SISTEMAS DE ADQUISICIÓN Y CONTROL

Grado en Diseño de Videojuegos

Versión 1.0

Telmo Miguel-Medina

UNIVERSIDAD DE BURGOS

Universidad de Burgos

Grado en Diseño de Videojuegos

Autor:

Telmo Miguel-Medina

SISTEMAS DE ADQUISICIÓN Y CONTROL

Versión 1.0

UNIVERSIDAD DE BURGOS

ISBN: en tramitación
DOI: en tramitación
Depósito legal: en tramitación

Primera edición, 2026
Universidad de Burgos
Grado en Diseño de Videojuegos

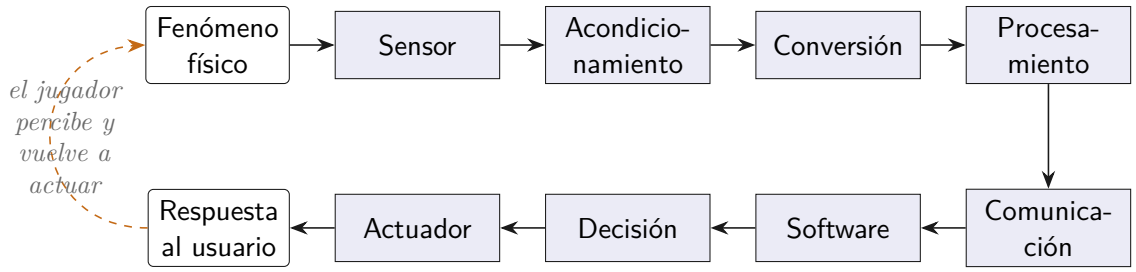
Contacto: tmiguel@ubu.es

Declaración de uso de Inteligencia Artificial (IA). En la preparación de este material se han empleado herramientas de IA generativa (modelos Claude Sonnet y Claude Opus, de Anthropic) como apoyo para la revisión de estilo y coherencia del texto, la generación de figuras vectoriales en LaTeX a partir de bocetos del autor, la verificación y el formateo de la bibliografía, y tareas de maquetación. Todas las decisiones de contenido, la selección y validación de fuentes y la redacción final son responsabilidad del autor, que asume la integridad del texto resultante.

ÍNDICE

PREFACIO

Este libro es el material de teoría de la asignatura *Sistemas de Adquisición y Control* del Grado en Diseño de Videojuegos de la Universidad de Burgos. Su hilo conductor es una cadena que va del fenómeno físico a la respuesta al jugador, y vuelve a empezar:



Cada capítulo desarrolla un tramo de la cadena; el último la recorre entera.

Entender esa cadena permite razonar cómo un mando, un periférico o una interfaz convierten una acción del cuerpo en información que un juego puede usar, y cómo el juego devuelve una respuesta física al jugador. Cada capítulo desarrolla un tramo de la cadena; el último la recorre entera, ya con un modelo que interpreta los datos y una interfaz que se adapta a quien juega.

A QUIÉN SE DIRIGE

A estudiantes de diseño de videojuegos, no de ingeniería electrónica. El objetivo no es calcular circuitos, sino **comprender los principios y poder tomar decisiones de diseño**: elegir un tipo de entrada, prever el efecto de añadir vibración o un enlace inalámbrico, especificar y prototipar un periférico. Por eso el texto evita el desarrollo matemático —la única fórmula que se usa con soltura es la ley de Ohm— y se apoya en analogías, esquemas y ejemplos de dispositivos reales. Los términos técnicos en inglés se mantienen con una breve explicación la primera vez que aparecen, y hay un glosario al final.

CÓMO ESTÁ ORGANIZADO

Tres bloques y once capítulos, en una progresión de *comprender* a *construir* y a *integrar*:

- **Bloque I — Principios de la electrónica aplicados a la gamificación y la interfaz hombre-máquina** (capítulos 1–5): qué es un sistema de adquisición y control, fundamentos de electrónica, sensores, acondicionamiento y conversión de señales, y microcontroladores. El capítulo 1 incluye una breve historia de los mandos como introducción.
- **Bloque II — Personalización de hardware y periféricos** (capítulos 6–8): los dispositivos de entrada (botones, palancas, ratones, pantallas), los actuadores y la retroalimentación háptica, y el proceso completo de diseño de un periférico personalizado.
- **Bloque III — Nuevas tecnologías en los sistemas de adquisición y control de software** (capítulos 9–11): la comunicación entre el hardware y el software, las tecnologías emergentes de interacción, y los sistemas que interpretan los datos para adaptarse al jugador, con sus límites y su dimensión ética.

Cada capítulo abre con sus *objetivos de aprendizaje* y su *alcance*, y cierra con un *resumen*, unas *ideas clave*, la lista de términos introducidos y su *bibliografía*. A lo largo del texto, los recuadros marcan conexiones con el diseño de videojuegos, avisos de seguridad y ampliaciones opcionales; las figuras son diagramas propios, y algunas fotografías muestran los componentes reales.

BLOQUE I

PRINCIPIOS DE LA ELECTRÓNICA APLICADOS A LA GAMIFICACIÓN Y LA INTERFAZ HOMBRE-MÁQUINA

CAPÍTULO 1

Introducción a los sistemas de adquisición y control

Cada vez que mueves un *stick*, aprietas un gatillo o notas la vibración de un impacto, hay un sistema trabajando por debajo: algo ha convertido el movimiento de tu mano en un número, ese número ha viajado hasta el juego, el juego ha decidido una respuesta y otro componente la ha devuelto a tus manos en forma de vibración. Ese «algo» es un sistema de adquisición y control. Este capítulo lo presenta y fija el vocabulario que se usará en todo el libro. Todavía no abriremos ningún aparato: primero conviene ver el mapa completo.

Objetivos de aprendizaje

- Explicar qué es un sistema de adquisición y control y nombrar sus componentes.
- Distinguir sistema físico y sistema digital, y variable física de variable de información.
- Describir la cadena *entrada física* → *adquisición* → *procesamiento* → *decisión* → *actuación* y reconocerla en un videojuego.
- Diferenciar los sistemas de lazo abierto y de lazo cerrado con ejemplos de interacción.

Alcance

Este capítulo ofrece una visión general de la arquitectura de estos sistemas. No se entra todavía en la implementación electrónica —eso empieza en el capítulo 2—: aquí solo se construye el marco conceptual y el lenguaje común.

ÍNDICE DEL CAPÍTULO

1.1 SISTEMAS DE ADQUISICIÓN Y CONTROL

> **Definición 1.1: sistema de adquisición y control**
>
> Conjunto de elementos que **miden** magnitudes del mundo físico, **deciden** a partir de esas medidas y **actúan** sobre el entorno.

En el nombre hay dos verbos. *Adquirir* es observar el mundo: recoger una magnitud física —una posición, una fuerza, un nivel de luz— y convertirla en un dato con el que se pueda trabajar. *Controlar* es lo contrario: tomar una decisión y provocar un efecto físico —encender una luz, mover un motor, hacer vibrar una carcasa—. Un *sistema de adquisición y control* hace las dos cosas y, a menudo, las encadena: mide, decide y actúa una y otra vez.

Estos sistemas están por todas partes. Un termostato mide la temperatura de una habitación y enciende o apaga la calefacción. Una cámara de fotos mide la luz que entra y ajusta la exposición y el enfoque. El sistema de frenado de un coche mide si una rueda se ha bloqueado y libera la presión del freno decenas de veces por segundo. En todos los casos aparece la misma idea: un lazo de medir, decidir y actuar [1].

Un mando y su consola forman también un sistema de adquisición y control. El mando *adquiere* tus acciones (qué botones pulsas, cuánto inclinas cada *stick*, cómo mueves el mando en el aire) y el conjunto *controla* la salida que percibes: la imagen en la pantalla, el sonido, la vibración, la resistencia de un gatillo. La diferencia con el termostato es que aquí, en mitad del lazo, hay una persona.

1.2 BREVE HISTORIA DE LOS SISTEMAS DE ADQUISICIÓN EN LOS VIDEOJUEGOS

El mando ha sido siempre un sistema de adquisición: un aparato que convierte las acciones de una persona en datos para una máquina. Su evolución es, punto por punto, la de la cadena que estudia este libro. Este apartado la recorre con los

hitos principales (figura 1.1), mirándolos como *soluciones técnicas*; el libro *Toma el Control* desarrolla esta historia con detalle [6].

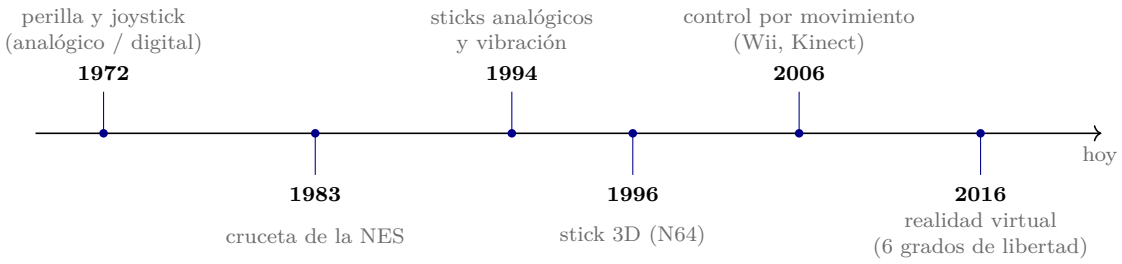


Figura 1.1. Hitos del control en los videojuegos. Cambian los componentes de cada época, pero no la estructura de la cadena de adquisición y control.

Los primeros controles: analógico y digital (años 70)

Los mandos de diales de la Magnavox Odyssey (1972) y las perillas de *Pong* generaban una tensión eléctrica que subía y bajaba con suavidad al girar la rueda: una **señal analógica** pura. Poco después, el joystick de cuatro contactos y el botón de disparo de *Space Invaders* (1978) producían lo contrario: hay contacto o no lo hay, una **señal digital**. La distinción entre analógico y digital —el eje de toda la asignatura— nace aquí, obligada por el coste y la sencillez de la electrónica de la época.

Figura 1.2. La perilla de la Magnavox Odyssey y de las máquinas de *Pong*: girarla mueve el cursor de forma continua, generando una señal analógica. *(foto: Evan-Amos · dominio público)*

El mando doméstico y la cruceta (años 80)

El joystick de la Atari 2600 llevó el control digital al salón. La NES (1983) introdujo la **cruceta** de Gunpei Yokoi y los botones A/B: un puñado de contactos que el microcontrolador del mando lee *escaneando* una pequeña matriz de filas y columnas (capítulo 6). Su disposición se convirtió en el estándar que han seguido casi todos los mandos posteriores.

Figura 1.3. El mando de la NES: la cruceta y los botones A/B son un conjunto de contactos digitales; su disposición marcó décadas de diseño de mandos. *(foto: Evan-Amos · dominio público)*

Más botones y la vuelta de lo analógico (años 90)

La Super Nintendo añadió gatillos de hombro y cuatro botones de cara; la PlayStation trajo los **sticks analógicos** —dos potenciómetros por palanca, cuya tensión hay que *digitalizar con un ADC* (capítulo 4)— y, con el DualShock, la **vibración**: por primera vez el mando era también una salida (capítulo 7). El lazo empezaba a cerrarse.

Figura 1.4. El DualShock de PlayStation: dos palancas analógicas, que exigen un convertidor analógico-digital, y el primer motor de vibración integrado en un mando. *(foto: Evan-Amos · dominio público)*

El salto al 3D (mediados de los 90)

La Nintendo 64 montó la primera palanca analógica en una consola. *Super Mario 64* dejó claro que moverse por un escenario tridimensional necesita **dirección e intensidad** —andar, trotar o correr según la inclinación—, algo que ocho contactos no pueden dar.

Figura 1.5. El mando de la Nintendo 64 y su palanca analógica, el primer *stick* de una consola doméstica. *(foto: Evan-Amos · dominio público)*

Inalámbrico y control por movimiento (años 2000)

El Wii Remote (2006) combinó acelerómetros, un puntero por infrarrojos y comunicación Bluetooth (capítulos 3 y 9): el mando pasó a ser un objeto que se mueve en el aire. El Kinect fue un paso más: eliminó el mando y reconstruía el cuerpo del jugador con una cámara de profundidad y visión artificial (capítulo 10).

Figura 1.6. El Wii Remote y el Nunchuk: acelerómetros para detectar el movimiento, un sensor infrarrojo para apuntar a la pantalla y enlace inalámbrico por Bluetooth. *(foto: Evan-Amos · dominio público)*

Móvil, inmersión y personalización (2010 – hoy)

Llegaron la pantalla táctil capacitiva, los mandos de realidad virtual con **seis grados de libertad** (capítulo 10), los mandos «Pro» y «Elite» personalizables y, sobre todo, el Xbox Adaptive Controller: un concentrador con jacks de 3,5 mm —cada uno, una entrada— que permite a cada jugador montar su propia interfaz (capítulo 8). La háptica de banda ancha y los gatillos adaptativos del DualSense (capítulo 7) llevaron la salida física un paso más allá.

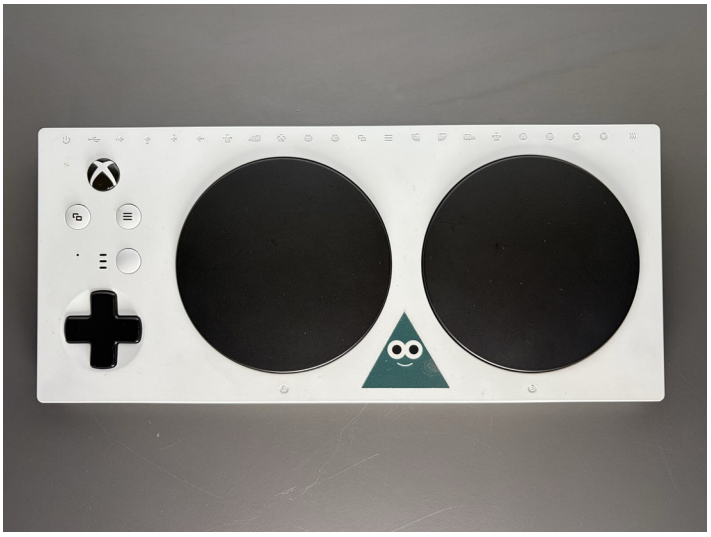


Figura 1.7. El Xbox Adaptive Controller: un concentrador de accesibilidad con jacks de 3,5 mm y puertos USB a los que se conectan pulsadores, pedales y palancas de terceros. *(foto: InclusiveGameLab · CC BY-SA 4.0)*

Los hilos que recorren el libro

De toda esta evolución se pueden extraer cuatro tendencias, que reaparecen en cada capítulo:

- **De lo analógico a lo digital**: de la perilla de *Pong* a los paquetes de datos de un mando USB.
- **De unidireccional a bidireccional**: el mando dejó de ser solo entrada cuando apareció la vibración, y hoy también da fuerza, sonido y luz.
- **De genérico a personal y accesible**: del mando único para todos a los perfiles, el remapeo y los concentradores adaptativos.
- **Del mando al cuerpo**: de pulsar botones a inclinar, agitar, señalar, mirar y, en el extremo, pensar.

Cambian los componentes de cada época, pero la estructura de la cadena —sensor, acondicionamiento, conversión, procesamiento, comunicación, actuación— es siempre la misma (sección 1.10).

1.3 SISTEMAS FÍSICOS Y SISTEMAS DIGITALES

El mundo físico es **continuo**. Cuando empujas un *stick*, no salta de «centrado» a «a tope»: pasa por todas las posiciones intermedias, y entre dos posiciones cualesquiera siempre hay otra. La fuerza con que aprietas un gatillo, la cantidad de luz que recibe un sensor o el ángulo con que giras un volante son magnitudes que pueden tomar infinitos valores.

Un sistema digital —un microcontrolador, un ordenador, el motor de un videojuego— es lo contrario: trabaja con **números finitos**, expresados en *bits*, y solo puede distinguir una cantidad limitada de valores distintos. No maneja «la posición exacta del *stick*», sino, por ejemplo, «un número entero entre 0 y 65 535».

Entre ambos mundos hay una frontera, y cruzarla tiene un coste. Igual que un reloj de agujas muestra el tiempo como un movimiento continuo y un reloj de dígitos lo parte en saltos de un segundo, convertir una magnitud física en un dato digital obliga a *redondear*: a quedarse con el valor discreto más cercano. Cuánto se pierde en ese redondeo, y cómo se hace bien, es el tema del capítulo 4. Por ahora basta con retener la idea: **todo sistema de adquisición vive a caballo entre una parte física y continua y una parte digital y discreta, y su trabajo es traducir entre las dos**.

1.4 VARIABLES FÍSICAS Y VARIABLES DE INFORMACIÓN

Llamamos *variable física* a la magnitud del mundo real que nos interesa: posición, ángulo, fuerza, presión, luz, sonido, temperatura, aceleración, proximidad, ritmo cardíaco… Y llamamos *variable de información* a su representación dentro del sistema digital: un número, un bit («pulsado / no pulsado») o un evento («se ha pulsado A»).

Medir es precisamente eso: asignar a una variable física una variable de información según una escala acordada. La escala importa tanto como el número. Decir

que un gatillo «marca 128» no significa nada si no se sabe que la escala va de 0 (sin apretar) a 255 (a fondo); con esa escala, 128 quiere decir «a medio recorrido».

Tabla 1.1. En un mando, cada acción física del jugador acaba representada como una variable de información.

Variable física	Ejemplo en un mando	Variable de información
Contacto	Pulsar un botón	1 bit (sí / no)
Posición	Inclinación de un *stick*	Dos números (ejes X e Y)
Fuerza / recorrido	Presión sobre un gatillo analógico	Un número (p. ej. 0 a 255)
Aceleración	Mover el mando en el aire	Tres números (ejes X, Y, Z)
Sonido	Hablar por el micrófono	Una secuencia de números

Ejemplo 1.1

En un mando de carreras, el pedal de acelerador es un potenciómetro. La *variable física* es el recorrido del pedal; la *variable de información* que llega al juego es un número, por ejemplo entre 0 y 1 023. El juego no «sabe» cuántos milímetros has bajado el pedal: solo recibe ese número y lo interpreta como «acelera al 60 %».

1.5 SEÑALES ANALÓGICAS Y DIGITALES

Una *señal* es una magnitud que varía a lo largo del tiempo y que transporta información. La señal es el vehículo; el dato es la carga.

Una señal **analógica** varía de forma continua, tanto en el tiempo como en el valor que toma: piensa en una rampa suave, sin escalones. La rueda giratoria (*paddle*) de los primeros *Pong* generaba una señal así —una tensión eléctrica que subía y bajaba con suavidad al girar la rueda—. Una señal **digital** solo toma unos pocos valores discretos, normalmente dos: «hay contacto» o «no lo hay». Un botón produce una señal digital.

La diferencia tiene una consecuencia práctica. Toda señal analógica se *ensucia* por el camino: los cables, los motores cercanos y la propia electrónica añaden pequeñas

variaciones —ruido— que se confunden con la señal. Una señal digital, al tener que distinguir solo entre dos niveles bien separados, aguanta mucho mejor ese ruido: un pequeño temblor no llega a convertir un «0» en un «1». Volveremos sobre el ruido y cómo se combate en el capítulo 4.

> **Conexión con el diseño de videojuegos**
>
> Un D-pad (cruceta) entrega señales digitales: cuatro contactos que, combinados, dan ocho direcciones. Es suficiente para un plataformas 2D clásico, donde el personaje corre a una única velocidad. Para mover a un personaje por un escenario 3D hace falta una señal analógica: el *stick* no solo dice *hacia dónde*, sino *cuánto* —caminar, trotar o esprintar según lo inclines—. La naturaleza de la señal de entrada condiciona qué mecánicas de movimiento son posibles.

1.6 SENSORES, PROCESAMIENTO Y ACTUADORES

Todo sistema de adquisición y control reparte el trabajo en tres papeles.

Sensores
Convierten una variable física en una señal eléctrica que el sistema puede leer. Son la puerta de entrada.

Procesamiento
Interpreta las señales, decide y coordina. En un mando es un microcontrolador; en el conjunto, también el procesador de la consola y el motor del juego.

Actuadores
Convierten una señal eléctrica en un efecto físico: movimiento, luz, sonido, fuerza, calor. Son la puerta de salida.

Sensores y actuadores son dos caras de la misma moneda: un *transductor* es cualquier dispositivo que convierte energía de una forma en otra. Un *sensor* es un transductor de entrada (energía del mundo → señal eléctrica) y un *actuador* es un transductor de salida (señal eléctrica → energía en el mundo).

Tabla 1.2. Los tres papeles, vistos en un mando moderno.

Papel	Ejemplos en el mando
Sensores	botones, *sticks*, gatillos, giróscopo y acelerómetro, panel táctil, micrófono
Procesamiento	microcontrolador del mando; procesador de la consola; motor del juego
Actuadores	motores de vibración, LED e iluminación, altavoz, gatillos con resistencia

1.7 SISTEMAS DE LAZO ABIERTO Y LAZO CERRADO

Un sistema funciona en *lazo abierto* cuando actúa **sin comprobar el resultado** de su actuación. Una tostadora con temporizador es lazo abierto: cuenta un tiempo y salta, sin mirar si el pan está dorado o carbonizado. La vibración de un mando cuando recibes un golpe funciona igual: el juego envía la orden «vibra fuerte medio segundo» y el mando la ejecuta sin más; nadie verifica el efecto.

Un sistema funciona en *lazo cerrado* cuando **mide el resultado y lo usa para corregirse**. Esa medida de vuelta se llama *realimentación*. Un termostato es lazo cerrado: mide la temperatura, la compara con la deseada y decide; mientras haga frío, sigue calentando. Un volante con *force-feedback* (fuerza de retorno) se acerca a esta idea: endurece o afloja el giro según lo que «siente» el coche en la pista, y tú respondes a esa resistencia moviendo las manos [1].

En un videojuego, el lazo se cierra normalmente **a través de la persona**. El sistema no comprueba por sí mismo si has apuntado bien; eres tú quien mira la pantalla, compara con lo que querías y corrige. La interfaz —el mando y la pantalla— es el punto por donde ese lazo pasa por tus manos y tus ojos [3].

Ejemplo 1.2

Apuntar en un *shooter*: mueves el ratón → el punto de mira se desplaza → ves que aún no está sobre el enemigo → corriges el movimiento → disparas. Es un lazo cerrado en el que el ser humano hace de sensor (los ojos), de procesamiento (el cerebro) y de actuador (la mano).

1.8 INTERFAZ HOMBRE-MÁQUINA (HMI)

La *interfaz hombre-máquina* (HMI) es el conjunto de entradas y salidas que conectan a una persona con un sistema. Un mando es una HMI; también lo son el cuadro de mandos de un avión, el cajero de un banco o el volante y los pedales de un coche.

Una HMI es buena cuando [3]:

- la relación entre la acción y su efecto es **clara** (pulsar aquí hace esto);
- hay **realimentación inmediata** (el sistema responde y se nota que te ha oído);
- la **latencia** es baja (la respuesta llega enseguida);
- es **coherente** (acciones parecidas producen efectos parecidos);
- es **accesible** (se puede usar con capacidades y preferencias distintas).

Las interfaces con las máquinas han cambiado mucho. De los paneles de interruptores y luces se pasó al teclado; después llegaron el ratón y las interfaces gráficas; el mando de consola introdujo la cruceta y, más tarde, los *sticks* analógicos; luego vinieron las pantallas táctiles y, hoy, el control por movimiento, por voz y por mirada. Cada salto amplía *qué* se puede expresar y *para quién*. El libro *Toma el Control* recorre esa historia con detalle [6]; aquí solo nos interesa quedarnos con la tendencia, porque marca hacia dónde van los sistemas de adquisición que estudiaremos.

1.9 SISTEMAS INTERACTIVOS Y VIDEOJUEGOS

Un *sistema interactivo* es aquel que responde de forma continua a las acciones de una persona, cerrando un *bucle de interacción*: el usuario actúa, el sistema responde, el usuario percibe esa respuesta y vuelve a actuar. Un videojuego es un sistema interactivo llevado al extremo: ese bucle se repite decenas o cientos de veces por segundo.

Internamente, un juego ejecuta una y otra vez el mismo ciclo —el *game loop* (bucle de juego)—: leer las entradas, actualizar el estado del mundo, y dibujar la imagen y generar el sonido [2]. La frecuencia con que se repite (30, 60 o 120 veces por segundo) marca cada cuánto el juego «te escucha» y te responde.

Estos sistemas funcionan en *tiempo real*. Tiempo real no significa «instantáneo», sino «responder dentro de un plazo lo bastante corto como para que la interacción se perciba inmediata». El retardo entre que actúas y que ves el efecto es la *latencia*; de dónde sale y cuánta se tolera lo veremos en los capítulos 4 y 9. Lo esencial ahora es que un videojuego se comporta como un **sistema de adquisición y control en tiempo real con un ser humano dentro del lazo** [5].

1.10 ARQUITECTURA GENERAL DE UN SISTEMA DE ADQUISICIÓN Y CONTROL

Todas las piezas anteriores se ordenan en una cadena (figura 1.8). Tiene un sentido de *ida* —de la acción física al dato— y un sentido de *vuelta* —del dato al efecto físico—.

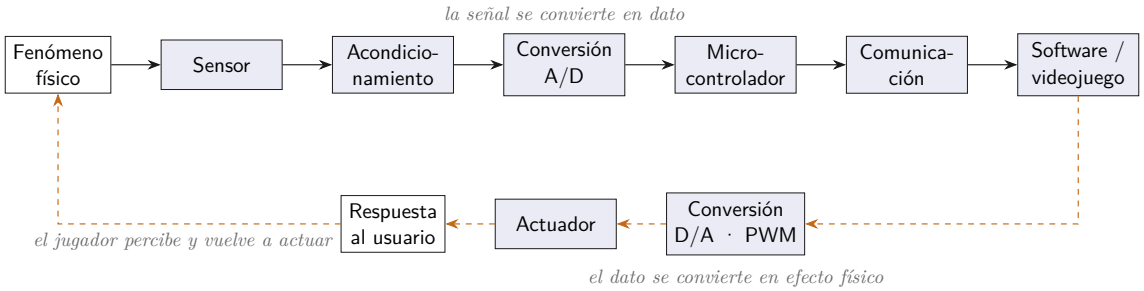


Figura 1.8. La cadena de un sistema de adquisición y control. Arriba, el sentido de ida (adquisición); abajo, el de vuelta (actuación y realimentación). La misma estructura aparece en un mando de cinco euros y en uno de gama alta: cambian los componentes, no las etapas.

Recorrámosla etapa por etapa, indicando en qué capítulo se estudia cada una:

Fenómeno físico
La acción del jugador: pulsar, inclinar, girar, mover, hablar.

Sensor (capítulo 3)
Convierte esa acción en una señal eléctrica.

Acondicionamiento (capítulo 4)
Adapta la señal para poder medirla bien: la amplifica, la filtra, ajusta su nivel.

Conversión A/D (capítulo 4)
Pasa de la señal continua a un número: es donde se cruza la frontera físico / digital.

Microcontrolador (capítulo 5)
Lee los números, los interpreta, decide y los empaqueta.

Comunicación (capítulo 9)
Lleva los datos hasta el ordenador o la consola y, dentro de ellos, hasta el motor del juego.

Software / videojuego
Usa los datos para actualizar el mundo y decide la respuesta.

Conversión D/A y PWM (capítulos 4 y 7)
Traduce la decisión de vuelta a una señal capaz de mover algo.

Actuador (capítulo 7)
Produce el efecto físico: vibración, luz, sonido, fuerza.

Respuesta al usuario
El jugador la percibe, la compara con lo que pretendía y vuelve a actuar. El lazo se cierra.

En instrumentación, a la parte de ida se la llama *cadena de medida*: la sucesión sensor → acondicionamiento → conversión que lleva de una magnitud física a un dato fiable [4]. En este libro la ampliamos con la parte de vuelta, porque en un mando la salida —la háptica, la luz, el sonido— es tan importante como la entrada.

1.11 EJEMPLOS DE APLICACIÓN EN VIDEOJUEGOS Y GAMIFICACIÓN

Mover un personaje con el *stick*. El pulgar inclina el *stick* (fenómeno físico); dos potenciómetros o sensores de efecto Hall generan dos señales (sensores); se adaptan y se convierten en dos números (conversión A/D); el microcontrolador los empaqueta y los envía (comunicación); el juego los interpreta como una dirección y una velocidad y mueve al personaje; lo ves en pantalla y ajustas la inclinación. Toda la cadena, decenas de veces por segundo.

Disparar con un botón. Aquí la variable física es un simple contacto y la variable de información es un **evento** digital: «se ha pulsado». No hay que convertir ningún valor continuo; basta con detectar el cambio de estado y comunicarlo a tiempo.

Vibración al recibir daño. El juego decide «el jugador ha sido alcanzado» y envía una orden de vibración (sentido de vuelta). El mando la ejecuta en **lazo abierto**: activa el motor el tiempo indicado sin comprobar nada. Es el canal de salida trabajando solo.

Gamificación y *exergames*. Una alfombra de baile, una bicicleta estática con sensor de pedaleo o un pulsómetro que sube o baja la dificultad según tu ritmo cardíaco usan el **cuerpo entero** como fuente de variables físicas. La cadena es la misma; solo cambian los sensores.

Accesibilidad. Un pulsador grande conectado a un adaptador que hace de puente con la consola sustituye a un botón del mando. La entrada física es distinta, pero el resto de la cadena —conversión, comunicación, juego— no cambia.

> **Conexión con el diseño de videojuegos**
>
> En los cinco ejemplos, el diseño del sistema de entrada y de salida decide *qué mecánicas son posibles* y *para quién*. Elegir los sensores, los actuadores y la forma de comunicarlos no es una cuestión solo técnica: es una decisión de diseño de juego.

RESUMEN

Un sistema de adquisición y control mide magnitudes del mundo físico, decide y actúa sobre el entorno. El mundo físico es continuo y los sistemas digitales trabajan con números finitos: por eso hace falta traducir entre *variables físicas* y *variables de información*. Esa información viaja en *señales*, que pueden ser analógicas (continuas, sensibles al ruido) o digitales (discretas, más robustas). El trabajo se reparte entre *sensores* (entrada), *procesamiento* (decisión) y *actuadores* (salida); sensores y actuadores son transductores. Un sistema es de *lazo abierto* si actúa sin comprobar el resultado y de *lazo cerrado* si se realimenta con la medida de ese resultado; en un videojuego, el lazo suele cerrarse a través del jugador. La interfaz hombre-máquina es el punto de contacto entre la persona y el sistema, y un videojuego es un sistema interactivo en tiempo real. Toda la asignatura recorre una única cadena: fenómeno físico → sensor → acondicionamiento → conversión → procesamiento → comunicación → software → decisión → actuador → respuesta al usuario.

Ideas clave

- Adquirir es convertir el mundo en datos; controlar es convertir datos en efectos sobre el mundo.
- Lo físico es continuo y lo digital es discreto: medir siempre implica redondear.
- Analógico frente a digital: el segundo pierde matices, pero resiste mucho mejor el ruido.
- Lazo abierto: actúo y no miro. Lazo cerrado: mido el resultado y me corrijo.
- En un videojuego, el jugador está *dentro* del lazo de control.

