "accion": {"topico": "casa/modo", "mensaje": "noche"},
      "respuesta": "Buenas noches. Apago las luces y activo el modo noche."
    }
  ]
}
```

Paso 6: el código. Este script escucha, convierte tu voz a texto, busca qué intent coincide, ejecuta la acción por MQTT y te responde hablando. Guárdalo como asistente.py, en ASCII:

```
#!/usr/bin/env python3
# Asistente de voz casero - Raspberry Pi
# Escucha en español, interpreta intents y controla la casa por MQTT

import json
import time
import unicodedata
import speech_recognition as sr
import pyttsx3
import paho.mqtt.client as mqtt

MQTT_BROKER = "192.168.1.50"
```



```

TOPICO_CLIMA = "casa/clima/medicion"

with open("intents.json", "r", encoding="utf-8") as f:
    INTENTS = json.load(f)["intents"]

ultimo_clima = "aun no tengo datos del clima"

motor_voz = pyttsx3.init()
motor_voz.setProperty("rate", 160)

def hablar(texto):
    print("Asistente:", texto)
    motor_voz.say(texto)
    motor_voz.runAndWait()

def escuchar():
    rec = sr.Recognizer()
    with sr.Microphone() as fuente:
        rec.adjust_for_ambient_noise(fuente, duration=1)
        print("Te escucho...")
        audio = rec.listen(fuente, timeout=5, phrase_time_limit=6)
    try:
        texto = rec.recognize_google(audio, language="es-MX")
        print("Escuche:", texto)
        return texto.lower()
    except sr.UnknownValueError:
        return ""
    except sr.RequestError as e:
        print("Error de red:", e)
        return ""

def normalizar(t):
    nfkd = unicodedata.normalize("NFD", t)
    return "".join(c for c in nfkd if not unicodedata.combining(c))

def buscar_intent(texto):
    for intent in INTENTS:
        for ejemplo in intent["ejemplos"]:
            if ejemplo in texto:
                return intent
    return None

def ejecutar(intent):
    accion = intent["accion"]
    if accion["topico"]:

```

```

        cliente = mqtt.Client(client_id="asistente_voz")
        cliente.connect(MQTT_BROKER, 1883, 60)
        cliente.publish(accion["topico"], accion["mensaje"])
        cliente.disconnect()
    respuesta = intent["respuesta"]
    if respuesta == "clima_actual":
        hablar("Segun mi estacion: " + ultimo_clima)
    else:
        hablar(respuesta)

def on_clima(cliente, datos, msg):
    global ultimo_clima
    try:
        j = json.loads(msg.payload.decode("utf-8"))
        ultimo_clima = "%.1f grados y %d por ciento de humedad" % (j["temperatura"],
j["humedad"])
    except Exception:
        pass

def main():
    cliente = mqtt.Client(client_id="asistente_oyente")
    cliente.on_message = on_clima
    cliente.connect(MQTT_BROKER, 1883, 60)
    cliente.subscribe(TOPICO_CLIMA)
    cliente.loop_start()
    hablar("Hola. Ya estoy listo. Dime que necesitas.")
    while True:
        texto = normalizar(escuchar())
        if not texto:
            continue
        if "adios" in texto or "callate" in texto:
            hablar("Hasta luego.")
            break
        intent = buscar_intent(texto)
        if intent:
            ejecutar(intent)
        else:
            hablar("No entendí. Puedes pedirme prender la luz, el clima o buenas
noches.")
            time.sleep(1)

if __name__ == "__main__":
    main()

```

Fíjate en cómo se conectan los proyectos: el asistente se suscribe al tópico de tu estación meteorológica (casa/clima/medicion) y así responde "qué clima hace" con TUS datos, no con los de internet. Esa es la magia de la fusión: cada proyecto hace más valiosos a los demás.

Paso 7: la rutina de "buenas noches" en YAML. Cuando el asistente publica "noche" en casa/modo, Home Assistant apaga las luces y baja la intensidad:

