

Fundamentos de Arduino y Captura de Sensores en Python

Prácticas Intermedias 2do Semestre 2025

INTEGRANTES:

Kevin Santos - 202101007

Javier Monterroso - 202102140

Raúl Yoque - 202103988

ÍNDICE

PARTE

TEORÍA



01. Introducción

02. Conceptos Básicos

03. Uso e Importancia

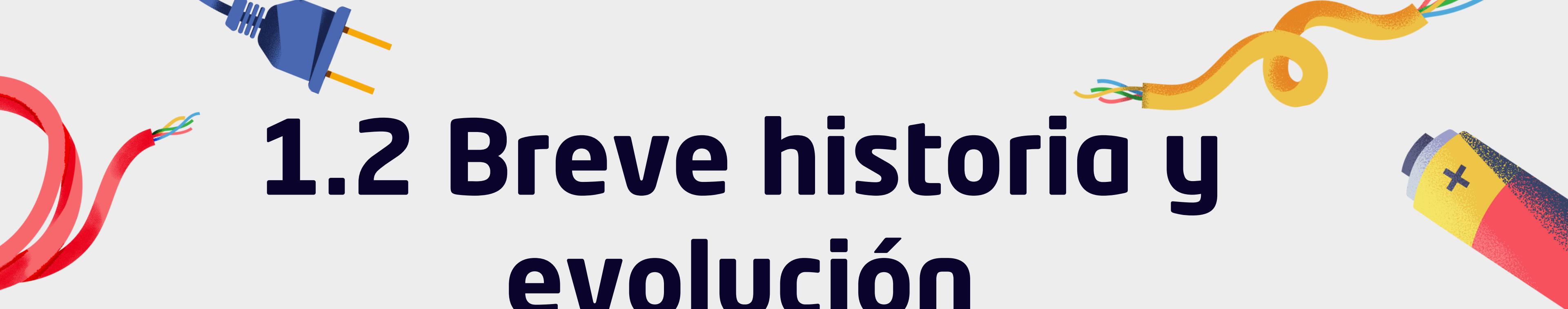
04. Sensores

05. Cierre Parte teoría

06. Desarrollo de Parte Práctica

1.1 ¿Qué es Arduino?

- Plataforma de hardware y software libre.
- Microcontroladores fáciles de programar (C/C++).
- Entorno de desarrollo: Arduino IDE.



1.2 Breve historia y evolución

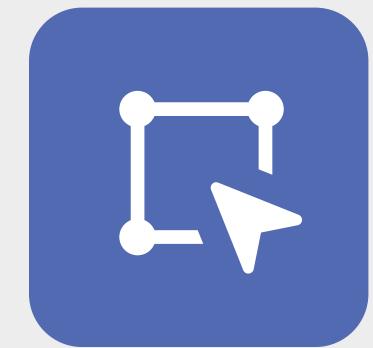


1.2 Breve historia y evolución

Nació en Italia, 2005 en el Instituto Ivrea.

Proyecto para estudiantes con bajo costo y facilidad de uso.

Expansión hacia una comunidad global de makers, docentes y empresas.



1.3 Diferencias con otros microcontroladores

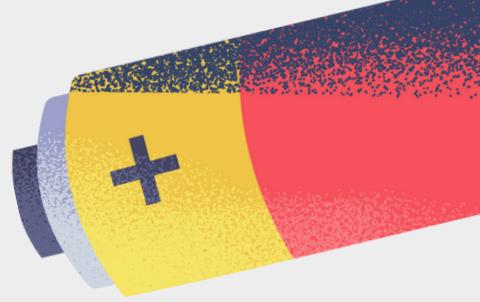
- Arduino vs. PIC / ARM:
- Arduino: fácil de usar, gran comunidad, plug & play.
- Otros microcontroladores: más potentes, pero requieren más conocimientos técnicos.
- Ejemplo: programar un LED en PIC vs. Arduino → Arduino es 3 líneas de código.

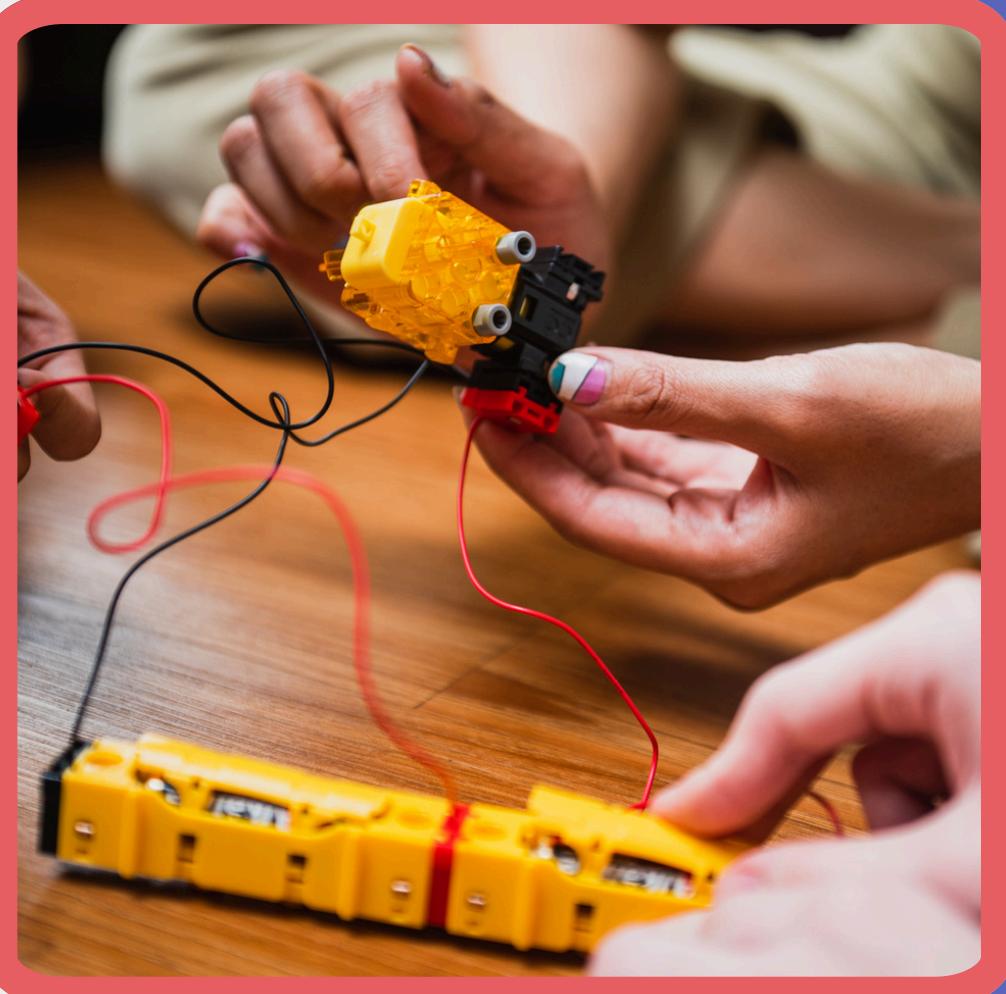
Parte 2: Conceptos Básicos

2.1 ¿Qué es un microcontrolador?

2.2 Diferencia microcontrolador vs.
microprocesador

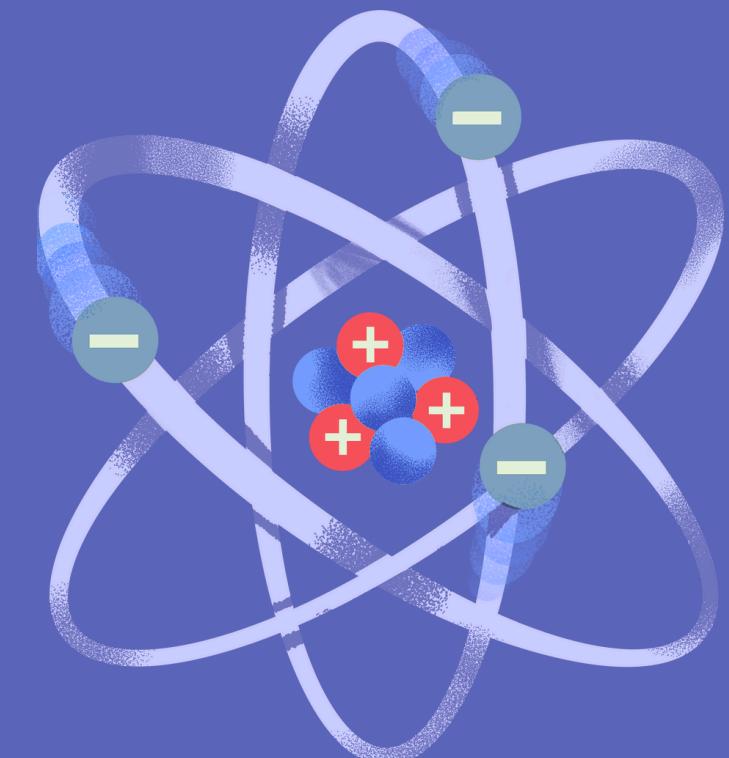
2.3 Partes principales de una placa Arduino





Qué es un microcontrolador?

- Chip que integra CPU, memoria y periféricos en un solo dispositivo.
- Es el “cerebro” que controla sensores, actuadores, motores



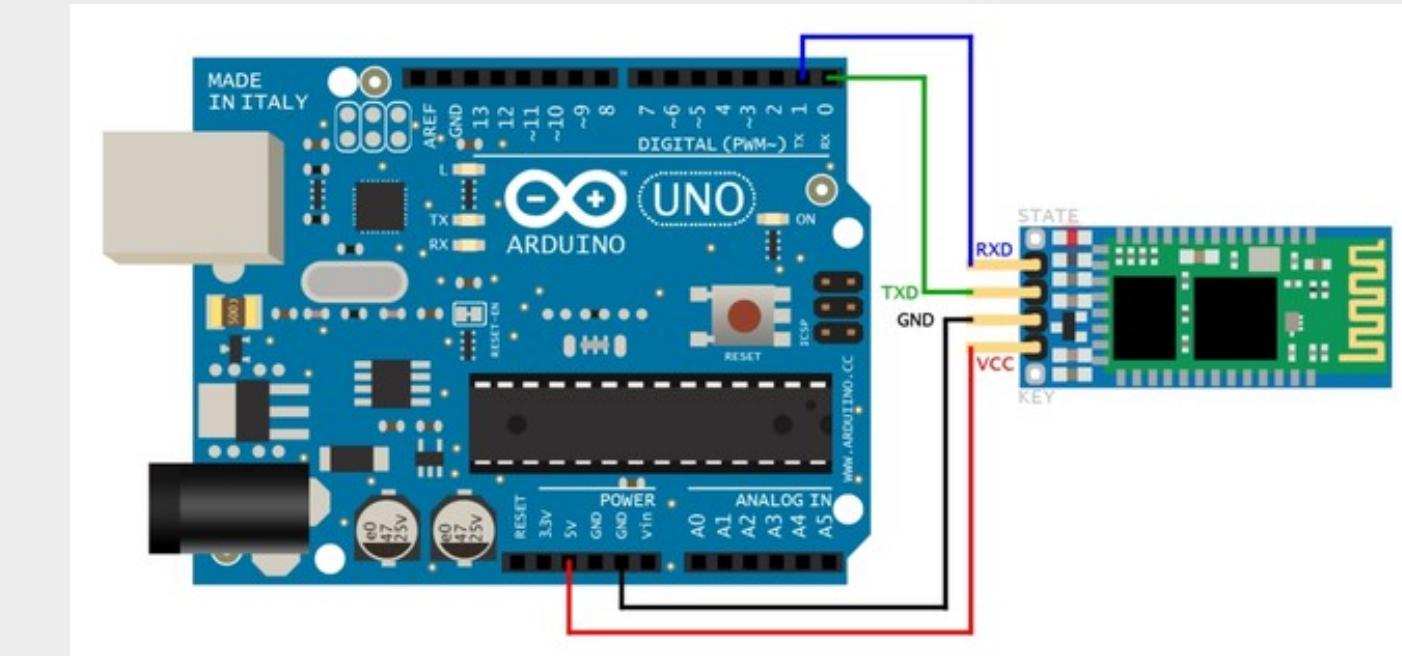
Diferencia microcontrolador vs. microporcesador

- Microcontrolador: pensado para controlar dispositivos, tareas específicas (ej: Arduino UNO → ATmega328p).
- Microporcesador: pensado para cálculos generales, multitarea (ej: procesador de PC).