TÉRMINOS DEL CAPÍTULO

Sistema de adquisición y control, variable física, variable de información, señal, señal analógica, señal digital, sensor, transductor, actuador, procesamiento, lazo abierto, lazo cerrado, realimentación, interfaz hombre-máquina (HMI), sistema interactivo, bucle de interacción, tiempo real, latencia.

PARA SABER MÁS

- Sánchez Ortega [6] recorre la historia de los mandos y de las interfaces de juego; útil para ampliar la panorámica de la sección 1.8.
- Norman [3] es la referencia clásica sobre por qué unas interfaces se entienden y otras no (correspondencia, realimentación, restricciones).
- Bolton [1], capítulos iniciales, para la visión de sistemas de medida y control y los conceptos de lazo abierto y cerrado fuera del videojuego.

BIBLIOGRAFÍA DEL CAPÍTULO

[1] William Bolton. *Mechatronics: Electronic Control Systems in Mechanical and Electrical Engineering*. 6.ª ed. Harlow: Pearson Education, 2015. ISBN: 978-1-292-07668-3.

[2] Jason Gregory. *Game Engine Architecture*. 3.ª ed. Boca Raton, FL: CRC Press, 2018. ISBN: 978-1-138-03545-4.

[3] Donald A. Norman. *The Design of Everyday Things*. Revised and expanded. New York: Basic Books, 2013. ISBN: 978-0-465-05065-9.

[4] Ramon Pallàs-Areny y John G. Webster. *Sensors and Signal Conditioning*. 2.ª ed. New York: John Wiley & Sons, 2001. ISBN: 978-0-471-33232-9.

[5] Katie Salen y Eric Zimmerman. *Rules of Play: Game Design Fundamentals*. Cambridge, MA: The MIT Press, 2004. ISBN: 978-0-262-24045-1.

[6] Pedro Luis Sánchez Ortega. *Toma el control: breve historia de los videojuegos con la evolución de sus mandos*. Material docente. Universidad de Burgos, 2023. URL: http://hdl.handle.net/10259/9588 (visitado 08-09-2026).

CAPÍTULO 2

Fundamentos de electrónica para sistemas interactivos

En el capítulo 1 vimos que un sistema de adquisición y control convierte acciones físicas en datos y datos en efectos físicos. Todas esas conversiones ocurren con electricidad: por los cables de un mando circulan corrientes minúsculas que representan la posición de un *stick* o la orden de hacer vibrar un motor. Este capítulo reúne el mínimo de electricidad y electrónica necesario para entender los sensores (capítulo 3), la conversión de señales (capítulo 4) y los microcontroladores (capítulo 5).

Objetivos de aprendizaje

- Manejar las magnitudes eléctricas básicas: corriente, tensión, resistencia, potencia y energía, con sus unidades y sus órdenes de magnitud.
- Aplicar la ley de Ohm y el concepto de divisor de tensión a casos sencillos.
- Reconocer los componentes electrónicos básicos y decir para qué sirve cada uno.
- Explicar los niveles lógicos y el papel de las resistencias *pull-up* y *pull-down* en una entrada digital.
- Aplicar las reglas básicas de seguridad al conectar circuitos de baja tensión.

Alcance

Se tratan las magnitudes eléctricas y los circuitos sencillos con un enfoque cualitativo. La única fórmula que se usa con soltura es la ley de Ohm; no se analizan circuitos con varias mallas, ni régimen transitorio, ni componentes en detalle. El objetivo es *entender de qué se habla* para poder elegir componentes y leer un esquema, no diseñar electrónica.

ÍNDICE DEL CAPÍTULO

2.1 MAGNITUDES ELÉCTRICAS FUNDAMENTALES

Cuatro magnitudes bastan para casi todo lo que haremos: corriente, tensión, resistencia y potencia. Una analogía ayuda a ordenarlas: un circuito eléctrico se parece a un circuito de agua. La *tensión* es la diferencia de presión que empuja el agua; la *corriente* es el caudal que circula; la *resistencia* es lo estrecha que es la tubería; la *potencia* es el trabajo que el agua puede hacer al moverse. La analogía tiene límites —la electricidad no «se derrama»—, pero sirve para fijar las ideas [4].

2.1.1 Carga y corriente

La materia tiene una propiedad llamada *carga eléctrica*, que llevan sobre todo los electrones. Cuando esas cargas se mueven de forma ordenada por un conductor, hay *corriente eléctrica*. La corriente se mide en **amperios** (A); en la electrónica de un mando o de una placa Arduino se trabaja con milésimas de amperio, los **miliamperios** (mA).

Conviene tener en la cabeza algunos órdenes de magnitud:

- un LED indicador consume del orden de 10 mA;
- un motor de vibración pequeño, entre 60 a 100 mA;
- un pin de un microcontrolador puede entregar como mucho unos 20 a 40 mA sin dañarse;
- un puerto USB 2.0 proporciona hasta 500 mA.

Por convenio, se dibuja la corriente saliendo del polo positivo de la fuente, aunque los electrones se muevan en sentido contrario; para lo que nos ocupa, ese detalle no cambia nada.

2.1.2 Tensión

La *tensión eléctrica* (o *voltaje*) es la diferencia de potencial entre dos puntos: el «empuje» que hace circular la corriente. Se mide en **voltios** (V) y **siempre se mide entre dos puntos**: hablar de «la tensión en un punto» solo tiene sentido si se sobreentiende respecto a qué otro punto —normalmente, la masa (sección 2.3)—.

Órdenes de magnitud habituales: una pila AA da 1,5 V; una celda de batería de litio, unos 3,7 V; la electrónica de una placa Arduino funciona a 5 V o a 3,3 V; un puerto USB entrega 5 V.

2.1.3 Resistencia

La *resistencia eléctrica* es la oposición al paso de la corriente. Se mide en **ohmios** (Ω); en la práctica se usan mucho los **kiloohmios** (kΩ). Según cómo se opongan al paso de la corriente, los materiales son *conductores* (la dejan pasar: metales), *aislantes* (la bloquean: plástico, aire) o *semiconductores* (un término medio que se puede controlar; son la base del diodo y el transistor).

Como componente, una resistencia se usa sobre todo para dos cosas: **limitar** la corriente que pasa por otro componente (por ejemplo, proteger un LED) y **fijar** la tensión de un punto del circuito (los divisores y los *pull-up* de más adelante).

2.1.4 Potencia y energía

La *potencia eléctrica* es el ritmo al que un circuito consume o entrega energía. Se calcula como el producto de la tensión por la corriente y se mide en **vatios** (W). En un mando se manejan décimas de vatio.

La *energía* es la potencia acumulada a lo largo del tiempo. En las baterías se expresa como *capacidad*, en miliamperios-hora (mA h): una batería de 500 mA h puede entregar 500 mA durante una hora, o 250 mA durante dos. De ahí sale la autonomía de un dispositivo, que se trata en el capítulo 8.

La potencia importa por tres motivos prácticos: el **calor** (un componente que disipa mucha potencia se calienta), el **consumo** (fija la autonomía) y los **límites** (cada componente y cada pin aguantan una corriente y una potencia máximas).

2.2 LEY DE OHM Y CIRCUITOS BÁSICOS

2.2.1 Ley de Ohm

Definición 2.1: ley de Ohm

En un componente resistivo, la tensión entre sus extremos es igual a la corriente que lo atraviesa multiplicada por su resistencia:

$$V = I \cdot R.$$

Es la única fórmula que usaremos con frecuencia, y se lee de forma intuitiva: para una resistencia dada, **más tensión implica más corriente**; para una tensión dada, **más resistencia implica menos corriente**. Despejando se obtienen las otras dos formas, $I = V/R$ y $R = V/I$.

Ejemplo 2.1

Resistencia para un LED. Quieres encender un LED desde una salida de 5 V. El LED «consume» por sí mismo unos 2 V y quieres que lo atraviesen 10 mA. La resistencia en serie debe absorber los 3 V restantes:

$$R = \frac{V}{I} = \frac{3\,\text{V}}{0{,}01\,\text{A}} = 300\,\Omega.$$

Como 300 Ω no es un valor comercial, se usa el más cercano, 330 Ω. Es el cálculo eléctrico más repetido en todo el libro.

En la teoría no resolveremos circuitos: basta con entender qué dice la ley de Ohm y cuándo aplicarla —por ejemplo, para elegir la resistencia que acompaña a un LED.

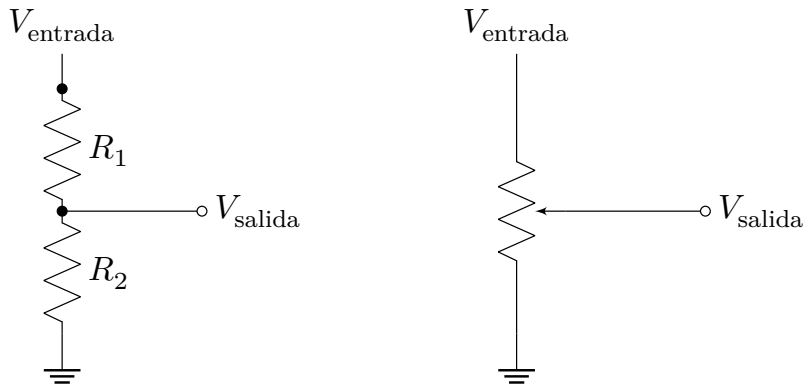
2.2.2 Circuitos serie y paralelo

Dos componentes están **en serie** cuando la corriente solo tiene un camino y pasa por los dos, uno detrás de otro. En serie, la corriente es la misma en todos los componentes y la tensión total *se reparte* entre ellos. Están **en paralelo** cuando comparten sus dos extremos y la corriente se divide entre ellos: en paralelo, todos tienen la misma tensión y las corrientes *se suman*.

En un mando, casi todos los subsistemas cuelgan **en paralelo** de la misma línea de alimentación: por eso la batería «ve» la suma de todos los consumos a la vez, y encender la iluminación o la vibración reduce la autonomía.

2.2.3 Divisores de tensión

Si se conectan dos resistencias en serie entre una tensión y la masa, el punto intermedio queda a una tensión que es una *fracción* de la de entrada, fijada por la proporción entre las dos resistencias (figura 2.1 a). Eso es un *divisor de tensión*.



(a) resistencias fijas: salida fija

(b) potenciómetro: salida ajustable con el cursor

Figura 2.1. Un divisor de tensión entrega en su salida una fracción de la tensión de entrada. Con resistencias fijas (a) esa fracción es fija; con un potenciómetro (b) el cursor la hace variable, y así funcionan un *stick* y un gatillo analógicos.

El divisor aparece por todas partes:

- Un **potenciómetro** es un divisor *ajustable*: al girar su cursor cambia la proporción y, con ella, la tensión de salida (figura 2.1 b). Así es como un *stick* o un gatillo analógicos convierten una posición en una tensión variable (capítulo 3).
- También sirve para **adaptar niveles**: bajar la tensión de un sensor de 5 V para que la pueda leer una entrada de 3,3 V (capítulo 4).

2.3 COMPONENTES ELECTRÓNICOS Y ALIMENTACIÓN

2.3.1 Componentes básicos

Con muy pocos tipos de componente se explica casi todo el hardware de interacción. Lo que hace falta recordar de cada uno es su *función*, no su física interna [5].

Tabla 2.1. Los componentes básicos y su papel en el hardware de un mando.

Componente	Qué hace	Dónde aparece
Resistencia	Limita la corriente; fija niveles de tensión	LED, *pull-up*, divisores
Condensador	Almacena carga; estabiliza y filtra	Alimentación de cada chip; filtros
Diodo	Deja pasar la corriente en un solo sentido	Protección; un LED es un diodo que emite luz
Transistor	Interruptor o amplificador controlado	Encender motores o LEDs desde el microcontrolador
Circuito integrado	Miles de transistores en una pastilla	El microcontrolador (capítulo 5)

El *transistor* merece una nota: es lo que permite que un microcontrolador, que solo puede dar unos pocos miliamperios, *gobierne* un motor de vibración que necesita muchos más. El microcontrolador controla el transistor con una señal minúscula, y

el transistor deja pasar la corriente grande desde la alimentación (capítulo 7).

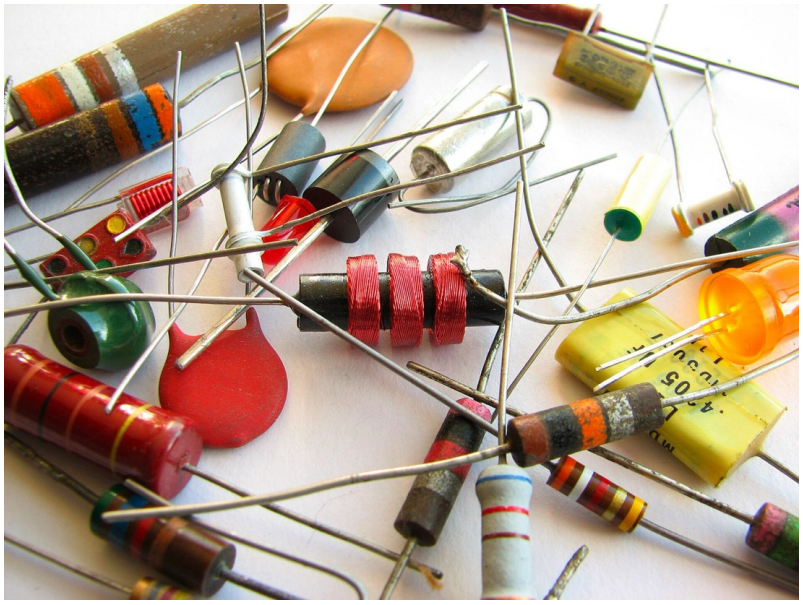


Figura 2.2. Los componentes básicos: resistencias (con bandas de color), condensadores, un diodo, LEDs y un transistor de tres patas. Con estos pocos tipos se explica casi todo el hardware de interacción. *(foto: Windell Oskay · CC BY 2.0)*

2.3.2 Alimentación de circuitos

Toda la electrónica necesita una **tensión de alimentación estable** —típicamente 3,3 V o 5 V— y una **masa común**.

> **Definición 2.2: masa (GND)**
>
> Punto del circuito que se toma como referencia de 0 V y respecto al cual se miden todas las demás tensiones. Se representa con el símbolo de tierra y se abrevia GND.

La masa común es imprescindible: si dos dispositivos se comunican pero no comparten su masa, sus «cero voltios» no coinciden y ninguna señal se interpreta bien (capítulo 9) [2].

Las fuentes de energía —pilas, baterías recargables, USB, adaptador de red— se estudian en el capítulo 8. Entre la fuente y el circuito suele haber un *regulador de tensión*, que convierte una entrada variable (una batería que se va descargando) en

una salida fija. Y junto a cada chip se coloca un *condensador* de desacoplo, para que los picos de consumo no «hundan» momentáneamente la alimentación.

2.4 ELECTRÓNICA ANALÓGICA Y DIGITAL

En el capítulo 1 distinguimos señal analógica y digital por su forma. Ahora la diferencia se ve en las tensiones.

2.4.1 Niveles lógicos

En un circuito digital, «0» y «1» no son tensiones exactas, sino *niveles lógicos*: rangos. En un sistema de 3,3 V, una tensión cercana a 0 V se lee como «0», una cercana a 3,3 V como «1», y entre ambas hay una franja «prohibida» que no debería usarse. Precisamente porque solo hay que distinguir dos rangos bien separados, lo digital **tolera el ruido**: un pequeño temblor de la tensión no convierte un «0» en un «1».

Esto tiene una consecuencia práctica: dos dispositivos que trabajan a tensiones distintas —uno a 5 V y otro a 3,3 V— pueden no entenderse, o incluso dañarse. Hay que **adaptar los niveles**, con un divisor de tensión o con un componente específico llamado convertidor de nivel (capítulos 4 y 9) [3].

2.4.2 Entradas digitales, *pull-up* y *pull-down*

Un pin configurado como **entrada digital** se limita a comprobar si la tensión que hay en él corresponde a un «0» o a un «1». El problema aparece cuando el pin no está conectado a nada: entonces *flota* y lee valores al azar, según la interferencia que capte.

La solución es una resistencia que «ate» el pin a un nivel conocido mientras nada lo cambie. Una resistencia pull-up lo conecta a la alimentación (lo mantiene en «1»); una pull-down lo conecta a masa (lo mantiene en «0»).

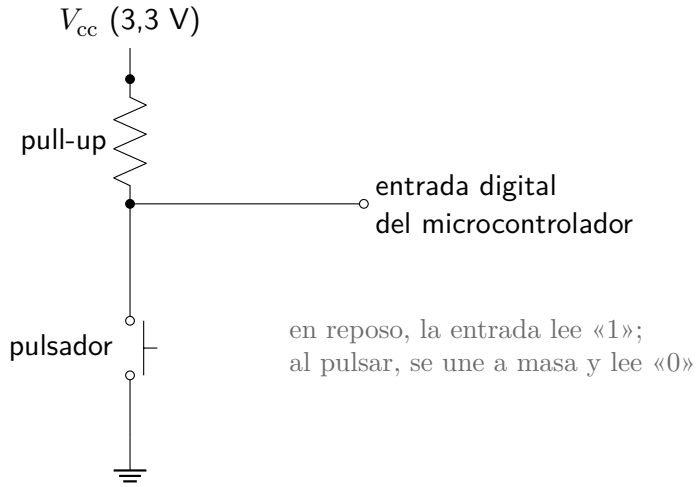


Figura 2.3. Un pulsador con resistencia *pull-up*. Mientras el botón está sin pulsar, la resistencia mantiene la entrada en «1»; al pulsar, el pin se conecta directamente a masa y la entrada pasa a «0».

Ejemplo 2.2

Botón con *pull-up* (figura 2.3). En reposo, la resistencia *pull-up* conecta la entrada a 3,3 V: el microcontrolador lee «1». Al pulsar, el botón une la entrada a masa: ahora lee «0». La lógica queda «al revés» (pulsado = 0), algo muy habitual. La mayoría de microcontroladores llevan *pull-up* internos que se activan por software (en Arduino, el modo `INPUT_PULLUP`), lo que ahorra la resistencia externa.

2.4.3 Analógico frente a digital, en tensiones

La diferencia esencial es esta: en una entrada **analógica** importa *el valor concreto* de la tensión —y por eso el ruido la degrada—; en una entrada **digital** solo importa *a qué lado del umbral* está la tensión, lo que la hace robusta pero le impide representar matices. Convertir de una a otra —muestrear y cuantificar— es el tema del capítulo 4.

2.5 SEGURIDAD Y BUENAS PRÁCTICAS DE CONEXIÓN

Las tensiones de este libro (3,3 a 5 V) no son peligrosas para las personas, pero sí para los componentes y, sobre todo, para las baterías de litio.

Aviso

- **No superes la tensión de alimentación de cada dispositivo.** Aplicar 5 V a un pin de 3,3 V lo puede destruir.
- **No pidas demasiada corriente a un pin.** Un LED necesita su resistencia; un motor necesita transistor o *driver* y su propia alimentación (capítulo 7).
- **No cortocircuites la alimentación** (unir + y − directamente): se calienta, y una batería de litio puede incendiarse.
- **Masa común** entre todos los dispositivos que se conecten entre sí.
- **Desconecta la alimentación antes de recablear** y revisa el montaje antes de volver a darle corriente.

RESUMEN

La electricidad se describe con cuatro magnitudes: corriente (flujo de carga, en amperios), tensión (el empuje, en voltios, siempre entre dos puntos), resistencia (la oposición, en ohmios) y potencia (el ritmo de consumo de energía, en vatios). La ley de Ohm, $V = I \cdot R$, las relaciona y basta para los cálculos del curso, como dimensionar la resistencia de un LED. En serie la corriente es común y la tensión se reparte; en paralelo la tensión es común y las corrientes se suman. Un divisor de tensión entrega una fracción de la tensión de entrada, y un potenciómetro es un divisor ajustable: así un *stick* traduce posición en tensión. Con pocos componentes —resistencia, condensador, diodo, transistor y circuito integrado— se explica casi todo el hardware de interacción. Todo circuito necesita alimentación estable y masa común. En digital, «0» y «1» son rangos de tensión, lo que da robustez frente al ruido pero obliga a adaptar dispositivos de tensiones distintas; una entrada digital sin conectar «flota», y una resistencia *pull-up* o *pull-down* la fija en un nivel conocido. Al conectar circuitos, respeta las tensiones, las corrientes y la masa común, y trata

las baterías de litio con cuidado.

Ideas clave

- Tensión entre dos puntos, corriente a través de un componente, resistencia que se opone: $V = I \cdot R$ los une.
- La potencia manda en el calor, el consumo y los límites de cada componente.
- Un potenciómetro es un divisor de tensión ajustable: convierte posición en tensión.
- El transistor deja que una señal pequeña gobierne una corriente grande.
- Una entrada digital sin fijar «flota»; el *pull-up* o el *pull-down* la atan a un nivel conocido.

TÉRMINOS DEL CAPÍTULO

Corriente eléctrica, tensión eléctrica, resistencia eléctrica, potencia eléctrica, energía y capacidad (mA h), ley de Ohm, circuito serie, circuito paralelo, divisor de tensión, potenciómetro, condensador, diodo, transistor, circuito integrado, regulador de tensión, masa (GND), nivel lógico, entrada digital, resistencia *pull-up*, resistencia *pull-down*.

PARA SABER MÁS

- Platt [4] enseña electrónica desde cero con montajes, sin matemática; muy adecuado para este nivel.
- Scherz y Monk [5] es un manual de referencia amplio para consultar componentes y circuitos concretos.
- Horowitz e Hill [1], como obra de consulta avanzada cuando se quiera profundizar en un tema puntual.

BIBLIOGRAFÍA DEL CAPÍTULO

[1] Paul Horowitz y Winfield Hill. *The Art of Electronics*. 3.ª ed. Cambridge: Cambridge University Press, 2015. ISBN: 978-0-521-80926-9.

[2] Simon Monk. *Electronics Cookbook: Practical Electronic Recipes with Arduino and Raspberry Pi*. Sebastopol, CA: O'Reilly Media, 2017. ISBN: 978-1-491-95340-2.

[3] Ramon Pallàs-Areny y John G. Webster. *Sensors and Signal Conditioning*. 2.ª ed. New York: John Wiley & Sons, 2001. ISBN: 978-0-471-33232-9.

[4] Charles Platt. *Make: Electronics: Learning by Discovery*. 3.ª ed. San Francisco, CA: Make Community, 2021. ISBN: 978-1-68045-687-4.

[5] Paul Scherz y Simon Monk. *Practical Electronics for Inventors*. 4.ª ed. New York: McGraw-Hill Education, 2016. ISBN: 978-1-259-58754-2.

CAPÍTULO 3

Sensores y adquisición de señales

El sensor es la puerta de entrada de la cadena que vimos en el capítulo 1: el punto donde una magnitud del mundo —una posición, una fuerza, un nivel de luz— se convierte en una señal eléctrica. Este capítulo recorre las familias de sensores más útiles para construir interfaces de juego y explica las características que hay que mirar para elegir uno.

Objetivos de aprendizaje

- Explicar qué es un sensor y qué es un transductor, y clasificarlos por magnitud, principio y tipo de salida.
- Relacionar cada familia de sensores con la magnitud que mide y la señal que entrega.
- Interpretar las características de un sensor: rango, sensibilidad, resolución, exactitud, precisión y tiempo de respuesta.
- Explicar para qué sirve la calibración y qué es la deriva.

Alcance

Se estudian las familias de sensores y sus características con un enfoque práctico: qué mide cada uno, qué señal entrega y qué esperar de su hoja de datos. No se analiza la física interna de cada sensor. El *dispositivo* de entrada completo (un mando, un ratón) y su relación con el diseño de juego se tratan en el capítulo 6.

ÍNDICE DEL CAPÍTULO

3.1 SENSORES Y TRANSDUCTORES

3.1.1 Sensores y transductores

Un *transductor* convierte energía de una forma en otra; un *sensor* es un transductor de entrada, que traduce una magnitud física en una señal eléctrica medible. Es la primera etapa de la adquisición.

Conviene distinguir el *elemento sensor* (la pieza que responde a la magnitud) del *módulo sensor* que se compra: muchos módulos ya incluyen la electrónica de acondicionamiento (capítulo 4) e incluso un pequeño procesador que entrega el valor ya digitalizado. Cuanto más trabajo hace el módulo, menos hay que hacer después [5].

3.1.2 Clasificación de sensores

Los sensores se ordenan de varias maneras a la vez:

- **Por la magnitud que miden**: posición, fuerza, luz, temperatura, sonido, proximidad, movimiento, magnitudes del cuerpo…
- **Por el principio físico**: resistivo, capacitivo, magnético (efecto Hall), óptico, piezoeléctrico, inercial (MEMS)…
- **Por si necesitan alimentación**: *activos* (necesitan energía para funcionar; son la mayoría) o *pasivos* (generan ellos mismos una pequeña señal, como un piezoeléctrico).
- **Por el contacto**: con contacto (un potenciómetro) o sin contacto (un sensor Hall, un ultrasónico).

3.1.3 Sensores analógicos y digitales

Según cómo entregan la información, hay tres casos (tabla 3.1):

Tabla 3.1. Las tres formas en que un sensor entrega su medida y cómo se lee cada una.

Tipo de salida	Qué entrega	Cómo se lee
Analógica	Una tensión continua	Se digitaliza con un ADC
Digital simple	Un nivel «sí / no»	Se lee como una entrada digital
Digital por bus	El valor ya digitalizado	Se lee por bus (I^2C o SPI)

El potenciómetro de un *stick* es analógico; un final de carrera es digital simple; una IMU moderna es digital por bus. La conversión analógico-digital se ve en el capítulo 4; las entradas digitales, en el capítulo 2; los buses, en el capítulo 9.

Ante un sensor nuevo, el primer paso es siempre identificar cuál es su salida (analógica, digital simple o por bus) y cómo se lee [2].

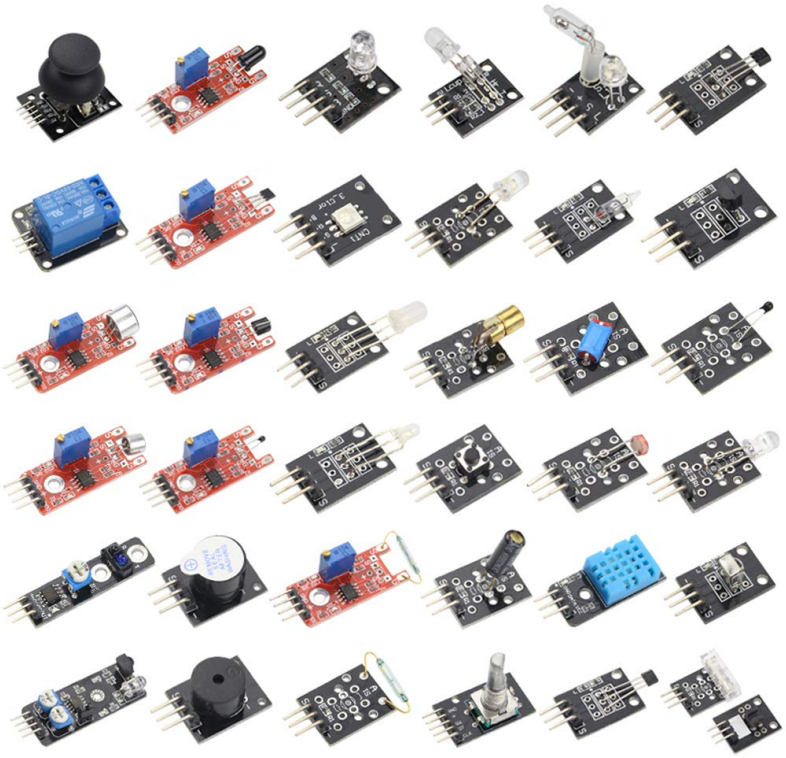


Figura 3.1. Un kit de módulos de sensor: cada placa lleva el elemento sensor, su acondicionamiento y unos pines de conexión. *(foto: Sjilla · CC BY-SA 4.0)*

3.2 FAMILIAS DE SENSORES

3.2.1 Posición y desplazamiento

- *Potenciómetro* (resistivo): un divisor de tensión ajustable (capítulo 2); la tensión de salida es proporcional a la posición o al giro. Es lo que hay dentro de un *stick*, un gatillo o una rueda. Su punto débil es el desgaste mecánico, que con el tiempo produce *deriva*.
- *Efecto Hall* (magnético): detecta un imán cercano, así que mide posición o giro *sin contacto* y sin desgaste. Los *sticks* y gatillos Hall de los mandos recientes lo usan para evitar la deriva.
- Encoder: un disco ranurado y un sensor óptico generan pulsos al girar; contándolos se conoce el movimiento. La rueda de un ratón es un *encoder*.

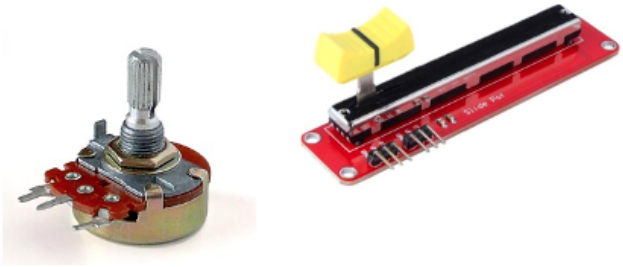


Figura 3.2. Un potenciómetro rotatorio y uno deslizante (*slider*). El eje o el cursor mueven un contacto sobre una pista resistiva; la salida es un divisor de tensión.

3.2.2 Fuerza y presión

- *Resistor sensible a la fuerza* (FSR): su resistencia baja al apretarlo, así que la tensión de salida crece con la fuerza. Botones y superficies sensibles a la presión.
- *Galga extensométrica*: mide una deformación minúscula; es la base de las básculas y de la Wii Balance Board.
- Piezoeléctrico: genera un pico de tensión al recibir un golpe; sirve para detectar impactos o toques, como en los *pads* de una batería electrónica.

Figura 3.3. Un resistor sensible a la fuerza (FSR), plano y redondo, y un sensor de flexión en tira. Ambos cambian de resistencia al deformarse y se leen con un divisor. *(foto: oomlout / Bomazi · CC BY-SA 2.0)*

3.2.3 Luz

- Fotorresistencia (LDR): su resistencia baja con la luz. Barata y lenta.
- Fotodiodo y fototransistor: rápidos; detección de luz, barreras y recepción de infrarrojos. El receptor de un mando a distancia y la cámara infrarroja del Wiimote funcionan así.
- Sensor de imagen: una matriz de fotodetectores. El sensor de un ratón óptico es, en realidad, una cámara diminuta que mira la superficie muchas veces por segundo.

3.2.4 Temperatura, sonido, proximidad

- **Temperatura** (termistor, sensor de silicio): poco habitual como entrada de juego, pero presente para proteger la batería y la electrónica (capítulo 8) y en algunos *wearables*.
- **Sonido** (micrófono *electret* o MEMS): convierte la presión sonora en una señal analógica que se muestrea muy rápido (capítulo 4). Voz, chat, detección de soplido.
- **Proximidad y presencia**: infrarrojo de reflexión (objeto cercano), ultrasónico (distancia por el eco), PIR (detecta el calor de una persona en movimiento) y *capacitivo* (detecta el dedo sin pulsar: botones táctiles, *touchpad*, pantalla del

móvil).