```
automation:
  - alias: "Modo noche del asistente"
    trigger:
      - platform: mqtt
        topic: "casa/modo"
        payload: "noche"
    action:
      - service: light.turn_off
        target:
          entity_id: light.sala
      - service: notify.telegram_casa
        data:
          message: "Modo noche activado por el asistente de voz."
```

Paso 8: pruebas y ajustes. La primera semana anota cada comando fallido: casi siempre es por ruido de fondo o porque dijiste el comando con otras palabras. Agrega esas variantes a los ejemplos de tu intents.json; ese archivo es tu verdadero "entrenamiento" de IA. Si el reconocimiento falla mucho, acerca el micrófono y baja el ruido (no lo pongas junto a la tele). Primera iteración natural: agregar temporizadores ("apaga la luz en 10 minutos") y una palabra de activación para no tener que tocar nada.

Práctica

En esta sección diseñas tú, con ejercicios guiados. Cada ejercicio trae su solución explicada paso a paso: intenta resolverlo primero y luego compara. Los ejercicios 1 y 2 completan los proyectos 4 y 5 de la unidad.

Ejercicio 1: Diseña el sistema de seguridad con IA (proyecto 4)

Enunciado: Diseña un sistema que vigile la entrada de tu casa: detecte personas con una cámara, te mande una foto a Telegram cuando haya movimiento, y distinga entre personas conocidas y desconocidas. Define el alcance del MVP, las seis capas, la lista de materiales con presupuesto y el criterio de éxito.

Solución paso a paso:

1. Alcance del MVP. Sí incluye: una cámara en la entrada, detección de personas con visión por computadora, foto a Telegram cuando detecte una persona, grabación de 10 segundos del evento. NO incluye (todavía): reconocimiento facial de conocidos, múltiples cámaras, sirena. Criterio de éxito: detectar al 95 por ciento de las personas que pasan en una semana, con menos de 2 falsas alarmas por día.
2. Las seis capas. Física: cámara Raspberry Pi Camera o USB. Control: Raspberry Pi 4 (la detección de personas necesita más poder que un ESP32). Conectividad: Wi-Fi y MQTT para avisar, Telegram para la foto. Datos: las fotos se guardan localmente 7 días y se borran solas. Presentación: mensaje de Telegram con foto y hora. Energía: eliminador de 5V; si la cámara va lejos del enchufe, un cable USB largo de buena calidad o PoE.
3. Materiales y presupuesto. Raspberry Pi 4 4GB (\$1,800), cámara oficial v2 o USB 1080p (\$600), caja estanca con visor (\$250), eliminador 5V 3A (\$180), microSD 64GB (\$200), soporte para pared (\$150). Subtotal: \$3,180 más 20 por ciento: \$3,816.
4. Cómo funciona la IA. Usa un modelo ligero de detección de personas (por ejemplo MobileNet SSD con OpenCV o la librería Frigate): la Pi analiza un cuadro por segundo; si el modelo dice "persona" con más del 60 por ciento de confianza, guarda la foto y la manda. Para no saturarte, después de una alerta espera 5 minutos antes de la siguiente.
5. Robustez. De noche la cámara normal no ve: agrega un reflector con sensor de movimiento (barato) o una cámara infrarroja. Si se va el internet, las fotos se guardan local y se mandan cuando regresa. Aviso de privacidad: la cámara solo apunta a TU entrada, nunca a la calle ni al vecino.
6. Primera iteración. Cuando el MVP funcione un mes, agrega reconocimiento facial de tu familia para que solo alerte por desconocidos.

Ejercicio 2: Diseña el monitor de energía del hogar (proyecto 5)

Enunciado: Diseña un aparato que mida cuánta electricidad consume tu casa en tiempo real, suba los datos a la nube, muestre el costo estimado en pesos en un tablero, y use una carcasa impresa en 3D. Define alcance, arquitectura, cómo medir sin tocar cables de alto voltaje y la carcasa.

Solución paso a paso:

1. Alcance del MVP. Sí incluye: medir la corriente total de la casa cada 10 segundos, calcular potencia y consumo acumulado del día, tablero con gráfica y costo estimado en pesos, alerta si el consumo pasa de cierto límite. NO incluye (todavía): medir por circuito,

controlar aparatos. Criterio de éxito: que su medición del consumo diario difiera menos del 5 por ciento de lo que marca tu medidor de luz en una semana.

2. Cómo medir con seguridad. NUNCA peles cables de 110V. Usa un sensor de pinza (CT clamp, como el SCT-013): se abre, se cierra alrededor de UN cable del medidor o del centro de carga, y mide por inducción sin contacto eléctrico. Es la forma segura y estándar. El sensor se conecta a un ESP32 con un pequeño circuito de acondicionamiento (resistencia de carga y divisor de voltaje) o a un módulo PZEM-004T que ya entrega los valores listos por serial.
3. Las seis capas. Física: pinza CT más ESP32. Control: ESP32 que calcula potencia cada 10 segundos. Conectividad: Wi-Fi y MQTT. Datos: la nube guarda el historial por día y mes. Presentación: tablero con consumo actual en watts, acumulado del día en kWh y costo estimado (kWh por la tarifa de tu recibo). Energía: eliminador USB; va junto al centro de carga.
4. Materiales y presupuesto. ESP32 (\$180), pinza SCT-013 de 100A (\$350), módulo PZEM-004T opcional (\$280), caja plástica (\$120), eliminador USB (\$150), impresión 3D de la carcasa a medida (\$200 si la mandas a imprimir). Subtotal: \$1,280 más 20 por ciento: \$1,536.
5. La carcasa en 3D. Mide tu caja y diseña en Tinkercad o FreeCAD: base con orificios para tornillos, tapa con ventilación, una ventana para el LED del ESP32 y una salida para el cable de la pinza. Imprímela en PETG (aguanta mejor el calor que el PLA). Esta es la fusión completa: electrónica, software, nube y fabricación digital en un solo aparato.
6. Calibración. Compara una semana contra tu medidor oficial y ajusta el factor de calibración en el código. Si tu tarifa es DAC o tiene escalones, configura los rangos en el tablero para que el costo estimado sea realista.

Ejercicio 3: El modo vacaciones (fusión de fusiones)

Enunciado: Combina tus proyectos en un "modo vacaciones": cuando sales de viaje, el sistema cuida la casa solo. Diseña qué hace cada proyecto, cómo se comunican y qué pasa si algo falla.

Solución paso a paso:

1. Define el disparador. Un tópico MQTT casa/modo con valor "vacaciones", activado desde tu celular o por el asistente de voz ("me voy de viaje"). Todo el ecosistema lo escucha.
2. Qué hace cada proyecto. El invernadero sigue regando solo y te reporta cada día por Telegram. La estación meteorológica vigila tormentas y heladas. El sistema de seguridad

sube su sensibilidad: cualquier persona detectada te manda foto inmediata. El asistente de voz prende y apaga luces a horarios aleatorios para simular que hay gente. El monitor de energía te avisa si el consumo se dispara (algo se quedó prendido) o cae a cero (se fue la luz).

3. Cómo se comunican. Todo por MQTT con tópicos claros: casa/modo para el modo, casa/alertas para avisos urgentes. Un tópico de "latido" (heartbeat): cada dispositivo publica "sigo vivo" cada hora; si uno deja de latir, te llega un aviso de "el invernadero no responde desde hace 3 horas".
4. Plan de fallas. Si se va el internet: cada sistema sigue funcionando en local y guarda los eventos para reportarlos cuando regreses la conexión. Si se va la luz: el monitor lo detecta (consumo en cero) y, si tienes un UPS pequeño en el router, te manda el aviso antes de apagarse. Deja a alguien de confianza con llave y con tu número, porque la tecnología avisa pero no abre puertas.
5. Prueba antes de viajar. Activa el modo vacaciones un fin de semana que estés en casa de un familiar y verifica que todo reporte bien. Nunca estrenes el sistema el día de tu vuelo.
6. Documenta el modo. Agrega una página a tu README: cómo se activa, qué hace cada cosa y el plan de fallas. Si algún día vendes este sistema, esa página vale dinero.

Ejercicio 4: Cotiza de verdad (caza los costos ocultos)

Enunciado: Toma el invernadero inteligente y haz su cotización real con precios de hoy en tu país: busca cada componente en dos tiendas, suma envíos, agrega herramientas que no tengas y calcula el presupuesto final con colchón. Luego identifica tres costos ocultos.

Solución paso a paso:

1. Arma tu tabla. Haz una tabla con columnas: componente, tienda A, tienda B, precio elegido, envío. Busca cada uno de los 10 componentes del proyecto 1. No uses precios de memoria: los precios cambian y la memoria miente.
2. Compara con envío incluido. Un sensor de \$90 con \$120 de envío cuesta \$210. A veces conviene comprar todo en una sola tienda aunque cada pieza salga un poco más cara, porque pagas un solo envío. Calcula el total por tienda antes de decidir.
3. Suma herramientas faltantes. ¿Tienes multímetro? ¿Cautín y soldadura? ¿Pinzas de corte? Anota lo que te falte con su precio; son inversión, porque te sirven para todos los proyectos futuros.

4. Aplica el colchón del 20 por ciento sobre el subtotal con envíos. Ese es tu presupuesto real. Si te da \$2,200 en vez de los \$1,920 del ejemplo, no pasa nada: el ejemplo era aproximado y tu cotización es la que manda.
5. Caza tres costos ocultos típicos: primero, los repuestos por aprendizaje (un ESP32 extra por si quemas el primero, \$180 bien invertidos); segundo, los consumibles (estaño, cinta aislante, bridas); tercero, el tiempo de espera de los envíos, que no cuesta dinero pero retrasa tu calendario dos o tres semanas si pides fuera del país.
6. Decide la compra por etapas con tu cotización en mano: etapa 1 (prototipo en protoboard), etapa 2 (versión robusta), etapa 3 (lujos). Comprar por etapas con precios reales es la diferencia entre un proyecto financiado y una tarjeta sobregirada.

Ejercicio 5: Plan de robustez (diseña para el peor día)

Enunciado: Toma tu estación meteorológica y escribe su plan de contingencia: qué pasa si se va la luz, si se cae el Wi-Fi, si se cae el internet, si falla un sensor y si se llena la memoria. Para cada caso: cómo lo detecta, qué hace el sistema y cómo te enteras tú.

Solución paso a paso:

1. Si se va la luz. Detección: el monitor de energía (proyecto 5) reporta consumo en cero, o un sensor simple de voltaje en la Pi. Qué hace: nada puede hacer sin energía, así que el plan es previo: un UPS pequeño (\$600) le da 30 minutos a la Pi y al router para mandar el aviso "se fue la luz a las 14:32" y apagarse ordenadamente sin corromper la memoria SD. Cómo te enteras: mensaje de Telegram enviado con esos últimos minutos de batería.
2. Si se cae el Wi-Fi. Detección: el script no puede publicar por MQTT. Qué hace: guarda cada medición en un archivo CSV local con fecha y hora, y reintenta la conexión cada 5 minutos (sin trabarse: el reintento va en un hilo o con tiempo límite). Cómo te enteras: cuando el Wi-Fi regresa, sube los datos atrasados a la nube y te manda "estuve sin Wi-Fi 2 horas, ya subí 12 mediciones pendientes".