Partes principales de una placa Arduino

01 Visualizar la placa a la derecha

- Microcontrolador (ATmega).
- Pines digitales y analógicos.
- Puerto USB.
- Regulador de voltaje.
- Cristal oscilador.
- Botón de reset.



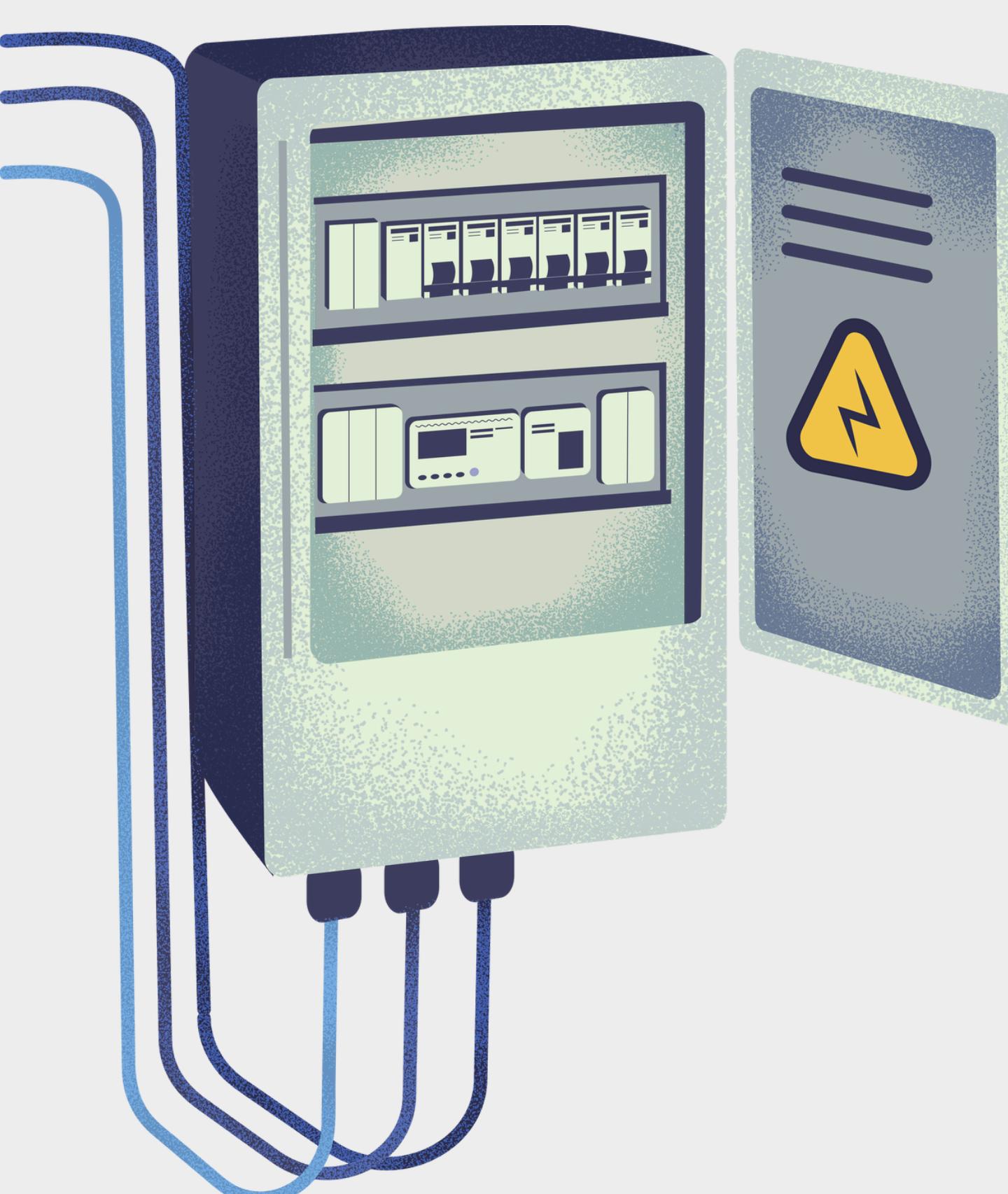
Parte 3: Uso e Importancia

3.1 Aplicaciones típicas

- Educación: primeras prácticas de electrónica.
- Prototipado rápido en IoT.
- Robótica: control de motores, sensores.
- Domótica: control de luces, puertas, sensores de temperatura.

3.2 Importancia en enseñanza e industria maker

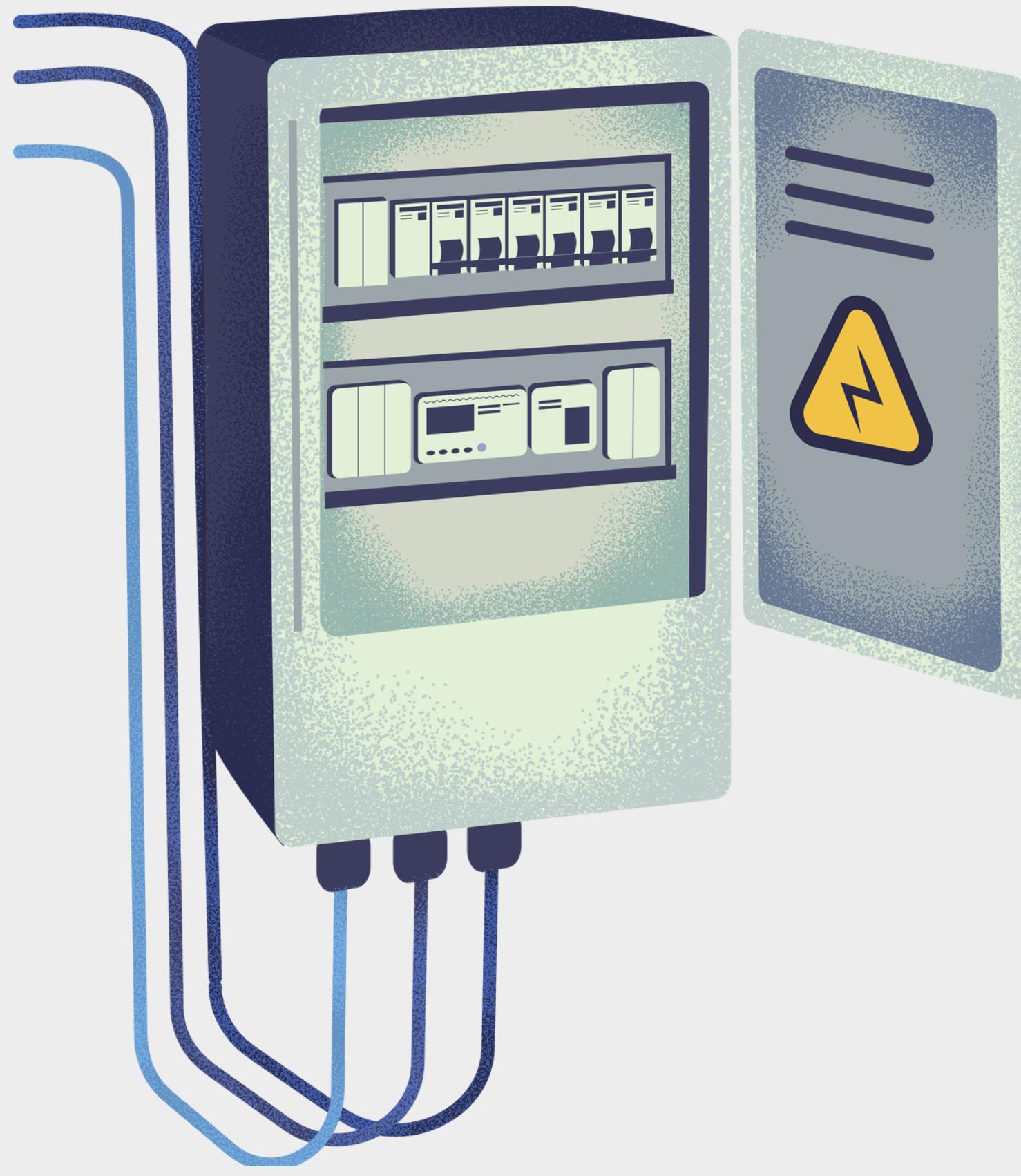
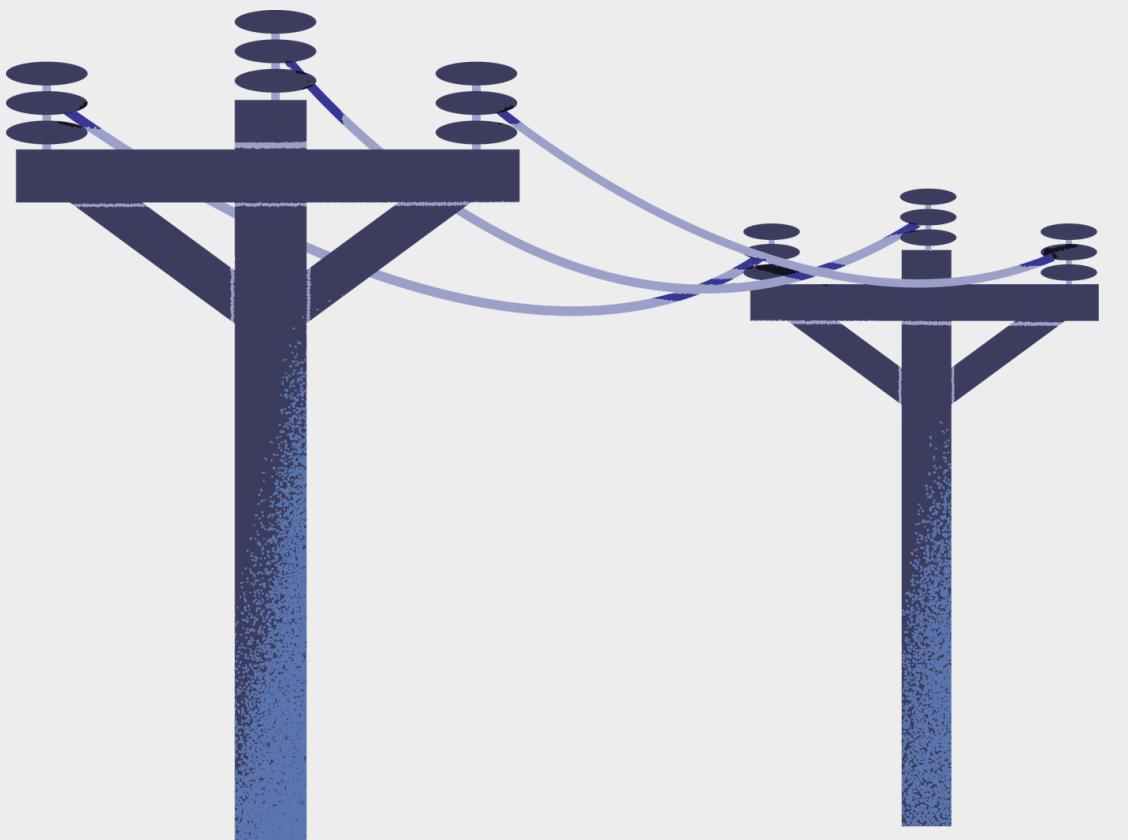
- Democratización del hardware.
- Aprendizaje accesible y práctico.
- Comunidad global con miles de ejemplos.