3.2.5 Movimiento: acelerómetros, giróscopos e IMU

Un *acelerómetro* mide la aceleración; como en reposo «siente» la gravedad, permite conocer la inclinación. Un *giróscopo* mide la velocidad de giro. Ambos son *MEMS*: micromecanismos tallados en silicio, con tres ejes cada uno.

Una *unidad de medida inercial* (IMU) reúne acelerómetro y giróscopo —y a veces un magnetómetro (brújula)— en un solo chip con salida digital por bus. Combinar sus datos (*fusión de sensores*) da una orientación estable; es la base del apuntado por giro y del seguimiento en realidad virtual (capítulos 9 y 10).

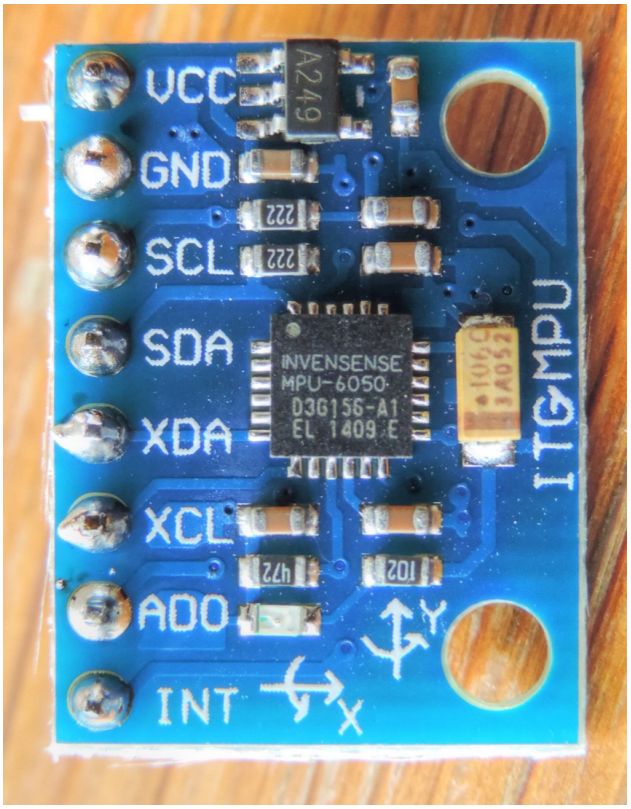


Figura 3.4. Un módulo de unidad de medida inercial (IMU): el acelerómetro y el giróscopo van en el chip pequeño del centro; la placa añade la alimentación y los conectores del bus. *(foto: Nevit Dilmen · CC BY-SA 3.0)*

3.2.6 Sensores biométricos

Miden magnitudes del cuerpo: ritmo cardíaco (un LED y un fotodetector sobre la piel), actividad eléctrica del músculo (EMG) o del cerebro (EEG), conductividad de la piel, temperatura corporal, dirección de la mirada. Son señales pequeñas y ruidosas que necesitan mucho acondicionamiento y filtrado (capítulo 4), y se usan cada vez más en *wearables*, *exergames* y adaptación de dificultad (capítulos 10 y 11).

> **Conexión con el diseño de videojuegos**
>
> Usar el pulso o la respiración como entrada cambia el diseño: un juego de terror puede volverse más agresivo cuanto más tranquilo estás, y un *exergame* puede ajustar la intensidad a tu ritmo cardíaco. A cambio, se manejan datos personales sensibles, con las implicaciones éticas que se ven en el capítulo 11.

Tabla 3.2. Familias de sensores y su presencia en el hardware de juego.

Familia	**Mide**	**Salida típica**	**Ejemplo**
Potenciómetro	posición, giro	analógica	*stick*, gatillo, rueda
Efecto Hall	posición sin contacto	analógica o digital	*stick* Hall
Encoder	giro (pulsos)	digital	rueda del ratón
FSR / piezo	fuerza, impacto	analógica	botón de presión, *pad* de percusión
Fotodiodo	luz	analógica o digital	receptor de infrarrojos
Micrófono	sonido	analógica	voz, soplido
Ultrasónico	distancia	digital (tiempo)	sensor de distancia del kit
Capacitivo	contacto del dedo	digital	*touchpad*, pantalla táctil
Acelerómetro y giróscopo	movimiento y giro	digital por bus	Wiimote, Joy-Con
Biométrico	pulso, músculo, mirada	analógica (débil)	pulsómetro, banda EMG

3.3 CARACTERÍSTICAS DE LOS SENSORES

Elegir un sensor es leer su hoja de características y comprobar que encaja con lo que se necesita. Estas son las magnitudes clave [4].

3.3.1 Rango, sensibilidad y resolución

El *rango de medida* es el intervalo de la magnitud física que el sensor puede medir; por encima o por debajo, el sensor *satura* y ya no distingue nada (figura 3.5). La *sensibilidad* es cuánto cambia la salida por cada unidad de cambio de la magnitud —la pendiente de la curva—: mucha sensibilidad hace que se noten los cambios pequeños, pero también el ruido. La *resolución* es el cambio más pequeño que el sensor puede distinguir; la fijan el ruido y, tras digitalizar, el número de bits del ADC (capítulo 4).

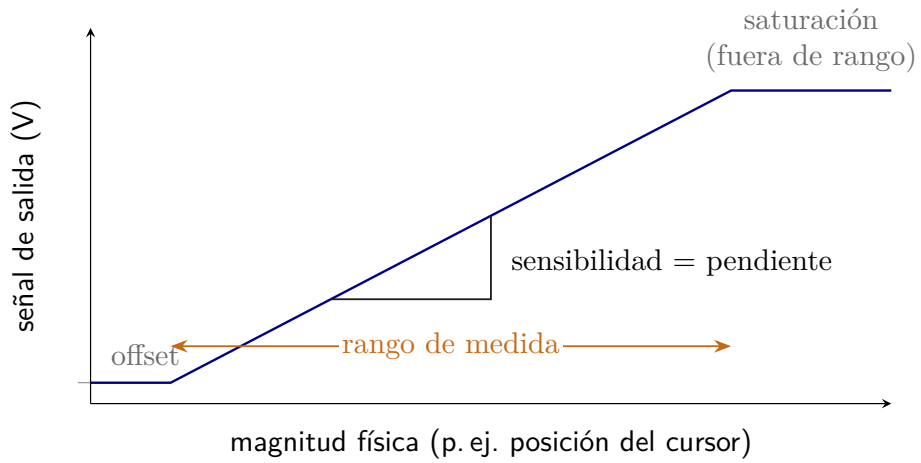


Figura 3.5. Curva de respuesta típica de un sensor. En el tramo útil (el rango de medida) la salida sube de forma aproximadamente lineal; su pendiente es la sensibilidad. Fuera del rango, la salida se aplana: el sensor satura.

Una curva que no sea una recta indica falta de *linealidad*: la relación entre magnitud y salida no es constante y hay que corregirla por software.

3.3.2 Exactitud, precisión y repetibilidad

Son tres cosas distintas (figura 3.6):

- *Exactitud*: cómo de cerca está la medida del valor real.
- *Precisión*: cómo de parecidas son entre sí varias medidas de lo mismo, aunque estén todas desviadas.
- *Repetibilidad*: dar la misma lectura ante la misma entrada en medidas sucesivas.

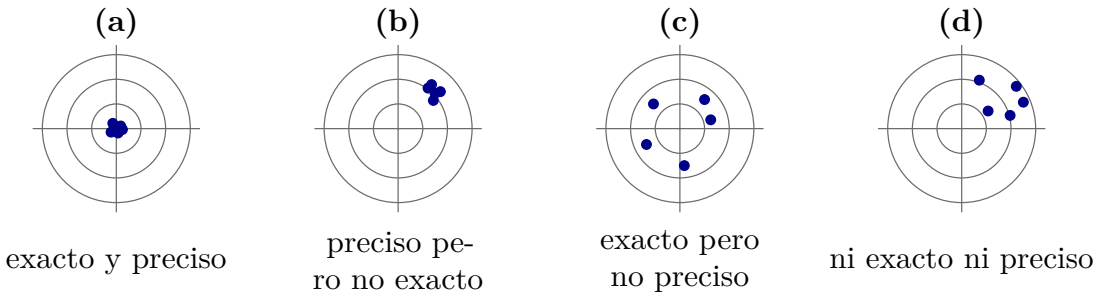


Figura 3.6. Exactitud frente a precisión. Un sensor puede ser preciso (repetible) sin ser exacto, y al revés. En un mando suele importar más la precisión y la baja latencia que la exactitud absoluta.

En un videojuego, muchas veces importa más la precisión y la rapidez que la exactitud: da igual el ángulo real del *stick* mientras la lectura sea consistente y responda al instante.

3.3.3 Tiempo de respuesta, deriva y ruido

El *tiempo de respuesta* es lo que tarda la salida en reflejar un cambio brusco de la magnitud. Un sensor lento añade latencia (capítulo 1), algo crítico en tiempo real.

La *deriva* es un cambio lento de la lectura con el tiempo, la temperatura o el desgaste, aunque la magnitud no cambie; el *offset* es un desvío fijo respecto al cero. El conocido *stick drift* es deriva de un potenciómetro gastado. El *ruido* es la variación aleatoria de la salida, y es lo que marca el límite real de resolución.

También figuran en la hoja de datos el **consumo** y la **tensión de alimentación**,

necesarios para el presupuesto de energía (capítulo 8).

3.3.4 Calibración

Calibrar es ajustar la relación entre la señal del sensor y la magnitud real, comparándola con una referencia conocida. Puede hacerse en fábrica, en el arranque del dispositivo o a mano por el usuario.

Ejemplo 3.1

Calibración de un *stick*. Al encender el mando, el firmware toma la lectura del *stick* en reposo y la guarda como «centro»; al mover el *stick* a los topes, registra los valores máximo y mínimo. Después, toda lectura se reescala a ese recorrido real y se aplica una *zona muerta* alrededor del centro para que la deriva no mueva al personaje solo.

RESUMEN

Un sensor traduce una magnitud física en una señal eléctrica; es la primera etapa de la adquisición. Se clasifica por lo que mide, por su principio físico, por si necesita alimentación y por si trabaja con contacto. Según su salida, hay sensores analógicos (una tensión continua que hay que digitalizar), digitales simples (un «sí/no») y digitales por bus (el valor ya digitalizado). Las familias más útiles para interfaces de juego son las de posición (potenciómetro, efecto Hall, *encoder*), fuerza (FSR, piezoeléctrico), luz (fotodiodo, sensor de imagen), sonido (micrófono), proximidad (infrarrojo, ultrasónico, capacitivo), movimiento (acelerómetro, giróscopo e IMU) y biométrica. Para elegir un sensor se leen sus características: rango (fuera de él, satura), sensibilidad (la pendiente), resolución (el cambio más pequeño que distingue), linealidad, exactitud (acierto), precisión y repetibilidad (consistencia), tiempo de respuesta (latencia que añade), deriva y ruido. Calibrar es ajustar la relación señal-magnitud con una referencia, y es lo que hace un mando con el centro y los topes de su *stick*.

Ideas clave

- Un sensor convierte una magnitud del mundo en una señal eléctrica: es la puerta de entrada de la cadena.
- Tres tipos de salida: analógica (a digitalizar), digital simple y digital por bus.
- Exactitud es acertar; precisión es ser consistente. En un juego suele pesar más la segunda, junto con la baja latencia.
- Todo sensor tiene rango, y fuera de él satura; y tiene ruido, que limita su resolución real.
- La calibración ajusta la lectura a la realidad; la deriva la va estropeando con el tiempo.

TÉRMINOS DEL CAPÍTULO

Sensor, transductor, sensor activo y pasivo, salida analógica, salida digital, salida por bus, potenciómetro, efecto Hall, *encoder*, resistor sensible a la fuerza (FSR), piezoeléctrico, fotodiodo, micrófono MEMS, sensor de proximidad, sensor capacitivo, acelerómetro, giróscopo, unidad de medida inercial (IMU), fusión de sensores, sensor biométrico, rango de medida, saturación, sensibilidad, resolución, linealidad, exactitud, precisión, repetibilidad, tiempo de respuesta, deriva, *offset*, ruido, calibración, zona muerta.

PARA SABER MÁS

- Platt [5] es un catálogo práctico de sensores, orientado a montar cosas, con el nivel de este curso.
- Fraden [1], como obra de consulta cuando interese el principio físico de un sensor concreto.
- Morris y Langari [3], para profundizar en las características de medida (exactitud, incertidumbre, calibración).

BIBLIOGRAFÍA DEL CAPÍTULO

[1] Jacob Fraden. *Handbook of Modern Sensors: Physics, Designs, and Applications*. 5.ª ed. Cham: Springer, 2016. ISBN: 978-3-319-19302-1.

[2] Simon Monk. *Electronics Cookbook: Practical Electronic Recipes with Arduino and Raspberry Pi*. Sebastopol, CA: O'Reilly Media, 2017. ISBN: 978-1-491-95340-2.

[3] Alan S. Morris y Reza Langari. *Measurement and Instrumentation: Theory and Application*. 2.ª ed. London: Academic Press, 2016. ISBN: 978-0-12-800884-3.

[4] Ramon Pallàs-Areny y John G. Webster. *Sensors and Signal Conditioning*. 2.ª ed. New York: John Wiley & Sons, 2001. ISBN: 978-0-471-33232-9.

[5] Charles Platt. *Encyclopedia of Electronic Components, Volume 3: Sensors*. San Francisco, CA: Maker Media, 2016. ISBN: 978-1-4493-3431-1.

CAPÍTULO 4

Acondicionamiento, conversión y procesamiento de señales

En el capítulo 3 un sensor entregaba una señal eléctrica. Esa señal casi nunca está lista para usarse: hay que *prepararla*, *convertirla* en números y *procesarla*. Este capítulo es el corazón técnico de la adquisición: explica cómo una tensión que varía se transforma en la secuencia de números que un juego puede leer, y cómo esos números vuelven a ser una señal capaz de mover algo.

Objetivos de aprendizaje

- Explicar por qué una señal necesita acondicionamiento y qué operaciones lo componen.
- Describir el muestreo y la cuantificación, y qué significa la resolución en bits.
- Aplicar el criterio de Nyquist y reconocer el aliasing.
- Explicar qué hacen un ADC, un DAC y la generación de señal por PWM.
- Situar el filtrado digital, la latencia y la frecuencia de actualización en la cadena.

Alcance

Se tratan muestreo, cuantificación, conversión, ruido, filtrado y generación de señales de forma conceptual. No hay transformadas ni diseño de filtros: el criterio de Nyquist se usa como regla («más del doble») y la resolución, con potencias de dos.

ÍNDICE DEL CAPÍTULO

4.1 SEÑALES Y SU ACONDICIONAMIENTO

4.1.1 Tipos y características de las señales

Recordando el capítulo 1, una señal puede ser analógica (varía de forma continua) o digital (toma pocos valores). De una señal analógica interesan cuatro rasgos: su **amplitud** (cómo de grande es), su **nivel de reposo**, su **rapidez de cambio** (con qué velocidad sube y baja) y su **ruido**. La rapidez de cambio es la que decide cada cuánto hay que mirarla: la posición de un *stick* cambia despacio, pero el sonido de un micrófono cambia miles de veces por segundo.

4.1.2 Acondicionamiento de señal

> **Definición 4.1: acondicionamiento de señal**
>
> Conjunto de operaciones que preparan la señal de un sensor para que el convertidor pueda medirla bien: adaptar su nivel y su rango, amplificarla, filtrarla y protegerla.

Hace falta acondicionar porque la señal del sensor suele ser demasiado pequeña, estar desplazada respecto al cero, llevar ruido, o moverse en un rango que no coincide con el de la entrada del convertidor [3]. El acondicionamiento es una etapa de la cadena (figura 1.8), entre el sensor y el ADC.

4.1.3 Adaptación de niveles y de rango

A veces basta con **encajar** la señal en la ventana del convertidor. Si un sensor da 0 a 5 V y la entrada admite 0 a 3,3 V, se baja con un divisor de tensión (capítulo 2); en el caso contrario se usa un convertidor de nivel. También se ajusta el *rango*: que el recorrido útil de la señal ocupe todo el intervalo del ADC y no solo un trozo.

> **Aviso**
>
> Aplicar a una entrada más tensión de la que admite la daña. Antes de conectar un sensor a un convertidor, comprueba su tensión de salida máxima.

4.1.4 Amplificación

Las señales muy pequeñas —un micrófono, un sensor biométrico: milivoltios— hay que *amplificarlas* antes de digitalizarlas. Un amplificador multiplica la señal por una *ganancia*; así se aprovecha todo el rango del ADC y el ruido que se añada después pesa mucho menos en comparación. En teoría lo vemos como una «caja que multiplica»; el circuito no nos ocupa.

4.1.5 Divisores de tensión como acondicionadores

El divisor de tensión (capítulo 2) no solo atenúa: es la forma habitual de *leer un sensor resistivo*. Una fotorresistencia, un FSR o un termistor, puestos en serie con una resistencia fija, forman un divisor cuya salida es una tensión que varía con la magnitud medida.

> **Ejemplo 4.1**
>
> **Leer una fotorresistencia (LDR).** Se conecta la LDR entre 3,3 V y un punto intermedio, y una resistencia fija entre ese punto y masa. Con mucha luz, la LDR tiene poca resistencia y el punto intermedio sube; a oscuras, baja. Esa tensión se lleva al ADC y se obtiene un número proporcional a la luz.

4.1.6 Filtrado analógico

Un *filtro* deja pasar unas frecuencias y bloquea otras. El más usado es el **paso bajo**: deja pasar lo lento y atenúa lo rápido, es decir, el ruido de alta frecuencia. Se construye con una resistencia y un condensador (capítulo 2); aquí interesa *qué consigue*, no cómo se calcula.

Su uso más importante viene a continuación: colocado **antes del ADC**, impide que entren frecuencias demasiado altas para la velocidad de muestreo y evita el aliasing (sección 4.2). Por eso se le llama *filtro antialiasing* y nunca falta en una cadena de adquisición bien hecha.

4.1.7 Ruido e interferencias

El *ruido* es una variación aleatoria que se suma a la señal; una *interferencia* es contaminación de una fuente concreta —la red eléctrica, la radio del propio mando, los motores de vibración, la electrónica digital conmutando—. Se combaten con cables cortos y apantallados, una masa común bien hecha, separando la parte analógica de la digital, filtrando y promediando (sección 4.4). La relación entre la señal y el ruido decide cuántos bits del ADC son de verdad aprovechables: no sirve de nada un convertidor de 16 bit si el ruido ya tapa los de abajo.

4.2 DIGITALIZACIÓN DE LA SEÑAL

Convertir una señal analógica en números tiene dos ejes independientes: el *tiempo* (muestreo) y la *amplitud* (cuantificación) (figura 4.1).

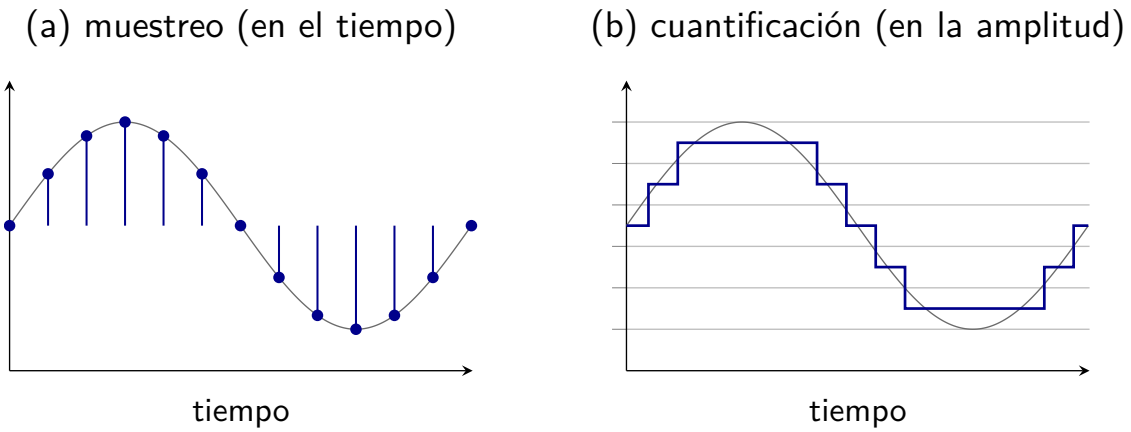


Figura 4.1. Los dos ejes de la conversión analógico-digital. (a) El muestreo toma el valor de la señal a intervalos regulares de tiempo. (b) La cuantificación redondea cada valor al nivel disponible más cercano.

4.2.1 Muestreo

Muestrear es tomar el valor de la señal a intervalos regulares y quedarse con esa secuencia de números. La *frecuencia de muestreo* (en Hz) dice cuántas muestras se toman por segundo; el intervalo entre muestras es su inversa. Para un *stick* bastan

unos cientos de muestras por segundo; para la voz, unas 16 000; para música, 44 100.

4.2.2 Frecuencia de muestreo y aliasing

> **Definición 4.2: criterio de Nyquist**
>
> Para no perder información, hay que muestrear a **más del doble** de la frecuencia más alta presente en la señal.

Si se muestrea más despacio, aparece *aliasing*: la señal que se reconstruye es *falsa* y parece más lenta de lo que realmente es (figura 4.2). Es el mismo efecto por el que, en el cine, las ruedas de un coche parecen girar hacia atrás [5].

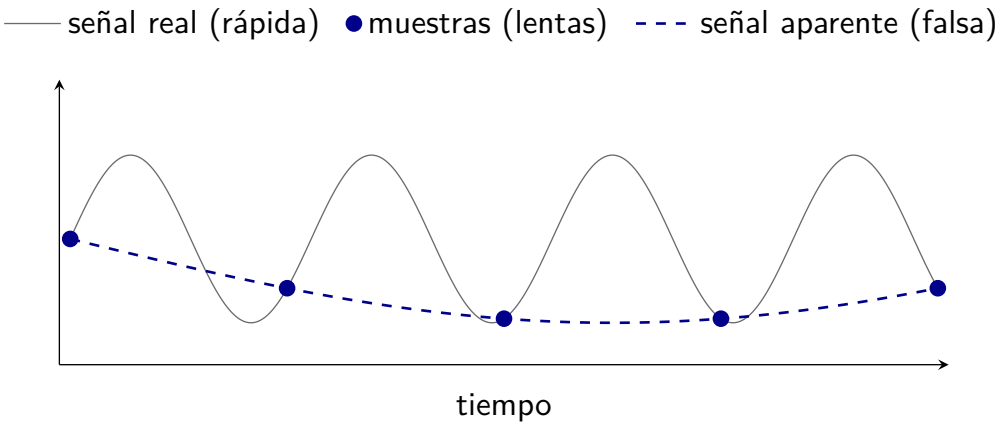


Figura 4.2. Aliasing. La señal real (gris) oscila deprisa, pero se muestrea una vez por período largo; las muestras encajan en una señal aparente mucho más lenta (azul discontinuo), que es la que el sistema «cree» haber medido.

Para evitarlo, el filtro antialiasing (sección 4.1) quita de la señal todo lo que esté por encima de la mitad de la frecuencia de muestreo, *antes* de digitalizar.

4.2.3 Cuantificación y resolución

Cuantificar es redondear cada muestra a uno de un número finito de niveles. La *resolución* se da en **bits**: con n bits hay 2^n niveles.

Bits	Niveles	Uso típico
8	256	PWM, sensores toscos
10	1 024	ADC de un microcontrolador (`analogRead`)
12	4 096	ADC de gama media, *sticks* de precisión
16	65 536	audio

La diferencia entre el valor real y el nivel más cercano es el **error de cuantificación**: más bits, menos error y mayor *rango dinámico* (distancia entre la señal más pequeña y la más grande representables). Que un *stick* vaya «de 0 a 1023» significa que su ADC es de 10 bit.

4.2.4 Conversión analógico-digital (ADC)

Un ADC junta las dos operaciones: entra una tensión, sale un número. De su hoja de datos importan el número de **bits**, la **tensión de referencia** (fija el rango de entrada: el número es proporcional a la tensión de entrada dividida por V_{ref}), la **velocidad** (muestras por segundo) y el número de **canales** (un solo ADC con un multiplexor lee varios sensores por turnos). Suele estar dentro del microcontrolador (capítulo 5); para más calidad —audio— se usa un chip aparte.

Ejemplo 4.2

Un ADC de 12 bit con una tensión de referencia de 3,3 V tiene 4 096 niveles, así que cada paso vale unos 0,8 mV. Una lectura de 2 048 corresponde a unos 1,65 V, la mitad del rango.

Conexión con el diseño de videojuegos

En Arduino, `analogRead()` devuelve un número de 0 a 1 023 (10 bit). Sobre esa lectura de un potenciómetro o un *joystick* se aplican zona muerta y un reescalado (`map()`) para ajustar el recorrido, tal como hace el firmware de un mando (capítulo 3).

4.3 GENERACIÓN DE SEÑALES

4.3.1 Conversión digital-analógico (DAC)

Un DAC recorre el camino inverso: convierte un número en una tensión. Se usa para generar sonido (la voz o los tonos de un mando) y señales de control analógicas. No todos los microcontroladores llevan DAC; muchos imitan la salida analógica con PWM.

4.3.2 PWM como mecanismo de generación de señales

La PWM enciende y apaga la salida muy deprisa. La fracción de tiempo que está encendida —el *ciclo de trabajo*— fija su **valor medio** (figura 4.3). Si la conmutación es lo bastante rápida, un LED, un motor (por su inercia) o un filtro paso bajo «ven» solo esa media.

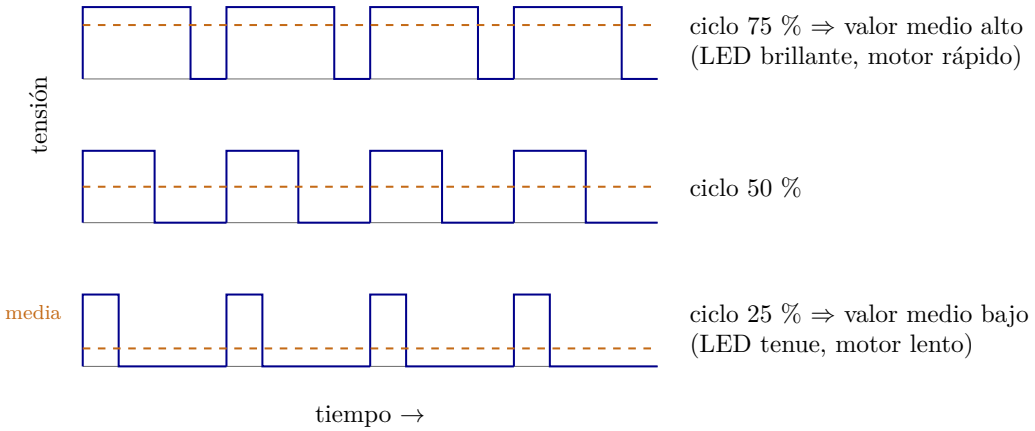


Figura 4.3. PWM. Cuanto mayor es el ciclo de trabajo (fracción de tiempo en nivel alto), mayor es el valor medio de la señal (línea discontinua): más brillo en un LED o más velocidad en un motor.

Se usa para el brillo de un LED, la velocidad de un motor y la intensidad de la vibración (capítulo 7). En Arduino, `analogWrite()` es PWM de 8 bit (0 a 255). La frecuencia de conmutación debe estar por encima de lo perceptible: si es baja, se ve el parpadeo o se oye un zumbido [4].

4.4 PROCESAMIENTO Y TIEMPO REAL

4.4.1 Filtrado digital

Cuando la señal ya son números, se puede filtrar por software, sin componentes.

- *Media móvil*: sustituir cada muestra por la media de las últimas N. Quita ruido y temblor; a cambio, retrasa la señal y suaviza los cambios bruscos.
- **Filtro exponencial**: una media que da más peso a las muestras recientes; parecido, pero más barato de calcular.
- **Antirrebote** (*debounce*): es un filtro digital: ignorar los cambios de un botón que duran menos de unos milisegundos (capítulo 2).
- **Fusión de sensores**: combinar el acelerómetro (estable a largo plazo) y el giróscopo (rápido pero con deriva) de una IMU para obtener una orientación buena (capítulos 3 y 9).

La regla general: más filtrado significa menos ruido, pero también más latencia y menos respuesta [5].

4.4.2 Latencia y frecuencia de actualización

La *frecuencia de actualización* es cada cuánto el sistema repite el ciclo de leer, procesar y responder: la tasa de sondeo del mando (125, 250, 1 000 Hz), los fotogramas por segundo del juego, el refresco de la pantalla.

La *latencia* es la suma de todos los retardos entre la acción y verla en pantalla (capítulo 1). Un desglose típico:

Etapa	**Retardo aprox.**
Lectura del sensor y antirrebote	1 a 3 ms
Conversión y proceso en el microcontrolador	< 1 ms
Comunicación (cable o inalámbrica)	1 a 8 ms
Espera al siguiente fotograma (a 60 fps)	≈ 8 ms
Render y envío a la pantalla	8 a 16 ms
Respuesta del panel	3 a 10 ms
Total	$\approx$ 30 a 60 ms

Además, la latencia no es constante: su variación es el jitter, y molesta más que un retardo alto pero fijo. Se reduce subiendo la tasa de sondeo, con un enlace de baja latencia (capítulo 9), filtrando menos, sincronizando bien con el fotograma y con una pantalla de refresco alto [1]. El filtrado del apartado anterior y la latencia tiran en sentidos opuestos: hay que decidir cuánto suavizar.

RESUMEN

La señal de un sensor se acondiciona (se adapta su nivel y su rango, se amplifica, se filtra y se protege), se convierte en números y se procesa. La conversión analógico-digital tiene dos ejes: el muestreo, que toma valores a intervalos regulares de tiempo, y la cuantificación, que redondea cada valor a uno de 2^n niveles según los bits de resolución. El criterio de Nyquist obliga a muestrear a más del doble de la frecuencia más alta de la señal; si no, aparece aliasing y la señal reconstruida es falsa, por eso se pone un filtro paso bajo antes del ADC. Un ADC entrega un número proporcional a la tensión de entrada respecto a su tensión de referencia. El camino inverso lo hacen un DAC o, más a menudo, PWM, cuya señal tiene un valor medio proporcional al ciclo de trabajo. Ya en software, la media móvil y otros filtros digitales quitan ruido a cambio de añadir latencia. La frecuencia de actualización marca cada cuánto responde el sistema, y la latencia total —suma de todas las etapas, del orden de decenas de milisegundos— junto con su variación (el *jitter*) determinan lo directo que se siente el control.

Ideas clave

- Acondicionar es dejar la señal a la medida del convertidor: nivel, rango, amplitud, ruido.
- Digitalizar tiene dos ejes: muestreo (tiempo) y cuantificación (amplitud, en bits).
- Nyquist: muestrea a más del doble de lo más rápido que haya en la señal, o tendrás aliasing.
- PWM finge una salida analógica: el valor medio sale del ciclo de trabajo.
- Filtrar más reduce el ruido pero añade latencia; hay que elegir el punto medio.

TÉRMINOS DEL CAPÍTULO

Acondicionamiento de señal, adaptación de nivel y de rango, amplificación, ganancia, filtro paso bajo, filtro antialiasing, ruido, interferencia, relación señal-ruido, muestreo, frecuencia de muestreo, criterio de Nyquist, aliasing, cuantificación, resolución en bits, error de cuantificación, rango dinámico, convertidor analógico-digital (ADC), tensión de referencia, multiplexor, convertidor digital-analógico (DAC), PWM, ciclo de trabajo, filtrado digital, media móvil, antirrebote, fusión de sensores, frecuencia de actualización, latencia, *jitter*.