3. Si se cae el internet pero hay Wi-Fi local. Detección: MQTT local funciona pero ThingSpeak no responde. Qué hace: sigue publicando en local (tu tablero de casa funciona) y acumula lo pendiente para la nube. Cómo te enteras: el tablero local muestra un aviso de "sin conexión a la nube desde las 09:15".
4. Si falla un sensor. Detección: lecturas imposibles (temperatura de -40 o de 85 grados, que son los valores de error típicos del BME280) o el sensor no responde por I2C. Qué hace: publica un mensaje de error en casa/clima/estado en vez de datos falsos, y reintenta leer cada ciclo. Cómo te enteras: Telegram con "el sensor de clima no responde, revisa las conexiones".

5. Si se llena la memoria. Detección: el script revisa el espacio libre cada día. Qué hace: borra automáticamente los archivos de más de 30 días (rotación de registros). Cómo te enteras: no necesitas enterarte, pero queda anotado en el registro del sistema.
6. Regla general que aprendiste: todo sistema fusión debe degradarse con gracia. "Degradarse con gracia" significa que cuando algo falla, el sistema hace lo mejor que puede con lo que le queda y te avisa, en vez de quedarse callado o romperse del todo. Escríbelo en tu README: es la marca de un profesional.

Ejercicio 6: De proyecto a negocio (vende tu invernadero)

Enunciado: Convierte el invernadero inteligente en un producto que puedas vender: define a quién se lo vendes, cuánto cobras, qué incluye, qué garantía das y cómo lo instalas. Haz los números.

Solución paso a paso:

1. Elige tu cliente. No le vendas a "todo el mundo". Un buen primer cliente: personas con jardín o huerto urbano que viajan seguido, o pequeños negocios como cafeterías con plantas. Tienen el problema (plantas que se mueren), tienen dinero para pagarlo y te recomiendan con otros.
2. Define el producto, no los componentes. El cliente no compra "un ESP32 con MQTT"; compra "tus plantas vivas cuando regreses de vacaciones". Tu paquete incluye: el sistema instalado, la app configurada en su celular, una visita de calibración a los 15 días y un manual simple de una página. Eso es lo que vale dinero.
3. Ponle precio con fórmula. Costo de materiales (\$1,920 según tu cotización) más tu tiempo de instalación y configuración (por ejemplo 6 horas) más garantía y soporte. Una fórmula sana: $\text{precio} = (\text{materiales} \times 2) + (\text{horas} \times \text{tu tarifa por hora})$. Con materiales de \$1,920 y 6 horas a \$250: $(1,920 \times 2) + (6 \times 250) = 3,840 + 1,500 = \$5,340$. Redondea a \$5,500. El "materiales x 2" cubre repuestos, traslados e imprevistos.
4. La garantía vende. Ofrece 3 meses de garantía: si algo falla, lo reparas gratis. Te protege a ti también: solo puedes ofrecerla porque tu sistema tiene reconexión automática, bitácora y plan de robustez. La garantía es la prueba de que confías en tu trabajo.
5. El proceso de venta e instalación. Visita 1 (gratis, 30 minutos): ves el lugar, tomas medidas, defines cuántas macetas y dónde va el agua. Cotización por escrito con el precio fijo. Visita 2: instalas (2 a 3 horas). Día 15: regresas a calibrar con datos reales y entregas el manual. Esa visita de calibración es tu diferenciador: ningún producto de tienda la incluye.

6. Escala con cabeza. No compres 20 kits antes de vender el primero. Vende uno, instálalo, aprende, mejora tu manual y tu proceso. Al quinto cliente ya sabes exactamente cuánto tardas y cuánto ganas por hora. Si la demanda crece, ahí sí compras componentes por volumen (más baratos) y estandarizas la caja impresa en 3D con tu logo. Empezaste regando tus plantas; terminaste con un negocio. Esa es la fusión completa: tecnología, metodología y visión.

Resumen de la unidad

- Un proyecto fusión combina dos o más tecnologías para resolver un problema real de una forma que ninguna lograría sola. No es un proyecto más grande: es un sistema donde cada tecnología hace más valiosas a las demás.
- Todo sistema fusión se descompone en seis capas: física (sensores y actuadores), control (microcontrolador o computadora), conectividad (Wi-Fi y MQTT), datos (nube o local), presentación (tablero, app, notificaciones) y energía (baterías o panel solar). Si te pierdes, pregunta qué va en cada capa.
- La metodología tiene cinco fases: idea (define el problema, no la tecnología, con un criterio de éxito medible), alcance (listas de "sí incluye" y "todavía no", con un MVP primero), presupuesto (precios reales, envíos, herramientas y 20 por ciento de colchón), prototipo (feo, por capas, sin soldar el primer día) e iteración (mide, cambia una sola cosa, vuelve a medir).
- El presupuesto se hace antes de comprar, por etapas, y cazando costos ocultos: envíos, herramientas, repuestos por aprendizaje y tiempos de espera.
- Prototipa de adentro hacia afuera: primero sensores y control, luego conectividad, después energía definitiva y al final la presentación. Cada capa se prueba sola antes de integrarse.
- Itera con datos, no con corazonadas: registra todo desde el día uno, cambia una sola cosa por ciclo y declara terminado el proyecto cuando cumple su criterio de éxito de forma estable.
- La robustez es lo que separa un juguete de un sistema: reconexión automática de Wi-Fi, watchdog timer, deep sleep para batería, respaldo local cuando no hay nube y degradación con gracia ante cualquier falla.
- Calibra tus sensores siempre: un número sin calibrar no significa nada. Tierra seca y tierra regada te dan el 0 y el 100 del sensor de humedad; una jeringa con agua conocida calibra el pluviómetro.

- Documenta mientras construyes: bitácora semanal, README con materiales, conexiones, código e instrucciones para replicarlo, y una sección honesta de problemas conocidos. Tu portafolio cuenta la historia en tres actos: problema, construcción y resultado con datos.
- Los tres proyectos contruidos paso a paso (invernadero inteligente, estación meteorológica y asistente de voz) se conectan entre sí por MQTT: el asistente responde el clima con los datos de tu estación, y el modo vacaciones los pone a todos a cuidar tu casa.
- Los proyectos 4 y 5 (seguridad con IA y monitor de energía con carcasa 3D) los diseñaste tú en la práctica, aplicando la misma metodología.
- Un proyecto se vuelve negocio cuando defines un cliente específico, vendes el resultado (no los componentes), pones precio con fórmula, das garantía y estandarizas tu proceso de instalación.
- Cerramos el curso donde empezamos: la tecnología fusión no es coleccionar aparatos, es resolver problemas reales combinando lo que sabes. Tienes la metodología, tienes cinco proyectos y tienes un portafolio que habla por ti. Lo que sigue es construir. Nos vemos en tu próximo sistema.

Tu proyecto final: el examen que sí importa

Este curso no termina con un examen de opción múltiple, porque en el mundo real nadie te pregunta definiciones: te piden resultados. Tu examen final es construir un proyecto fusión completo, de tu propia idea, aplicando todo lo de esta unidad. Aquí va el reto, con reglas claras.

El reto: elige un problema real de tu casa, tu familia o tu comunidad, y resuélvelo con un sistema que combine al menos tres tecnologías de las que viste en el curso (por ejemplo: sensores + energía solar + nube, o IA + IoT + impresión 3D). Tienes 60 días. Al final debes entregar: el sistema funcionando en su lugar definitivo, un README completo con materiales, conexiones y código, un mes de datos o de bitácora que demuestre que cumple su criterio de éxito, y una página de portafolio con fotos del proceso y del resultado.

Algunas ideas para inspirarte, por si la página en blanco te paraliza: un comedero automático para mascotas que avisa cuando se acaba la comida (IoT + automatización + app); un sistema que cuide a una persona mayor sola, detectando si no hay movimiento en horas y avisando a la familia (sensores + nube + alertas); un riego para un huerto comunitario con panel solar y reporte semanal (el invernadero llevado a escala); un detector de fugas de agua con válvula de cierre automático (sensores + automatización + notificaciones); o una lámpara solar para la banqueta de tu calle con sensor de movimiento (energía solar + automatización + fabricación). Cualquiera de estas, bien ejecutada, vale más que un diploma.

Usa esta lista de verificación antes de darlo por terminado, como un piloto antes de despegar:

- Tengo el problema escrito en dos frases y un criterio de éxito medible.
- Tengo mis listas de "sí incluye" y "todavía no", y respeté el MVP.
- Hice el presupuesto con precios reales y compré por etapas.
- Cada capa se probó por separado antes de integrarse.
- Los sensores están calibrados con valores de mi propio entorno.
- El sistema se reconecta solo al Wi-Fi y sobrevive a un corte de luz sin corromperse.
- Llevé bitácora y el README está completo, incluyendo problemas conocidos.
- El sistema cumplió su criterio de éxito durante el tiempo que definí.
- Publiqué el portafolio y el código está versionado en un repositorio.

Si marcas las nueve, no solo aprobaste el curso: tienes un sistema real, un portafolio real y una historia real que contar. Y si alguna no se cumple, ya sabes qué hacer: no es un fracaso, es tu siguiente iteración. Así trabajan los profesionales, y ahora tú eres uno de ellos.

Cierro con esto: la tecnología fusión no se trata de tener los componentes más caros ni el código más elegante. Se trata de mirar un problema cotidiano -plantas que se mueren, luz que se desperdicia, una casa sola en vacaciones- y decir "yo puedo resolver eso". Esa frase, dicha en serio y respaldada con método, es lo que convierte a alguien que "sabe de tecnología" en alguien que construye el futuro con sus manos. Gracias por llegar hasta aquí. Ahora ve y construye algo que te haga sentir orgulloso.

EXAMEN FINAL

Este es tu examen final del libro "Curso Completo de Tecnología Fusión" de Lucas EGA Academy (lucasegaacademy.com). Si llegaste hasta aquí es porque ya recorriste las diez unidades del curso, desde el panorama de tecnologías emergentes hasta los proyectos fusión donde todo se une. Este examen mide lo que aprendiste, no lo que memorizaste: muchas preguntas piden que apliques los conceptos a situaciones reales.