Continuación Parte 3

3.3 Mercado y comunidad

- Más de millones de placas vendidas.
- Compatibilidad con librerías open source.
- Alternativas: ESP32, NodeMCU, Raspberry Pi Pico.

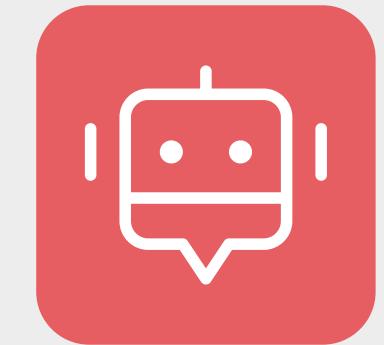


Parte 4: Sobre Sensores



¿Qué es un sensor?

Dispositivo que detecta fenómenos físicos y los convierte en señales eléctricas.



Clasificación general

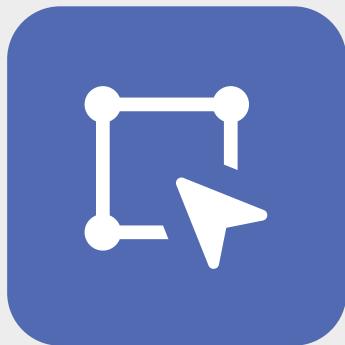
Analógicos:

Producen un valor continuo (ejemplo: de 0V a 5V).
El Arduino los lee en los pines A0–A5 con `analogRead()`.

Ejemplo: sensor de temperatura LM35.

Digitales:

Solo tienen dos estados: 0 (apagado) o 1 (encendido).
Se conectan a pines digitales y se leen con `digitalRead()`.
Ejemplo: sensor PIR de movimiento.



Tipos comunes

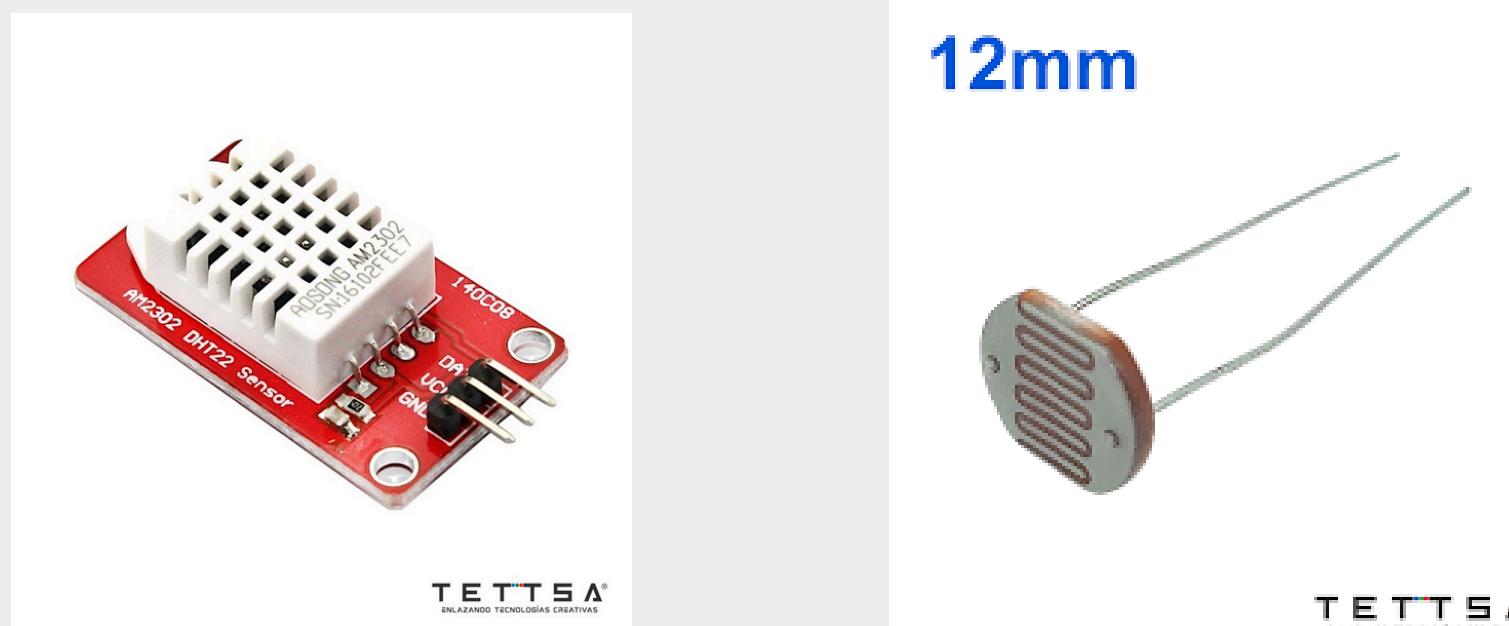
- Temperatura (LM35, DHT11, DHT22).
- Movimiento (PIR).
- Distancia (Ultrasonido HC-SR04).
- Luz (LDR).
- Humedad de suelo.

¿Qué pueden medir los sensores?

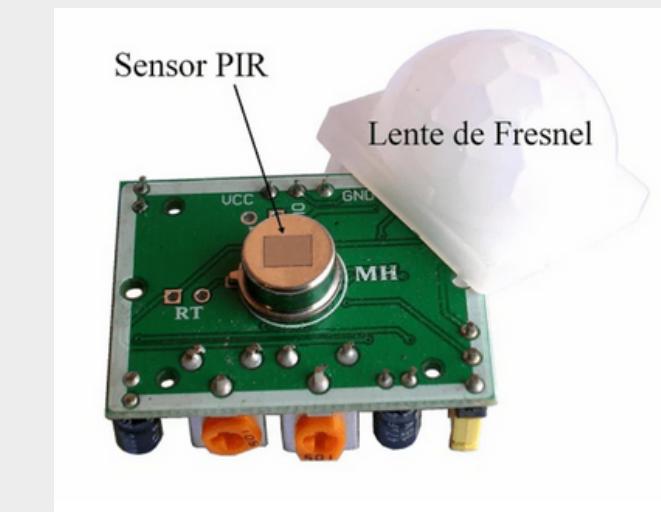
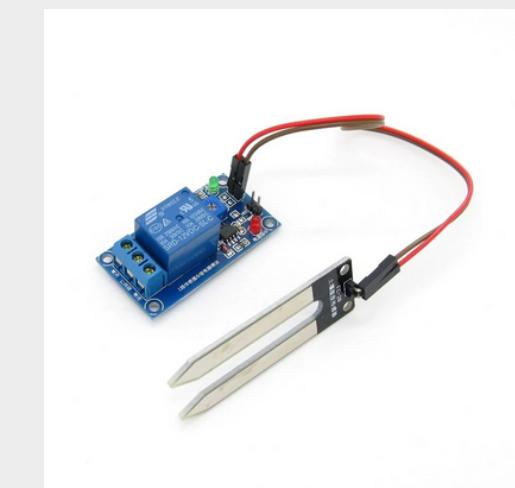
- Físicos: luz, temperatura, humedad, presión, distancia.
- Químicos: gas, alcohol, CO₂, humo.
- Biométricos: ritmo cardíaco, huella dactilar.
- Mecánicos: inclinación, vibración, aceleración.

Ejemplos de sensores usados en Arduino

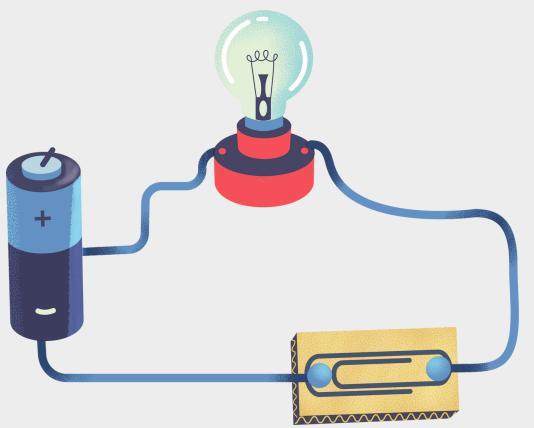
- LDR (resistencia dependiente de luz): mide luminosidad.
- DHT11/DHT22: temperatura y humedad ambiental.
- LM35: temperatura con precisión de 0.5 °C.



- HC-SR04 (ultrasonido): mide distancia en centímetros.
- PIR: detecta movimiento de personas/animales.
- Sensor de humedad de suelo: mide humedad en plantas.
- MQ-2/MQ-7: detectores de gas (propano, metano, humo, CO).



Importancia de los sensores en proyectos



- Sin sensores, el Arduino solo ejecuta instrucciones fijas.
- Con sensores, el sistema se vuelve interactivo y autónomo:
- Una lámpara automática enciende cuando hay oscuridad.
- Un sistema de riego activa la bomba cuando la tierra está seca.
- Un robot móvil evita obstáculos con un sensor ultrasónico.
- Valor agregado: sensores permiten automatización, base de la Industria 4.0 y el IoT.



Cómo Arduino procesa la información de un sensor

- El sensor genera una señal (analógica o digital).
- El pin del Arduino recibe esa señal.
- El microcontrolador la traduce a un valor binario.
- El programa (código en C++) decide qué hacer con ella.
- Arduino envía una acción: encender un LED, mover un motor, enviar datos al PC.