PARA SABER MÁS

- Smith [5] explica el muestreo, la cuantificación y el filtrado con muy poca matemática y muchos ejemplos; disponible en línea.
- Scherz y Monk [4], para los detalles prácticos de ADC, DAC y PWM en circuitos reales.
- Lyons [2], como referencia si más adelante se quiere entrar en el procesado digital con rigor.

BIBLIOGRAFÍA DEL CAPÍTULO

[1] Jason Gregory. *Game Engine Architecture*. 3.ª ed. Boca Raton, FL: CRC Press, 2018. ISBN: 978-1-138-03545-4.

[2] Richard G. Lyons. *Understanding Digital Signal Processing*. 3.ª ed. Upper Saddle River, NJ: Prentice Hall, 2010. ISBN: 978-0-13-702741-5.

[3] Ramon Pallàs-Areny y John G. Webster. *Sensors and Signal Conditioning*. 2.ª ed. New York: John Wiley & Sons, 2001. ISBN: 978-0-471-33232-9.

[4] Paul Scherz y Simon Monk. *Practical Electronics for Inventors*. 4.ª ed. New York: McGraw-Hill Education, 2016. ISBN: 978-1-259-58754-2.

[5] Steven W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. Disponible en línea en https://www.dspguide.com. San Diego, CA: California Technical Publishing, 1997. ISBN: 978-0-9660176-3-2.

CAPÍTULO 5

Microcontroladores e interfaces de adquisición

En la cadena del capítulo 1, entre la conversión y la comunicación había una etapa de *procesamiento*: algo que lee los números, decide y prepara la respuesta. En un mando, ese papel lo hace un microcontrolador. Este capítulo cierra el Bloque I explicando qué es, qué lleva dentro y cómo se programa para leer sensores y gobernar salidas.

Objetivos de aprendizaje

- Describir la arquitectura básica de un microcontrolador: CPU, memoria y periféricos.
- Explicar el papel de los GPIO, las entradas analógicas, los temporizadores, las interrupciones y el PWM.
- Entender la estructura de un programa embebido y el ciclo de lectura de sensores.
- Situar Arduino y las plataformas con conectividad (ESP32) como herramientas de prototipado.

Alcance

Se estudia la arquitectura de un microcontrolador y sus periféricos con el nivel necesario para leer sensores y programar sistemas sencillos. No hay ensamblador ni programación a nivel de registros; el único código es un *sketch* de Arduino de una docena de líneas.

ÍNDICE DEL CAPÍTULO

5.1 MICROCONTROLADORES Y SISTEMAS EMBEBIDOS

5.1.1 Qué es un microcontrolador

> **Definición 5.1: microcontrolador**
>
> Circuito integrado que reúne en una sola pastilla un procesador, memoria y periféricos de entrada y salida. Es el «cerebro» de un sistema embebido.

Un *sistema embebido* es un pequeño computador dedicado a una función concreta dentro de un aparato que no parece un ordenador: un mando, un microondas, un termostato, un coche. La diferencia con un PC es de propósito: un ordenador es general, tiene sistema operativo, mucha potencia y mucho consumo; un microcontrolador está dedicado a una tarea, arranca al instante, gasta muy poco y cuesta céntimos [1].

En un mando moderno puede haber *varios*: uno principal que lee los controles y otros dedicados a la radio, a la háptica o a la gestión de la batería.

5.1.2 Arquitectura básica: CPU, memoria y periféricos

Un microcontrolador tiene tres partes (figura 5.1).

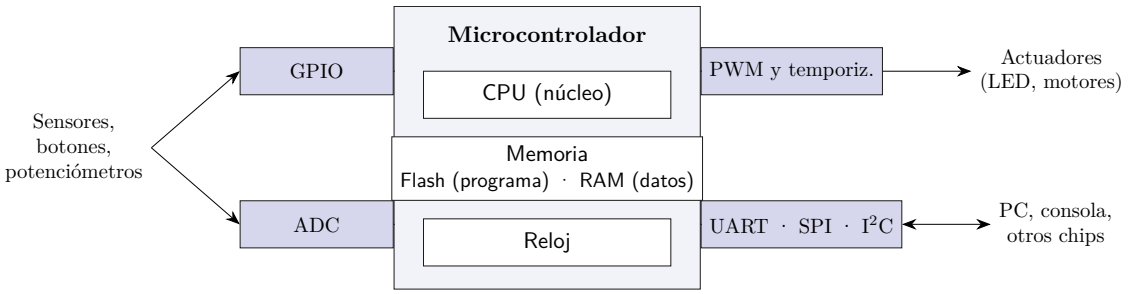


Figura 5.1. Arquitectura básica de un microcontrolador. La CPU ejecuta el programa guardado en la memoria, al ritmo del reloj; los periféricos (GPIO, ADC, temporizadores y PWM, interfaces de comunicación) conectan con el mundo exterior.

CPU (núcleo)
Ejecuta las instrucciones del programa una tras otra, al ritmo del **reloj** (unos pocos a unos cientos de MHz). Maneja los datos en palabras de 8, 16 o 32 bits según el modelo.

Memoria
La *memoria Flash* guarda el **programa** y no se borra al apagar; la *RAM* guarda los **datos** mientras el programa se ejecuta y se borra al quitar la alimentación. Ambas son pequeñas: se miden en kB, no en GB.

Periféricos
Bloques especializados junto a la CPU que hacen tareas concretas sin ocuparla: GPIO, ADC, temporizadores y PWM, e interfaces de comunicación (UART, SPI, I^2C). A veces también DAC, USB o radio.

La consecuencia práctica es que el microcontrolador *ya trae dentro* casi todo lo de los capítulos 3 y 4: el ADC para leer sensores y el generador de PWM para las salidas.

5.2 PERIFÉRICOS DE ENTRADA Y SALIDA

5.2.1 GPIO y entradas y salidas digitales

Los **GPIO** (pines de propósito general) se configuran por software, uno a uno, como entrada o como salida:

- Como **salida** digital, el pin se pone a nivel alto o bajo: enciende un LED o activa el transistor que gobierna un motor (capítulo 7).
- Como **entrada** digital, el pin informa de si la tensión que hay es alta o baja: así se lee un botón, con su resistencia *pull-up* (capítulo 2).

Un uso típico: configurar un pin como entrada con `pinMode(pin, INPUT_PULLUP)` y leer un pulsador con `digitalRead()`, y otro pin como salida para un LED.

5.2.2 Entradas analógicas

Algunos pines están conectados al **ADC** interno (capítulo 4). En Arduino, `analogRead()` devuelve un número de 0 a 1 023 (10 bit); muchos microcontroladores modernos usan 12 bit. Un solo ADC con un multiplexor lee varios pines analógicos por turnos, así que leer un *stick* son dos lecturas seguidas, una por eje.

5.2.3 Temporizadores

Un *temporizador* cuenta pulsos del reloj. Sirve para medir cuánto tarda algo, para ejecutar una acción cada cierto tiempo sin detener el programa, y para generar señales periódicas. En Arduino, `millis()` y `micros()` dan el tiempo transcurrido desde el arranque; comparándolo se programan acciones *sin* usar `delay()`, que bloquea el programa mientras cuenta.

5.2.4 Interrupciones

Hay dos maneras de enterarse de un suceso (figura 5.2):

- *Sondeo*: comprobar la entrada una y otra vez dentro del bucle. Simple, pero puede tardar en reaccionar o perderse un suceso muy corto.
- *Interrupción*: el suceso avisa y la CPU lo atiende al instante, dejando lo que estuviera haciendo.

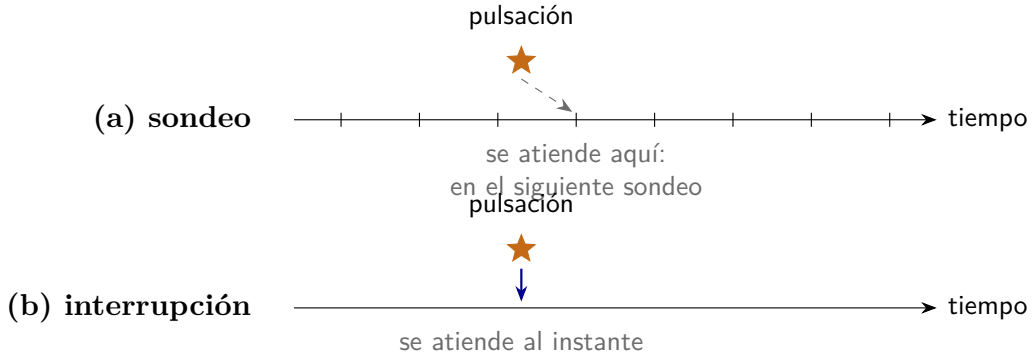


Figura 5.2. Con sondeo (a), una pulsación entre dos comprobaciones no se atiende hasta la siguiente. Con interrupción (b), se atiende en el momento en que ocurre.

Se usa interrupción para sucesos poco frecuentes pero urgentes —un pulso de un *encoder*, un dato que llega por el puerto serie, un botón que no se puede perder— y sondeo para lo que cambia despacio, como un potenciómetro. La rutina que atiende una interrupción debe ser brevísima: normalmente, apuntar que ha pasado y salir.

5.2.5 PWM

El generador de PWM (capítulo 4) es un periférico basado en un temporizador: una vez configurado, produce la señal él solo, sin que la CPU esté pendiente. En Arduino, `analogWrite(pin, valor)` fija el ciclo de trabajo con un `valor` de 0 a 255. Se usa para el brillo de un LED, la velocidad de un motor y la intensidad de la vibración.

5.3 ADQUISICIÓN Y PROCESAMIENTO DE DATOS

5.3.1 Lectura de sensores

Juntando lo anterior: un sensor digital se lee por un GPIO de entrada; uno analógico, por el ADC; uno «de bus», por SPI o I^2C (capítulo 9). La frecuencia de lectura debe ajustarse a lo rápido que cambie la señal (capítulo 4): leer de más gasta CPU y energía sin ganar nada.

5.3.2 Procesamiento básico

Lo que el programa hace con los números cabe en unas pocas líneas y no necesita potencia:

- **escalar** un rango a otro (`map()`) y aplicar una *zona muerta*;
- **filtrar** el ruido con una media móvil (capítulo 4);
- **comparar** con umbrales, **contar** y **detectar flancos**: distinguir que un botón *acaba de pulsarse* de que *está pulsado*, con antirrebote.

5.3.3 Estructura de un programa embebido

Un programa embebido tiene dos partes (figura 5.3): una de **configuración**, que se ejecuta una sola vez al arrancar (configurar los pines, iniciar las comunicaciones), y un **bucle** que se repite sin fin: leer las entradas, procesarlas, actuar sobre las salidas y esperar al ritmo fijado. En Arduino son las funciones `setup()` y `loop()`, y el programa se llama sketch [3].

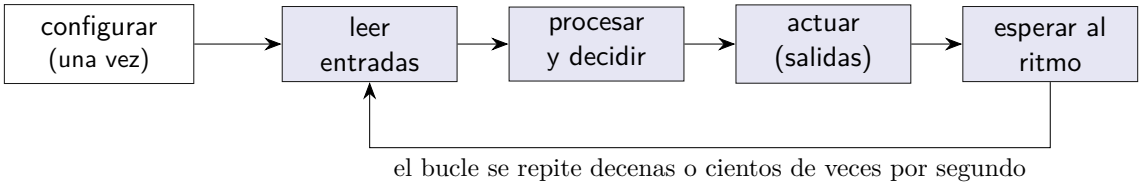


Figura 5.3. Estructura de un programa embebido: una configuración inicial y un bucle que se repite. Es la misma idea que el bucle de juego del capítulo 1.

La regla de oro del tiempo real es que **el bucle no debe bloquearse**: cada `delay()` es tiempo en el que el sistema no atiende nada. Para hacer varias cosas «a la vez» se usa `millis()`; para programas que funcionan por fases, una máquina de estados [4].

5.4 PLATAFORMAS DE PROTOTIPADO Y EJEMPLO INTEGRADOR

5.4.1 Arduino y plataformas de prototipado

Arduino es a la vez una placa, un entorno de programación y una comunidad, pensados para prototipar sin ser experto. La placa lleva un microcontrolador, un regulador de tensión, un conector USB —que programa *y* alimenta— y todos los pines accesibles en tiras de conectores. Se programa en un C++ simplificado, con funciones como `digitalWrite()`, `analogRead()` o `Serial.print()`, y hay librerías hechas para casi cualquier sensor o módulo. Plataformas del mismo estilo son micro:bit, Raspberry Pi Pico, STM32 y Teensy.

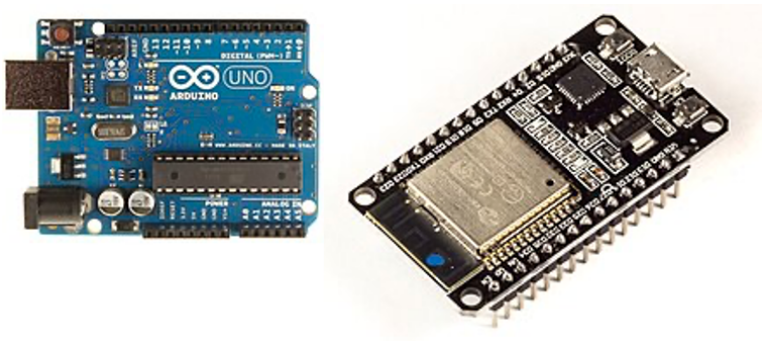


Figura 5.4. Una placa Arduino y una placa ESP32. Ambas llevan el microcontrolador, el regulador, el conector USB y los pines en tiras; la ESP32 añade, además, Wi-Fi y Bluetooth.

5.4.2 ESP32 y microcontroladores con conectividad

El ESP32 es una familia de microcontroladores que, además de lo anterior, integran **Wi-Fi y Bluetooth**, más memoria y más potencia, a un precio parecido. Para esta asignatura importan por dos motivos: permiten construir un mando o un periférico *inalámbrico* sin añadir un módulo de radio aparte (capítulo 9), y conectar el proyecto a una red o a un servicio en línea (capítulo 10). Se programan con el mismo entorno de Arduino [2].

5.4.3 Ejemplo completo de un sistema de adquisición

El listado 5.1 recorre toda la cadena del Bloque I en una docena de líneas: lee un potenciómetro, lo procesa y lo usa para dos salidas —el brillo de un LED por PWM y el envío de datos a un PC—.

```
const int PIN_POT = A0;      // entrada analogica
const int PIN_LED = 9;       // salida con PWM

void setup() {
  pinMode(PIN_LED, OUTPUT);
  Serial.begin(115200);              // enlace serie con el PC
}

void loop() {
  int crudo = analogRead(PIN_POT);           // 0..1023   (ADC,
      cap. 4)
```

```
  int valor = map(crudo, 0, 1023, 0, 255);    // escalado  (cap.
      3 y 4)
  if (valor < 8) valor = 0;                   // zona muerta

  analogWrite(PIN_LED, valor);                // brillo por PWM
     (cap. 4 y 7)
  Serial.println(valor);                      // envio al PC   (
     cap. 9)
  delay(10);                                  // ~100 lecturas
     por segundo
}
```

Listado 5.1. Sketch que lee un potenciómetro y lo lleva a un LED (por PWM) y al PC (por el puerto serie).

Cada línea corresponde a una etapa de la cadena: `analogRead` es la conversión analógico-digital; `map` y la zona muerta son el procesamiento; `analogWrite` es la generación de señal por PWM; `Serial` es la comunicación con el exterior.

Este mismo esquema —leer un sensor, procesar y enviar el resultado— es la base del sistema completo que se arma en el capítulo 9.

RESUMEN

Un microcontrolador reúne en una pastilla un procesador, memoria y periféricos, y es el cerebro de un sistema embebido: un computador dedicado, de arranque inmediato, bajo consumo y bajo coste. La CPU ejecuta el programa —guardado en la memoria Flash— al ritmo del reloj, y usa la RAM para los datos. Los periféricos hacen el trabajo de contacto con el mundo sin ocupar la CPU: los GPIO leen y escriben niveles digitales; el ADC convierte tensiones en números; los temporizadores miden tiempos y generan PWM; las interfaces de comunicación hablan con otros chips y con el PC. Un suceso se puede detectar por sondeo (comprobar en cada vuelta del bucle) o por interrupción (que avisa al instante). Un programa embebido tiene una configuración que se ejecuta una vez y un bucle que se repite sin fin y que no debe bloquearse. Arduino facilita prototipar todo esto, y familias como el ESP32 añaden Wi-Fi y Bluetooth para hacerlo inalámbrico. Un *sketch* de una docena de líneas basta para recorrer toda la cadena: leer un sensor, procesarlo y llevarlo a una

salida y a un PC.

Ideas clave

- Un microcontrolador es un computador entero —CPU, memoria y periféricos— dedicado a una tarea.
- Los periféricos (GPIO, ADC, temporizadores/PWM, buses) hacen su trabajo sin ocupar la CPU.
- Sondeo: se comprueba por turnos. Interrupción: el suceso avisa y se atiende al instante.
- Todo programa embebido es configuración una vez + bucle que se repite y no se bloquea.
- El mismo bucle leer → procesar → actuar aparece en un *sketch* y en un motor de videojuego.

TÉRMINOS DEL CAPÍTULO

Microcontrolador, sistema embebido, CPU, reloj, memoria Flash, memoria RAM, periférico, GPIO, entrada digital, salida digital, entrada analógica, temporizador, `millis()`, interrupción, sondeo, PWM, detección de flanco, máquina de estados, *sketch*, `setup()` y `loop()`, Arduino, ESP32.

PARA SABER MÁS

- Monk [3] para empezar a programar Arduino paso a paso.
- Blum [1], más completo, entra en sensores, motores y comunicación.
- White [4], sobre cómo estructurar el software de un sistema embebido para que sea fiable.

BIBLIOGRAFÍA DEL CAPÍTULO

[1] Jeremy Blum. *Exploring Arduino: Tools and Techniques for Engineering Wizardry*. 2.ª ed. Indianapolis, IN: John Wiley & Sons, 2019. ISBN: 978-1-119-40537-5.

[2] Neil Cameron. *Electronics Projects with the ESP8266 and ESP32*. Berkeley, CA: Apress, 2021. ISBN: 978-1-4842-6335-8.

[3] Simon Monk. *Programming Arduino: Getting Started with Sketches*. 3.ª ed. New York: McGraw-Hill Education, 2022. ISBN: 978-1-264-67698-5.

[4] Elecia White. *Making Embedded Systems: Design Patterns for Great Software*. Sebastopol, CA: O'Reilly Media, 2011. ISBN: 978-1-449-30214-6.

BLOQUE II

PERSONALIZACIÓN DE HARDWARE Y PERIFÉRICOS

CAPÍTULO 6

Dispositivos de entrada y periféricos

El Bloque I estudió las piezas: señales, sensores, conversión, microcontroladores. Este capítulo abre el Bloque II juntándolas en *dispositivos* completos —mandos, teclados, ratones, periféricos especializados— y analiza cómo cada uno funciona por dentro y cómo condiciona el diseño de juego.

Objetivos de aprendizaje

- Analizar los dispositivos de entrada convencionales y el sensor que llevan dentro.
- Explicar cómo se leen teclados y matrices de botones, ratones y pantallas táctiles.
- Describir la entrada por movimiento, por cámara y por gestos, y sus límites.
- Enumerar criterios para diseñar un dispositivo de entrada personalizado.

Alcance

Se ve el dispositivo de entrada *completo* —sus sensores (capítulo 3), su lectura (capítulo 5) y su efecto en el juego—, no la física de cada sensor. El proceso de diseño y construcción de un periférico a medida es el capítulo 8.

ÍNDICE DEL CAPÍTULO

6.1 DISPOSITIVOS DE ENTRADA CONVENCIONALES

Un dispositivo de entrada es uno o varios sensores, su electrónica de lectura y, a menudo, un microcontrolador, dentro de una carcasa pensada para el cuerpo. Traduce acciones físicas en datos para el juego [1].

6.1.1 Botones e interruptores

Un botón es un contacto: pulsado o no pulsado, un bit (capítulo 1). Los hay de membrana (baratos, silenciosos, con poco tacto), mecánicos (con clic y sensación definidos, más duraderos) y táctiles de placa (los pequeños). Tienen *rebote*, que se filtra por antirrebote (capítulos 2 y 4), y un *punto de actuación*: cuánto hay que apretar para que registre; acortarlo hace el botón más rápido (capítulo 8). En diseño, un botón es una acción: el número y la disposición de los botones definen qué géneros son posibles.

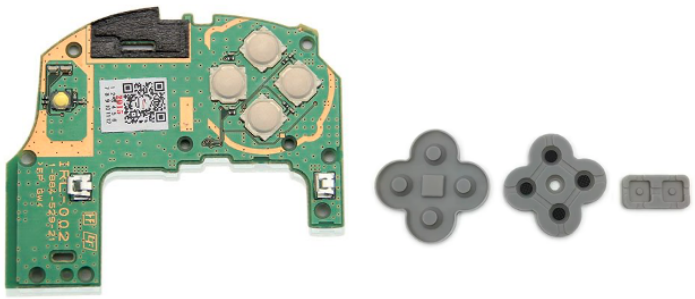


Figura 6.1. Por dentro de una cruceta: la almohadilla de goma conductora (derecha) y las pistas de contacto de la placa (izquierda). Al presionar, la goma une las dos mitades de una pista y cierra el circuito.

6.1.2 Joysticks y palancas analógicas

Una palanca sobre dos ejes son dos potenciómetros (o dos sensores de efecto Hall): dos valores analógicos que van al ADC (capítulos 3 y 4). La de un mando lleva un muelle que la devuelve al centro; la de un *joystick* de vuelo, no. Sobre la lectura cruda se aplican zona muerta y calibración, y muchas llevan un botón que se activa

al presionar la palanca. En diseño aportan *dirección e intensidad*: caminar o correr, mover una cámara, apuntar.

6.1.3 Gamepads

El gamepad es la integración estándar: cruceta, cuatro botones de cara, dos palancas analógicas, dos gatillos, botones de hombro, botones de sistema y, cada vez más, una IMU, un panel táctil, un altavoz y motores de vibración. Visto en conjunto, un mando **es un pequeño sistema de adquisición y control** (figura 6.2): sensores que leen las manos, un microcontrolador que arma una trama de datos y un enlace que la lleva a la consola, con un canal de vuelta para la háptica.

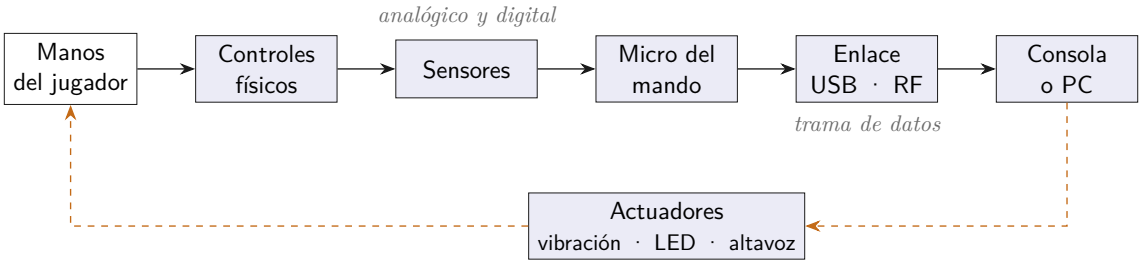


Figura 6.2. Un mando reproduce la cadena del capítulo 1: los controles accionan sensores, el microcontrolador del mando los empaqueta y un enlace los envía a la consola, que devuelve órdenes al canal de actuadores.

Tabla 6.1. Qué sensor lleva dentro cada control de un mando.

Control	Sensor	Señal
Cruceta / botones de cara	contactos	digital (1 bit cada uno)
Palanca analógica	2 potenciómetros o sensores Hall	analógica → ADC
Gatillo analógico	potenciómetro + contacto de fin	analógica + digital
Panel táctil	sensor capacitivo	posición del dedo
Movimiento del mando	acelerómetro + giróscopo (IMU)	digital por bus

El sistema operativo presenta el mando de forma uniforme —«ejes y botones»—

mediante HID o XInput (capítulo 9), así que el juego no necesita saber la marca.

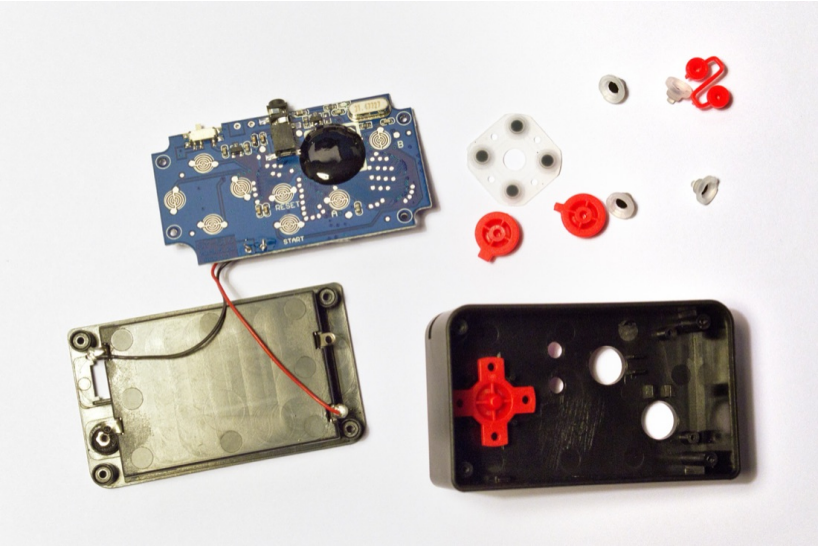


Figura 6.3. Un mando sencillo desmontado: la placa, con las pistas de contacto en forma de anillo y el circuito integrado bajo la gota de resina; la cruceta de plástico; las almohadillas de goma conductora que cierran cada contacto al pulsar; y las dos mitades de la carcasa. *(foto: Multicherry · CC BY-SA 4.0)*

6.1.4 Potenciómetros y encoders

Ambos miden giro o desplazamiento, pero de forma distinta:

- El *potenciómetro* tiene topes y da la *posición absoluta* como una tensión (capítulo 2): ruedas, deslizadores, volantes sencillos.
- El encoder gira sin fin y entrega *pulsos* que se cuentan para saber el movimiento (capítulo 3): la rueda del ratón, las ruedas de *jog*, los volantes buenos, los *spinners* de arcade. Un *encoder* absoluto da además la posición directamente.

6.1.5 Teclados y matrices de botones

Un teclado son muchos contactos digitales. Como no hay un pin por tecla, se disponen en una *matriz* de filas y columnas: el microcontrolador activa una fila cada vez y comprueba qué columnas responden, repitiéndolo muy deprisa (figura 6.4). Así, doce botones se leen con siete pines, y cien, con veinte.

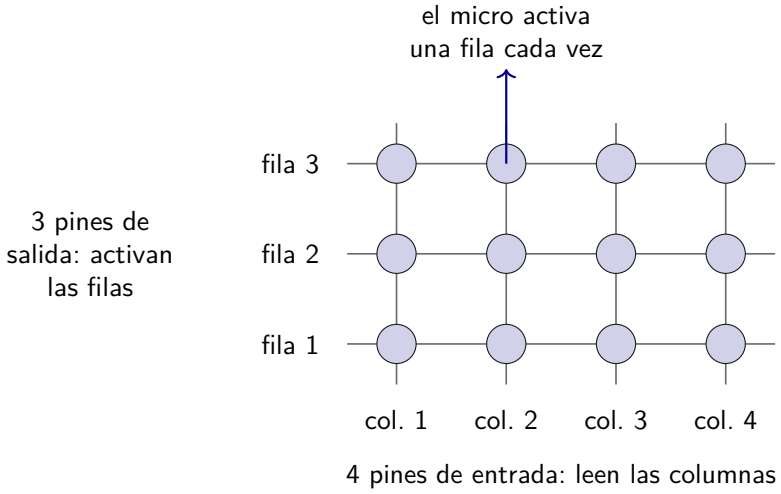


Figura 6.4. Matriz de botones. El microcontrolador pone activa una fila y lee las columnas; barriendo las tres filas muy rápido conoce el estado de los doce botones con solo siete pines.

La matriz tiene un problema: al pulsar varias teclas a la vez pueden aparecer pulsaciones *fantasma*. Se resuelve con un diodo por tecla (*anti-ghosting*), lo que permite el N-key rollover: registrar muchas teclas simultáneas, algo necesario en juegos. Como todo dispositivo, el teclado tiene su tasa de sondeo (capítulo 4), de 125 a 1 000 Hz.

6.1.6 Ratones y dispositivos apuntadores

Un ratón óptico ilumina la superficie con un LED o un láser, una cámara diminuta toma cientos o miles de fotos por segundo y un procesador compara fotos consecutivas para deducir el desplazamiento (capítulo 3). Sus dos parámetros clave:

- *CPI/DPI*: cuentas por pulgada de movimiento; a más CPI, el puntero recorre más pantalla con el mismo gesto. Se ajusta (capítulo 8).
- Tasa de sondeo: de 125 Hz (8 ms) a varios kHz; a 125 Hz el puntero se nota «a tirones» en un juego de acción (capítulo 4).

La *aceleración del ratón* —una ganancia que depende de la velocidad del gesto— suele desactivarse en juego para que el mismo movimiento dé siempre el mismo resultado (repetibilidad, capítulo 3). Otros apuntadores: *trackball*, *trackpad*,

lápiz y tableta, y los mandos por puntero como el Wiimote [2].

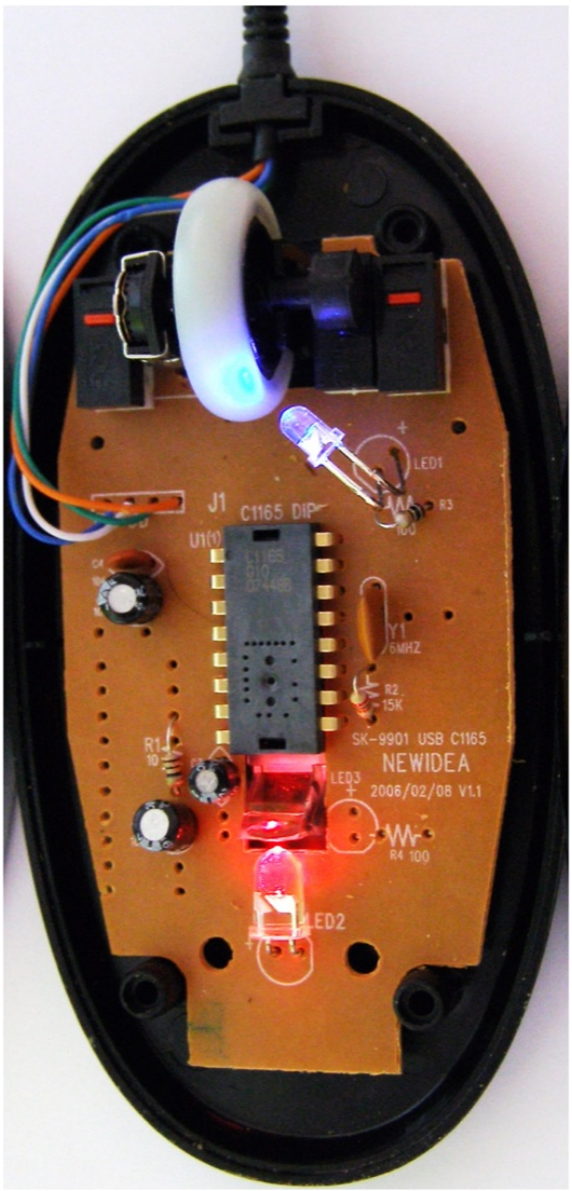


Figura 6.5. La cara inferior de un ratón óptico: el LED ilumina la superficie en ángulo y una lente lleva su imagen al sensor, que compara fotogramas sucesivos para medir el desplazamiento. *(foto: Wikiwide · dominio público)*

6.1.7 Pantallas táctiles

Una pantalla táctil capacitiva mide dónde y con cuántos dedos se la toca (capítulo 3). No hay retorno táctil —no «notas» el botón—, así que el diseño lo

compensa con señales visuales y zonas amplias, y hay que contar con la *oclusión*: el dedo tapa lo que toca. Sobre ella se construyen controles virtuales (cruceta y botones dibujados) o mecánicas pensadas para el táctil: arrastrar, trazar, pellizcar.