INSTRUCCIONES: Aparta un bloque de 90 a 120 minutos sin interrupciones y resuelve el examen de corrido. Las 50 preguntas valen dos puntos cada una, para un total de 100 puntos. Califica así: 90 a 100 puntos, dominio total, felicidades; 75 a 89 puntos, buen dominio, repasa las unidades donde fallaste; 60 a 74 puntos, dominio básico, vuelve a leer las unidades débiles; menos de 60 puntos, repite el libro completo. Para las preguntas abiertas (36 a 50) compara tu respuesta con la respuesta modelo de la sección de soluciones: no se espera que coincidan palabra por palabra, sino que incluyan las ideas clave que cada respuesta modelo explica. Anota tus respuestas en una hoja aparte y solo después consulta la sección "## Respuestas y soluciones".

Preguntas de opción múltiple

PREGUNTA 1 (opción múltiple): En el contexto de la Unidad 1, cuando decimos que dos tecnologías "convergen" nos referimos a que:

a) Compiten entre si por el mismo mercado hasta que una desaparece b) Se combinan y potencian mutuamente para crear soluciones que ninguna lograría sola c) Se usan una después de la otra en un proceso secuencial d) Son reemplazadas por una tercera tecnología mas barata

PREGUNTA 2 (opción múltiple): Cual de los siguientes enunciados describe mejor la diferencia entre una IA de propósito específico y una IA de propósito general:

a) La IA específica es mas cara que la IA general b) La IA específica resuelve una sola tarea con alto rendimiento, mientras la IA general pretende manejar muchas tareas distintas con flexibilidad c) La IA general ya existe y la específica es solo un proyecto de investigación d) No hay diferencia, son sinonimos

PREGUNTA 3 (opción múltiple): En un sistema de IoT, el componente que captura una magnitud física del entorno y la convierte en una senal digital se llama:

a) Actuador b) Gateway c) Sensor d) Broker

PREGUNTA 4 (opción múltiple): Una empresa de videovigilancia necesita que sus cámaras detecten intrusos en tiempo real sin depender de internet. La mejor decisión arquitectónica es:

a) Enviar todos los videos a la nube y procesarlos allá b) Procesar las imágenes en el edge, cerca de las cámaras, y enviar a la nube solo las alertas c) Guardar los videos en discos duros locales sin ningún procesamiento d) Contratar personal humano para vigilar los monitores todo el día

PREGUNTA 5 (opción múltiple): El principio de "cero confianza" (zero trust) en ciberseguridad establece que:

a) No se debe confiar en ningún software de ningún proveedor b) Ningún usuario o dispositivo se considera confiable por defecto, incluso dentro de la red, y cada acceso debe verificarse c) Las contraseñas ya no son necesarias d) Solo se confía en el administrador de la red

PREGUNTA 6 (opción múltiple): En impresión 3D, la técnica FDM (modelado por deposición fundida) consiste en:

a) Curar resina líquida con luz ultravioleta b) Fundir un filamento de plástico y depositarlo capa por capa c) Sinterizar polvo metálico con laser d) Cortar bloques de material con un torno

PREGUNTA 7 (opción múltiple): Un panel solar fotovoltaico genera electricidad porque:

a) El calor del sol calienta un líquido que mueve una turbina b) Los fotones de la luz solar liberan electrones en un material semiconductor, generando corriente eléctrica c) Las baterías internas convierten energía química en eléctrica d) El viento mueve las celdas del panel

PREGUNTA 8 (opción múltiple): En un hogar inteligente, el protocolo que permite que dispositivos de distintas marcas se comuniquen entre sí sin depender de un único fabricante se conoce como:

a) Un protocolo propietario cerrado b) Un estándar de interoperabilidad, como Zigbee, Z-Wave o Matter c) Un cable ethernet directo d) Una contraseña compartida

PREGUNTA 9 (opción múltiple): La principal ventaja de un dron cuadricóptero frente a un robot terrestre con ruedas es:

a) Que consume menos energía siempre b) Que puede moverse en tres dimensiones y llegar a lugares elevados o inaccesibles c) Que no necesita ningún sensor para operar d) Que nunca se estrella

PREGUNTA 10 (opción múltiple): Un "proyecto fusión", según la filosofía de este curso, es aquel que:

a) Usa una sola tecnología muy avanzada b) Integra dos o mas tecnologías de las unidades del curso para resolver un problema real c) Copia un proyecto que ya existe en internet d) Solo usa componentes gratuitos

PREGUNTA 11 (opción múltiple): Según la Unidad 1, cual es la principal razón por la que una tecnología "emergente" es difícil de evaluar:

a) Porque siempre es muy cara b) Porque su potencial real aun no se conoce del todo y su impacto depende de como converja con otras tecnologías c) Porque solo la entienden los científicos d) Porque nadie la quiere usar

PREGUNTA 12 (opción múltiple): En aprendizaje automático, el "sobreajuste" (overfitting) ocurre cuando:

a) El modelo es demasiado simple para los datos b) El modelo memoriza los datos de entrenamiento y falla con datos nuevos c) La computadora se calienta demasiado d) Hay demasiados datos de entrenamiento

PREGUNTA 13 (opción múltiple): En IoT, el protocolo MQTT es muy usado porque:

a) Es el mas rápido para transmitir video en alta definición b) Es ligero, eficiente en ancho de banda y funciona bien con dispositivos de pocos recursos c) No necesita ningún servidor intermediario d) Solo funciona con dispositivos caros

PREGUNTA 14 (opción múltiple): La principal diferencia entre nube publica y edge computing es:

a) La nube es gratis y el edge es de paga b) La nube centraliza el procesamiento en centros de datos lejanos, mientras el edge procesa los datos cerca de donde se generan c) El edge no usa internet nunca d) No hay diferencia, son lo mismo

PREGUNTA 15 (opción múltiple): Un ataque de "phishing" consiste en:

a) Dañar físicamente un servidor con un martillo b) Engañar a una persona para que revele datos confidenciales haciéndose pasar por una entidad de confianza c) Acelerar la conexión a internet d) Instalar antivirus en los equipos

PREGUNTA 16 (opción múltiple): Para imprimir una pieza que debe soportar altas temperaturas y esfuerzo mecánico, el material mas adecuado entre filamentos comunes seria:

a) PLA, porque es el mas barato b) ABS o PETG, por su mayor resistencia térmica y mecánica c) Papel d) Madera

PREGUNTA 17 (opción múltiple): Una turbina eólica genera electricidad porque:

a) El viento mueve las aspas, que giran un generador que convierte energía mecánica en eléctrica b) Las aspas atraen los rayos c) El movimiento calienta un cable d) El viento carga las baterías directamente

PREGUNTA 18 (opción múltiple): En domótica, un "escenario" o "rutina" es:

a) Una obra de teatro sobre tecnología b) Un conjunto de acciones automáticas encadenadas que se ejecutan cuando ocurre un evento (por ejemplo, al anochecer se encienden luces y se cierran cortinas) c) Un tipo de cable eléctrico d) Un sensor de movimiento

PREGUNTA 19 (opción múltiple): En robótica, el "efector final" es:

a) El último robot de una línea de producción b) La parte del robot que interactúa con el entorno para realizar una tarea, como una pinza o un soldador c) El botón de apagado d) La batería del robot

PREGUNTA 20 (opción múltiple): Si quieres construir un proyecto fusión de invernadero automatizado, la combinación mínima coherente de tecnologías sería:

a) Impresión 3D, ciberseguridad, nube, energías renovables e IoT b) Solo una app de celular c) Solo impresión 3D d) Solo energías renovables

Preguntas de verdadero o falso

PREGUNTA 21 (verdadero o falso): La convergencia tecnológica significa que las tecnologías dejan de evolucionar por separado y se combinan para crear soluciones nuevas.

PREGUNTA 22 (verdadero o falso): Un modelo de IA entrenado solo con datos de un país funcionará igual de bien en cualquier otro país sin ajustes.

PREGUNTA 23 (verdadero o falso): Un actuador es un dispositivo que ejecuta una acción física, como abrir una válvula o encender un motor, a partir de una señal de control.

PREGUNTA 24 (verdadero o falso): El edge computing siempre elimina por completo la necesidad de usar la nube.

PREGUNTA 25 (verdadero o falso): Usar la misma contraseña en todos tus servicios es una práctica segura si la contraseña es larga.

PREGUNTA 26 (verdadero o falso): En impresión 3D es necesario orientar bien la pieza y, en muchos casos, agregar soportes para las partes que quedan "en el aire".

PREGUNTA 27 (verdadero o falso): La energía solar y la eólica son intermitentes, por lo que un sistema renovable confiable necesita almacenamiento o respaldo.

PREGUNTA 28 (verdadero o falso): En un hogar inteligente, todos los dispositivos deben ser de la misma marca para funcionar.

PREGUNTA 29 (verdadero o falso): Un dron necesita sensores como giroscopio y acelerómetro para mantenerse estable en el aire.

PREGUNTA 30 (verdadero o falso): Un proyecto fusión bien planteado empieza definiendo el problema a resolver y después elige las tecnologías, no al revés.

PREGUNTA 31 (verdadero o falso): Las tecnologías emergentes siempre reemplazan por completo a las tecnologías anteriores en menos de un año.

PREGUNTA 32 (verdadero o falso): El aprendizaje supervisado usa datos etiquetados para entrenar al modelo.

PREGUNTA 33 (verdadero o falso): Un gateway en IoT puede traducir protocolos y filtrar datos antes de enviarlos a la nube.

PREGUNTA 34 (verdadero o falso): El ransomware es un tipo de malware que cifra los archivos de la víctima y pide un rescate para liberarlos.

PREGUNTA 35 (verdadero o falso): Una impresora 3D de resina (SLA) suele lograr más detalle que una FDM, pero sus materiales son más delicados de manejar.

Preguntas abiertas

PREGUNTA 36 (abierta): Explica con tus propias palabras que es la convergencia tecnológica y da dos ejemplos de productos o servicios actuales donde se note claramente que dos o más tecnologías emergentes se potencian entre sí.

PREGUNTA 37 (abierta): Una clínica rural quiere usar IA para apoyar el diagnóstico de radiografías, pero tiene pocos datos etiquetados y conexión a internet intermitente. Propón una estrategia realista que combine aprendizaje automático con despliegue en edge, explicando las decisiones técnicas y sus riesgos.

PREGUNTA 38 (abierta): Diseña un sistema IoT completo para monitorear la calidad del aire en una escuela: que sensores usarías, como se comunicarían, donde procesarías los datos y como presentarías la información a los maestros. Justifica cada decisión.

PREGUNTA 39 (abierta): Una fábrica quiere migrar su sistema de control de inventario a la nube, pero su proceso de producción no puede detenerse ni un minuto. Explica como combinarías nube y edge computing para lograr la migración sin interrumpir la operación, y que riesgos mitigarías con esa arquitectura.

PREGUNTA 40 (abierta): Imagina que eres responsable de ciberseguridad de una pequeña empresa con 20 empleados. Describe un plan de seguridad básico pero completo que incluya autenticación, respaldos, capacitación del personal y respuesta ante incidentes, explicando por que cada medida importa.