Sensores y el Internet de las Cosas (IoT)

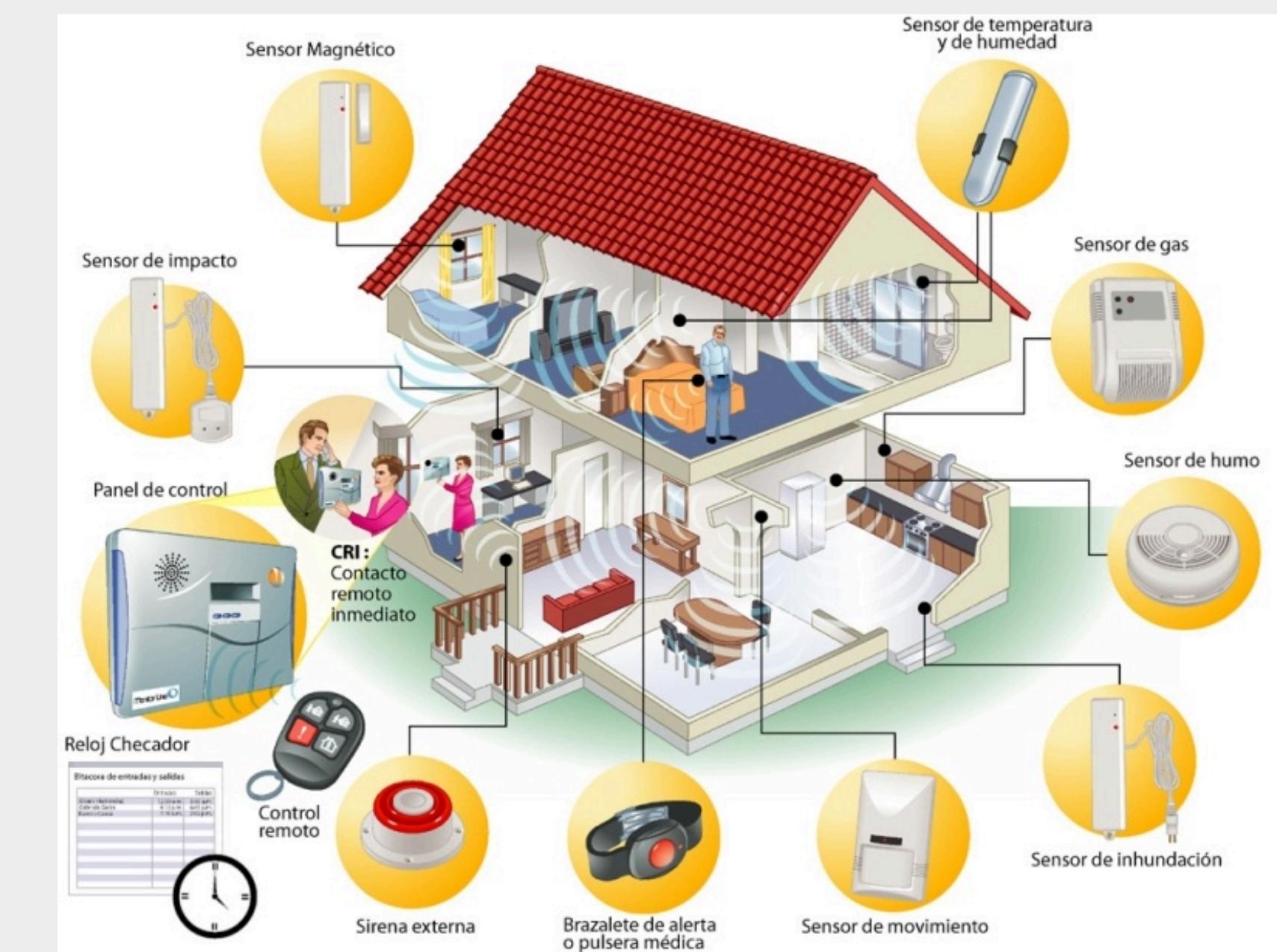
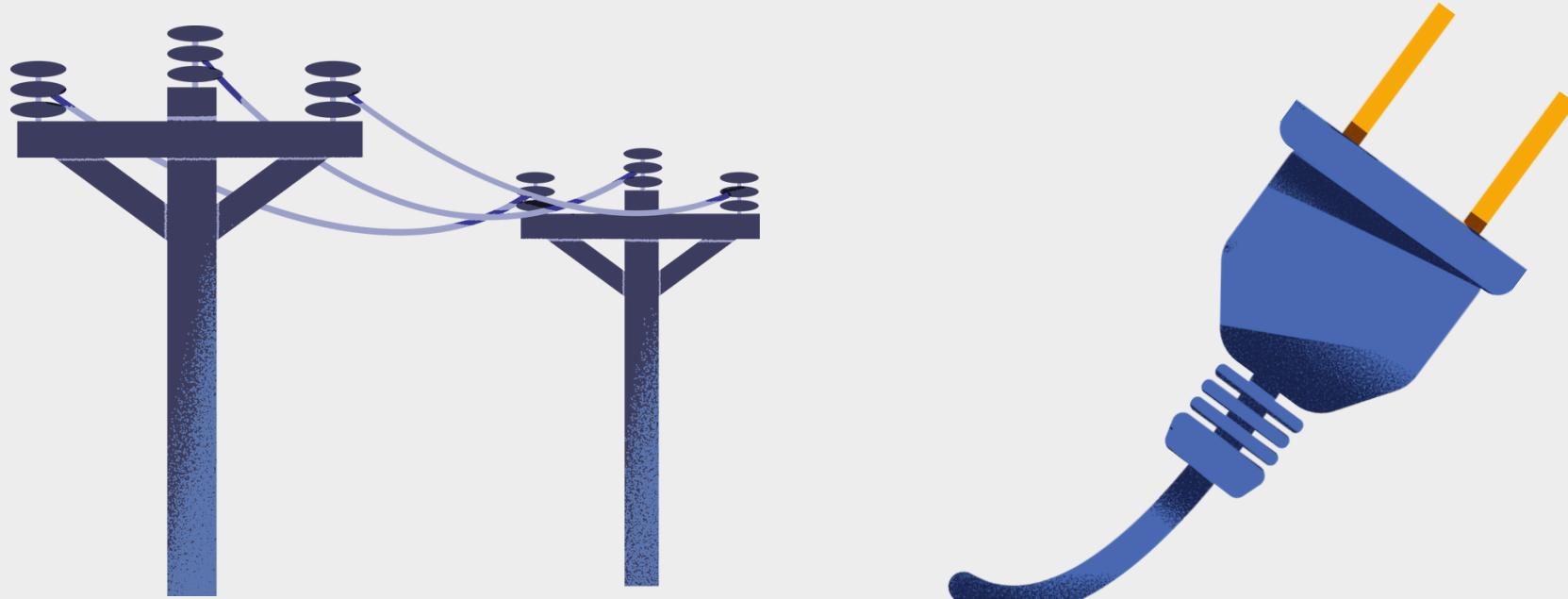
- Los sensores son la base del IoT.
- Ejemplo:
- Sensor de temperatura → mide 28 °C.
- Arduino procesa → envía dato a internet vía WiFi.
- App móvil muestra el valor.
- Aplicaciones: casas inteligentes, ciudades inteligentes, agricultura de precisión.

Ventajas de usar sensores con Arduino

- Gran variedad de sensores baratos y fáciles de conectar.
- Comunidad global con librerías listas para usar.
- Se pueden combinar varios sensores en un mismo proyecto.
- Posibilidad de prototipar proyectos reales (estaciones meteorológicas, alarmas, domótica).

Resumen Parte Teórica

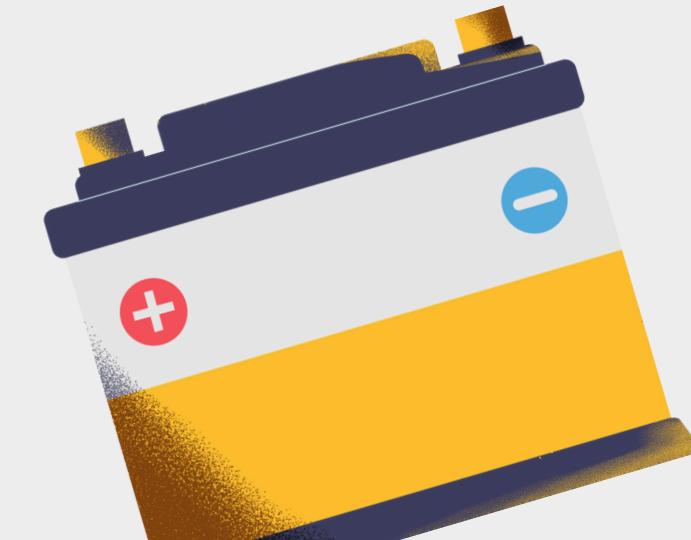
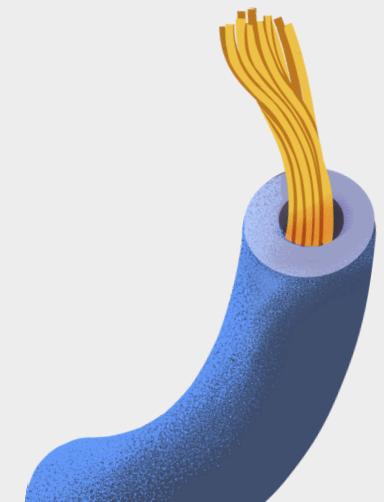
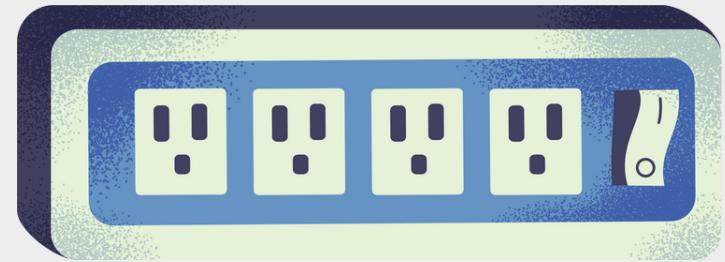
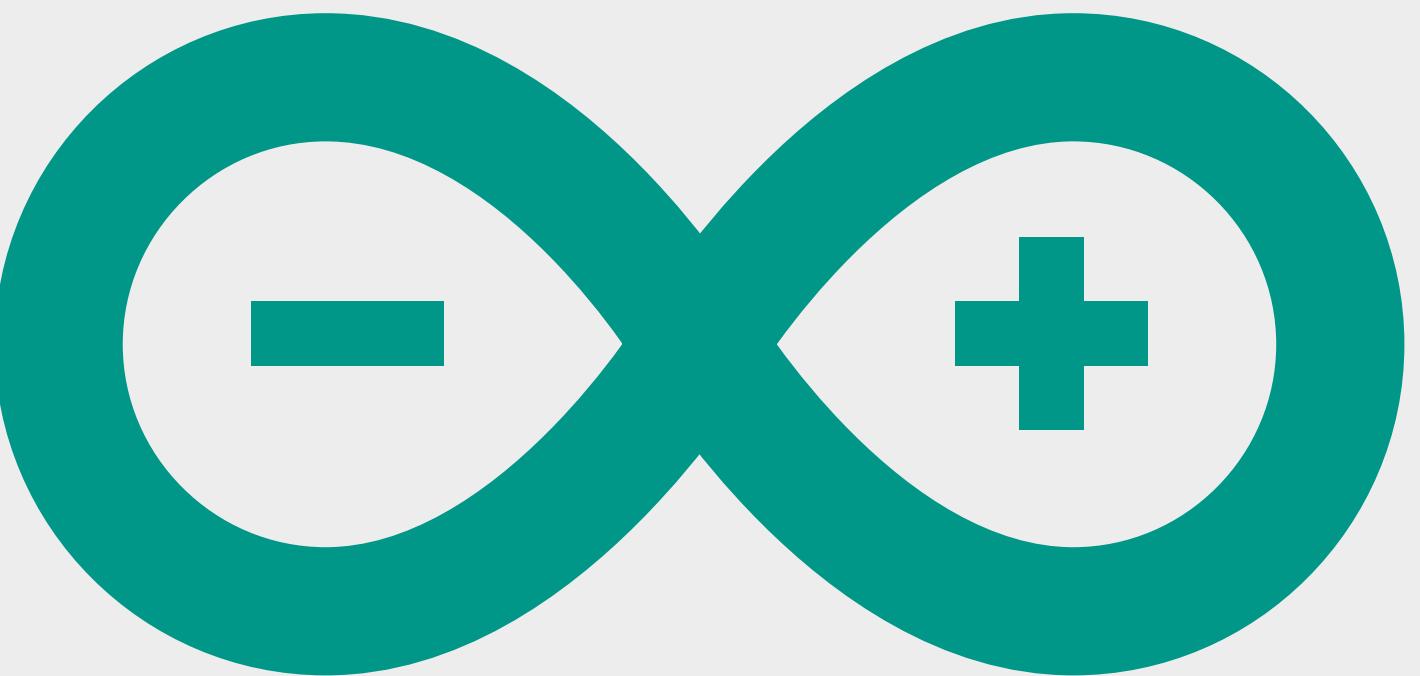
- ¿Qué es Arduino y por qué es importante.
- Historia y evolución de la plataforma.
- Conceptos básicos: microcontrolador, microprocesador, partes de la placa.
- Usos y aplicaciones en educación, prototipado, IoT y robótica.
- Sensores: tipos, ejemplos y su papel en proyectos reales.