6.2 ENTRADA POR MOVIMIENTO E IMAGEN

6.2.1 Sensores de movimiento: acelerómetros y giróscopos

Un acelerómetro mide aceleración e inclinación; un giróscopo, velocidad de giro; la IMU los combina y entrega el dato por bus (capítulos 3 y 4). Con ellos, el mando se convierte en un objeto que se mueve: el Wiimote, los Joy-Con, un móvil, o el *gyro aiming* (afinar la puntería con el giro mientras el *stick* hace el movimiento grueso). Su límite es importante: la IMU sabe cómo está *orientado* el mando y cómo *acelera*, pero no *dónde está* en la habitación —esa información se pierde por acumulación de error—.

6.2.2 Sistemas de *tracking*

El *seguimiento* (*tracking*) determina la posición, y normalmente la orientación, de un mando o del cuerpo en el espacio. Hay dos enfoques, que se detallan en el capítulo 10: *outside-in* (cámaras o estaciones fijas en la sala observan al usuario) e *inside-out* (cámaras en el propio dispositivo observan la sala). Ambos combinan óptica e IMU —la óptica corrige la deriva de la IMU, la IMU rellena entre fotogramas— y dan seis grados de libertad: posición en los tres ejes más orientación.

6.2.3 Cámaras como dispositivos de entrada

Una cámara normal más software detecta la cara, las manos, el color de un mando o códigos impresos. Una *cámara de profundidad* (como Kinect) mide además la distancia de cada punto de la escena, con lo que reconstruye el esqueleto del jugador sin necesidad de mando. El coste es la sensibilidad a la iluminación, la

oclusión, la latencia de procesamiento y la privacidad (capítulo 11).

6.2.4 Interfaces gestuales

En una *interfaz gestual* la orden es un movimiento del cuerpo —agitar, dibujar una forma, señalar, pellizcar en el aire— reconocido como comando, a partir de una IMU, una cámara o el seguimiento de manos. Sus problemas son los falsos positivos y negativos, la fatiga del brazo, la falta de retorno y la dificultad de descubrir qué gestos existen. Funciona bien como complemento y mal como entrada única.

6.3 INTERFACES NO CONVENCIONALES Y DISEÑO

6.3.1 Interfaces no convencionales

Cuando el hardware define el género aparecen periféricos especializados: volantes y pedales, *HOTAS* y palancas de vuelo, *arcade sticks*, pistolas, alfombras de baile, instrumentos (guitarra, batería, micrófono de canto). Y hay interfaces poco habituales que además abren la puerta a más jugadores: control por soplido y sorbo, pedales, sensores de flexión en un guante, botones de gran tamaño, voz o mirada (capítulos 8 y 10). Cada una captura una acción física concreta y habilita una experiencia; el coste es fragmentar el mercado.

Figura 6.6. Un volante con pedales: el giro se mide con un potenciómetro o un *encoder*, los pedales con potenciómetros, y un motor devuelve fuerza al volante (capítulo 7). *(foto: FreeMediaKid! · CC BY 4.0)*

6.3.2 Diseño de dispositivos de entrada personalizados

El proceso completo es el capítulo 8; los principios son:

- Partir de la **acción** que se quiere capturar y de **quién** la hará (una mano, un pie, un soplido; precisión o fuerza limitadas).
- Elegir el **sensor** adecuado (capítulo 3) y cómo leerlo (capítulo 5).
- Cuidar la **ergonomía**: postura, alcance, fuerza necesaria, tamaño de los controles, retorno.
- Definir el **mapeo** —qué hace cada control— con correspondencia clara, coherencia y posibilidad de remapear (capítulo 8).
- Priorizar **latencia y repetibilidad** sobre exactitud (capítulos 3 y 4): que responda igual y deprisa.
- Incluir la **accesibilidad** desde el principio.

La sensación de control —el *game feel*— nace tanto del hardware (recorrido, resistencia, punto de actuación) como del software (curvas de respuesta, filtrado, asistencias) [3].

RESUMEN

Un dispositivo de entrada combina sensores, electrónica de lectura y, a menudo, un microcontrolador en una carcasa ergonómica. Un botón es un contacto digital; una palanca analógica son dos potenciómetros o sensores Hall que van al ADC; un *gamepad* integra ambos más gatillos, IMU, panel táctil y actuadores, y se comporta como un pequeño sistema de adquisición y control que el sistema operativo presenta de forma uniforme. Los potenciómetros dan posición absoluta y los *encoders*, movimiento. Un teclado organiza sus contactos en una matriz que se lee por barrido, con diodos para permitir muchas teclas a la vez. Un ratón óptico es una cámara que compara fotos de la superficie; sus parámetros son el CPI y la tasa de sondeo. Una pantalla táctil capacitiva no da retorno, y el diseño lo compensa. Los acelerómetros y giróscopos hacen del mando un objeto que se mueve, pero no saben dónde está: para eso hace falta *tracking*, que combina óptica e IMU. Las cámaras, con o sin profundidad, permiten jugar sin mando, y las interfaces gestuales, dar órdenes con el cuerpo, con sus límites. Al diseñar un dispositivo se parte de la acción y de quién la hará, se elige el sensor, se cuida la ergonomía y el mapeo, y se prioriza que responda igual y deprisa.

Ideas clave

- Detrás de cada control hay un sensor del capítulo 3: contacto, potenciómetro, capacitivo, inercial.
- Un *gamepad* es un sistema de adquisición y control en miniatura.
- La matriz permite leer muchos botones con pocos pines; los diodos evitan pulsaciones fantasma.
- Un acelerómetro sabe cómo te mueves, no dónde estás; la posición la da el *tracking*.
- La sensación de control es mitad hardware (recorrido, resistencia) y mitad software (curvas, filtrado).

TÉRMINOS DEL CAPÍTULO

Dispositivo de entrada, botón, punto de actuación, palanca analógica, *joystick*, *gamepad*, cruceta (D-pad), gatillo analógico, potenciómetro, *encoder* incremental y

absoluto, matriz de botones, escaneo, *anti-ghosting*, *N-key rollover*, ratón óptico, CPI/DPI, tasa de sondeo, aceleración del ratón, *trackball*, pantalla táctil capacitiva, oclusión, acelerómetro, giróscopo, IMU, *gyro aiming*, seguimiento (*tracking*), *outside-in* e *inside-out*, seis grados de libertad, cámara de profundidad, interfaz gestual, periférico especializado, mapeo, *game feel*.

PARA SABER MÁS

- Swink [3] analiza cómo el hardware y el software del control producen la sensación de jugar.
- Dix et al. [1], capítulo de dispositivos de entrada, para el marco general de interacción persona-ordenador.
- Sánchez Ortega [2] recorre la historia de cada tipo de mando y periférico.

BIBLIOGRAFÍA DEL CAPÍTULO

[1] Alan Dix et al. *Human-Computer Interaction*. 3.ª ed. Harlow: Pearson Prentice Hall, 2004. ISBN: 978-0-13-046109-4.

[2] Pedro Luis Sánchez Ortega. *Toma el control: breve historia de los videojuegos con la evolución de sus mandos*. Material docente. Universidad de Burgos, 2023. URL: http://hdl.handle.net/10259/9588 (visitado 08-09-2026).

[3] Steve Swink. *Game Feel: A Game Designer's Guide to Virtual Sensation*. Burlington, MA: Morgan Kaufmann, 2009. ISBN: 978-0-12-374328-2.

CAPÍTULO 7

Actuadores y retroalimentación

Hasta ahora la cadena iba en un solo sentido: del cuerpo al dato. Este capítulo recorre el sentido de vuelta —del dato al efecto físico— y los dispositivos que lo hacen posible: LEDs, motores, solenoides, relés y, sobre todo, la vibración y el retorno de fuerza que convierten un mando en algo que también responde.

Objetivos de aprendizaje

- Clasificar los actuadores eléctricos habituales y su función como salida.
- Explicar el control por PWM y el papel de los *drivers* y las etapas de potencia.
- Describir la vibración y la retroalimentación háptica, y qué información pueden transmitir.
- Reconocer las limitaciones de corriente y potencia al alimentar actuadores.

Alcance

Se ve qué hace cada actuador, cómo se controla y para qué sirve en un juego o un periférico. No se diseñan puentes en H ni etapas de potencia, ni se estudian las ecuaciones de los motores: el control de PWM y la potencia se tratan de forma cualitativa.

ÍNDICE DEL CAPÍTULO

7.1 ACTUADORES COMO ELEMENTOS DE SALIDA

7.1.1 Qué es un actuador

Un *actuador* es un transductor de salida: convierte una señal eléctrica en un efecto físico —movimiento, luz, sonido, fuerza, calor— (capítulo 1). Es la etapa de vuelta de la cadena, y la que hace que un mando no sea solo una entrada.

Casi ningún actuador se conecta directamente a un pin del microcontrolador: un pin entrega unas decenas de miliamperios y la mayoría de los actuadores necesitan mucho más, así que entre medias va una *etapa de potencia* (sección 7.4) —igual que el transistor que vimos en el capítulo 2—.

7.1.2 LEDs y dispositivos de señalización

El LED es un diodo que emite luz; necesita su resistencia en serie (capítulo 2) y su brillo se regula por PWM (capítulo 4). Es la salida más simple, y también la más usada para informar: jugador 1 o 2, nivel de batería, notificación, barra de luz, iluminación RGB. Un paso más son los *displays* (siete segmentos, OLED, la pantalla de un móvil) y, para el sonido más básico, un zumbador piezoeléctrico.

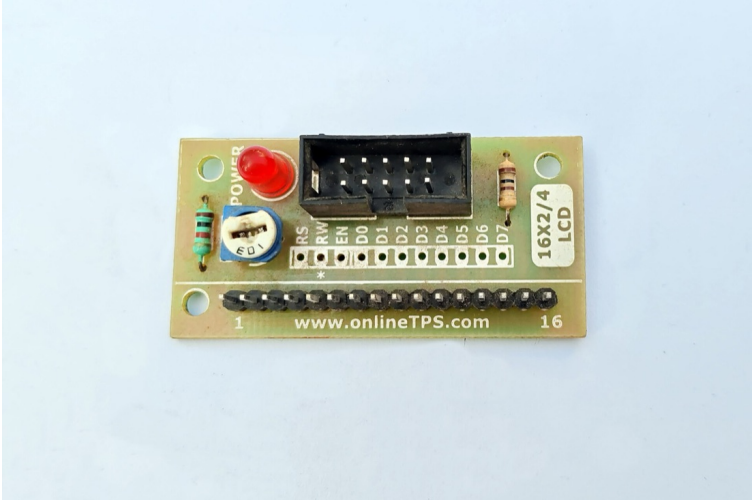


Figura 7.1. Módulos de *display* habituales: un indicador de siete segmentos, una pantalla OLED pequeña y un panel LCD. Todos son salidas visuales que el microcontrolador gobierna por un bus. *(foto: Suyash Dwivedi · CC BY-SA 4.0)*

7.2 ACTUADORES ELECTROMECÁNICOS

Tabla 7.1. Actuadores electromecánicos y su uso en juegos y periféricos.

Actuador	**Qué produce / cómo se controla**	**Ejemplo**
Motor de corriente continua	giro continuo; velocidad por PWM, sentido invirtiendo la polaridad	vibración, volantes, ventiladores
Servomotor	un ángulo concreto que mantiene; se le indica la posición	gatillos adaptativos, mecanismos de arcade
Motor paso a paso	giro en pasos fijos; posición contando pasos, sin sensor	impresoras 3D, periféricos de precisión
Solenoide	un golpe o un empuje lineal breve; todo o nada	*pinball*, efectos de golpe, cerrojos
Relé	conmuta circuitos de mucha potencia con una señal pequeña	encender luces o máquinas en una instalación

Un **motor de corriente continua** gira mientras le llega corriente; con más tensión gira más rápido y, invirtiendo la polaridad, gira al revés. No conoce su posición ni su velocidad exacta salvo que se le añada un *encoder* (capítulo 6). Un *servomotor* es un motor con reductora, sensor de posición y electrónica en el mismo bloque: se le indica un ángulo y va a él y lo mantiene. Un *motor paso a paso* avanza en incrementos fijos, lo que da control de posición sin sensor, a costa de ser más lento y de consumir aunque esté parado [4].

Un *solenoide* transforma un pulso de corriente en un movimiento lineal corto —un émbolo que sale o golpea—; es la base de los *flippers* y *bumpers* de un *pinball*. Un *relé* es un interruptor accionado por una pequeña corriente que permite al microcontrolador conectar y desconectar circuitos de mucha más potencia, con aislamiento; es lento, hace clic y se desgasta, así que para conmutar rápido y a

menudo se usa un transistor [3].

7.3 VIBRACIÓN Y HÁPTICA

7.3.1 Motores de vibración

Hay dos tecnologías principales:

- *Motor de masa excéntrica (ERM)*: un peso descentrado en el eje de un motor; al girar, todo el conjunto vibra. Barato y robusto, pero arranca y para despacio y no permite separar la frecuencia de la intensidad. Es el *rumble* clásico.
- *Actuador resonante lineal (LRA)*: una masa que oscila en una sola dirección, empujada por un electroimán, a una frecuencia fija. Arranca y para deprisa —«golpes» definidos— y la intensidad se controla. Habitual en móviles y mandos recientes.

Al ERM se le manda PWM (el ciclo de trabajo es la intensidad); al LRA, una señal más elaborada.

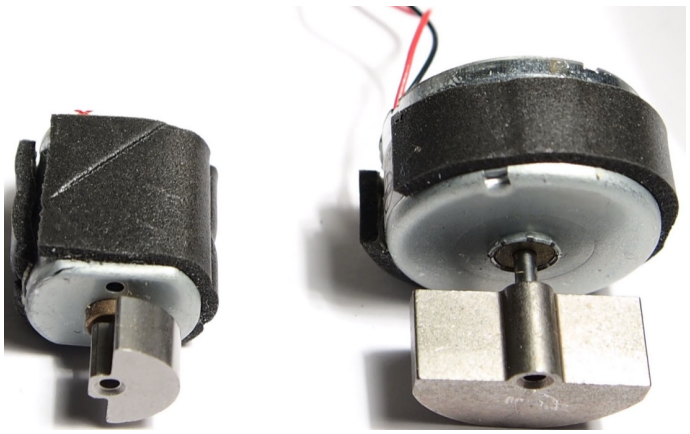


Figura 7.2. Un motor de masa excéntrica «de moneda» (el disco plano, con su peso descentrado) y un actuador resonante lineal (la cápsula rectangular). Son lo que vibra dentro de un mando o un móvil. *(foto: Tiia Monto · CC BY-SA 3.0)*

7.3.2 Retroalimentación háptica

La *háptica de banda ancha* usa un actuador de bobina de voz —casi un altavoz— capaz de reproducir muchas frecuencias e intensidades a la vez. Por eso transmite «texturas» y no solo un zumbido (figura 7.3): es lo que llevan el DualSense y el HD Rumble de Switch.

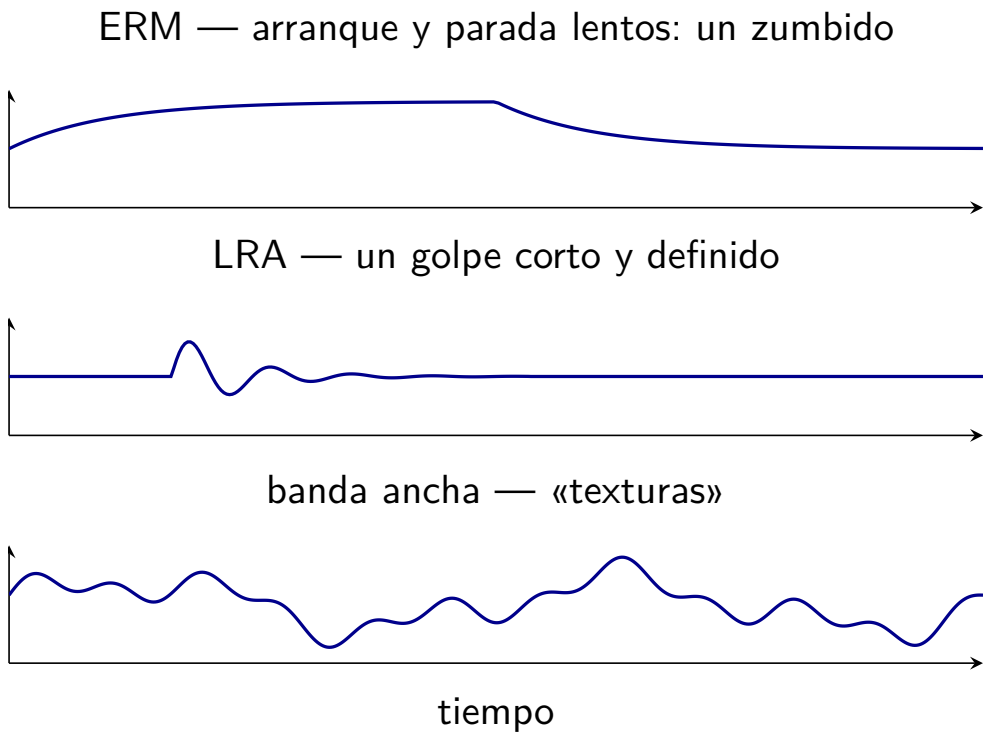


Figura 7.3. Lo que cada actuador puede transmitir. Un motor de masa excéntrica solo produce un zumbido que sube y baja despacio; un actuador resonante da golpes cortos y definidos; uno de banda ancha reproduce señales complejas que se perciben como texturas.

El *retorno de fuerza* (*force-feedback*) va un paso más allá: el actuador se *opone* al movimiento del jugador o lo empuja. Es lo que endurece el giro de un volante según el agarre del coche, lo que resiste en un *joystick* de vuelo y lo que cambia la dureza de un gatillo adaptativo (un motorcito con engranaje dentro del gatillo). Más allá del mando —trajes, guantes, actuadores de temperatura, ultrasonidos para tacto sin contacto— se trata en el capítulo 10.

Bien usada, la háptica comunica información —dirección, superficie, peligro, un disparo— sin ocupar la pantalla ni el sonido [1]. Pero cansa y se ignora si es

constante: conviene que sea puntual y significativa.

7.4 CONTROL Y POTENCIA DE ACTUADORES

7.4.1 Control mediante PWM

El PWM (capítulos 4 y 5) sirve igual para la salida que para medir: el ciclo de trabajo fija el brillo de un LED, la velocidad de un motor o la intensidad de una vibración, y el temporizador del microcontrolador genera la señal por su cuenta. Para invertir el *sentido* de giro de un motor hace falta además un circuito que invierta la polaridad, el *puente en H*.

7.4.2 *Drivers* y etapas de potencia

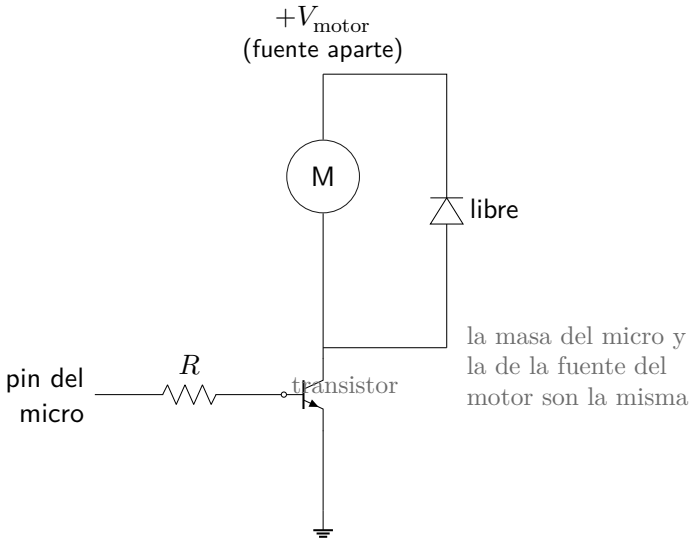


Figura 7.4. Etapa de potencia. El pin del microcontrolador solo controla la base de un transistor; es el transistor el que deja pasar la corriente grande desde una fuente aparte. El diodo «libre» absorbe el pico de tensión al cortar la corriente al motor.

Un driver es un circuito con transistores que recibe la señal débil del microcontrolador y con ella gobierna la corriente que necesita el actuador (figura 7.4). Hace

falta porque un pin da unos 20 mA, un motor de vibración pide 100 a 500 mA y el de un volante, amperios: conectar el motor al pin lo destruye.

Un motor y un relé son bobinas: al cortarles la corriente generan un pico de tensión que dañaría el transistor. El *diodo de rueda libre*, en paralelo con el actuador, lo absorbe. Los módulos habituales son un transistor MOSFET con su diodo, o un chip de puente en H cuando se quiere controlar el sentido [2].

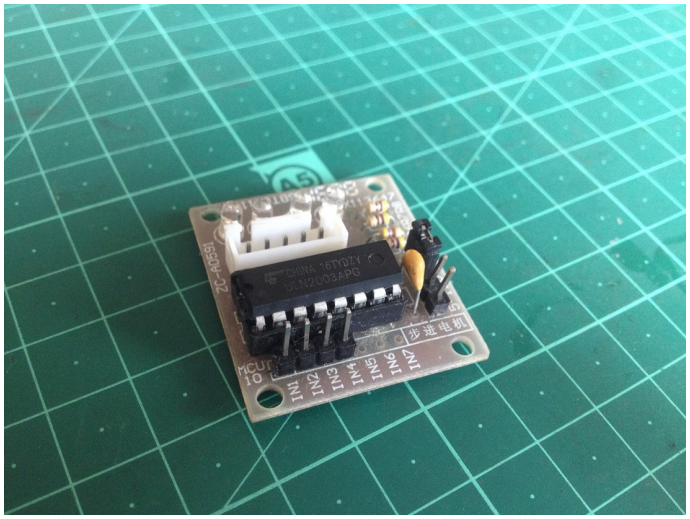


Figura 7.5. Un módulo *driver*: una placa pequeña con el transistor de potencia (o un chip de puente en H), su diodo y los bornes para el motor y para su alimentación aparte. *(foto: Kushagra Keshari · CC BY-SA 4.0)*

7.4.3 Alimentación de actuadores

Los actuadores consumen mucho más que la lógica, y a golpes. Conviene alimentarlos de una **línea aparte** —o con buen desacoplo— para que sus picos no hundan la tensión del microcontrolador y lo reinicien. Eso sí, la línea de los actuadores y la del microcontrolador deben compartir **masa** (capítulo 2), o la señal de control no tiene referencia. En un mando inalámbrico todo sale de la batería: activar la vibración y la iluminación reduce la autonomía (capítulo 8).

7.4.4 Limitaciones de corriente y potencia

Cada elemento de la cadena tiene un máximo: el pin, el transistor, la fuente, los cables y la batería. La potencia que no se aprovecha se convierte en **calor**: un *driver* que trabaja fuerte necesita disipador, y un motor bloqueado consume el máximo y se calienta. La regla práctica es dimensionar el *driver* y la fuente para el *pico* de consumo, no para la media, y proteger el circuito con un fusible o una limitación de corriente.

7.5 DISEÑO DE SISTEMAS DE RESPUESTA FÍSICA

Diseñar la salida física empieza por elegir el actuador según el **efecto** buscado —aviso luminoso, golpe, textura, fuerza, movimiento— y según **quién** lo percibe y **dónde**: en la mano, en el cuerpo, a distancia.

Después está la cuestión del lazo (capítulo 1):

- La háptica de un mando suele ser **lazo abierto**: el juego manda «vibra así» y confía en que se note. Es simple y suficiente para avisos.
- El retorno de fuerza de un volante se acerca al **lazo cerrado**: el juego mide la posición del volante y calcula la fuerza en función de ella y de la física (figura 7.6). Si el lazo va lento o el sensor es pobre, el volante se siente «gomoso» u oscila.

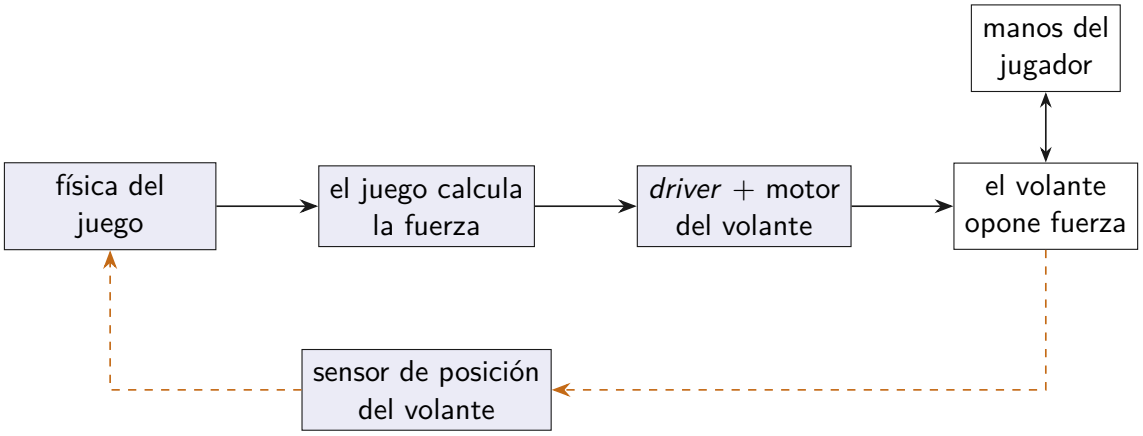


Figura 7.6. El retorno de fuerza de un volante es un lazo cerrado: la física del juego fija una fuerza, el motor la aplica, el volante se mueve, un sensor mide su posición y esa medida vuelve a entrar en el cálculo. El jugador forma parte del lazo.

Por último, la salida física llega tarde si se acumulan retardos (capítulo 4): una vibración de disparo que llega 100 ms tarde no «pega». Y debe ser **coherente** con lo que ocurre, **consistente** y **moderada**: si todo vibra, nada comunica. El canal de vuelta es parte del diseño de juego, no un adorno (capítulo 6).

RESUMEN

Un actuador convierte una señal eléctrica en un efecto físico y cierra el canal de vuelta de la cadena. El LED, con su resistencia y regulado por PWM, es la salida más simple y la más usada para informar. Entre los actuadores electromecánicos, el motor de corriente continua gira según la tensión y cambia de sentido con la polaridad; el servomotor va a un ángulo y lo mantiene; el motor paso a paso avanza en pasos contables; el solenoide da un golpe; el relé conmuta potencia con aislamiento. La vibración se produce con un motor de masa excéntrica (un zumbido lento) o con un actuador resonante lineal (golpes definidos), y la háptica de banda ancha, con un actuador de bobina de voz, transmite texturas; el retorno de fuerza, además, se opone al jugador. Casi ningún actuador se conecta a un pin: entre medias va un *driver* de transistores, con su diodo de rueda libre, alimentado de una línea aparte pero con masa común, y dimensionado para el pico de consumo, porque la potencia sobrante se vuelve calor. Al diseñar la salida física se elige el actuador por el efecto

buscado, se decide si el lazo es abierto o cerrado, se vigila la latencia y se usa la háptica de forma puntual y coherente.

Ideas clave

- El actuador es un transductor de salida: la etapa de vuelta de la cadena.
- Un pin no mueve un motor: hace falta un *driver* de transistores y una fuente aparte con masa común.
- Motor de masa excéntrica = zumbido; resonante lineal = golpe; banda ancha = textura; *force-feedback* = fuerza.
- Dimensiona el *driver* y la fuente para el pico de consumo; la potencia sobrante es calor.
- La háptica comunica si es puntual, coherente y a tiempo; constante, se ignora.

TÉRMINOS DEL CAPÍTULO

Actuador, transductor de salida, LED, zumbador, motor de corriente continua, puente en H, servomotor, motor paso a paso, solenoide, relé, motor de masa excéntrica (ERM), actuador resonante lineal (LRA), háptica de banda ancha, retorno de fuerza (*force-feedback*), gatillo adaptativo, PWM, ciclo de trabajo, *driver* o etapa de potencia, transistor MOSFET, diodo de rueda libre, masa común, pico de consumo, lazo abierto, lazo cerrado.

PARA SABER MÁS

- Scherz y Monk [4], para los detalles prácticos de motores, *drivers* y etapas de potencia.
- Platt y Jansson [3], como catálogo de relés, solenoides, interruptores y transistores.
- Choi y Kuchenbecker [1], una panorámica de la tecnología y la percepción de la vibración táctil.

BIBLIOGRAFÍA DEL CAPÍTULO

[1] Seungmoon Choi y Katherine J. Kuchenbecker. «Vibrotactile Display: Perception, Technology, and Applications». En: *Proceedings of the IEEE* 101.9 (2013), págs. 2093-2104. DOI: 10.1109/JPROC.2012.2221071.

[2] Simon Monk. *Electronics Cookbook: Practical Electronic Recipes with Arduino and Raspberry Pi*. Sebastopol, CA: O'Reilly Media, 2017. ISBN: 978-1-491-95340-2.

[3] Charles Platt y Fredrik Jansson. *Encyclopedia of Electronic Components, Volume 1: Resistors, Capacitors, Inductors, Switches, Encoders, Relays, Transistors*. Sebastopol, CA: Maker Media, 2012. ISBN: 978-1-4493-3389-5.

[4] Paul Scherz y Simon Monk. *Practical Electronics for Inventors*. 4.ª ed. New York: McGraw-Hill Education, 2016. ISBN: 978-1-259-58754-2.

CAPÍTULO 8

Personalización y diseño de hardware para videojuegos

Los capítulos anteriores del Bloque II presentaron las piezas —dispositivos de entrada (capítulo 6) y actuadores (capítulo 7)—. Este capítulo cierra el bloque con el *proceso*: cómo pasar de una idea de interacción a un periférico construido, integrando también los sensores (capítulo 3) y el microcontrolador (capítulo 5).

Objetivos de aprendizaje

- Traducir una interacción deseada en requisitos técnicos de un periférico.
- Seleccionar sensores, actuadores, sistema de alimentación y microcontrolador de forma justificada.
- Describir el prototipado electrónico y las nociones básicas de diseño de PCB.
- Incorporar ergonomía, accesibilidad y diseño centrado en el usuario al proceso.