PREGUNTA 41 (abierta): Necesitas imprimir en 3D una prótesis sencilla de mano para un niño. Explica paso a paso el proceso completo: desde la elección de la tecnología y el material, pasando por el diseño y los ajustes de impresión, hasta las pruebas de seguridad antes de usarla.

PREGUNTA 42 (abierta): Una comunidad rural sin acceso a la red eléctrica quiere tener luz y cargar celulares usando energías renovables. Diseña una solución con paneles solares y baterías: calcula los componentes básicos, explica como dimensionarías el sistema y que harías en días nublados.

PREGUNTA 43 (abierta): Diseña un hogar inteligente accesible para una persona mayor que vive sola: que dispositivos instalarías, que rutinas automáticas programarías y que medidas de seguridad y privacidad tomarías para que la tecnología la ayude sin complicarle la vida.

PREGUNTA 44 (abierta): Explica como funciona un dron cuadricoptero para mantenerse estable en el aire y propón un uso práctico y responsable de drones en tu ciudad, incluyendo los riesgos y las regulaciones que deberías considerar.

PREGUNTA 45 (abierta): Propón un proyecto fusión completo que integre al menos cuatro tecnologías de este curso para resolver un problema real de tu comunidad. Describe el problema, las tecnologías elegidas, como se conectan entre si y como medirías si el proyecto fue exitoso.

PREGUNTA 46 (abierta): Analiza los riesgos éticos y sociales de desplegar cámaras con reconocimiento facial en espacios públicos de tu ciudad. Presenta argumentos a favor y en contra, y concluye con tu postura fundamentada.

PREGUNTA 47 (abierta): Compara la nube publica, la nube privada y el edge computing en términos de costo, latencia, control de los datos y escalabilidad. Para cada uno da un ejemplo de caso de uso donde sea la mejor opción.

PREGUNTA 48 (abierta): Explica por que la seguridad de un sistema IoT no puede depender solo de una buena contraseña del wifi. Describe al menos cinco capas o medidas de seguridad que aplicarías a una red de sensores domésticos y por que cada una es necesaria.

PREGUNTA 49 (abierta): Quieres fabricar con impresión 3D un repuesto de plástico para una licuadora que ya no se vende. Describe el flujo de trabajo completo: como obtendrias el modelo 3D, que ajustes harías, como probarias la pieza y que limitaciones del material deberías aceptar.

PREGUNTA 50 (abierta): Imagina que es el año 2035 y la convergencia de las diez tecnologías de este curso ya es parte de la vida diaria. Escribe un análisis de como seria un día normal tuyo en ese futuro, mencionando al menos seis de las tecnologías y explicando como interactuan entre si.

Respuestas y soluciones

RESPUESTA 1: b - La convergencia tecnológica es justamente la combinación de tecnologías que se potencian entre si para crear soluciones nuevas que ninguna lograría por separado, como cuando la IA se une al IoT y a la nube en un hogar inteligente. La opción a describe competencia, no convergencia; la c describe un proceso secuencial simple, y la d describe sustitución. Esta pregunta evalúa el concepto central de la Unidad 1, que es la base de todo el curso: entender que el valor real de las tecnologías emergentes aparece cuando se fusionan. Además, quien entiende la convergencia puede anticipar oportunidades, porque los productos mas disruptivos de la próxima década naceran justo en la interseccion de dos o mas de estas tecnologías.

RESPUESTA 2: b - Una IA de propósito específico, también llamada IA débil o estrecha, esta diseñada para rendir al máximo en una sola tarea, como reconocer rostros o traducir textos, mientras que una IA de propósito general buscaría la flexibilidad de una mente humana para muchas tareas distintas. La opción a es falsa porque el costo depende del caso; la c invierte la realidad, ya que las IA específicas son las que existen y funcionan hoy, mientras la IA general sigue siendo un objetivo de investigación. Esta pregunta evalúa la Unidad 2 y verifica que el lector no caiga en el mito de que toda IA es

igual ni en la confusión entre lo que ya existe y lo que aun es ciencia ficción. En la practica esto significa que antes de comprar una solución de IA debes preguntar para que tarea específica fue entrenada y con que datos, en lugar de suponer que sirve para todo.

RESPUESTA 3: c - El sensor es el componente que mide una magnitud física del entorno, como temperatura, humedad, luz o movimiento, y la convierte en una senal digital que el sistema puede procesar. El actuador hace lo contrario: recibe una orden digital y ejecuta una acción física; el gateway es el intermediario que conecta los dispositivos con la red; y el broker es el servidor que distribuye mensajes en protocolos como MQTT. Esta pregunta evalúa el vocabulario básico de la Unidad 3 y es clave porque confundir sensores con actuadores es uno de los errores mas comunes al diseñar un sistema IoT. En tu kit de inicio conviene etiquetar físicamente cada módulo como sensor o actuador, porque esa claridad evita errores de cableado que después son difíciles de diagnosticar.

RESPUESTA 4: b - Procesar las imagenes en el edge, junto a las cámaras, permite detectar intrusos en tiempo real sin depender de la conexión a internet ni saturar el ancho de banda con video constante, y solo se envían a la nube las alertas importantes. Enviar todo a la nube fallaría justo cuando se corte internet, que es cuando mas se necesita la vigilancia; guardar sin procesar no detecta nada; y depender de personas es caro y poco confiable. Esta pregunta evalúa la Unidad 4 y mide la capacidad de elegir entre nube y edge según los requisitos de latencia y autonomía, que es la decisión arquitectonica mas importante de esa unidad. Este mismo patrón se repite en fabricas, hospitales y ciudades inteligentes: lo critico se procesa en el edge y lo historico se analiza en la nube.

RESPUESTA 5: b - El modelo de cero confianza parte de que ningún usuario ni dispositivo es confiable por el simple hecho de estar dentro de la red corporativa, por lo que cada acceso a cada recurso debe autenticarse y autorizarse de forma continua. La opción a es una exageración impractica, la c es falsa porque las contraseñas y otros factores de autenticación siguen siendo necesarios, y la d describe el modelo antiguo de confianza implicita que zero trust precisamente quiere eliminar. Esta pregunta evalúa la Unidad 5 y verifica la comprensión del cambio de paradigma mas importante de la ciberseguridad moderna. Para aplicarlo en la vida real empieza por lo mas sencillo: activa el doble factor de autenticación en tus cuentas mas importantes y desconfía de cualquier acceso que no hayas solicitado tu.

RESPUESTA 6: b - La técnica FDM funde un filamento de plástico, como PLA o ABS, y lo deposita capa por capa siguiendo el diseño del modelo 3D hasta formar la pieza. La

opción a describe la tecnología SLA de resina, la c describe el sinterizado selectivo por laser de metales, y la d describe un proceso sustractivo de mecanizado, que es lo opuesto a la impresión 3D. Esta pregunta evalúa la Unidad 6 y comprueba que el lector distingue las principales tecnologías de impresión 3D, algo indispensable antes de elegir máquina y material para un proyecto. Si vas a comprar tu primera impresora, que sea FDM: es la mas barata, la mas fácil de mantener y tiene la comunidad mas grande para resolver dudas.

RESPUESTA 7: b - El efecto fotovoltaico ocurre cuando los fotones de la luz solar golpean un material semiconductor, generalmente silicio, y liberan electrones que fluyen generando corriente eléctrica directa. La opción a describe la energía termosolar de concentracion, que es otra tecnología distinta; la c confunde el panel con una batería; y la d mezcla conceptos de la energía eólica. Esta pregunta evalúa la Unidad 7 y verifica el principio físico fundamental de los paneles solares, sin el cual no se puede dimensionar ni entender un sistema fotovoltaico. Con este principio claro ya puedes entender las fichas técnicas de los paneles, donde la eficiencia indica que porcentaje de la luz solar se convierte en electricidad util.

RESPUESTA 8: b - Los estándares de interoperabilidad como Zigbee, Z-Wave y Matter definen un lenguaje común para que focos, sensores y cerraduras de distintas marcas trabajen juntos sin quedar atrapados con un solo fabricante. Un protocolo propietario cerrado hace exactamente lo contrario, te encierra con una marca; el cable ethernet es solo un medio físico; y la contraseña compartida no tiene nada que ver con interoperabilidad. Esta pregunta evalúa la Unidad 8 y mide si el lector entiende que la clave de un hogar inteligente escalable es evitar el encierro con un solo proveedor. Antes de comprar dispositivos para tu casa revisa que soporten Matter, porque ese sello te garantiza que seguiran funcionando aunque cambies de asistente de voz o de marca.

RESPUESTA 9: b - Un cuadricoptero vuela y puede desplazarse en tres dimensiones, lo que le permite inspeccionar techos, sobrevolar obstáculos y llegar a lugares donde un robot con ruedas simplemente no puede entrar. La opción a es falsa porque volar consume mucha energía; la c es falsa porque un dron depende totalmente de sus sensores para estabilizarse; y la d es obviamente falsa. Esta pregunta evalúa la Unidad 9 y verifica que el lector comprende las ventajas y limitaciones reales de cada tipo de plataforma robótica. Por eso en la Unidad 10 aprenderás a elegir la plataforma según el terreno y la misión, en lugar de enamorarte de una sola tecnología.

RESPUESTA 10: b - Según la filosofía del curso, un proyecto fusión integra dos o mas tecnologías de las unidades estudiadas para resolver un problema real, porque el valor esta en la combinación. La opción a describe un proyecto de una sola tecnología; la c

describe copiar sin crear; y la d pone una restriccion de costo que no define al proyecto. Esta pregunta evalúa la Unidad 10 y comprueba que el lector interiorizo la idea central del libro: la fusión de tecnologías al servicio de problemas reales. Guarda esta definición porque será tu criterio para evaluar tu proyecto final: si usa una sola tecnología, todavía no es un proyecto fusión.

RESPUESTA 11: b - Una tecnología emergente es difícil de evaluar porque su potencial todavía no se conoce por completo y su impacto real depende de como se combine con otras tecnologías en el futuro. La opción a es falsa porque muchas tecnologías emergentes empiezan caras pero no es esa la razón de la dificultad; la c es un prejuicio; y la d es falsa porque el interés suele ser alto. Esta pregunta evalúa la Unidad 1 y mide la comprensión de por que predecir el futuro tecnológico es tan complicado. Esta humildad ante el futuro es también una ventaja competitiva: quien experimenta con tecnologías emergentes aprende mas rápido que quien espera a que maduren.