Conclusiones

- Arduino democratiza el acceso a la electrónica y programación.
- Es una herramienta accesible, económica y respaldada por una gran comunidad.
- La combinación de sensores + Arduino = proyectos interactivos y útiles.
- El futuro del IoT y la industria maker depende de estas tecnologías.

ARDUINO IDE



¿Qué es el Arduino IDE?

01

Es el entorno oficial para escribir, compilar, subir y depurar “sketches” a placas Arduino y compatibles. En IDE 2 se integran: Editor moderno con autocompletado, Boards Manager, Library Manager, Serial Monitor, Serial Plotter y depuración.

02

Diferencias clave vs 1.8.x (“Legacy”): interfaz renovada, autocompletado, salto a definición, depuración integrada y herramientas de serie mejoradas.

¿Qué es exactamente IDE 2?

01

Arquitectura: está construido sobre Eclipse Theia y Electron; la compilación y subida las realiza un servidor arduino-cli en modo daemon bajo el capó. Es una reescritura mayor respecto a IDE 1.x.

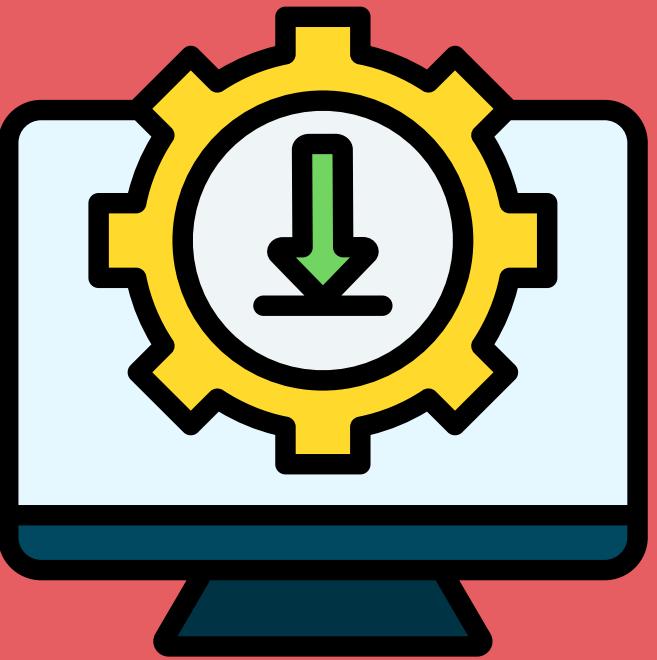
02

Descarga/instaladores: Windows, macOS y Linux desde el sitio oficial de Arduino (página “Software”).

02

Historial de versiones y notas: repositorio y “Releases” en GitHub. Útil para confirmar compatibilidad por SO.

Instalación y primer arranque



- Instalar IDE 2: guía oficial con pasos por sistema operativo. [Documentación de Arduino](#)
- Boards Manager (cores oficiales): abre Tools → Board → Boards Manager, busca el paquete (p. ej., “Arduino AVR Boards” para UNO/Nano/Mega) y pulsa Install. [Arduino Soporte+1](#)
- Cores de terceros (ESP32, ESP8266, etc.): agrega su URL en File → Preferences → Additional Boards Manager URLs, luego instálalo desde Boards Manager. (Ejemplo ESP32 de Espressif). [Arduino Soporte+1](#)
- Library Manager: Sketch → Include Library → Manage Libraries..., busca e instala; desde allí abres los Ejemplos de cada librería. [Documentación de Arduino+1](#)

TOUR DEL IDE

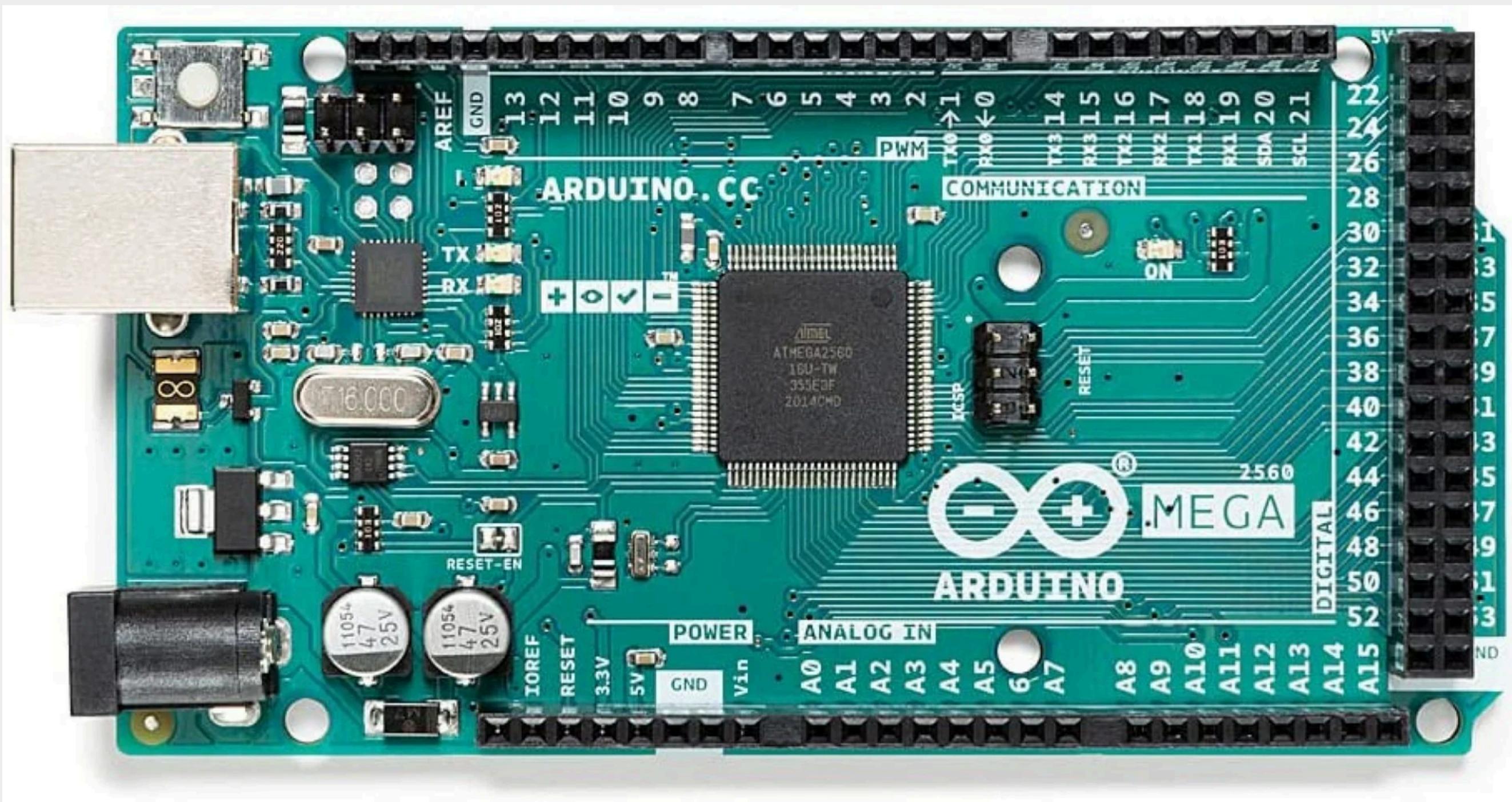
ARDUINO IDE

Arduino IDE (Integrated Development Environment) es el entorno de desarrollo oficial y gratuito creado por Arduino para que cualquier persona, desde principiantes hasta expertos en electrónica, pueda escribir, compilar y cargar programas en las placas Arduino y en muchas otras placas compatibles.