Alcance

Se recorre el proceso de diseño de un periférico personalizado con un enfoque de criterios y buenas prácticas. No se enseñan herramientas de esquemático ni reglas de trazado de PCB, ni electroquímica de baterías: el diseño de PCB y la elección de batería se tratan a nivel de concepto.

ÍNDICE DEL CAPÍTULO

8.1 DEL CONCEPTO A LOS REQUISITOS

8.1.1 Concepto de periférico personalizado

Un *periférico personalizado* es un dispositivo de entrada o salida hecho a medida para una interacción o un usuario concretos que un mando estándar no cubre bien. Se hacen para probar una mecánica nueva (un instrumento, un control físico), para **accesibilidad**, para una instalación artística o simplemente para aprender. Con el Bloque II ya están las piezas; falta el método para juntarlas [2].

8.1.2 Identificación de requisitos de interacción

Antes de tocar hardware hay que describir la **interacción** —qué hace el jugador, con qué parte del cuerpo, con cuánta fuerza y precisión, cuántas veces por minuto, en qué postura— y **para quién** —qué habilidades y en qué contexto—. De ahí salen los requisitos técnicos: qué magnitudes medir y en qué rango, qué salida hace falta, qué latencia y repetibilidad se toleran, cuánta autonomía, qué tamaño y peso, con o sin cable, y qué presupuesto y plazo.

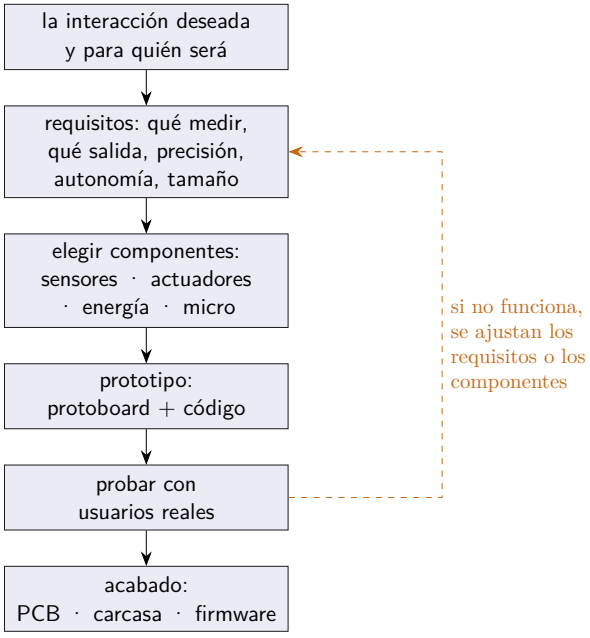


Figura 8.1. El proceso de diseño de un periférico. Los requisitos casi nunca son correctos a la primera: hay que prototipar, probar con usuarios reales y volver atrás a ajustar.

El proceso es iterativo (figura 8.1): los requisitos malos se descubren probando, así que se prototipa pronto, se prueba con gente y se corrige.

8.2 ARQUITECTURA Y SELECCIÓN DE COMPONENTES

8.2.1 Arquitectura hardware

El primer paso técnico es dibujar el diagrama de bloques del periférico (figura 8.2): qué sensores van a qué entradas del microcontrolador, qué actuadores con qué *drivers*, de dónde sale la energía y cómo se comunica con el juego. Es la cadena del capítulo 1 aplicada a un aparato concreto.

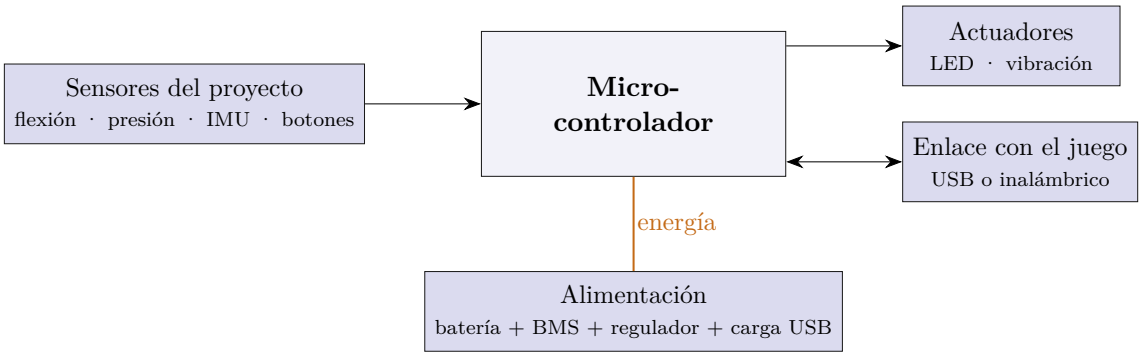


Figura 8.2. Arquitectura de un periférico personalizado: los sensores del proyecto entran en el microcontrolador, que gobierna los actuadores y habla con el juego; todo se alimenta de un mismo sistema de energía.

8.2.2 Selección de sensores

Los criterios, recogiendo el capítulo 3: qué magnitud y en qué rango; salida analógica, digital o por bus; resolución y tiempo de respuesta suficientes; tensión de alimentación y consumo; tamaño, montaje, precio y disponibilidad; y si hay una librería hecha. Por tipo de acción: agarrar → sensor de fuerza o de flexión; inclinar o agitar → IMU; girar → potenciómetro o *encoder*; acercarse → infrarrojo o ultrasónico; soplar → sensor de presión.

8.2.3 Selección de actuadores

Criterios, recogiendo el capítulo 7: qué efecto (luz, vibración, fuerza, movimiento, sonido); lazo abierto o cerrado; corriente y tensión, que fijan el *driver*; ruido acústico; y consumo, que afecta a la batería. Muchos periféricos no necesitan más actuador que un LED de estado.

8.2.4 Selección del sistema de alimentación (baterías)

Tres opciones básicas: alimentación por **USB** (sin batería, más simple, con cable); **pilas** sustituibles (NiMH o alcalinas); o **batería recargable integrada** (Li-ion o LiPo, más densidad de energía y sin residuos, pero requiere electrónica de protección).

De una batería importan la **capacidad** (en mA h), la tensión por celda (3,7 V en Li-ion), el *C-rate* (a qué ritmo admite carga y descarga), la curva de descarga (la tensión cae con el uso) y los ciclos de vida. Toda batería de litio lleva un *BMS* que la protege de sobrecarga, sobredescarga, cortocircuito y temperatura, y un regulador que convierte su tensión variable en la fija que necesita la electrónica (capítulo 2). La carga por USB sigue un perfil de corriente constante y luego tensión constante.

> **Aviso**
>
> Las baterías de litio (LiPo) son delicadas: no las perfores ni las aplastes, no las cortocircuites, no las cargues sin vigilancia y retíralas de inmediato si se hinchan o se calientan.

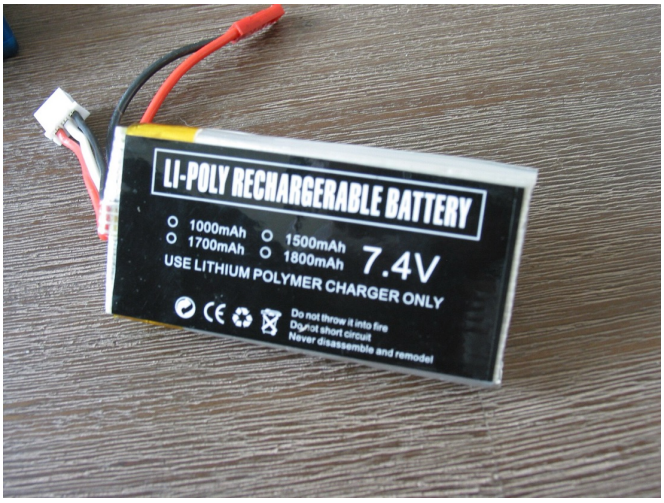


Figura 8.3. Una batería LiPo con su placa de protección (BMS) soldada, y un módulo de carga por USB. La batería nunca se usa sin esa electrónica. *(foto: Daniel van den Ouden · CC BY 2.0)*

8.2.5 Selección del microcontrolador

Criterios, recogiendo el capítulo 5: número de entradas y salidas, y cuántas analógicas; si hace falta un ADC de más bits; si necesita radio (entonces un ESP32) o le basta USB; si tiene potencia y memoria para el filtrado y el protocolo previstos; su consumo; si se conoce y tiene comunidad y librerías; y el tamaño de la placa. Para prototipar casi siempre sirve una placa Arduino o similar; para un producto, a veces se pasa al chip suelto.

8.3 DISEÑO Y PROTOTIPADO ELECTRÓNICO

8.3.1 Diseño de circuitos sencillos

El circuito de un periférico suele reducirse a: el microcontrolador, resistencias (para LEDs, *pull-up* y divisores), condensadores de desacoplo, un *driver* por cada actuador de potencia y un conector de alimentación. Conviene dibujar el esquema antes de montar nada.

8.3.2 Prototipado, *breadboard* y conexiones

La *protoboard* tiene orificios conectados en tiras y permite montar y rehacer sin soldar: ideal para probar, mala para algo definitivo (los contactos se aflojan y captan ruido). Buenas prácticas al montar: cables cortos, colores coherentes (rojo para el positivo, negro para la masa), una línea de masa común bien hecha, un condensador de desacoplo junto al microcontrolador y junto a cada *driver*, y **nunca** alimentar un motor desde el riel de la lógica (capítulo 7). Del protoboard se pasa a una placa perforada soldada o a una PCB.

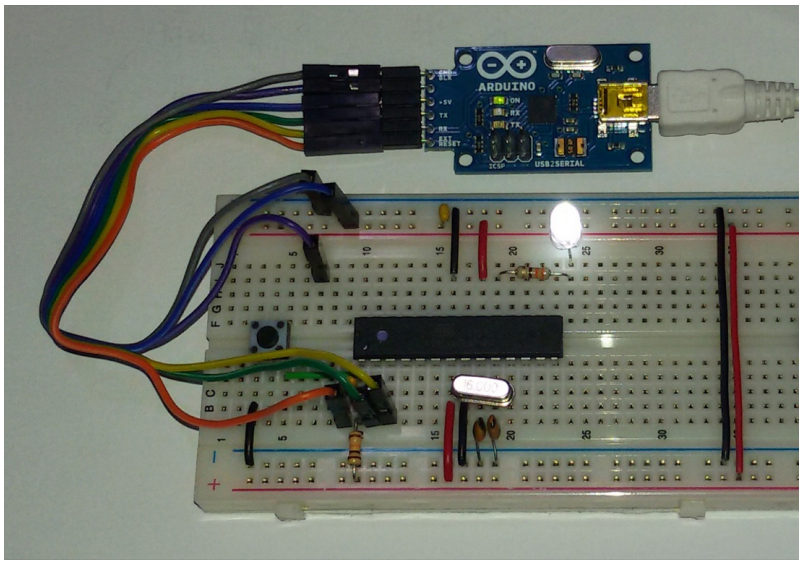


Figura 8.4. Un prototipo en protoboard: la placa Arduino, un sensor, unas resistencias y los cables de colores que unen todo sin soldar. *(foto: Dolicom · CC BY-SA 3.0)*

8.3.3 Introducción al diseño de PCB

Una PCB es una placa en la que las conexiones son pistas de cobre grabadas: fiable, compacta y repetible. Se diseña con un programa —del esquema a la colocación de componentes y al trazado de pistas— y se fabrica fuera. Solo compensa cuando el diseño ya está probado y se van a hacer varias unidades; para una pieza única, la placa perforada basta.

8.3.4 Alimentación y autonomía

La autonomía se estima con un *presupuesto de energía*: se suma el consumo de cada subsistema y se divide la capacidad de la batería entre ese consumo medio (tabla 8.1).

Tabla 8.1. Presupuesto de energía de ejemplo para un mando inalámbrico sencillo.

Subsistema	**Consumo aprox.**
Microcontrolador y sensores	15 mA
Radio (media, con envíos)	25 mA
LED de estado	5 mA
Vibración (solo mientras actúa)	90 mA
Consumo medio en juego (sin vibración continua)	$\approx$ 50 mA
Autonomía con una batería de 500 mA h	$\approx$ 10 h

Para alargarla se usan modos de bajo consumo: dormir entre lecturas, apagar la radio entre paquetes, atenuar los LEDs y apagar el dispositivo por inactividad.

8.4 ASPECTOS FÍSICOS Y DE USUARIO

8.4.1 Carcasas y componentes físicos

La carcasa protege la electrónica, la sujeta y da forma a la interacción. Puede hacerse por impresión 3D, corte láser, con cajas comerciales o con materiales reaprovechados. Hay que prever la sujeción de la placa, el acceso al conector de carga, la ventilación si hay potencia y un interruptor de encendido. Los componentes que toca el jugador —botones, palancas, superficies— importan tanto como la electrónica: su recorrido, su resistencia, su tacto y su tamaño.

8.4.2 Ergonomía

La *ergonomía* adapta el dispositivo al cuerpo: postura neutra, controles al alcance sin forzar, fuerza de accionamiento razonable, tamaño y separación adecuados, peso y equilibrio, y un buen agarre. En sesiones largas, las posturas forzadas, los botones duros y los gestos amplios repetidos producen fatiga.

8.4.3 Accesibilidad

La accesibilidad es un requisito desde el principio, no un parche, y abarca la diversidad motora, visual, auditiva y cognitiva. Estrategias habituales: reducir el número de entradas necesarias, usar entradas grandes y de bajo esfuerzo, permitir temporizaciones configurables, sustituir un eje analógico por pulsadores, y admitir control por soplido, voz o mirada (capítulo 10) con remapeo total.

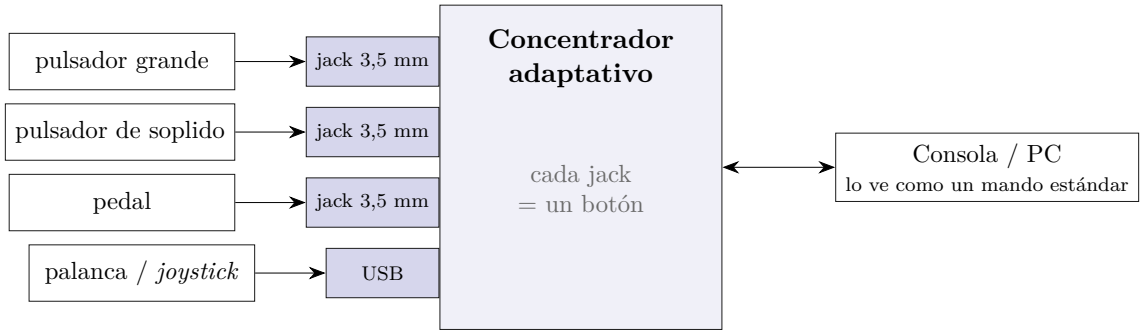


Figura 8.5. Un concentrador adaptativo (como el Xbox Adaptive Controller) ofrece muchas entradas conmutadas: cada jack de 3,5 mm equivale a un botón, y por los puertos USB se conectan palancas. El jugador arma su propia interfaz con pulsadores, pedales y palancas de terceros, y la consola lo ve como un mando estándar.

El Xbox Adaptive Controller es un *concentrador* de este tipo (figura 8.5); el Logitech Adaptive Gaming Kit y el QuadStick son piezas del mismo ecosistema. En el juego, la accesibilidad se apoya además en menús de opciones y modos de un botón (capítulo 6) [1].

8.4.4 Diseño centrado en el usuario

El *diseño centrado en el usuario* consiste en involucrar a las personas que van a usar el dispositivo desde el principio y en cada iteración: observar, prototipar barato, probar y ajustar —el lazo de la figura 8.1—. En accesibilidad es imprescindible: nada sobre el usuario sin el usuario [3].

8.5 MODIFICACIÓN DE PERIFÉRICOS COMERCIALES

8.5.1 Modificación de periféricos comerciales

El modding es abrir un mando o periférico existente y cambiarlo: botones distintos, palancas, gatillos, pesos, o cablear entradas hacia conectores externos (el mando «adaptado»). La ventaja es aprovechar la carcasa, la ergonomía base, el enlace y la compatibilidad; el inconveniente, el poco espacio interno, la garantía y, a veces, medidas anti-manipulación. Hay que cuidar los cortocircuitos y el daño a la placa, y tener presente que los *mod-chips* y los firmware alternativos pueden vulnerar los términos de servicio o dar ventaja injusta: aquí solo tratamos la modificación física legítima.

8.5.2 Integración de sensores en dispositivos existentes

La vía más rápida para un «controlador raro» es añadir un sensor a un objeto que no es un mando —una raqueta, un volante de juguete, una figura, una prenda—. Un microcontrolador pequeño (o un ESP32, para que sea inalámbrico) lee el sensor y presenta el objeto al juego como un mando estándar o por el puerto serie (capítulo 9).

8.6 PROTOTIPO DE UN PERIFÉRICO PARA VIDEOJUEGOS

Reuniéndolo todo con un caso: se quiere «inclinar un objeto para dirigir». El proceso sería: requisitos (medir inclinación, sin salida más que un LED, latencia baja, inalámbrico, una sesión de autonomía) → componentes (IMU + ESP32 + batería pequeña con BMS + LED) → prototipo en protoboard con el código que lee la IMU, la filtra y la envía → carcasa impresa → pruebas con jugadores → iterar. La parte de comunicación con el motor de juego es el capítulo 9.

Este caso reúne todo el capítulo: un sensor sobre un microcontrolador, con su código, enviando datos a un motor de videojuego, con memoria y defensa.

Una lista de comprobación para dar el prototipo por terminado: responde igual y a tiempo, aguanta una sesión de juego, es cómodo, se entiende sin manual y lo ha probado alguien que no lo construyó.

RESUMEN

Diseñar un periférico personalizado es un proceso iterativo que empieza por la interacción deseada y por quién la hará, y de ahí deriva los requisitos técnicos: qué medir, qué salida, latencia, repetibilidad, autonomía, tamaño. Con los requisitos se dibuja la arquitectura —sensores, microcontrolador, actuadores, energía, enlace— y se eligen los componentes con criterios claros: el sensor por magnitud, rango y tipo de salida; el actuador por el efecto y la potencia; la alimentación entre USB, pilas o batería de litio (siempre con BMS y regulador); y el microcontrolador por entradas, radio, potencia y comunidad. Se prototipa en protoboard con buenas prácticas de conexión, y solo se pasa a PCB cuando el diseño está probado y hay varias unidades. La autonomía se estima con un presupuesto de energía. La carcasa, la ergonomía y, sobre todo, la accesibilidad —con concentradores como el Xbox Adaptive Controller— forman parte del diseño desde el principio, y se validan probando con usuarios reales. Modificar un periférico comercial o instrumentar un objeto cotidiano son atajos frecuentes para llegar antes a un prototipo.

Ideas clave

- Primero la interacción y para quién; los requisitos técnicos salen de ahí.
- Elegir cada componente con criterios: rango y salida (sensor), efecto y potencia (actuador), radio y consumo (micro).
- Una batería de litio nunca va sin BMS; la autonomía se calcula con un presupuesto de energía.
- El protoboard es para probar; la PCB, para cuando el diseño está cerrado y hay varias unidades.
- Ergonomía y accesibilidad desde el principio, y se validan probando con usuarios reales.

TÉRMINOS DEL CAPÍTULO

Periférico personalizado, requisitos de interacción, arquitectura hardware, selección de componentes, sistema de alimentación, pila, batería de litio (Li-ion / LiPo), capacidad (mA h), C-rate, curva de descarga, BMS, regulador de tensión, presupuesto de energía, modo de bajo consumo, protoboard (*breadboard*), placa perforada, PCB, esquemático, carcasa, ergonomía, accesibilidad, concentrador adaptativo, diseño centrado en el usuario, *modding*, instrumentación de objetos.

PARA SABER MÁS

- Igoe [2] y O'Sullivan e Igoe [3] cubren el diseño de dispositivos físicos interactivos de principio a fin.
- Game Accessibility Guidelines Working Group [1], una guía práctica y por niveles de accesibilidad en juegos.
- Yuan, Folmer y Harris [4], una panorámica académica de la accesibilidad en videojuegos.

BIBLIOGRAFÍA DEL CAPÍTULO

[1] Game Accessibility Guidelines Working Group. *Game Accessibility Guidelines*. 2024. URL: https://gameaccessibilityguidelines.com (visitado 08-09-2026).

[2] Tom Igoe. *Making Things Talk: Using Sensors, Networks, and Arduino to See, Hear, and Feel Your World*. 3.ª ed. San Francisco, CA: Make Community, 2017. ISBN: 978-1-68045-315-6.

[3] Dan O'Sullivan y Tom Igoe. *Physical Computing: Sensing and Controlling the Physical World with Computers*. Boston, MA: Thomson Course Technology, 2004. ISBN: 978-1-59200-346-1.

[4] Bei Yuan, Eelke Folmer y Frederick C. Harris. «Game accessibility: a survey». En: *Universal Access in the Information Society* 10.1 (2011), págs. 81-100. DOI: 10.1007/s10209-010-0189-5.

BLOQUE III

NUEVAS TECNOLOGÍAS EN LOS SISTEMAS DE ADQUISICIÓN Y CONTROL DE SOFTWARE

CAPÍTULO 9

Comunicación e integración hardware-software

El Bloque III conecta el hardware con el software. Este capítulo cierra la cadena del capítulo 1 por su parte central: cómo el dato sale del microcontrolador (capítulo 5), viaja por un cable o por el aire, y llega a un motor de videojuego que lo usa.

Objetivos de aprendizaje

- Distinguir los conceptos de dato, protocolo y trama.
- Describir los principales buses cableados (serie, UART, I^2C, SPI, USB) y sus usos.
- Comparar Bluetooth y Wi-Fi y los compromisos de la comunicación inalámbrica.
- Explicar la integración de un microcontrolador con un motor de videojuego y el papel de la latencia y la sincronización.

Alcance

Se ve qué bus hay detrás de cada función y por qué, con un enfoque funcional. No hay cronogramas de bits, ni tramas a nivel de campo, ni cálculos de enlace de radio.

ÍNDICE DEL CAPÍTULO

9.1 FUNDAMENTOS DE COMUNICACIÓN ENTRE DISPOSITIVOS

9.1.1 Comunicación entre dispositivos

Comunicar dos dispositivos es tener un **medio** —un cable o el aire— y unas **reglas** para usarlo. Las reglas tienen dos capas: la *física* (qué hilos, qué tensiones, qué frecuencia) y la *lógica* (cómo se ordenan los bits, quién habla cuándo, cómo se detectan errores) [3].

9.1.2 Datos y protocolos

Un *protocolo* es el conjunto de reglas que dos dispositivos acuerdan para entenderse: la velocidad, el formato de los datos, quién inicia la conversación y qué respuesta se espera. Hay protocolos de *bajo nivel*, que dicen cómo se transmite un byte (UART), y de *alto nivel*, que dicen qué significan los bytes (HID, o un formato propio).

9.1.3 Paquetes de datos

Los datos no se envían a chorro, sino en *tramas* (o paquetes) con estructura fija (figura 9.1): una **cabecera** (marca de inicio, identificador, longitud), una **carga útil** (los valores: ejes, botones) y una **comprobación** —un checksum— que el receptor recalcula para descartar lo que llega corrupto.

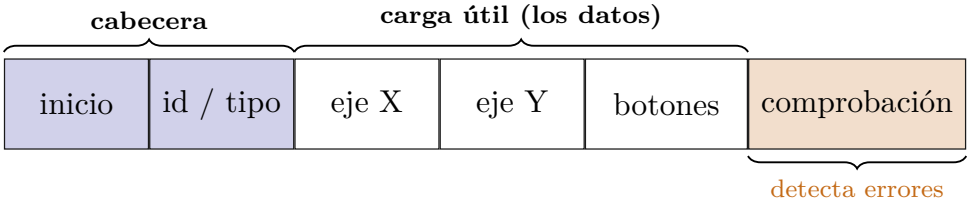


Figura 9.1. Anatomía de una trama. La cabecera dice dónde empieza el envío y de qué tipo es; la carga útil son los datos; la comprobación permite detectar si la trama llegó dañada.

La comunicación es casi siempre **serie** (los bits van uno tras otro por un hilo) y puede ser **síncrona** (con una línea de reloj compartida) o **asíncrona** (sin reloj: ambos extremos pactan la velocidad de antemano).

9.2 BUSES Y COMUNICACIÓN CABLEADA

9.2.1 Comunicación serie y UART

La *comunicación serie* envía los bits uno tras otro. El **UART** es su forma más simple: dos hilos de datos (TX y RX, cruzados entre los dos dispositivos) más la masa común. Es *asíncrono*: no hay reloj, así que ambos extremos se ponen de acuerdo en la velocidad, en *baudios* (9 600, 115 200…). Es punto a punto. Un Arduino usa UART para hablar con el PC: un chip convierte UART en USB, y en el ordenador aparece como un puerto serie (capítulo 5).

9.2.2 I^2C y SPI

Son buses *internos* del dispositivo: conectan el microcontrolador con la IMU, las memorias, los expansores de pines y las pantallas (capítulos 3 y 6).

- **I^2C**: dos hilos compartidos —datos y reloj— más la masa. Varios dispositivos en el mismo bus, cada uno con una **dirección**. Lento a medio, con muy pocos hilos.
- **SPI**: tres hilos comunes —reloj, salida y entrada— más **una línea de selección por dispositivo**, más la masa. Rápido, a cambio de más hilos.

9.2.3 USB

El **USB** es el bus universal para conectar periféricos a un ordenador o una consola. Lleva energía y datos por el mismo cable (capítulo 2), con conector reversible (USB-C). Al enchufar, el dispositivo se identifica —*enumeración*— y el sistema carga el soporte adecuado.

La **clase HID** (*Human Interface Device*) es un «idioma común» para teclados, ratones y mandos: el dispositivo describe sus ejes y botones en un *descriptor* y funciona **sin instalar controladores**. Un mando puede presentarse como HID; un Arduino, por defecto, se presenta como un puerto serie, y con la librería adecuada, también como HID [1].

Tabla 9.1. Los buses cableados de uso habitual.

Bus	Hilos	Estructura	Uso típico
UART	2 datos + masa	punto a punto, sin reloj	Arduino ↔ PC
I^2C	2 + masa	bus compartido, por dirección	IMU, memorias, pantallas
SPI	3 + 1 por dispositivo + masa	bus compartido, por selección	sensores y pantallas rápidos
USB	2 datos + alimentación	clases (HID, serie…)	conectar periféricos al ordenador

9.3 COMUNICACIÓN INALÁMBRICA

9.3.1 Fundamentos y compromisos

Quitar el cable significa montar los datos sobre una onda de radio. La banda de 2,4 GHz es libre y está muy concurrida (Wi-Fi, microondas, otros mandos), de modo que hay interferencias; el alcance depende de la potencia y de los obstáculos. Todo enlace inalámbrico es un **compromiso entre latencia, consumo y alcance**: no se pueden optimizar los tres a la vez. Antes de comunicar hay que *emparejar* los dispositivos y, normalmente, cifrar el enlace para que nadie lea las pulsaciones.

9.3.2 Bluetooth, Wi-Fi y *dongles* propietarios

- **Bluetooth**: *Classic* (más ancho de banda, audio) y *BLE* (bajo consumo). Con el perfil HID, un mando genérico funciona sin cable en móvil, PC y consola. Su latencia es algo mayor y más variable que la de un cable.

- **Wi-Fi**: mucho ancho de banda y alcance, pero más consumo y latencia; se usa en el mando de algunas consolas y en el juego en la nube (capítulo 10). Un ESP32 lo lleva integrado (capítulo 5).
- Dongles *de baja latencia* (Xbox Wireless, Lightspeed): un receptor propio en un puerto USB con un protocolo optimizado en la banda de 2,4 GHz, con menos latencia y menos *jitter* que Bluetooth, a cambio de necesitar ese receptor.
- **NFC** (amiibo): alcance de centímetros, la etiqueta no lleva batería. **Infrarrojos**: caso histórico (barra del Wiimote), con línea de visión.

Tabla 9.2. Compromisos de las tecnologías inalámbricas para mandos.

Tecnología	Latencia	Consumo	Alcance
Bluetooth (BLE / HID)	media, variable	bajo	sala
Dongle 2,4 GHz	baja, estable	medio	sala
Wi-Fi	media-alta	alto	vivienda y más
NFC	–	nulo (etiqueta)	centímetros

9.4 INTEGRACIÓN CON SOFTWARE Y MOTORES

9.4.1 APIs e interfaces de comunicación

Una *API* es el conjunto de funciones que un sistema ofrece para que otros programas lo usen. El sistema operativo expone los mandos mediante una API, y el motor de juego añade otra por encima. En Windows conviven **XInput** (mandos «tipo Xbox», sencillo) y **DirectInput** (cualquier dispositivo, más general); en Linux, **evdev**; en consola, la API del fabricante.

9.4.2 Del microcontrolador al motor de videojuego

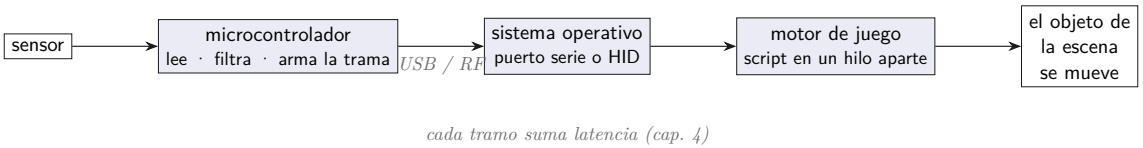


Figura 9.2. El camino de un dato desde el sensor hasta el objeto de la escena. El motor lee el puerto serie (o el dispositivo HID) desde un script, en un hilo aparte para no frenar el bucle de juego.

Un Arduino conectado por USB aparece como un **puerto serie**; el motor lo lee con la librería de puerto serie del lenguaje, interpreta la trama y actualiza el estado (figura 9.2). La otra opción es que el Arduino se presente como **HID** y entonces el juego lo vea como un mando, sin código extra. En cualquier caso hay que fijar un **protocolo de aplicación** sencillo: qué formato tiene cada envío (por ejemplo `ejeX,ejeY,botones\n`), a qué velocidad y cómo se reconecta si se desenchufa.

9.4.3 Unity, Unreal y otros motores

En **Unity**, el *Input System* unifica mando, teclado y táctil en «acciones»; para un dispositivo propio por serie se usa la clase de puerto serie de .NET o un plugin, o se declara un dispositivo de entrada personalizado [4]. En **Unreal**, el sistema de *Input* con *mappings* recibe los dispositivos HID directamente, y para serie se usa un plugin o un módulo en C++. En **Godot** y en los *frameworks* web el patrón es el mismo: leer el puerto serie, o un dispositivo HID/*gamepad*. La idea de fondo: si presentas tu periférico como HID, el motor no lo distingue de un mando comercial; si vas por serie, el «pegamento» lo pones tú [2].

> **Conexión con el diseño de videojuegos**
>
> Un montaje mínimo de extremo a extremo: un Arduino con un sensor envía una trama por el puerto serie y un script de Unity mueve un objeto de la escena en tiempo real.

Figura 9.3. Montaje de extremo a extremo: una placa Arduino con un sensor, conectada por un cable USB al ordenador donde corre el motor de juego.