RESPUESTA 12: b - El sobreajuste ocurre cuando el modelo aprende de memoria los datos de entrenamiento, incluyendo su ruido, y por eso falla cuando enfrenta datos nuevos que nunca vio. Un modelo demasiado simple sufre el problema contrario, el subajuste; el calentamiento de la computadora es un tema de hardware; y tener muchos datos de entrenamiento suele ayudar, no perjudicar. Esta pregunta evalúa la Unidad 2 y es fundamental porque el sobreajuste es el error mas común de quien entrena modelos por primera vez. La forma practica de detectarlo es dividir tus datos en entrenamiento y prueba desde el inicio, y desconfiar de cualquier modelo que rinda perfecto en unos y mal en otros.

RESPUESTA 13: b - MQTT es un protocolo ligero de publicación y suscripción diseñado para consumir poco ancho de banda y poca energía, ideal para sensores y dispositivos con recursos limitados. No es el mas rápido para video, porque para eso existen otros protocolos; si necesita un broker intermediario; y funciona precisamente con dispositivos baratos y sencillos. Esta pregunta evalúa la Unidad 3 y verifica que el lector entiende por que en IoT se eligen protocolos ligeros en lugar de los protocolos pesados de la web tradicional. Cuando programes tu primer sensor con un microcontrolador barato notarás la diferencia: con MQTT la batería dura meses, con protocolos pesados duraría días.

RESPUESTA 14: b - La nube publica centraliza el procesamiento en grandes centros de datos lejanos, mientras el edge computing acerca el procesamiento al lugar donde se generan los datos para reducir latencia y dependencia de la conexión. La opción a es falsa porque ambos modelos tienen costos; la c es falsa porque el edge suele coordinarse con la nube; y la d ignora una diferencia arquitectonica real. Esta pregunta

evalúa la Unidad 4 y comprueba la distinción conceptual mas importante de esa unidad. En los proyectos fusión del final del curso aplicarás esta distinción a diario, decidiendo que parte de tu sistema vive en la nube y que parte vive en el edge.

RESPUESTA 15: b - El phishing es un engaño social donde el atacante se hace pasar por un banco, una empresa o un contacto de confianza para que la víctima revele contraseñas o datos bancarios. Dañar hardware físicamente es vandalismo, no phishing; acelerar internet no tiene relación; e instalar antivirus es una defensa, no un ataque. Esta pregunta evalúa la Unidad 5 y es vital porque el phishing es la puerta de entrada de la mayoría de los ciberataques reales y la mejor defensa es el criterio del usuario. La regla de oro es sencilla: ningún banco ni empresa seria te pedirá tu contraseña por correo o mensaje, así que ante la duda verifica por un canal oficial.

RESPUESTA 16: b - El ABS y el PETG soportan mejor el calor y el esfuerzo mecánico que el PLA, que se deforma a temperaturas relativamente bajas. El PLA es barato y fácil de imprimir, pero no sirve para piezas exigentes; el papel y la madera no son filamentos de impresión 3D funcionales para este caso. Esta pregunta evalúa la Unidad 6 y mide la capacidad de elegir material según los requisitos mecanicos, que es una decisión de ingeniería real. Como regla practica, usa PLA para prototipos y piezas decorativas, y reserva el PETG o el ABS para piezas que trabajen bajo esfuerzo o calor.

RESPUESTA 17: a - El viento empuja las aspas de la turbina, ese giro mueve un generador que transforma la energía mecánica de rotación en energía eléctrica. Las aspas no atraen rayos, el movimiento no calienta cables para generar electricidad, y el viento no carga baterías directamente sin pasar por el generador. Esta pregunta evalúa la Unidad 7 y verifica el principio de conversión de energía de la eólica, base para entender su dimensionamiento. Este mismo principio de convertir movimiento en electricidad lo verás aplicado en presas hidroelectricas y hasta en linternas que se cargan agitandolas.

RESPUESTA 18: b - Un escenario o rutina encadena acciones automáticas ante un evento: al anochecer se encienden las luces, se cierran las cortinas y se activa la alarma, todo sin intervención manual. No es una obra de teatro, no es un cable y no es un sensor individual, sino la lógica que coordina varios dispositivos. Esta pregunta evalúa la Unidad 8 y comprueba que el lector entiende que la domótica va mas allá de prender focos con el celular: su valor esta en la automatización. El verdadero salto ocurre cuando dejas de controlar cada foco manualmente y diseñas rutinas que anticipan lo que necesitas, como la del adulto mayor de la pregunta 43.

RESPUESTA 19: b - El efector final es la herramienta en el extremo del brazo robótico que realiza el trabajo: una pinza que sujeta piezas, un soldador, una boquilla de pintura o una ventosa. No es el último robot de la línea, no es el botón de apagado y no es la batería. Esta pregunta evalúa la Unidad 9 y verifica el vocabulario técnico básico de la robótica. Cuando veas un brazo robótico industrial fíjate primero en su efector final, porque esa herramienta te dice exactamente para que tarea fue construido.

RESPUESTA 20: a - Un invernadero automatizado coherente combina sensores IoT para medir humedad y temperatura, la nube para guardar y analizar datos, ciberseguridad para proteger el acceso remoto, impresión 3D para fabricar soportes y piezas, y energías renovables para alimentar el sistema de forma autónoma. Las opciones b, c y d usan una sola tecnología y por tanto no son proyectos fusión. Esta pregunta evalúa la Unidad 10 y mide la capacidad de combinar tecnologías con sentido ante un problema concreto. Observa que la respuesta correcta no es la que usa más tecnologías, sino la que combina las necesarias con una lógica clara de cómo se conectan.

RESPUESTA 21: Verdadero - La convergencia tecnológica es exactamente eso: tecnologías que antes evolucionaban por separado empiezan a combinarse y dan origen a soluciones completamente nuevas, como el smartphone que fusiona teléfono, cámara, GPS e internet. Esta pregunta evalúa la Unidad 1 y refuerza la definición central del curso. Cada vez que uses tu celular recuerda que tienes en la mano una prueba viviente de esta definición.

RESPUESTA 22: Falso - Un modelo entrenado con datos de un solo país aprende los patrones de ese contexto y suele fallar en otros países por diferencias en equipos, poblaciones y prácticas médicas; esto se conoce como sesgo de datos. Antes de desplegarlo en otro lugar hay que validarlo y ajustarlo con datos locales. Esta pregunta evalúa la Unidad 2 y mide la comprensión de las limitaciones y sesgos de los modelos de IA. Por eso los proyectos serios de IA incluyen una fase de validación con datos del lugar donde el modelo va a operar, no solo del lugar donde se entrena.

RESPUESTA 23: Verdadero - El actuador convierte una señal de control en una acción física: abre una válvula, enciende un motor, mueve un brazo robótico. Es el complemento del sensor: el sensor mide el mundo y el actuador lo modifica. Esta pregunta evalúa la Unidad 3 y refuerza la distinción sensor-actuador. Un termostato inteligente resume el ciclo completo: el sensor mide la temperatura y el actuador enciende o apaga la calefacción.

RESPUESTA 24: Falso - El edge computing reduce la dependencia de la nube para tareas críticas de baja latencia, pero no la elimina: la nube sigue siendo necesaria para

almacenamiento masivo, entrenamiento de modelos de IA, análisis historico y coordinacion entre sitios. La arquitectura correcta es híbrida, nube y edge trabajando juntos. Esta pregunta evalúa la Unidad 4 y evita el error común de ver nube y edge como enemigos en lugar de complementos. Piensa en ellos como socios: el edge reacciona rápido y la nube recuerda todo, y juntos cubren lo que ninguno lograría solo.

RESPUESTA 25: Falso - Reutilizar la misma contraseña en todos los servicios es peligroso porque si un solo servicio sufre una filtración, el atacante probara esa contraseña en todos los demás. Lo correcto es usar contraseñas únicas por servicio, idealmente gestionadas con un administrador de contraseñas, y activar la autenticación de dos factores. Esta pregunta evalúa la Unidad 5 y ataca uno de los hábitos mas riesgosos y comunes. Un administrador de contraseñas elimina la excusa de la memoria, porque solo debes recordar una contraseña maestra y el programa cuida las demás.

RESPUESTA 26: Verdadero - La orientación de la pieza determina su resistencia y su acabado, y las partes que quedan volando en el aire necesitan soportes temporales que se retiran después, porque la impresora no puede depositar plástico sobre la nada. Ignorar los soportes provoca piezas deformadas o impresiones fallidas. Esta pregunta evalúa la Unidad 6 y refuerza una de las habilidades practicas mas importantes de la impresión 3D. Con la practica aprenderás a orientar las piezas para minimizar soportes, lo que ahorra material, tiempo y deja un mejor acabado.

RESPUESTA 27: Verdadero - El sol no siempre brilla y el viento no siempre sopla, por eso un sistema renovable confiable necesita baterías para almacenar excedentes, conexión a la red como respaldo, o una combinación de fuentes que se complementen. Diseñar un sistema renovable sin pensar en la intermitencia es el error mas caro de la Unidad 7. Esta pregunta evalúa precisamente ese concepto de dimensionamiento con almacenamiento. Esta es la razón por la que todo presupuesto de un sistema solar debe incluir baterías o un plan de respaldo desde el primer día.

RESPUESTA 28: Falso - Gracias a los estándares de interoperabilidad como Matter, Zigbee y Z-Wave, los dispositivos de distintas marcas pueden trabajar juntos en el mismo hogar inteligente. Quedarse con una sola marca es una decisión valida por comodidad, pero no una obligacion técnica. Esta pregunta evalúa la Unidad 8 y refuerza la importancia de los estándares abiertos. Esa libertad para mezclar marcas también protege tu inversión: si un fabricante desaparece, tu hogar sigue funcionando.

RESPUESTA 29: Verdadero - El giroscopio mide la orientación y el acelerómetro mide los cambios de velocidad, y el controlador de vuelo usa esos datos cientos de veces por

segundo para ajustar cada motor y mantener el dron estable. Sin estos sensores el dron no podría volar de forma controlada. Esta pregunta evalúa la Unidad 9 y verifica la comprensión de los fundamentos del vuelo estabilizado. La próxima vez que veas un video de un dron, recuerda que detrás de ese vuelo suave hay miles de correcciones por segundo.