HERRAMIENTAS A UTILIZAR

- Arduino MEGA



HERRAMIENTAS A UTILIZAR

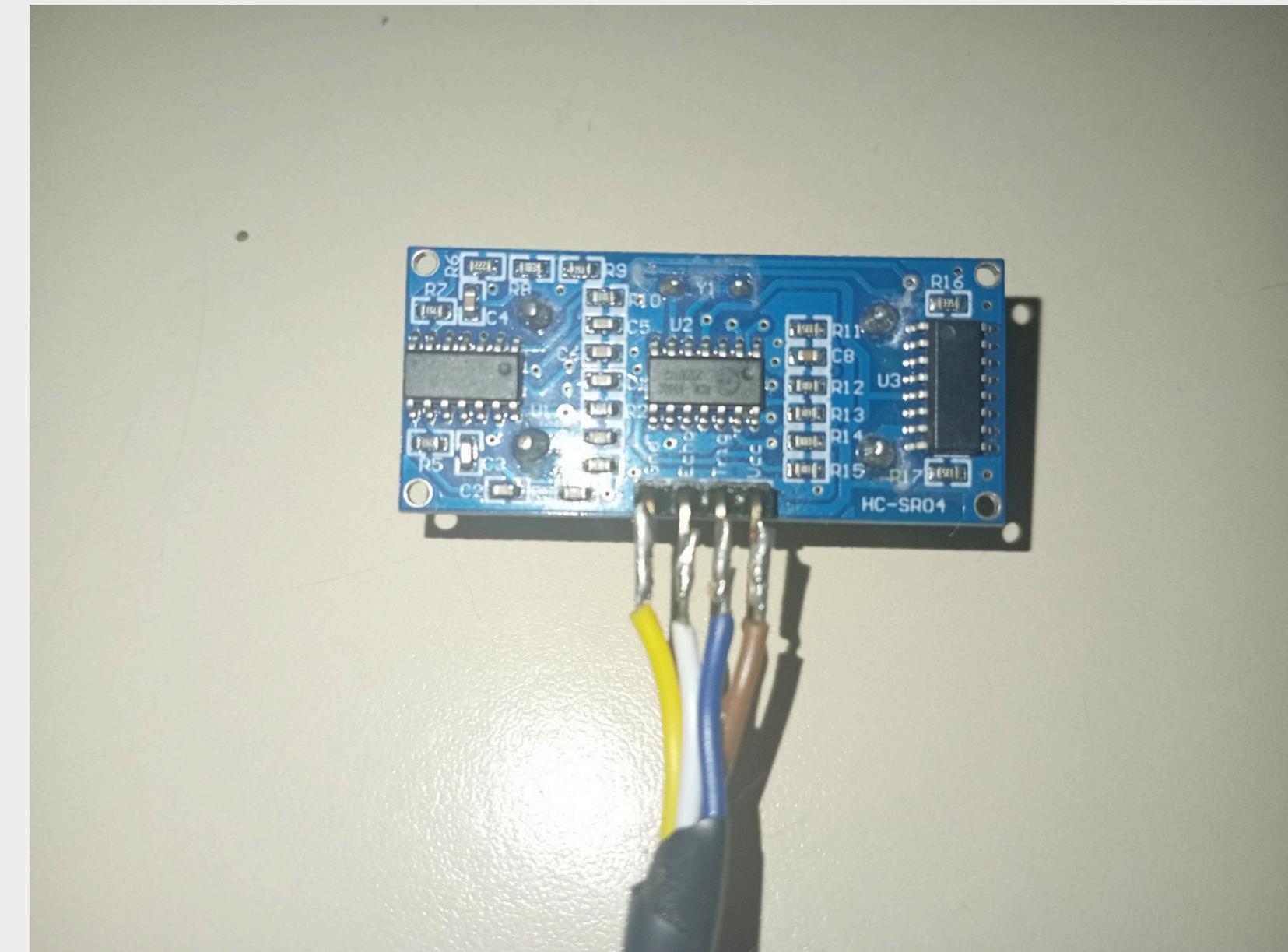
- Cable USB tipo A a USB tipo B



ULTRASÓNICO

Terminales del sensor

- VCC: Alimentación del sensor (5V).
- GND: Conexión a tierra (negativo).
- Trig (Trigger): Entrada que activa el envío del pulso ultrasónico.
- Echo: Salida que devuelve la duración del eco recibido, en microsegundos.



ULTRASÓNICO

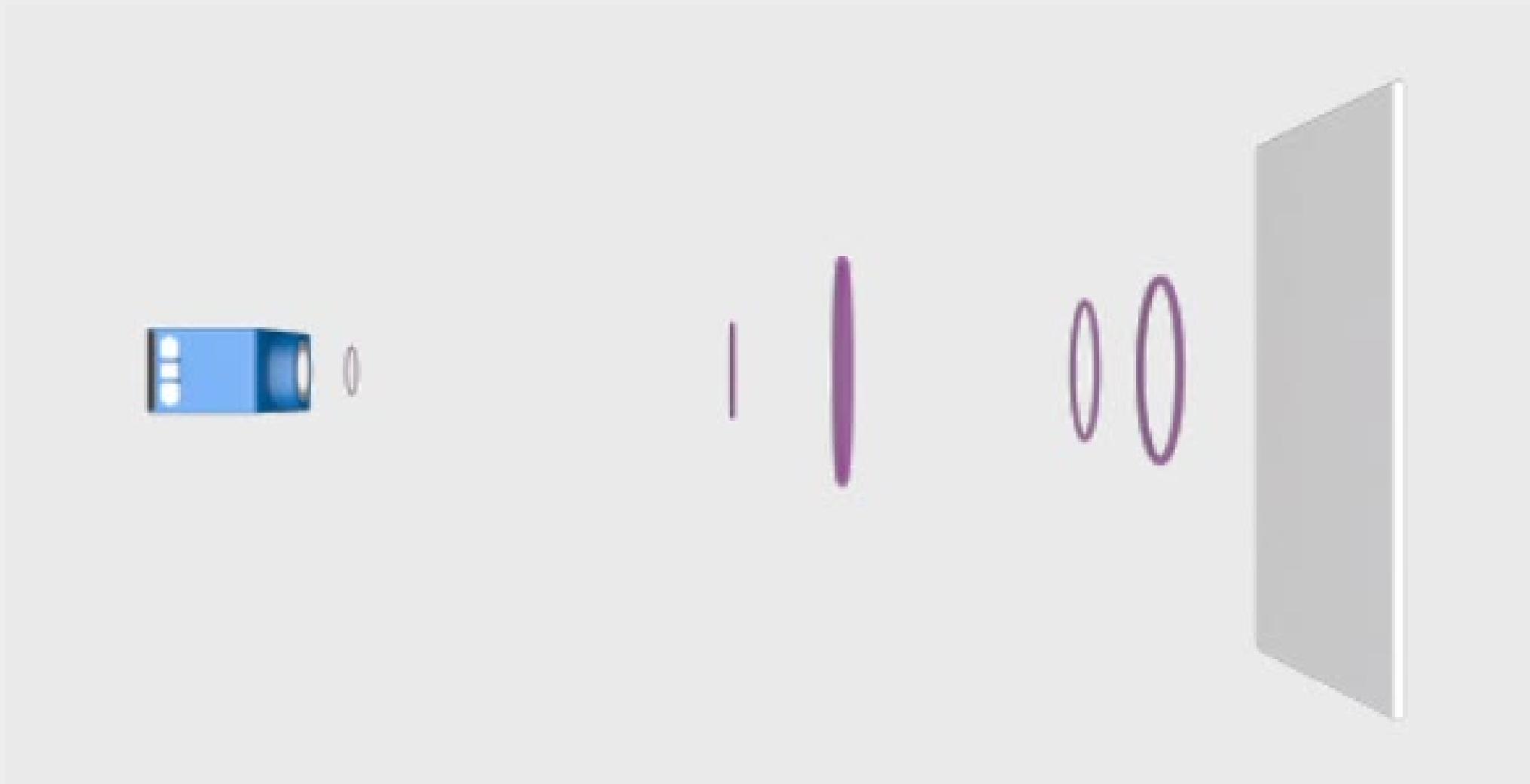
El sensor ultrasónico mide distancias utilizando ondas de sonido de alta frecuencia (ultrasonido).

Funciona así:

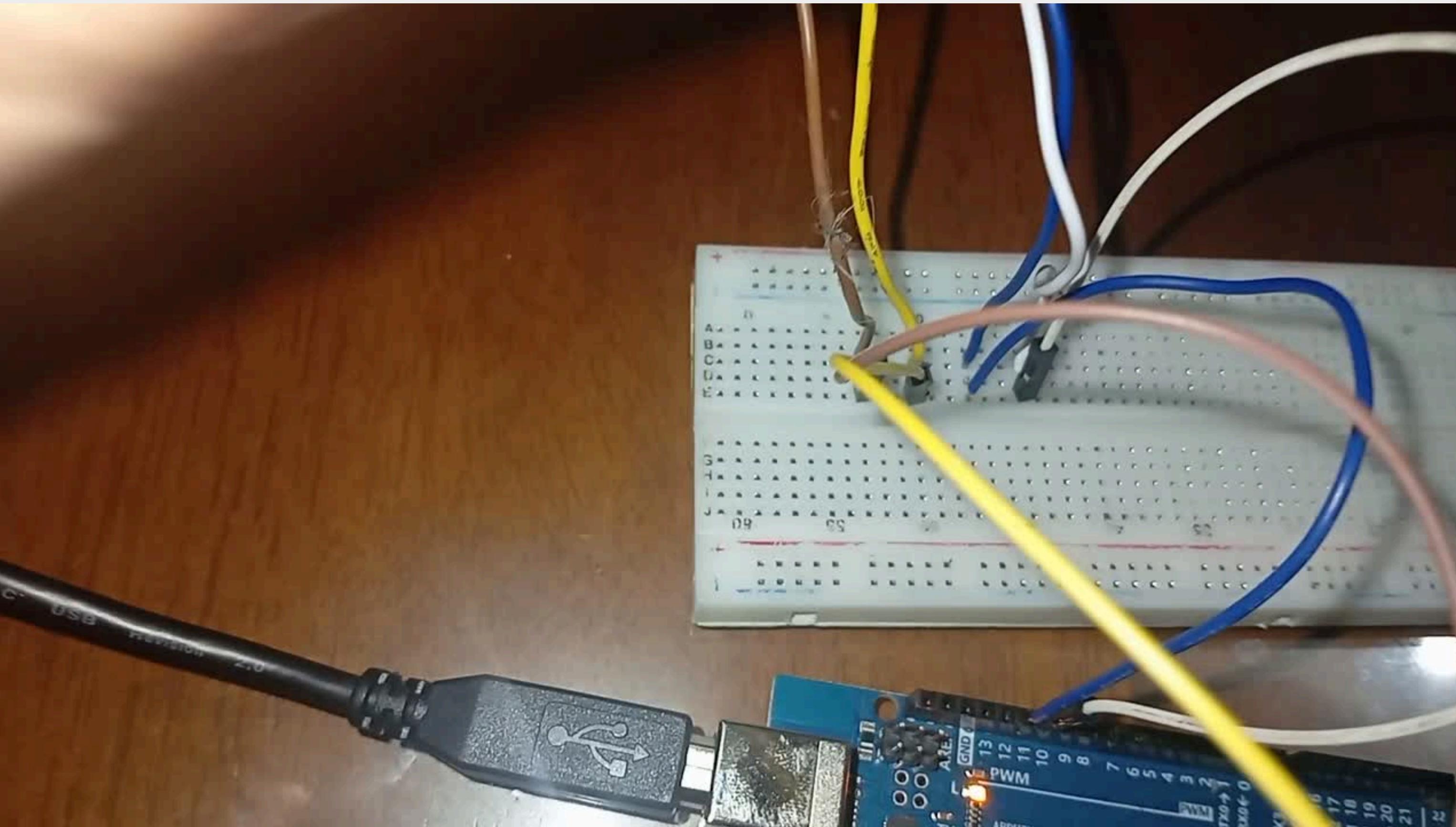
1. El Arduino envía un pulso eléctrico al pin Trig.
2. El sensor emite un tren de ondas ultrasónicas (40 kHz).
3. Estas ondas rebotan en el objeto más cercano.
4. El eco regresa al sensor y se recibe en el pin Echo.
5. Midiendo el tiempo que tarda en volver el eco, el Arduino calcula la distancia con la fórmula

$$d = (t \times 0.034) / 2$$

(0.034 cm/μs es la velocidad del sonido, y se divide entre 2 porque el sonido va y regresa).