9.5 RENDIMIENTO DE LA COMUNICACIÓN

9.5.1 Latencia y sincronización

La comunicación es un tramo del presupuesto de latencia (capítulo 4): un cable añade del orden de 1 ms; Bluetooth, 5 a 15 ms y de forma variable; el juego en la nube, mucho más. Además, el mando envía a su ritmo y el juego lee al suyo (por fotogramas): si no encajan, se pierden o se repiten muestras. Las marcas de tiempo (capítulo 4) permiten saber cuándo se tomó cada dato, y un pequeño *buffer* suaviza el *jitter* a costa de algo de retardo.

9.5.2 Frecuencia de actualización

Cada eslabón tiene su ritmo: el microcontrolador lee el sensor a una frecuencia, envía a otra, el sistema operativo sondea a otra y el juego corre a sus fotogramas por segundo. La cadena va al ritmo del eslabón más lento, y enviar más deprisa de lo que el juego consume solo gasta ancho de banda y energía.

9.5.3 Arquitecturas de comunicación para sistemas interactivos

Patrones habituales: **sondeo** (el juego pregunta el estado cada fotograma) frente a **eventos** (el dispositivo avisa de cada cambio); **estado completo** en cada trama (robusto ante pérdidas) frente a **solo cambios** (menos datos, más frágil); y un **hilo dedicado** a la comunicación, separado del bucle de juego. Para un periférico propio y local lo habitual es una trama con el estado completo, a una frecuencia fija cómoda (del orden de 100 a 250 Hz), leída en un hilo aparte y con reconexión automática. En sistemas distribuidos —varios jugadores, juego en la nube— la red añade latencia y pérdidas, que se compensan con predicción y reconciliación (capítulo 10).

RESUMEN

Comunicar dos dispositivos es tener un medio y unas reglas —un protocolo— con una capa física y otra lógica. Los datos viajan en tramas con cabecera, carga útil y una comprobación de errores, casi siempre en serie, de forma síncrona o asíncrona. Entre los buses cableados, el UART es punto a punto y sin reloj (Arduino con el PC); I^2C y SPI son buses internos que conectan el microcontrolador con sensores y pantallas, el primero por dirección y con dos hilos, el segundo por selección y más rápido; y el USB es el bus universal para periféricos, que lleva energía y datos y, mediante la clase HID, hace que un mando funcione sin controladores. Sin cable, los datos van por radio en la banda de 2,4 GHz, y cada tecnología —Bluetooth, dongle propietario, Wi-Fi, NFC— es un compromiso distinto entre latencia, consumo y alcance. El sistema operativo expone los mandos por una API (XInput, DirectInput, evdev) y el motor de juego por otra; un microcontrolador se integra presentándose

como HID —y entonces es transparente— o por el puerto serie, definiendo un protocolo de aplicación propio y leyéndolo en un hilo aparte. En rendimiento, la comunicación suma latencia y su variación, hay que sincronizar el ritmo del mando con el de los fotogramas, y el diseño habitual para un periférico local es enviar el estado completo a una frecuencia fija con reconexión automática.

Ideas clave

- Un protocolo son las reglas para entenderse; una trama, el bloque de datos con cabecera, carga útil y comprobación.
- UART = punto a punto sin reloj; I^2C = bus por dirección; SPI = bus por selección, más rápido; USB = universal, con clases como HID.
- HID hace que un mando funcione sin controladores; por serie, el pegamento con el motor lo pones tú.
- Todo enlace inalámbrico es un compromiso entre latencia, consumo y alcance.
- La cadena de comunicación va al ritmo del eslabón más lento; enviar de más no ayuda.

TÉRMINOS DEL CAPÍTULO

Medio y reglas, capa física y capa lógica, protocolo, trama (cabecera, carga útil, comprobación), *checksum*, comunicación serie, síncrono y asíncrono, UART, baudios, I^2C, SPI, línea de selección, USB, enumeración, clase HID, descriptor, banda de 2,4 GHz, emparejamiento, Bluetooth Classic y BLE, Wi-Fi, *dongle* de baja latencia, NFC, API, XInput, DirectInput, evdev, puerto serie virtual, protocolo de aplicación, hilo dedicado, sondeo frente a eventos, estado completo frente a cambios, sincronización, *buffer*.

PARA SABER MÁS

- Igoe [3] cubre la comunicación entre dispositivos y con el ordenador, con proyectos.
- Axelson [1], como referencia sobre USB y la clase HID.

- Unity Technologies [4], la documentación del sistema de entrada de Unity.

BIBLIOGRAFÍA DEL CAPÍTULO

[1] Jan Axelson. *USB Complete: The Developer's Guide*. 5.ª ed. Madison, WI: Lakeview Research, 2015. ISBN: 978-1-931448-28-4.

[2] Jason Gregory. *Game Engine Architecture*. 3.ª ed. Boca Raton, FL: CRC Press, 2018. ISBN: 978-1-138-03545-4.

[3] Tom Igoe. *Making Things Talk: Using Sensors, Networks, and Arduino to See, Hear, and Feel Your World*. 3.ª ed. San Francisco, CA: Make Community, 2017. ISBN: 978-1-68045-315-6.

[4] Unity Technologies. *Input System*. 2024. URL: https://docs.unity3d.com/Packages/com.unity.inputsystem@latest (visitado 08-09-2026).

CAPÍTULO 10

Nuevas tecnologías de adquisición e interacción

Este capítulo es una panorámica de las tecnologías que están ampliando lo que un sistema puede medir y cómo una persona puede interactuar con él. Casi todas son extensiones de lo ya visto —sensores (capítulo 3), actuadores (capítulo 7) y comunicación (capítulo 9)— aplicadas de formas nuevas. El objetivo no es dominar cada una, sino entender a grandes rasgos cómo funcionan, qué permiten y qué limitan, para poder elegirlas con criterio.

Objetivos de aprendizaje

- Explicar los principios de IoT, *wearables*, realidad virtual y aumentada.
- Describir el seguimiento de posición y movimiento y la captura de movimiento.
- Comprender el papel de la visión artificial, el reconocimiento de gestos y las interfaces de voz.
- Valorar posibilidades y limitaciones de estas tecnologías en videojuegos.

Alcance

Visión técnica y aplicada, honesta sobre lo que ya existe y lo que aún es promesa. No se entra en algoritmos de visión, filtros de fusión ni redes; la parte de aprendizaje automático se trata en el capítulo 11.

ÍNDICE DEL CAPÍTULO

10.1 DISPOSITIVOS CONECTADOS Y LLEVABLES

10.1.1 Internet de las cosas

El IoT son objetos cotidianos con sensores, un microcontrolador y conexión de red, que envían datos y reciben órdenes. Un ESP32 conectado por Wi-Fi (capítulos 5 y 9) ya es un dispositivo IoT. En juego aparece como mandos y juguetes conectados, tableros físicos, instalaciones y juegos que reaccionan a datos del entorno.

10.1.2 Dispositivos conectados y *wearables*

El patrón es siempre *sensor* → *micro* → *red* → *servicio* → *app o juego*. Conectar abre posibilidades —estado compartido, actualización, telemetría— y problemas —dependes de la red, latencia, seguridad, privacidad—.

Un wearable se lleva puesto: pulsera, reloj, anillo, banda, prenda. Usa los sensores del capítulo 3: IMU (pasos, gestos), pulso óptico, conductancia de la piel, temperatura, a veces EMG. En juego sirve para *exergames*, para juegos que reaccionan a tu estado y para el control por gesto de muñeca. Sus límites: batería minúscula, señales pequeñas y ruidosas, y el propio movimiento del cuerpo como fuente de ruido [2].

Figura 10.1. Una pulsera de actividad: dentro lleva una IMU, un sensor óptico de pulso, una batería diminuta y radio para enviar los datos al móvil. *(foto: PamD · CC BY-SA 4.0)*

10.1.3 Sensores portátiles

Son módulos de sensor autónomos —con su batería y su radio— que se colocan en un objeto o en el cuerpo y transmiten. Permiten instrumentar cualquier cosa sin cables (capítulo 8).

10.2 REALIDAD EXTENDIDA Y SEGUIMIENTO

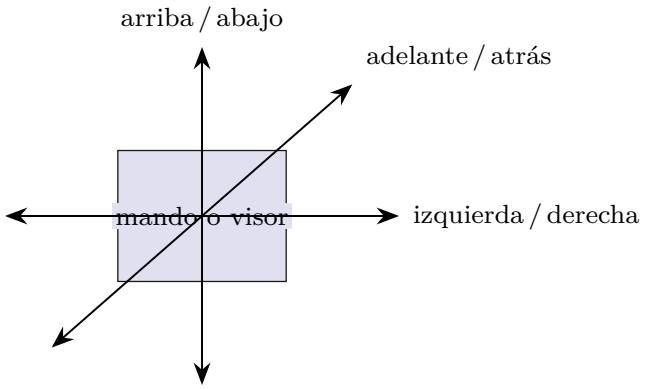
10.2.1 Realidad virtual y aumentada

En **realidad virtual** un visor sustituye la vista y el oído por un mundo generado; necesita imagen estéreo, baja latencia y frecuencia alta para no marear. En **realidad aumentada** se superponen elementos virtuales sobre el mundo real, a través de una pantalla o de gafas transparentes, lo que exige saber dónde está el usuario y qué hay a su alrededor. El término *realidad extendida* (XR) engloba ambas y la mixta [3].

Figura 10.2. Un visor de realidad virtual y sus dos mandos. Cada mando lleva una IMU, botones, un gatillo y elementos que el sistema de seguimiento localiza en el espacio con seis grados de libertad. *(foto: PB · CC BY-SA 4.0)*

10.2.2 Seguimiento de posición y movimiento

El seguimiento (*tracking*) da la posición y la orientación de un mando o del usuario. Con *seis grados de libertad* (figura 10.3) se conocen las tres coordenadas de posición y las tres de orientación.



3 de posición (moverse) + **3 de orientación** (girar sobre cada eje)

= 6 grados de libertad

Figura 10.3. Seis grados de libertad: moverse en tres direcciones y girar sobre tres ejes. Un acelerómetro y un giróscopo dan la orientación; la posición necesita una referencia externa.

Hay dos enfoques (figura 10.4): el outside-in, con estaciones fijas en la sala que observan al usuario —preciso, pero atado a un espacio montado—, y el inside-out, con cámaras en el propio visor que observan la sala —portátil, pero dependiente de que haya rasgos visibles y buena luz—. Ambos combinan óptica e IMU (capítulo 3): la óptica corrige la deriva de la IMU y la IMU rellena los huecos entre fotogramas de la cámara [4].

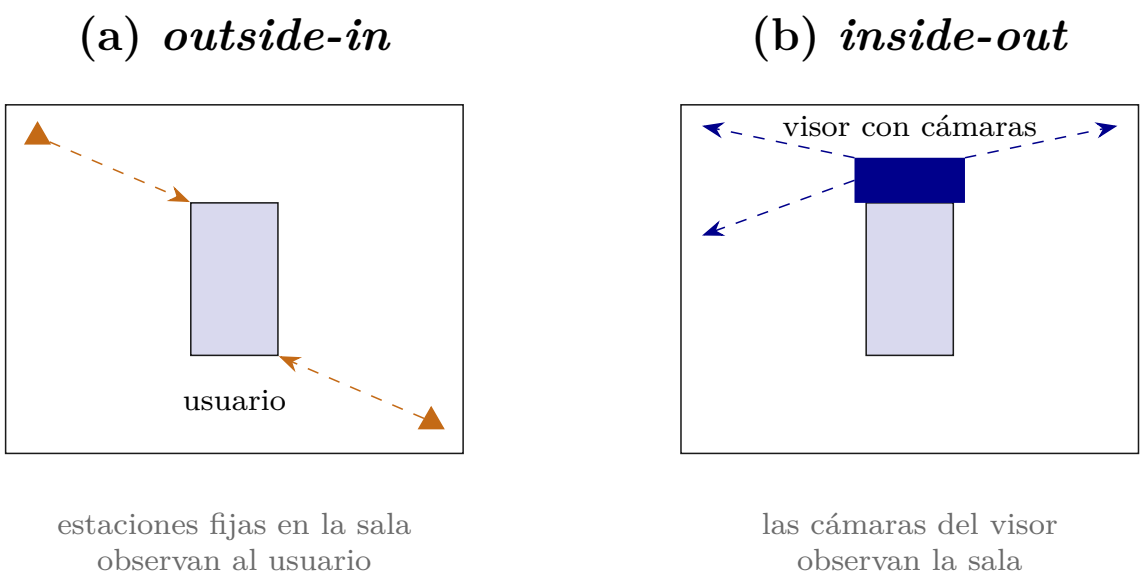


Figura 10.4. *Outside-in*: cámaras o estaciones fijas en la sala siguen al usuario. *Inside-out*: las cámaras van en el visor y miran la sala para deducir su propia posición.

10.2.3 Captura de movimiento e IMU

La *captura de movimiento* registra el movimiento del cuerpo entero: con marcadores y cámaras (cine y estudios), con un traje de sensores inerciales, o con cámaras de profundidad sin traje (menos preciso). Sirve para animar personajes y, en tiempo real, para avatares que imitan al jugador. En todos los casos la IMU (capítulo 3) aporta la orientación rápida y rellena los huecos del seguimiento óptico, pero la posición absoluta se pierde por deriva, así que siempre hace falta una referencia externa.

10.3 VISIÓN, GESTOS Y VOZ

10.3.1 Visión artificial

La *visión artificial* extrae información de imágenes: detectar una cara, unas manos, un cuerpo, un objeto, un código, o la distancia (con dos cámaras o con una de profundidad). Permite jugar sin mando, sostener la realidad aumentada y animar

avatares. A cambio es sensible a la luz y a la oclusión, exige cálculo y plantea un problema de privacidad: una cámara siempre encendida (capítulo 11) [6].

10.3.2 Reconocimiento de gestos

Interpretar un movimiento —de la mano, del cuerpo o del mando— como un comando. Se apoya en la IMU, en una cámara o en el seguimiento de manos, y normalmente en un modelo entrenado (capítulo 11). Sus problemas, ya vistos (capítulo 6): falsos positivos y negativos, fatiga y dificultad para descubrir qué gestos existen. Funciona bien como complemento.

10.3.3 Interfaces de voz

El micrófono (capítulo 3) más un reconocedor de voz da comandos, dictado y conversación, con las manos libres. Sus inconvenientes: el ruido ambiente, los acentos, la latencia —a menudo se procesa en la nube, con lo que implica para la privacidad— y que hablar en voz alta no siempre es apropiado.

10.4 INTERACCIÓN TÁCTIL, HÁPTICA Y MULTIMODAL

10.4.1 Interfaces táctiles avanzadas

Más allá de tocar y deslizar (capítulo 6): multitáctil con muchos dedos, detección de la fuerza con que se aprieta y del área de contacto, superficies táctiles curvas o flexibles, y *trackpads* con háptica que fingen un clic donde no hay ninguno.

10.4.2 Sistemas hápticos

De la vibración simple a la háptica de banda ancha y el retorno de fuerza (capítulo 7), llevados al cuerpo entero: chalecos y trajes con muchos actuadores,

guantes que resisten al cerrar la mano, actuadores de temperatura y **ultrasonidos** que crean sensaciones táctiles en el aire, sin contacto. El objetivo es aumentar la sensación de presencia; el límite, el coste, el peso y que demasiada información háptica satura [1].

10.4.3 Interfaces multimodales

Una *interfaz multimodal* combina varias entradas a la vez —mirar un objeto y decir «coge eso», señalar y hablar, gesto y voz—. Los datos de cada canal se *fusionan* para interpretar la intención (capítulo 11), y cada canal cubre las carencias del otro: la voz es ambigua en el «qué», la mirada lo resuelve. Es como funciona la comunicación entre personas; la dificultad está en sincronizar los canales y resolver sus conflictos.

10.5 DATOS BIOMÉTRICOS Y AMBIENTALES

10.5.1 Sensores biométricos

Miden magnitudes del cuerpo (capítulo 3): pulso y su variabilidad, respiración, conductancia de la piel (activación y estrés), actividad muscular (EMG) y cerebral (EEG), temperatura, y dirección de la mirada. En juego permiten dificultad adaptativa según el estrés, juegos de relajación y respiración, *biofeedback*, *exergames*, y —con el seguimiento de la mirada— apuntar, seleccionar y el *renderizado foveado*: dibujar con detalle solo la zona que el ojo está mirando. Son señales pequeñas y ruidosas, que exigen mucho acondicionamiento (capítulo 4), y son **datos personales sensibles** (capítulo 11).

10.5.2 Adquisición de datos ambientales

Sensores que miden el entorno, no al jugador: luz ambiente, sonido, temperatura, calidad del aire, ubicación, hora, tiempo atmosférico. Sirven para ambientación

dinámica, juegos basados en localización, instalaciones que reaccionan a la sala y ajustes de accesibilidad automáticos (por ejemplo, el brillo según la luz ambiente).

10.6 PROCESAMIENTO EN EL BORDE Y APLICACIONES

10.6.1 Procesamiento en el borde

El *procesamiento en el borde* (*edge computing*) hace el cálculo en el propio dispositivo, o cerca, en vez de enviarlo a la nube (figura 10.5). A favor del borde: menos latencia, más privacidad —los datos no salen—, funciona sin red y consume menos datos. A favor de la nube: más potencia de cálculo, modelos más grandes, actualización sencilla y estado compartido entre usuarios. Cada vez más, el reconocimiento de voz, de gestos o de intención (capítulo 11) se hace en el borde por latencia y privacidad, y lo más pesado se deja en la nube [5].

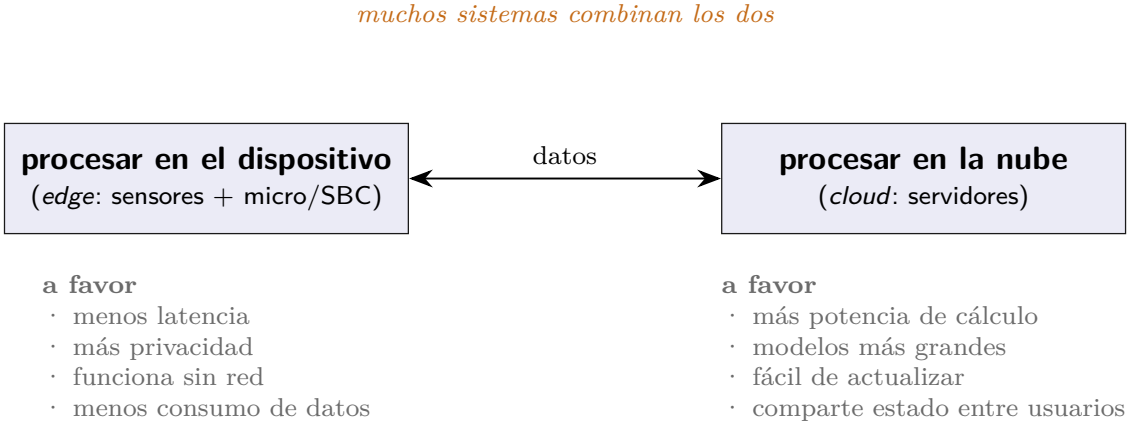


Figura 10.5. Procesar en el dispositivo o en la nube. La decisión depende de la latencia tolerable, de la privacidad de los datos y de la potencia de cálculo necesaria; muchos sistemas reparten el trabajo entre los dos.

10.6.2 Aplicaciones en videojuegos y experiencias inmersivas

Recapitulando: *exergames* y fitness; juegos que reaccionan al estado del jugador; realidad virtual y aumentada; instalaciones y experiencias de museo; juegos basados en localización; captura de movimiento para animación; y accesibilidad, con la voz, la mirada y el gesto como vías alternativas (capítulo 8). El criterio para elegir es siempre el mismo: cada tecnología capta una información concreta con un coste concreto —latencia, precisión, calibración, luz, privacidad, dinero—; se escoge la que resuelve la interacción con el menor coste, y se combinan cuando aportan.

RESUMEN

Las tecnologías emergentes de interacción son, en su mayoría, sensores y actuadores conocidos usados de formas nuevas. El IoT conecta objetos con sensores y red; los *wearables* llevan esos sensores puestos, con batería y señales minúsculas. La realidad virtual y la aumentada necesitan un seguimiento preciso: seis grados de libertad, con enfoque *outside-in* (estaciones fijas) o *inside-out* (cámaras en el visor), y siempre una combinación de óptica e IMU. La captura de movimiento registra el cuerpo entero. La visión artificial saca información de imágenes —caras, manos, distancia— a costa de luz, cálculo y privacidad. El reconocimiento de gestos y las interfaces de voz aportan vías naturales pero con falsos disparos y con problemas de ruido y contexto. La háptica avanzada lleva la sensación al cuerpo entero. Las interfaces multimodales fusionan varios canales para interpretar la intención. Los sensores biométricos permiten adaptar el juego al estado del jugador y habilitan el renderizado foveado, pero manejan datos sensibles. Y el procesamiento en el borde frente a la nube es un compromiso entre latencia y privacidad, por un lado, y potencia de cálculo, por otro. En todos los casos, elegir una tecnología es sopesar la información que capta contra su coste.

Ideas clave

- Casi toda «nueva tecnología» de interacción es un sensor o un actuador conocido en un uso nuevo.
- El seguimiento con seis grados de libertad combina óptica (posición) e IMU

(orientación); *outside-in* es preciso y fijo, *inside-out* es portátil.
- La visión y la voz son naturales pero frágiles: luz, ruido, oclusión, privacidad.
- Las interfaces multimodales funcionan porque cada canal tapa los huecos del otro.
- Procesar en el borde da latencia baja y privacidad; la nube da potencia; se combinan.

TÉRMINOS DEL CAPÍTULO

Internet de las cosas (IoT), dispositivo conectado, *wearable*, sensor portátil, realidad virtual, realidad aumentada, realidad extendida (XR), seguimiento (*tracking*), seis grados de libertad, *outside-in* e *inside-out*, captura de movimiento (*motion capture*), visión artificial, cámara de profundidad, reconocimiento de gestos, interfaz de voz, interfaz táctil avanzada, háptica de cuerpo completo, ultrasonidos táctiles, interfaz multimodal, fusión de canales, sensor biométrico, renderizado foveado, datos ambientales, juego basado en localización, procesamiento en el borde (*edge computing*), nube.

PARA SABER MÁS

- Jerald [3], referencia de diseño de realidad virtual centrado en la persona.
- LaValle [4], un curso completo de realidad virtual disponible en línea, con el seguimiento y la percepción bien explicados.
- Szeliski [6], para entrar en la visión artificial con rigor.

BIBLIOGRAFÍA DEL CAPÍTULO

[1] Seungmoon Choi y Katherine J. Kuchenbecker. «Vibrotactile Display: Perception, Technology, and Applications». En: *Proceedings of the IEEE* 101.9 (2013), págs. 2093-2104. DOI: 10.1109/JPROC.2012.2221071.

[2] Jacob Fraden. *Handbook of Modern Sensors: Physics, Designs, and Applications*. 5.ª ed. Cham: Springer, 2016. ISBN: 978-3-319-19302-1.

[3] Jason Jerald. *The VR Book: Human-Centered Design for Virtual Reality*. New York: ACM Books / Morgan & Claypool, 2015. ISBN: 978-1-97000-112-9.

[4] Steven M. LaValle. *Virtual Reality*. Disponible en línea en http://lavalle.pl/vr/. Cambridge: Cambridge University Press, 2023. ISBN: 978-1-107-15975-1.

[5] Weisong Shi et al. «Edge Computing: Vision and Challenges». En: *IEEE Internet of Things Journal* 3.5 (2016), págs. 637-646. DOI: 10.1109/JIOT.2016.2579198.

[6] Richard Szeliski. *Computer Vision: Algorithms and Applications*. 2.ª ed. Cham: Springer, 2022. ISBN: 978-3-030-34371-2.

CAPÍTULO 11

Sistemas inteligentes e interfaces de nueva generación

El libro empezó con una señal eléctrica y termina aquí, varios pasos más arriba: cómo esos datos se convierten en *información de alto nivel* —qué gesto, qué actividad, qué estado de ánimo—, cómo eso permite interfaces que se adaptan al jugador, y qué límites, riesgos y responsabilidades trae consigo.

Objetivos de aprendizaje

- Describir la cadena *adquisición* → *preprocesado* → *características* → *reconocimiento* en tiempo real.
- Explicar qué es un clasificador y qué significa entrenar un modelo.
- Comprender qué es una interfaz adaptativa y sensible al contexto.
- Analizar las limitaciones, los riesgos y la dimensión ética de los sistemas inteligentes, incluida la propia asignatura.

Alcance

Se ve el *proceso* —cómo se pasa de la señal a la decisión— sin la matemática de los algoritmos ni las arquitecturas de redes neuronales. La última sección trata la ética y el uso responsable de la IA, también como herramienta de estudio.

ÍNDICE DEL CAPÍTULO

11.1 DATOS DE INTERACCIÓN EN TIEMPO REAL

11.1.1 Datos de interacción

Todo lo visto en el libro produce datos: la posición de un *stick*, las pulsaciones, el movimiento del mando, la mirada, el pulso, la voz. Un dato aislado dice poco («el mando aceleró un instante»); muchos datos juntos y a lo largo del tiempo dicen mucho («el jugador ha hecho un gesto de saludo», «está frustrado», «va a girar a la izquierda»). Ese salto, de dato crudo a información con sentido, es el tema del capítulo.

11.1.2 Adquisición en tiempo real y preprocesado

Los datos llegan como un flujo continuo, a una frecuencia (capítulos 4 y 9). Se analizan por **ventanas**: trozos de medio segundo a un par de segundos, a veces solapados, que se procesan uno a uno. Hay que terminar cada ventana antes de que llegue la siguiente, y con poca latencia si controla el juego.

Antes de analizar, se hace *preprocesado*: filtrar el ruido (capítulo 4), quitar el *offset*, normalizar la escala —para que dé igual lo fuerte que se mueva el mando—, recortar la ventana y alinear varios sensores. Un preprocesado pobre estropea todo lo que viene después.

11.1.3 Extracción de características

Una *característica* es un número que resume un aspecto de la ventana: la media, el máximo, la energía, el número de picos, la duración, la relación entre ejes (figura 11.1). Se pasa de cientos de muestras a unos pocos números, y son esos números los que usa el modelo.

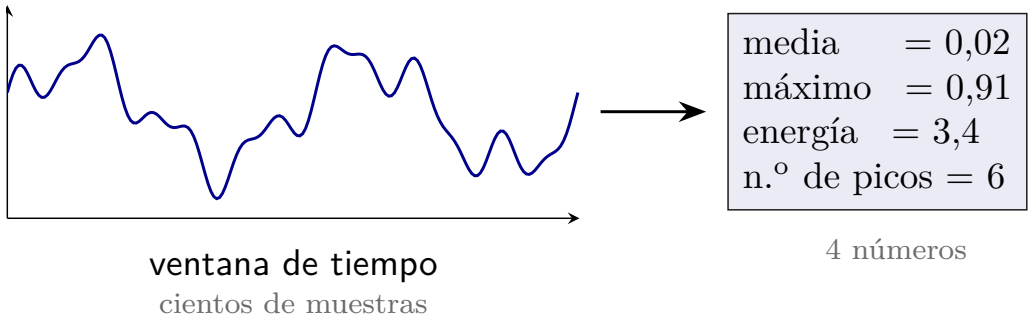


Figura 11.1. Extraer características: una ventana de señal con cientos de muestras se resume en unos pocos números que capturan su forma. El modelo trabaja con esos números.

Elegir buenas características —las que separan bien las categorías— es la mitad del trabajo. Algunos modelos (las redes profundas) aprenden las características por su cuenta, a cambio de necesitar muchos más datos y de ser menos transparentes.

11.2 RECONOCIMIENTO DE PATRONES Y APRENDIZAJE AUTOMÁTICO

11.2.1 Reconocimiento de patrones y clasificadores

El *reconocimiento de patrones* consiste en decidir, dadas unas características, a qué **categoría conocida** pertenecen: qué gesto, qué palabra, qué actividad. El modelo que toma esa decisión es un *clasificador*.

11.2.2 Aprendizaje automático aplicado a la interacción

Hay dos formas de construir el clasificador: **a mano**, con reglas escritas por una persona («si la energía supera un umbral y hay dos picos, es una sacudida»), o con *aprendizaje automático*, *entrenando* un modelo con muchos ejemplos etiquetados (figura 11.2).

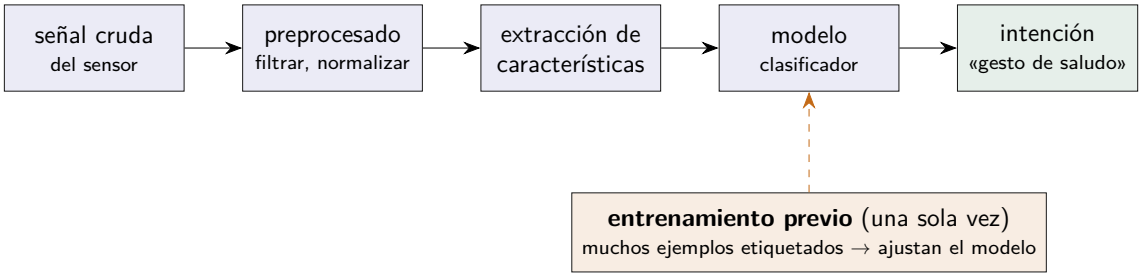


Figura 11.2. De la señal a la intención. El entrenamiento ocurre una sola vez, antes, con ejemplos etiquetados; el uso en tiempo real recorre solo la fila de arriba.

El proceso completo es siempre el mismo: señal → preprocesado → características → modelo → decisión. El entrenamiento ocurre una vez, antes de usar el sistema; la clasificación en tiempo real solo recorre esa cadena hacia delante [2].

11.2.3 Gestos, movimientos y actividad

- **Gestos y movimientos**: ventanas cortas y características de la forma del movimiento. Sacudir, dibujar un círculo, un golpe de raqueta.
- **Reconocimiento de actividad**: ventanas más largas, para distinguir andar, correr, estar quieto o subir escaleras a partir de la IMU de un *wearable* [1].

El mismo esquema sirve para la voz (características del sonido), para la mirada (patrones de fijación) y para el estado del jugador (pulso, conductancia de la piel y expresión, juntos).

11.3 INTERFACES ADAPTATIVAS

11.3.1 Interfaces adaptativas y personalización

Una *interfaz adaptativa* observa al jugador, estima su estado —habilidad, fatiga, estrés, intención— y ajusta algo: la dificultad, la asistencia de puntería, el tamaño de los elementos, el mapeo (figura 11.3). Es un **lazo cerrado con el jugador dentro** (capítulos 1 y 7): observar, estimar, ajustar y volver a observar [4].