RESPUESTA 30: Verdadero - La metodología correcta de un proyecto fusión es partir del problema real que se quiere resolver y después seleccionar las tecnologías que mejor lo atacan, porque empezar por la tecnología lleva a soluciones que buscan un problema. Esta pregunta evalúa la Unidad 10 y comprueba que el lector adopto el enfoque de diseño centrado en el problema. Aplícalo como un mantra en tu proyecto final: primero el problema en una frase, después la lista de tecnologías que lo atacan.

RESPUESTA 31: Falso - Las tecnologías emergentes tardan años o décadas en madurar y casi nunca reemplazan por completo a las anteriores; lo normal es que convivan, se complementen o que la tecnología vieja encuentre nuevos nichos. Pensar en reemplazos totales e inmediatos es uno de los mitos que la Unidad 1 desmonta. Esta pregunta evalúa la visión realista de la adopción tecnológica. La historia de la tecnología esta llena de supuestos reemplazos totales que terminaron conviviendo, como el cine y la television.

RESPUESTA 32: Verdadero - En el aprendizaje supervisado cada ejemplo de entrenamiento lleva su etiqueta correcta, como una foto marcada con "gato" o "perro", y el modelo aprende a predecir esas etiquetas para datos nuevos. Sin etiquetas no hay supervisión. Esta pregunta evalúa la Unidad 2 y refuerza la clasificación de los tipos de aprendizaje automático. Casi todas las aplicaciones de IA que usas a diario, desde el correo no deseado hasta el reconocimiento facial, se entrenaron de esta forma.

RESPUESTA 33: Verdadero - El gateway actúa como traductor entre los protocolos ligeros de los sensores y los protocolos de internet, y además puede filtrar, agregar y preprocesar datos para no saturar la conexión con la nube. Es una pieza clave de la arquitectura IoT. Esta pregunta evalúa las Unidades 3 y 4 juntas, mostrando como el gateway vive en la frontera entre el edge y la nube. Por eso el gateway suele ser el primer equipo que se instala en un despliegue IoT serio: sin el, los sensores quedan aislados.

RESPUESTA 34: Verdadero - El ransomware cifra los archivos de la víctima y exige un pago, normalmente en criptomonedas, a cambio de la clave de descifrado, y es una de las amenazas mas daninas para empresas y hospitales. La defensa incluye respaldos aislados, parches actualizados y capacitacion contra phishing. Esta pregunta evalúa la

Unidad 5 y verifica el conocimiento de las amenazas modernas. La lección practica es que ningún rescate garantiza recuperar tus archivos, así que los respaldos aislados son tu unico seguro real.

RESPUESTA 35: Verdadero - Las impresoras de resina curan capas muy finas con luz, lo que da un detalle superior al de FDM, pero la resina líquida es tóxica, requiere guantes y ventilación, y las piezas necesitan lavado y curado posterior. Cada tecnología tiene su lugar según el proyecto. Esta pregunta evalúa la Unidad 6 y mide la capacidad de comparar tecnologías de impresión. Si tu proyecto necesita piezas pequeñas con mucho detalle, como joyería o miniaturas dentales, la resina es tu aliada; para piezas funcionales grandes, la FDM.

RESPUESTA 36: Respuesta modelo - La convergencia tecnológica es el fenomeno por el cual tecnologías que antes avanzaban por separado se combinan y se potencian mutuamente, creando productos y servicios que ninguna podría lograr sola. Un ejemplo claro es el automóvil eléctrico autónomo: fusiona baterías avanzadas de la Unidad 7, sensores IoT de la Unidad 3, IA de la Unidad 2 para la conduccion y conectividad en la nube de la Unidad 4. Otro ejemplo es el hogar inteligente moderno, donde asistentes de voz con IA, sensores, edge computing y estándares de interoperabilidad trabajan juntos para automatizar la casa. Esta pregunta evalúa la Unidad 1 y verifica que el lector puede identificar la convergencia en el mundo real, no solo definirla. Si tu respuesta incluyó la definición y dos ejemplos bien explicados, date los dos puntos completos.

RESPUESTA 37: Respuesta modelo - La estrategia realista combina tres decisiones. Primero, ante la falta de datos etiquetados, usar aprendizaje por transferencia: partir de un modelo preentrenado con radiografías publicas y ajustarlo con los pocos casos locales, además de técnicas de aumento de datos para multiplicar el dataset. Segundo, desplegar el modelo en edge con una computadora local en la clinica, para que el diagnóstico funcione sin internet y los datos sensibles de los pacientes nunca salgan del lugar. Tercero, establecer que la IA solo sugiere y el médico decide, con un protocolo de validación continua. Los riesgos son el sesgo por pocos datos locales, que se mitiga validando con médicos, y el mantenimiento del equipo edge sin soporte técnico cercano. Esta pregunta evalúa las Unidades 2 y 4 y mide la capacidad de diseñar soluciones con restricciones reales. Date el punto completo si tu propuesta incluye transferencia de aprendizaje, despliegue en edge y la supervisión del médico como decisión final.

RESPUESTA 38: Respuesta modelo - Usaría sensores de CO2, partículas PM2.5 y compuestos organicos volatiles, porque son los contaminantes mas relevantes en aulas. Los sensores se comunicarian por LoRa o wifi a un gateway por salón, usando

MQTT por ser ligero, y el gateway agregaría los datos antes de enviarlos a la nube para no saturar la red. El procesamiento de alertas se haría en el edge para avisar de inmediato si el aire es malo, mientras la nube guardaría el historico para análisis por semana o mes. La información se presentaría en un tablero sencillo con semáforo de colores para los maestros y reportes automáticos a la dirección. Cada decisión se justifica: sensores específicos al riesgo real, MQTT por eficiencia, edge para alertas rápidas y nube para historial. Esta pregunta evalúa la Unidad 3 y verifica el diseño completo de un sistema IoT. Date los dos puntos si tu diseño incluye sensores adecuados, un protocolo ligero, procesamiento en edge para alertas y una forma clara de mostrar los datos.

RESPUESTA 39: Respuesta modelo - La arquitectura híbrida es la respuesta: mantener el control de inventario crítico corriendo en edge dentro de la fábrica, con servidores locales que responden en milisegundos y siguen operando aunque se caiga internet, mientras se replica la información a la nube para análisis, respaldos y visibilidad corporativa. La migración se haría por fases: primero se instala la capa edge en paralelo sin tocar el sistema actual, luego se sincroniza con la nube, y solo cuando ambas capas concuerdan se apaga el sistema viejo. Los riesgos mitigados son la latencia en la línea de producción, la dependencia de la conexión a internet y la pérdida de datos durante el cambio. Además se aplicaría ciberseguridad a la conexión entre edge y nube. Esta pregunta evalúa la Unidad 4 y mide la capacidad de diseñar arquitecturas híbridas con cero tiempo de inactividad. Date el punto completo si tu arquitectura mantiene la operación en edge, usa la nube como respaldo y describe una migración por fases sin apagar la fábrica.

RESPUESTA 40: Respuesta modelo - El plan incluye cuatro pilares. Autenticación: contraseñas únicas gestionadas con un administrador de contraseñas mas doble factor en cuentas críticas, porque las credenciales robadas son la causa mas común de incidentes. Respaldos: copias automáticas diarias siguiendo la regla de tres copias en dos medios distintos y una fuera del sitio, porque el ransomware solo se derrota con respaldos aislados. Capacitación: sesiones cortas trimestrales sobre phishing y manejo de datos, porque el empleado es la primera línea de defensa. Respuesta ante incidentes: un protocolo escrito con pasos claros, que incluye desconectar equipos, avisar al responsable y no pagar rescates, porque improvisar durante una crisis siempre sale mal. Cada medida se elige por ser barata y de alto impacto para una empresa pequeña. Esta pregunta evalúa la Unidad 5 y verifica la capacidad de armar un programa de seguridad completo y realista. Date los dos puntos si tu plan cubre los cuatro pilares y explica con tus palabras por que cada uno protege a la empresa.

RESPUESTA 41: Respuesta modelo - El proceso empieza eligiendo FDM con PLA o PETG de grado seguro, porque son fáciles de imprimir y no tóxicos al tacto, descartando resina por sus químicos. El diseño partiría de un modelo paramétrico de prótesis de mano de código abierto, ajustado a las medidas del niño, porque diseñar desde cero una prótesis es riesgoso. En la impresión se usaría alta densidad de relleno en las zonas de esfuerzo, orientación que favorezca la resistencia de los dedos y soportes solo donde sean inevitables. Después vendrían las pruebas: verificar que no haya bordes filosos, comprobar el agarre con objetos de distinto tamaño y peso, y revisar que los mecanismos no atrapen la piel. Finalmente, todo el proceso debe supervisarse con un profesional de la salud, porque una prótesis mal ajustada puede lastimar. Esta pregunta evalúa la Unidad 6 y mide el criterio de ingeniería aplicado a un caso sensible. Date el punto completo si tu proceso incluye elección de material seguro, ajustes al niño, pruebas de seguridad y supervisión profesional.

RESPUESTA 42: Respuesta modelo - La solución es un sistema solar autónomo dimensionado en tres pasos. Primero se calcula el consumo: por ejemplo, 4 focos LED de 10 watts por 5 horas más la carga de 5 celulares suman unos 250 watts-hora diarios. Segundo se dimensionan los paneles: con 5 horas de sol pico se necesita al menos un panel de 100 watts con margen de seguridad. Tercero se dimensionan las baterías para dos días de autonomía, usando baterías de ciclo profundo que no se dañan con descargas parciales. Para los días nublados el sistema cuenta con esa reserva de dos días, además de un controlador de carga que protege las baterías y la opción de priorizar consumos esenciales. También se explicaría a la comunidad el mantenimiento básico: limpiar los paneles y revisar conexiones. Esta pregunta evalúa la Unidad 7 y verifica la capacidad de dimensionar un sistema renovable real. Date los dos puntos si tu diseño calcula el consumo, dimensiona paneles y baterías, y explica que pasa en días nublados.