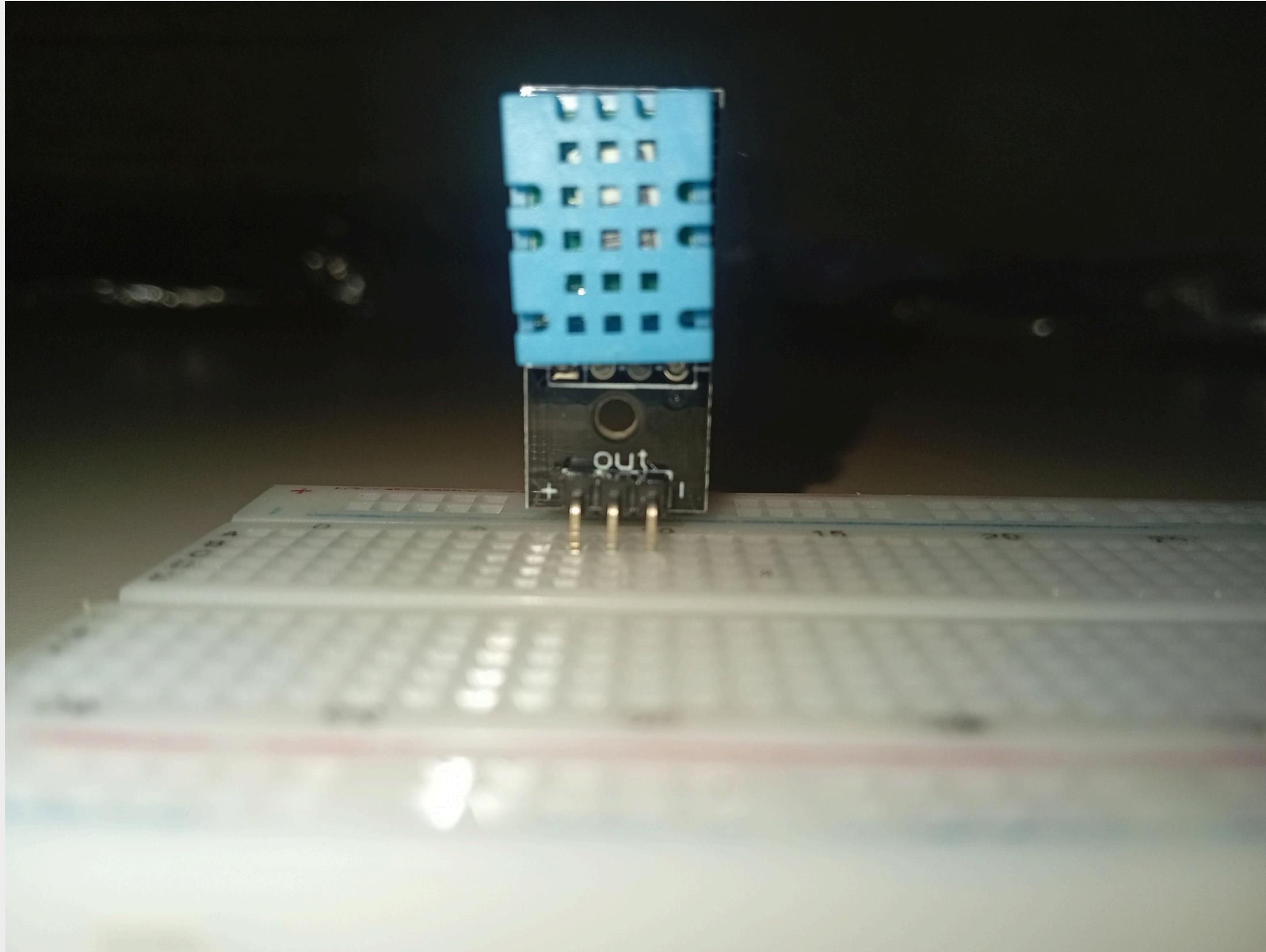
ULTRASÓNICO



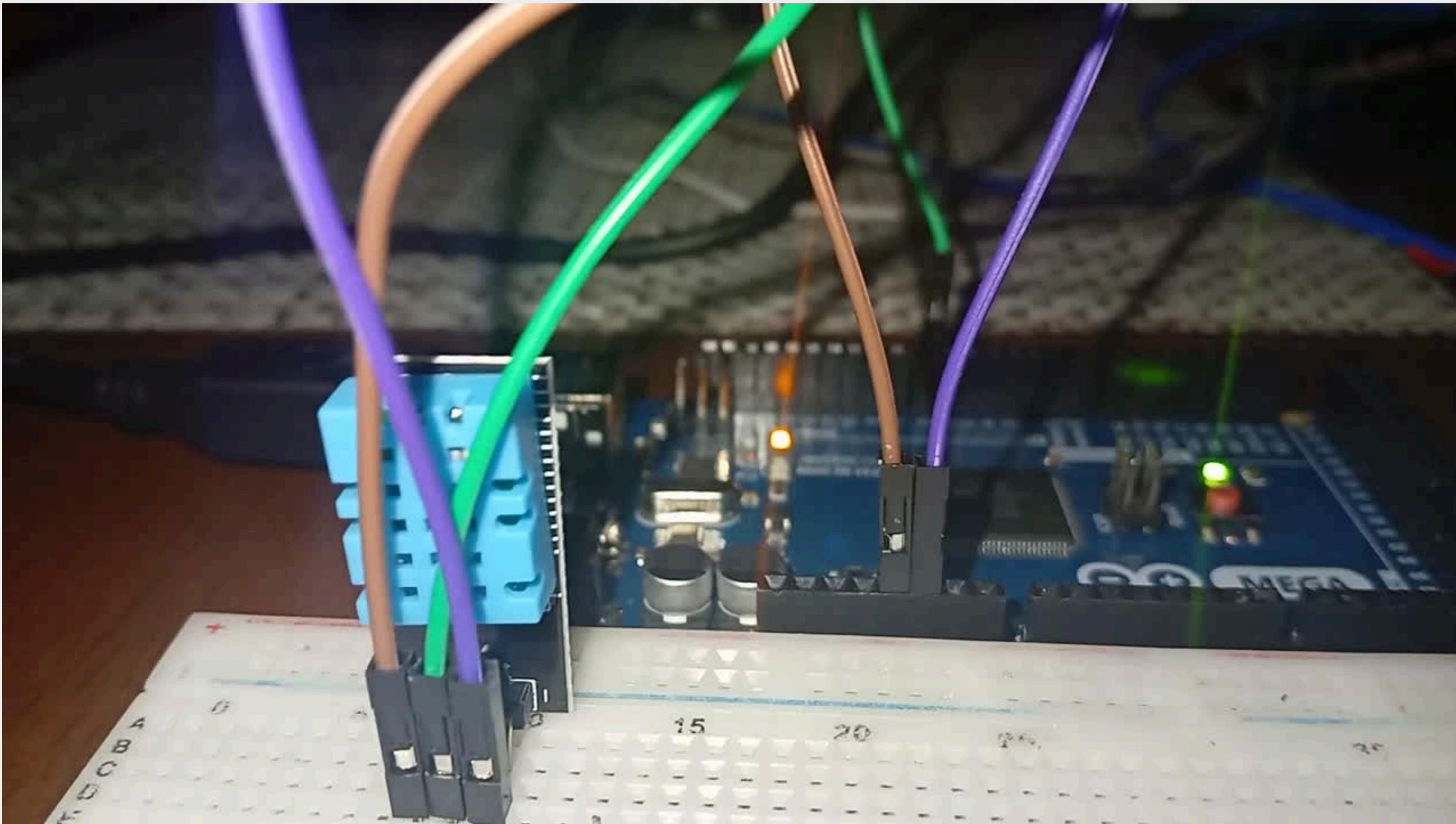
DHT11

Conexiones

- VCC
- out
- gnd



DHT11



LED

Terminales del LED

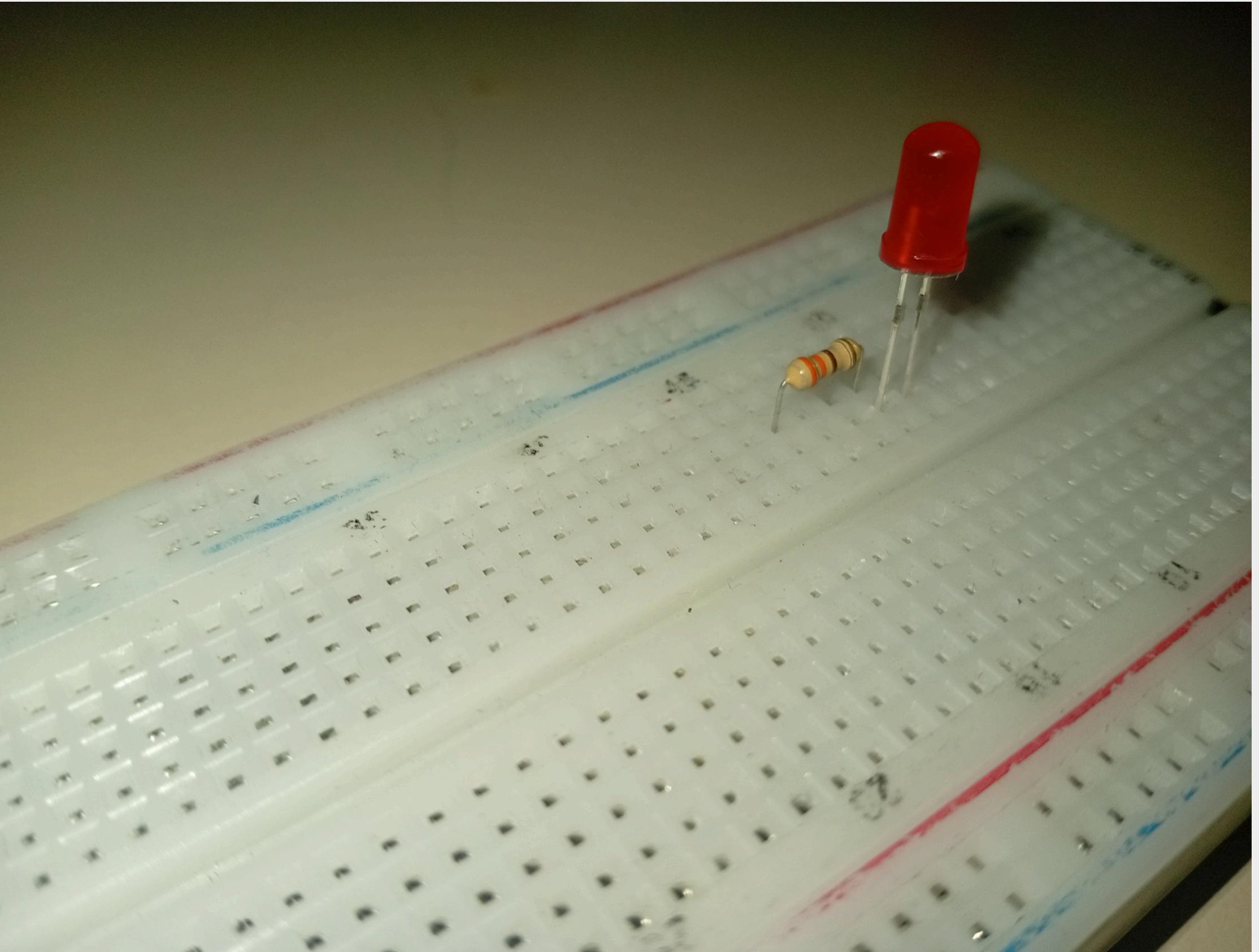
- Ánodo (+):