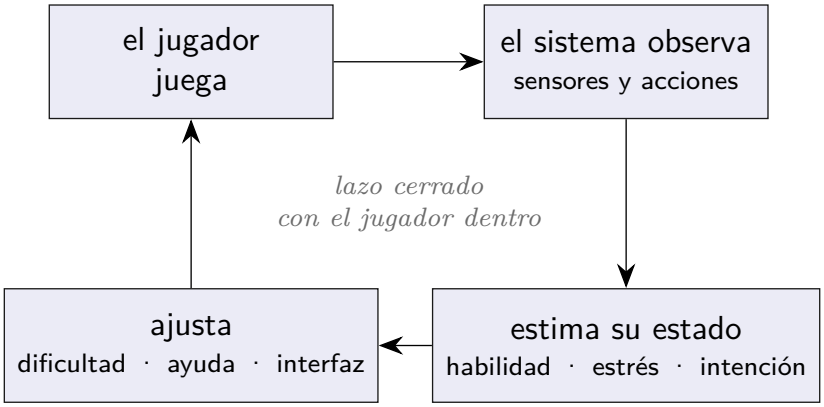


Figura 11.3. Una interfaz adaptativa es un lazo cerrado: el sistema observa al jugador, estima su estado y ajusta el juego; el jugador responde y el ciclo se repite.

La *personalización* añade memoria: recordar las preferencias y el rendimiento de cada jugador para arrancar ya ajustado.

11.3.2 Sistemas sensibles al contexto e IA en la interfaz

Un *sistema sensible al contexto* tiene en cuenta la situación —el lugar (capítulo 10), la hora, el dispositivo, si hay más gente, si el jugador va en movimiento—: la misma acción puede significar cosas distintas según el contexto.

La IA entra en la cadena de control de muchas formas (capítulos 7 y 10): asistencia de puntería que aprende el estilo del jugador, predicción de la siguiente acción, dificultad dinámica, filtrado inteligente del temblor, o traducir una señal difícil —EMG, EEG— en una intención. Y en las **interfaces multimodales inteligentes** (capítulo 10), un modelo fusiona voz, gesto y mirada y resuelve las ambigüedades entre canales. El resultado son interfaces que «entienden» más con menos esfuerzo del usuario.

11.4 LÍMITES, RIESGOS Y FUTURO

11.4.1 Limitaciones y riesgos de los sistemas inteligentes

- **Fiabilidad y sesgo**: un modelo acierta «casi siempre», y falla más con personas o situaciones distintas de las de su entrenamiento.
- **Datos**: entrenar necesita muchos ejemplos etiquetados, caros de conseguir; y los datos de interacción —movimiento, voz, pulso, mirada, y en el extremo las señales cerebrales de una BCI (capítulo 10)— son **datos personales sensibles**, difíciles de anonimizar.
- **Transparencia**: cuando el sistema decide por el jugador —le ayuda a apuntar, le cambia la dificultad—, a menudo no se puede explicar por qué. La *transparencia algorítmica* choca con la sensación de control y con la equidad: ¿es justa una asistencia que no todos reciben?
- **Privacidad y consentimiento**: qué se registra, quién lo guarda, para qué se usa después, y qué consentimiento es válido, sobre todo con menores.
- **Opacidad y dependencia**: una interfaz que se adapta sola puede volverse impredecible y restar agencia al jugador.

11.4.2 Uso responsable de la inteligencia artificial

La IA es también una herramienta de trabajo en esta asignatura, y se le aplica el mismo juicio crítico que a una interfaz inteligente: ¿qué datos usa?, ¿a quién puede perjudicar?, ¿se puede explicar lo que hace? [3]. Las reglas de uso:

- La IA puede ayudar a explorar conceptos, a autoevaluarse y a revisar la redacción.
- Todo lo que se entrega hay que **comprenderlo y saber defenderlo**; la defensa de la práctica es el mecanismo de verificación.
- Hay que **declarar** el uso de IA que se ha hecho en cada entrega.
- Riesgos que evitar: la falsa sensación de competencia, la dependencia, la homogeneización de las respuestas y los problemas éticos y académicos.

11.4.3 Tendencias futuras y nuevas interfaces para videojuegos

Modelos más eficientes que corren en el borde (capítulo 10), con lo que el reconocimiento de gestos, de voz o de intención no necesita enviar nada a la nube; fusión sensorial más rica —mando, IMU, mirada y biometría a la vez—; interfaces que se personalizan desde el primer minuto, lo que es también accesibilidad automática; y una háptica y una BCI que mejoran despacio —hoy la BCI es sobre todo una vía de accesibilidad, no un mando general—.

La tendencia de fondo es la misma que recorre el libro: **cada vez menos hardware visible entre la intención del jugador y su efecto en el juego**. Pero con más datos personales de por medio, lo que hace la responsabilidad más importante, no menos.

Cierre

Este libro ha recorrido una sola cadena: un fenómeno físico se convierte en una señal, la señal en un dato, el dato viaja hasta el software, el software decide y un actuador devuelve una respuesta al cuerpo del jugador. Cada capítulo ha sido un tramo de esa cadena —el sensor, el acondicionamiento, la conversión, el microcontrolador, el dispositivo, el actuador, la comunicación— y, al final, el modelo que interpreta y la interfaz que se adapta. Entender esa cadena, con sus costes y sus riesgos, es lo que permite diseñar interacciones que se sienten bien, que funcionan para más personas y que respetan a quien las usa.

RESUMEN

Convertir datos de sensores en información útil sigue siempre el mismo proceso: adquirir el flujo por ventanas en tiempo real, preprocesarlo (filtrar, normalizar), extraer características —unos pocos números que resumen cada ventana— y pasarlas a un modelo que decide a qué categoría conocida pertenecen. Ese modelo, el clasificador, se construye a mano con reglas o se entrena con ejemplos etiquetados; el entrenamiento

ocurre una vez y la clasificación, en tiempo real. El mismo esquema reconoce gestos, actividades, voz, patrones de mirada y estados del jugador. Con esa información se construyen interfaces adaptativas —un lazo cerrado que observa, estima y ajusta— y sensibles al contexto, y se mete IA en la cadena de control para asistir, predecir y fusionar canales. Todo ello tiene límites: los modelos fallan y tienen sesgos, necesitan muchos datos, esos datos son personales y sensibles, y sus decisiones no siempre se pueden explicar, lo que afecta al control, a la equidad y a la privacidad. La misma actitud crítica se aplica al uso de la IA como herramienta de estudio: ayuda permitida, pero comprensión y defensa obligatorias. La dirección del futuro es la de todo el libro: menos hardware visible entre la intención y su efecto, y por eso más responsabilidad.

Ideas clave

- El proceso es siempre: ventana → preprocesado → características → modelo → decisión.
- Una característica resume cientos de muestras en un número; elegir buenas características es media batalla.
- Un modelo se entrena una vez con ejemplos; luego clasifica en tiempo real.
- Una interfaz adaptativa es un lazo cerrado con el jugador dentro: observa, estima, ajusta.
- Los sistemas inteligentes fallan, sesgan y son opacos, y manejan datos personales: la responsabilidad crece con la potencia.

TÉRMINOS DEL CAPÍTULO

Datos de interacción, información de alto nivel, ventana de análisis, tiempo real, preprocesado, normalización, característica, extracción de características, reconocimiento de patrones, clasificador, aprendizaje automático, entrenamiento, ejemplo etiquetado, inferencia, red profunda, reconocimiento de gestos, reconocimiento de actividad, interfaz adaptativa, personalización, sistema sensible al contexto, dificultad dinámica, interfaz multimodal inteligente, sesgo, transparencia algorítmica, privacidad, consentimiento informado, uso responsable de la IA.

PARA SABER MÁS

- Yannakakis y Togelius [4], sobre inteligencia artificial en videojuegos, incluido el modelado del jugador y la adaptación; disponible en línea.
- Géron [2], una introducción práctica al aprendizaje automático.
- Bulling, Blanke y Schiele [1], un tutorial sobre reconocimiento de actividad con sensores inerciales.

BIBLIOGRAFÍA DEL CAPÍTULO

[1] Andreas Bulling, Ulf Blanke y Bernt Schiele. «A Tutorial on Human Activity Recognition Using Body-Worn Inertial Sensors». En: *ACM Computing Surveys* 46.3 (2014), págs. 1-33. DOI: 10.1145/2499621.

[2] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. 3.ª ed. Sebastopol, CA: O'Reilly Media, 2022. ISBN: 978-1-098-12597-4.

[3] Brent Daniel Mittelstadt et al. «The Ethics of Algorithms: Mapping the Debate». En: *Big Data & Society* 3.2 (2016). DOI: 10.1177/2053951716679679.

[4] Georgios N. Yannakakis y Julian Togelius. *Artificial Intelligence and Games*. Disponible en línea en https://gameaibook.org. Cham: Springer, 2018. ISBN: 978-3-319-63518-7.

Siglas y acrónimos

ADC convertidor analógico-digital (*analog-to-digital converter*) 66

BCI interfaz cerebro-computadora (*brain-computer interface*) 153, 154

DAC convertidor digital-analógico (*digital-to-analog converter*) 67

HID dispositivo de interfaz humana (*human interface device*) 88, 125, 127

HMI interfaz hombre-máquina (*human-machine interface*) 27

IMU unidad de medida inercial (*inertial measurement unit*) 50, 53, 68, 87, 91, 126, 140

IoT internet de las cosas (*internet of things*) 136

PCB placa de circuito impreso (*printed circuit board*) 115

PWM modulación por ancho de pulso (*pulse-width modulation*) 67, 77

Glosario

acelerómetro Sensor que mide la aceleración; en reposo detecta la gravedad, lo que permite conocer la inclinación 53

actuador Dispositivo que convierte una señal eléctrica en un efecto físico (movimiento, luz, sonido, fuerza, calor) 25, 98

actuador resonante lineal (LRA) Actuador de vibración que oscila en una dirección a una frecuencia fija; más rápido y preciso que un motor de masa excéntrica 101

aliasing Efecto de muestrear demasiado despacio: la señal reconstruida es falsa y parece más lenta de lo que realmente es 65

amplificación Multiplicación de una señal por una ganancia para aprovechar todo el rango del convertidor y reducir el peso relativo del ruido 63

API Conjunto de funciones que un programa o un sistema ofrece para que otros programas lo usen; el sistema operativo expone los mandos mediante una API 128

aprendizaje automático Conjunto de técnicas que construyen un modelo a partir de ejemplos, en lugar de programar las reglas a mano 150

baudios Unidad de velocidad de una comunicación serie asíncrona; aproximadamente, bits por segundo 126

BMS (gestión de batería) Electrónica que protege una batería de litio frente a sobrecarga, sobredescarga, cortocircuito y temperatura, y equilibra sus celdas 113

bucle de interacción Ciclo repetido de acción del usuario, respuesta del sistema y percepción del usuario 28

C-rate Medida del ritmo al que se carga o descarga una batería en relación con su capacidad; 1 C vacía una batería en una hora 113

calibración Ajuste de la relación entre la señal de un sensor y la magnitud física real, comparándolo con una referencia conocida 57

captura de movimiento (*motion capture*) Registro del movimiento del cuerpo completo mediante marcadores, trajes con sensores o cámaras, para animar un personaje o controlar una experiencia 140

característica Número que resume un aspecto de una señal (media, máximo, energía, número de picos…) y con el que trabaja el modelo en lugar de con la señal completa 149

ciclo de trabajo En una señal PWM, fracción del tiempo en que la señal está en nivel alto; determina su valor medio 67

clasificador Modelo que, dadas unas características, decide a cuál de varias categorías pertenecen 150

comprobación de errores (*checksum*) Valor calculado a partir de los datos de una trama y enviado con ella, que el receptor recalcula para detectar si la trama llegó corrupta 125

comunicación serie Transmisión de los bits de un dato uno tras otro por un mismo hilo 126

concentrador adaptativo Dispositivo, como el Xbox Adaptive Controller, que ofrece muchas entradas conmutadas (jacks de 3,5 mm) y puertos USB para que el jugador conecte pulsadores, pedales y palancas y arme su propia interfaz 117

condensador Componente que almacena carga eléctrica; se usa para estabilizar la alimentación y para filtrar señales 40, 42

corriente eléctrica Flujo de carga eléctrica a través de un conductor; se mide en amperios (A) 36

CPI/DPI Cuentas (o puntos) por pulgada de desplazamiento de un ratón; a más CPI, el puntero se mueve más por el mismo gesto de la mano 89

cuantificación Asignación de cada muestra a uno de un número finito de niveles; su finura depende de la resolución en bits 65

diodo de rueda libre Diodo colocado en paralelo con un motor o un relé para absorber el pico de tensión que se genera al cortarle la corriente y proteger el transistor 104

diseño centrado en el usuario Método de diseño que involucra a las personas que van a usar el dispositivo desde el principio y en cada iteración: observar, prototipar, probar y ajustar 118

divisor de tensión Montaje de dos resistencias en serie que entrega a su salida una fracción de la tensión de entrada, fijada por la proporción entre ambas 39

***dongle* de baja latencia** Receptor de radio propio que se conecta a un puerto USB y usa un protocolo optimizado para tener menos latencia y menos variabilidad que Bluetooth 128

***driver* (etapa de potencia)** Circuito intermedio, basado en transistores, que recibe la señal débil del microcontrolador y con ella gobierna la corriente grande que necesita un actuador 103

efecto Hall Fenómeno por el que un campo magnético genera una tensión en un material por el que circula corriente; se usa para detectar posición o presencia sin contacto 51

emparejamiento (*pairing*) Proceso de asociar dos dispositivos inalámbricos para que se reconozcan y, normalmente, cifren su comunicación 127

encoder Sensor que convierte un movimiento de rotación o desplazamiento en una serie de pulsos digitales que se cuentan para conocer la posición 51, 88

entrenamiento Proceso, previo al uso, de ajustar un modelo mostrándole muchos ejemplos etiquetados para que aprenda a distinguir las categorías 150

ergonomía Adaptación de un dispositivo al cuerpo humano: postura, alcance, fuerza, tamaño, peso y agarre, para que se use cómodamente y sin fatiga 117

exactitud Cercanía entre el valor medido y el valor real 56

filtro paso bajo Filtro que deja pasar las variaciones lentas de una señal y atenúa las rápidas; se usa para quitar ruido de alta frecuencia y para evitar el aliasing antes del ADC 63

frecuencia de actualización Cada cuánto un sistema repite el ciclo de leer entradas, procesar y responder (tasa de sondeo del mando, fotogramas por segundo del juego, refresco de la pantalla) 68

frecuencia de muestreo Número de muestras que se toman de una señal por segundo 64

gamepad Mando de videojuegos estándar que integra en una carcasa cruceta, botones, palancas analógicas, gatillos y, a menudo, sensores de movimiento y actuadores 87

giróscopo Sensor que mide la velocidad de giro alrededor de un eje 53

háptica de banda ancha Actuador de vibración (tipo bobina de voz) capaz de reproducir una gama amplia de frecuencias e intensidades, con lo que transmite «texturas» y no solo un zumbido 102

interfaz adaptativa Interfaz que observa al usuario, estima su estado y ajusta la dificultad, la ayuda o su propia disposición en consecuencia 151

interfaz gestual Interfaz en la que la orden es un movimiento del cuerpo reconocido como comando, captado por sensores inerciales, cámara o seguimiento de manos 92

interfaz multimodal Interfaz que combina varias vías de entrada a la vez —voz, gesto, mirada, tacto— y fusiona sus datos para interpretar la intención 142

interrupción Mecanismo por el que un suceso (una pulsación, un dato que llega) hace que el procesador deje lo que hace y atienda ese suceso al instante 76

jitter Variación de la latencia entre una medida y la siguiente; suele molestar más que una latencia alta pero constante 69

latencia Retardo entre una acción y el momento en que su efecto se percibe 28, 68

lazo abierto Sistema de control que actúa sin comprobar el resultado de su actuación 26

lazo cerrado Sistema de control que mide el resultado de su actuación y lo usa para corregirse (realimentación) 26

matriz de botones Disposición de los botones en filas y columnas para leer muchos con pocos pines: el microcontrolador activa una fila cada vez y comprueba qué columnas responden 88

media móvil Filtro digital que sustituye cada muestra por la media de las últimas N; reduce el ruido a costa de añadir retardo 68

memoria Flash Memoria no volátil de un microcontrolador donde se guarda el programa; conserva su contenido sin alimentación 75

memoria RAM Memoria volátil de un microcontrolador donde el programa guarda los datos mientras se ejecuta; se borra al quitar la alimentación 75

modding Modificación física de un mando o periférico comercial existente: cambiar botones, palancas o gatillos, o cablear entradas hacia conectores externos 118

motor de masa excéntrica (ERM) Motor con un peso descentrado en el eje; al girar produce una vibración genérica, con arranque y parada lentos 101

motor paso a paso Motor que gira en incrementos fijos (pasos), lo que permite controlar su posición sin necesidad de un sensor 100

muestreo Toma del valor de una señal analógica a intervalos regulares de tiempo para representarla como una secuencia de números 64

N-key rollover Capacidad de un teclado de registrar correctamente muchas teclas pulsadas a la vez; se consigue añadiendo un diodo por tecla para evitar pulsaciones fantasma 89

nivel lógico Rango de tensión que un circuito digital interpreta como «0» o como «1» 42

periférico personalizado Dispositivo de entrada o salida hecho a medida para una interacción o un usuario concretos que un mando estándar no cubre bien 111

potencia eléctrica Ritmo al que un circuito consume o entrega energía; producto de la tensión por la corriente; se mide en vatios (W) 37

potenciómetro Resistencia con un cursor móvil que actúa como divisor de tensión ajustable; convierte una posición o un giro en una tensión 51, 88

precisión Cercanía entre sí de varias medidas repetidas de la misma magnitud, con independencia de si aciertan el valor real 56

preprocesado Limpieza de una señal antes de analizarla: filtrar el ruido, quitar el *offset*, normalizar la escala y recortar la ventana de interés 149

presupuesto de energía Estimación de la autonomía de un dispositivo sumando el consumo de cada subsistema y dividiendo la capacidad de la batería entre ese consumo medio 116

procesamiento en el borde (*edge computing*) Procesar los datos en el propio dispositivo o cerca de él, en lugar de enviarlos a la nube; reduce la latencia y protege la privacidad 143

protoboard (*breadboard*) Placa con orificios conectados internamente en tiras que permite montar y rehacer circuitos sin soldar; útil para probar, poco fiable para algo definitivo 115

protocolo Conjunto de reglas que dos dispositivos acuerdan para entenderse: velocidad, formato de los datos, quién inicia la comunicación y qué respuesta se espera 125

rango de medida Intervalo de valores de la magnitud física que un sensor puede medir; fuera de él, el sensor satura 55

realidad extendida (XR) Término que engloba la realidad virtual, la aumentada y la mixta 138

realimentación Uso de la medida del resultado de una actuación para corregir esa misma actuación 26

reconocimiento de patrones Identificar a qué categoría conocida pertenece un conjunto de características: qué gesto, qué palabra, qué actividad 150

regulador de tensión Circuito que convierte una tensión de entrada variable en una tensión de salida fija y estable 41

relé Interruptor accionado por una pequeña corriente que permite a un microcontrolador conectar y desconectar circuitos de mucha más potencia 100

renderizado foveado (*foveated rendering*) Técnica que dibuja con detalle solo la zona de la pantalla que el ojo está mirando, gracias al seguimiento de la mirada, para ahorrar cálculo 142

repetibilidad Capacidad de un sensor de dar la misma lectura ante la misma entrada en medidas sucesivas 56

resistencia *pull-down* Resistencia que conecta una entrada digital a masa para fijarla en «0» mientras nada la lleve a «1» 42

resistencia *pull-up* Resistencia que conecta una entrada digital a la alimentación para fijarla en «1» mientras nada la lleve a «0» 42

resistencia eléctrica Oposición de un material o componente al paso de la corriente; se mide en ohmios 37

resolución Cambio más pequeño de la magnitud medida que un sensor o un convertidor pueden distinguir 55, 65

retorno de fuerza (*force-feedback*) Retroalimentación en la que el actuador se opone al movimiento del usuario o lo empuja, como en un volante que endurece el giro según el agarre del coche 102

seguimiento (*tracking*) Determinación de la posición, y a menudo la orientación, de un mando o del cuerpo del jugador en el espacio 91

seguimiento *inside-out* Seguimiento en el que las cámaras van en el propio dispositivo y observan la sala para deducir su posición 139

seguimiento *outside-in* Seguimiento en el que cámaras o estaciones fijas en la sala observan al usuario y a sus mandos 139

seis grados de libertad (6 GDL) Capacidad de un sistema de seguimiento de conocer las tres coordenadas de posición y las tres de orientación de un objeto 138

sensibilidad Cuánto cambia la señal de salida de un sensor por cada unidad de cambio de la magnitud medida 55

sensor Dispositivo que convierte una magnitud física en una señal eléctrica medible 25, 49

sensor capacitivo Sensor que detecta la proximidad o el contacto de un objeto (como un dedo) por el cambio que produce en una capacitancia 52

servomotor Motor con electrónica interna que lo lleva a un ángulo concreto y lo mantiene; se controla indicándole la posición deseada 100

señal Magnitud que varía a lo largo del tiempo y que transporta información 24

sistema de adquisición y control Conjunto de elementos que miden magnitudes del mundo físico, deciden a partir de esas medidas y actúan sobre el entorno 16

sistema embebido Sistema informático dedicado a una función concreta dentro de un dispositivo mayor, normalmente basado en un microcontrolador 74

sistema interactivo Sistema que responde de forma continua a las acciones de una persona, cerrando un bucle de acción y percepción 28

sistema sensible al contexto Sistema que tiene en cuenta la situación —lugar, hora, actividad, dispositivo, entorno— para decidir cómo comportarse 152

sketch Nombre que recibe un programa en el entorno Arduino; se compone de una función de configuración que se ejecuta una vez y un bucle que se repite 78

solenoide Actuador que convierte un pulso de corriente en un movimiento lineal breve; un émbolo que entra o sale 100

sondeo Comprobar repetidamente el estado de una entrada dentro del bucle del programa, en lugar de esperar a que un suceso avise 76

temporizador Periférico que cuenta pulsos del reloj para medir tiempos, generar señales periódicas o disparar acciones a intervalos fijos; base de la generación de PWM 76

tensión eléctrica Diferencia de potencial entre dos puntos; el «empuje» que mueve la carga; se mide en voltios (V) 37

tiempo de respuesta Tiempo que tarda la salida de un sensor en reflejar un cambio brusco de la magnitud medida 56

tiempo real Modo de funcionamiento en el que el sistema responde dentro de un plazo lo bastante corto como para que la interacción se perciba inmediata 28

trama Bloque de datos con estructura fija —cabecera, carga útil y comprobación— en que se agrupa la información para enviarla 125

transductor Dispositivo que convierte energía de una forma a otra; un sensor es un transductor de entrada y un actuador, uno de salida 25, 49

transistor Componente que actúa como interruptor o amplificador controlado por una señal pequeña; elemento básico de los circuitos integrados 40

transparencia algorítmica Posibilidad de entender y explicar por qué un sistema automático tomó una decisión concreta 153

variable de información Representación de una variable física como dato manejable por un sistema digital: un número, un bit o un evento 23

variable física Magnitud del mundo real que se puede medir: posición, fuerza, presión, luz, sonido, temperatura, aceleración, proximidad, etc. 23

visión artificial Conjunto de técnicas que extraen información de imágenes: detectar una cara, unas manos, un objeto o su distancia 140

wearable Dispositivo electrónico que se lleva puesto —pulsera, reloj, anillo, prenda— con sensores que miden actividad o magnitudes del cuerpo 136

zona muerta Intervalo de valores próximos al reposo de un mando que el sistema ignora para evitar deriva y ruido 57

Bibliografía general

[1] Jan Axelson. *USB Complete: The Developer's Guide*. 5.ª ed. Madison, WI: Lakeview Research, 2015. ISBN: 978-1-931448-28-4.

[2] Jeremy Blum. *Exploring Arduino: Tools and Techniques for Engineering Wizardry*. 2.ª ed. Indianapolis, IN: John Wiley & Sons, 2019. ISBN: 978-1-119-40537-5.

[3] William Bolton. *Mechatronics: Electronic Control Systems in Mechanical and Electrical Engineering*. 6.ª ed. Harlow: Pearson Education, 2015. ISBN: 978-1-292-07668-3.

[4] Andreas Bulling, Ulf Blanke y Bernt Schiele. «A Tutorial on Human Activity Recognition Using Body-Worn Inertial Sensors». En: *ACM Computing Surveys* 46.3 (2014), págs. 1-33. DOI: 10.1145/2499621.

[5] Neil Cameron. *Electronics Projects with the ESP8266 and ESP32*. Berkeley, CA: Apress, 2021. ISBN: 978-1-4842-6335-8.

[6] Seungmoon Choi y Katherine J. Kuchenbecker. «Vibrotactile Display: Perception, Technology, and Applications». En: *Proceedings of the IEEE* 101.9 (2013), págs. 2093-2104. DOI: 10.1109/JPROC.2012.2221071.

[7] Alan Dix et al. *Human-Computer Interaction*. 3.ª ed. Harlow: Pearson Prentice Hall, 2004. ISBN: 978-0-13-046109-4.

[8] Jacob Fraden. *Handbook of Modern Sensors: Physics, Designs, and Applications*. 5.ª ed. Cham: Springer, 2016. ISBN: 978-3-319-19302-1.

[9] Game Accessibility Guidelines Working Group. *Game Accessibility Guidelines*. 2024. URL: https://gameaccessibilityguidelines.com (visitado 08-09-2026).

[10] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. 3.ª ed. Sebastopol, CA: O'Reilly Media, 2022. ISBN: 978-1-098-12597-4.

[11] Jason Gregory. *Game Engine Architecture*. 3.ª ed. Boca Raton, FL: CRC Press, 2018. ISBN: 978-1-138-03545-4.

[12] Paul Horowitz y Winfield Hill. *The Art of Electronics*. 3.ª ed. Cambridge: Cambridge University Press, 2015. ISBN: 978-0-521-80926-9.

[13] Tom Igoe. *Making Things Talk: Using Sensors, Networks, and Arduino to See, Hear, and Feel Your World*. 3.ª ed. San Francisco, CA: Make Community, 2017. ISBN: 978-1-68045-315-6.

[14] Jason Jerald. *The VR Book: Human-Centered Design for Virtual Reality*. New York: ACM Books / Morgan & Claypool, 2015. ISBN: 978-1-97000-112-9.

[15] Steven M. LaValle. *Virtual Reality*. Disponible en línea en http://lavalle.pl/vr/. Cambridge: Cambridge University Press, 2023. ISBN: 978-1-107-15975-1.

[16] Richard G. Lyons. *Understanding Digital Signal Processing*. 3.ª ed. Upper Saddle River, NJ: Prentice Hall, 2010. ISBN: 978-0-13-702741-5.

[17] Brent Daniel Mittelstadt et al. «The Ethics of Algorithms: Mapping the Debate». En: *Big Data & Society* 3.2 (2016). DOI: 10.1177/2053951716679679.

[18] Simon Monk. *Electronics Cookbook: Practical Electronic Recipes with Arduino and Raspberry Pi*. Sebastopol, CA: O'Reilly Media, 2017. ISBN: 978-1-491-95340-2.

[19] Simon Monk. *Programming Arduino: Getting Started with Sketches*. 3.ª ed. New York: McGraw-Hill Education, 2022. ISBN: 978-1-264-67698-5.

[20] Alan S. Morris y Reza Langari. *Measurement and Instrumentation: Theory and Application*. 2.ª ed. London: Academic Press, 2016. ISBN: 978-0-12-800884-3.

[21] Donald A. Norman. *The Design of Everyday Things*. Revised and expanded. New York: Basic Books, 2013. ISBN: 978-0-465-05065-9.

[22] Dan O'Sullivan y Tom Igoe. *Physical Computing: Sensing and Controlling the Physical World with Computers*. Boston, MA: Thomson Course Technology, 2004. ISBN: 978-1-59200-346-1.

[23] Ramon Pallàs-Areny y John G. Webster. *Sensors and Signal Conditioning*. 2.ª ed. New York: John Wiley & Sons, 2001. ISBN: 978-0-471-33232-9.

[24] Charles Platt. *Encyclopedia of Electronic Components, Volume 3: Sensors*. San Francisco, CA: Maker Media, 2016. ISBN: 978-1-4493-3431-1.

[25] Charles Platt. *Make: Electronics: Learning by Discovery*. 3.ª ed. San Francisco, CA: Make Community, 2021. ISBN: 978-1-68045-687-4.

[26] Charles Platt y Fredrik Jansson. *Encyclopedia of Electronic Components, Volume 1: Resistors, Capacitors, Inductors, Switches, Encoders, Relays, Transistors*. Sebastopol, CA: Maker Media, 2012. ISBN: 978-1-4493-3389-5.

[27] Katie Salen y Eric Zimmerman. *Rules of Play: Game Design Fundamentals*. Cambridge, MA: The MIT Press, 2004. ISBN: 978-0-262-24045-1.

[28] Pedro Luis Sánchez Ortega. *Toma el control: breve historia de los videojuegos con la evolución de sus mandos*. Material docente. Universidad de Burgos, 2023. URL: http://hdl.handle.net/10259/9588 (visitado 08-09-2026).

[29] Paul Scherz y Simon Monk. *Practical Electronics for Inventors*. 4.ª ed. New York: McGraw-Hill Education, 2016. ISBN: 978-1-259-58754-2.

[30] Weisong Shi et al. «Edge Computing: Vision and Challenges». En: *IEEE Internet of Things Journal* 3.5 (2016), págs. 637-646. DOI: 10.1109/JIOT.2016.2579198.

[31] Steven W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. Disponible en línea en https://www.dspguide.com. San Diego, CA: California Technical Publishing, 1997. ISBN: 978-0-9660176-3-2.

[32] Steve Swink. *Game Feel: A Game Designer's Guide to Virtual Sensation*. Burlington, MA: Morgan Kaufmann, 2009. ISBN: 978-0-12-374328-2.

[33] Richard Szeliski. *Computer Vision: Algorithms and Applications*. 2.ª ed. Cham: Springer, 2022. ISBN: 978-3-030-34371-2.

[34] Unity Technologies. *Input System*. 2024. URL: https://docs.unity3d.com/Packages/com.unity.inputsystem@latest (visitado 08-09-2026).

[35] Elecia White. *Making Embedded Systems: Design Patterns for Great Software*. Sebastopol, CA: O'Reilly Media, 2011. ISBN: 978-1-449-30214-6.

[36] Georgios N. Yannakakis y Julian Togelius. *Artificial Intelligence and Games*. Disponible en línea en https://gameaibook.org. Cham: Springer, 2018. ISBN: 978-3-319-63518-7.

[37] Bei Yuan, Eelke Folmer y Frederick C. Harris. «Game accessibility: a survey». En: *Universal Access in the Information Society* 10.1 (2011), págs. 81-100. DOI: 10.1007/s10209-010-0189-5.

Colofón

Este libro se compuso con LaTeX (motor LuaLaTeX, clase `report`). Las figuras son vectoriales, generadas con Ti*k*Z, CircuiTikz y pgfplots; la bibliografía se gestiona con BIBLATEX/BIBER y el glosario con GLOSSARIES-EXTRA. La tipografía es Computer Modern (Latin Modern), con composición en español mediante `babel`.

Material docente de la asignatura *Sistemas de Adquisición y Control*, Grado en Diseño de Videojuegos, Universidad de Burgos.

Primera edición, 2026.

UNIVERSIDAD DE BURGOS

Material de teoría de la asignatura *Sistemas de Adquisición y Control* del Grado en Diseño de Videojuegos. Recorre la cadena que va del fenómeno físico a la respuesta al jugador —sensor, acondicionamiento, conversión, procesamiento, comunicación, software y actuador— con un enfoque técnico pensado para quien diseña videojuegos, no para quien calcula circuitos.