RESPUESTA 43: Respuesta modelo - Instalaría iluminación automática con sensores de movimiento en pasillos y baño para evitar caídas de noche, un asistente de voz con comandos simples para prender luces y llamar a familiares, sensores de humo y de fugas de gas conectados a alertas en el celular de un familiar, y una cerradura inteligente que un cuidador pueda abrir en emergencias. Las rutinas serían mínimas y automáticas: luces que se encienden solas al anochecer, recordatorios de voz para medicamentos y una rutina de "buenas noches" que apaga todo con una sola frase. En seguridad y privacidad elegiría dispositivos con procesamiento local cuando sea posible, desactivaría el almacenamiento innecesario de grabaciones de voz y configuraría todo yo mismo para que ella no tenga que lidiar con apps complicadas. La clave es que la tecnología sea invisible: ayuda sin pedir atención. Esta pregunta evalúa

la Unidad 8 y mide el diseño centrado en el usuario vulnerable. Date el punto completo si tus dispositivos y rutinas simplifican la vida de la persona mayor y tomas medidas concretas de privacidad.

RESPUESTA 44: Respuesta modelo - El cuadricoptero se mantiene estable gracias a un ciclo continuo: el giroscopio y el acelerómetro miden su orientación e inclinación cientos de veces por segundo, el controlador de vuelo compara esos datos con la posición deseada y ajusta la velocidad de cada uno de los cuatro motores de forma independiente para corregir cualquier desviación. Un uso práctico y responsable en la ciudad sería la inspección de techos y paneles solares en edificios, que hoy requiere andamios peligrosos y caros. Los riesgos incluyen la caída sobre personas, la invasión a la privacidad al grabar vecinos y las interferencias con otros equipos. Las regulaciones a considerar son las de la autoridad aeronáutica local: altura máxima, zonas prohibidas como aeropuertos, registro del dron y respeto a la privacidad de terceros. Esta pregunta evalúa la Unidad 9 y combina comprensión técnica con responsabilidad. Date los dos puntos si explicas el ciclo de estabilización, propones un uso responsable y mencionas riesgos y regulaciones.

RESPUESTA 45: Respuesta modelo - Problema: en mi colonia se desperdicia mucha agua por fugas en tinacos que nadie detecta a tiempo. Tecnologías: sensores IoT de nivel y flujo en cada tinaco, una red LoRa con gateway para comunicar los datos sin depender del wifi de cada casa, la nube para el tablero vecinal con alertas, paneles solares pequeños con batería para alimentar los sensores de forma autónoma, e impresión 3D para fabricar los soportes y carcasas de los sensores a medida. La conexión entre ellas: el sensor mide, LoRa transmite al gateway, el gateway sube a la nube, la nube genera alertas al celular del vecino y gráficas de consumo. El éxito se mediría comparando el consumo de agua antes y después, contando las fugas detectadas a tiempo y encuestando a los vecinos sobre la facilidad de uso. Esta pregunta evalúa la Unidad 10 y es la prueba final de la filosofía fusión: problema real primero, tecnologías después. Date el punto completo si tu proyecto integra al menos cuatro tecnologías con una conexión lógica entre ellas y defines como medir el éxito.

RESPUESTA 46: Respuesta modelo - A favor: el reconocimiento facial podría ayudar a encontrar personas desaparecidas, identificar delincuentes buscados y reaccionar más rápido ante emergencias, además de disuadir delitos en zonas vigiladas. En contra: crea vigilancia masiva que afecta a ciudadanos inocentes, los sistemas tienen más errores con ciertos grupos de personas lo que genera discriminación, los datos biométricos robados no se pueden cambiar como una contraseña, y abre la puerta a abusos por parte de gobiernos o empresas. Mi postura fundamentada es que el despliegue

generalizado en espacios públicos no se justifica hoy: los riesgos para la privacidad y la libertad superan los beneficios, y solo sería aceptable en casos puntuales con autorización judicial, supervisión independiente y borrado garantizado de los datos de inocentes. Esta pregunta evalúa las Unidades 2 y 5 desde el ángulo ético y mide el pensamiento crítico. Date los dos puntos si presentas argumentos de ambos lados y cierras con una postura propia bien fundamentada.

RESPUESTA 47: Respuesta modelo - La nube publica ofrece costo inicial bajo y escalabilidad casi infinita, pero con mayor latencia y menos control sobre los datos; ideal para una startup que lanza una app y necesita crecer rápido sin comprar servidores. La nube privada da control total sobre los datos y la seguridad, a cambio de mayor costo e inversión en personal; ideal para un hospital que maneja expedientes médicos sensibles bajo regulación estricta. El edge computing ofrece la latencia mas baja porque procesa junto a la fuente de datos, pero con capacidad limitada y costo por sitio; ideal para una fábrica donde los robots necesitan respuestas en milisegundos y no pueden esperar a un servidor lejano. La lección es que no hay una opción ganadora universal: cada caso se decide según que pese mas entre costo, velocidad, control y escala. Esta pregunta evalúa la Unidad 4 y verifica el análisis comparativo de arquitecturas. Date el punto completo si comparas las tres opciones en los cuatro criterios y das un caso de uso adecuado para cada una.

RESPUESTA 48: Respuesta modelo - La contraseña del wifi solo protege la puerta de entrada de la red, pero los dispositivos IoT tienen sus propias vulnerabilidades: firmware desactualizado, servicios innecesarios expuestos y datos viajando sin cifrar. Por eso aplicaria cinco capas. Uno: segmentar la red, poniendo los sensores en una red separada de las computadoras personales. Dos: cambiar todas las contraseñas de fábrica y desactivar cuentas por defecto. Tres: mantener el firmware actualizado, automatizando las actualizaciones cuando sea posible. Cuatro: cifrar las comunicaciones con protocolos seguros para que nadie intercepte los datos. Cinco: aplicar el principio de mínimo privilegio, dando a cada dispositivo solo los permisos que necesita. Cada capa cubre un hueco que la contraseña del wifi no tapa, y juntas forman la defensa en profundidad. Esta pregunta evalúa la Unidad 5 aplicada al IoT y mide la comprensión de la seguridad por capas. Date los dos puntos si explicas por que la contraseña del wifi no basta y describes al menos cinco medidas en capas.

RESPUESTA 49: Respuesta modelo - El flujo empieza obteniendo el modelo 3D: primero buscar en repositorios gratuitos si alguien ya modelo esa pieza, y si no existe, medir la pieza rota con un calibrador y modelarla en un programa CAD gratuito o escanearla con el celular. Después se ajusta el diseño dejando tolerancias para que las partes móviles

no queden pegadas y reforzando las zonas que soportan esfuerzo. En la impresión se elegiría PETG por su resistencia al calor y a los golpes, con buena densidad de relleno y orientación que favorezca la resistencia mecánica. La prueba incluiría montar la pieza, hacer funcionar la licuadora en vacío y luego con carga, revisando vibraciones y temperatura. Las limitaciones a aceptar son que el plástico impreso es menos resistente que el inyectado de fábrica, que puede deformarse con calor alto y que la pieza es para uso personal, no para vender. Esta pregunta evalúa la Unidad 6 y verifica el dominio del flujo completo de fabricación digital. Date el punto completo si tu flujo cubre obtención del modelo, ajustes, elección de material, pruebas y limitaciones aceptadas.

RESPUESTA 50: Respuesta modelo - Despierto y el hogar inteligente ya ajustó la temperatura y preparo el café según mi rutina aprendida por IA. En el baño, un espejo con sensores IoT revisa mis signos vitales y los compara en la nube con mi historial médico, alertando solo si algo sale de rango. Desayuno con energía 100 por ciento renovable: el techo solar y la batería doméstica cubrieron la noche y el excedente se vendió a la red automáticamente. Para ir al trabajo subo a un vehículo autónomo eléctrico que se coordina con drones de reparto y semáforos inteligentes para evitar tráfico. En la oficina, un asistente de IA resume mis correos mientras un robot colaborativo arma prototipos cuyas piezas se imprimieron en 3D durante la noche. Todo viaja cifrado bajo esquemas de cero confianza porque cada dispositivo verifica su identidad continuamente. Al volver, el sistema ya sabe que hoy hice ejercicio y ajusta la cena sugerida y la iluminación para descansar mejor. Las seis tecnologías interactúan: IA, IoT, nube y edge, renovables, robótica y drones, impresión 3D, hogar inteligente y ciberseguridad tejiendo una sola experiencia. Esta pregunta evalúa las diez unidades juntas y cierra el curso imaginando la convergencia hecha vida cotidiana. Date los dos puntos si tu análisis menciona al menos seis tecnologías y muestra como interactúan entre sí en tu día.

ÍNDICE

PRÓLOGO	4
Por qué este libro existe	4
Cómo está organizado el libro	5
Cómo estudiar este libro	6
Lo que vas a lograr	6
UNIDAD 1: Panorama de las tecnologías emergentes	8
Explicación detallada	11
Ejemplos paso a paso	21
Práctica	27
Resumen de la unidad	34
UNIDAD 2: Inteligencia artificial aplicada	36
Explicación detallada	38
Ejemplos paso a paso	50
Práctica	55
Resumen de la unidad	59
UNIDAD 3: Internet de las cosas (IoT)	61
Introducción	61
Explicación detallada	64
Ejemplos paso a paso	73
Práctica	80
Resumen de la unidad	89
UNIDAD 4: Nube y edge computing	92
Explicación detallada	95
Ejemplos paso a paso	107
Práctica	114
Resumen de la unidad	119
UNIDAD 5: Ciberseguridad moderna	121
Introducción	121
Explicación detallada	123
Ejemplos paso a paso	136
Práctica	144
Resumen de la unidad	147
UNIDAD 6: Impresión 3D y fabricación digital	150
Explicación detallada	152
Ejemplos paso a paso	164

Práctica	170
Resumen de la unidad	175
UNIDAD 7: Energías renovables y tecnología	177
Introducción	177
Explicación detallada	180
Ejemplos paso a paso	190
Práctica	198
Resumen de la unidad	203
UNIDAD 8: Automatización del hogar inteligente	205
Explicación detallada	208
Ejemplos paso a paso	218
Práctica	225
Resumen de la unidad	229
UNIDAD 9: Robótica y drones	232
Introducción	232
Explicación detallada	234
Ejemplos paso a paso	248
Práctica	257
Resumen de la unidad	262
UNIDAD 10: Proyectos que combinan varias tecnologías	264
Explicación detallada	267
Ejemplos paso a paso	272
Práctica	287
Resumen de la unidad	293
EXAMEN FINAL	296
Preguntas de opción múltiple	296
Preguntas de verdadero o falso	299
Preguntas abiertas	300
Respuestas y soluciones	302

Curso Completo de Tecnología Fusión

El panorama de las tecnologías que están cambiando el mundo: inteligencia artificial, internet de las cosas, nube y edge computing, ciberseguridad, impresión 3D, energías renovables, hogar inteligente, robótica y drones. 10 unidades con prácticas y examen final.

Colibrí Editorial EGA

lucasegaacademy.com

Hecho en México