- Es el positivo del LED.
- Debe conectarse al voltaje (por ejemplo, a un pin digital de Arduino o a 5V/3.3V a través de una resistencia).
- Es el pin más largo en la mayoría de los LEDs.

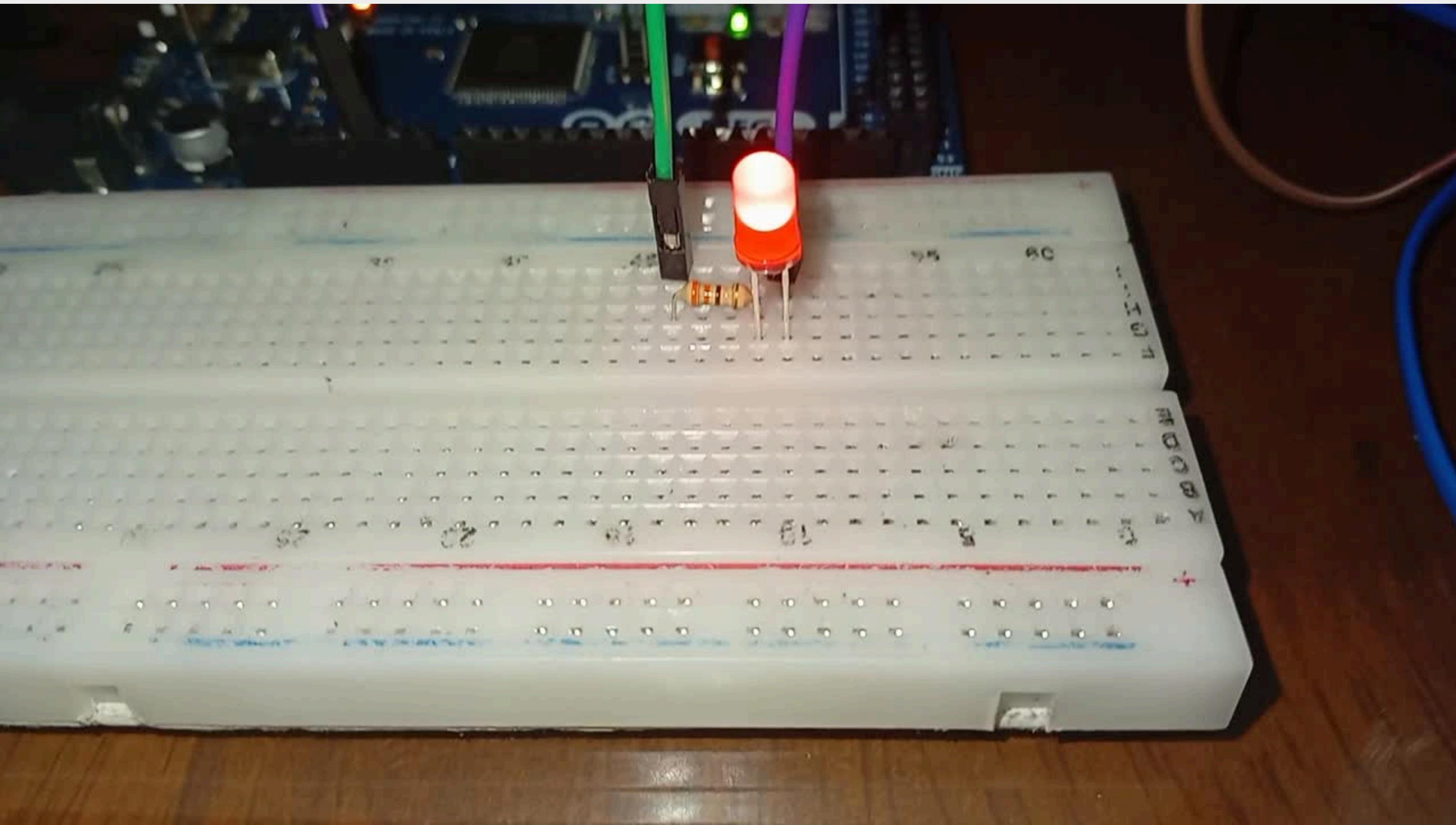
- Cátodo (-):

- Es el negativo del LED.
- Se conecta a GND (tierra) en el circuito.
- Es el pin más corto y en el encapsulado suele haber una parte plana en la base para identificarlo.

Resistencia de 330Ω



LED



LCD

Una LCD (Liquid Crystal Display) es una pantalla de cristal líquido que se utiliza para mostrar información de manera sencilla y económica en proyectos electrónicos. En el caso de Arduino, las más comunes son las de 16x2 (16 caracteres por 2 filas) y 20x4 (20 caracteres por 4 filas).

En este caso se utiliza una LCD 16x2 con adaptador I2C integrado.

- Esto significa que la pantalla tiene 16 columnas y 2 filas, permitiendo mostrar hasta 32 caracteres en total.
- El adaptador I2C que va en la parte trasera simplifica la conexión, reduciendo de 16 terminales a solo 4 terminales (GND, VCC, SDA, SCL).

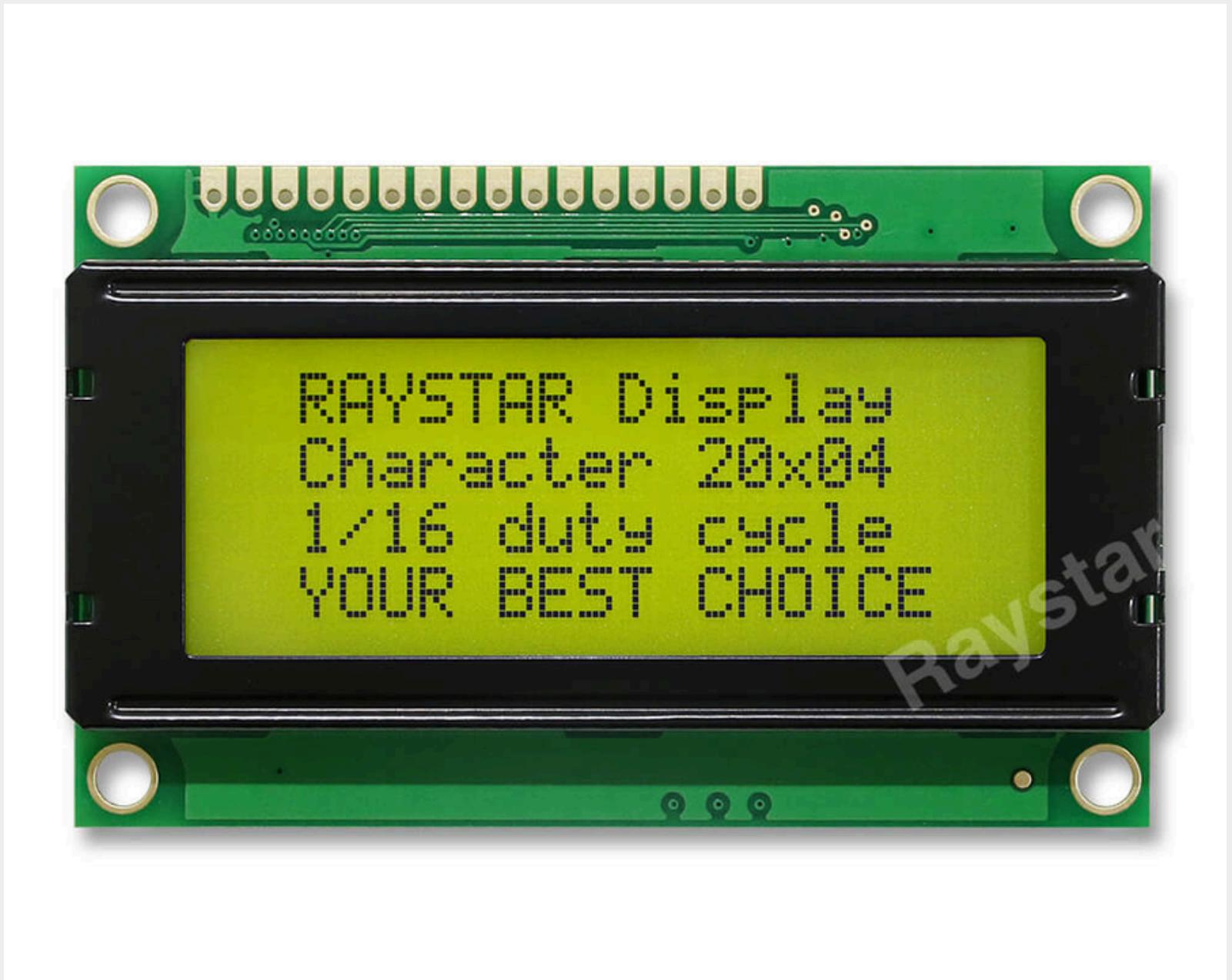
¿Qué es I2C?

I2C (Inter-Integrated Circuit) es un protocolo de comunicación en serie sincrónico, creado por Philips en los años 80, que se utiliza para conectar microcontroladores con periféricos (como pantallas, sensores, memorias o convertidores) usando únicamente dos líneas de comunicación compartidas.

- SDA (Serial Data): es la línea por donde viajan los datos.
- SCL (Serial Clock): es la línea de reloj que marca el ritmo de la comunicación y asegura que tanto el emisor como el receptor estén sincronizados.

LCD

20x4



16x2



LCD

Terminales de una LCD con módulo I2C

1. GND (Ground)

- Terminal de tierra o negativo de la alimentación.
- Se conecta al GND del Arduino.

2. VCC (Voltage Common Collector)

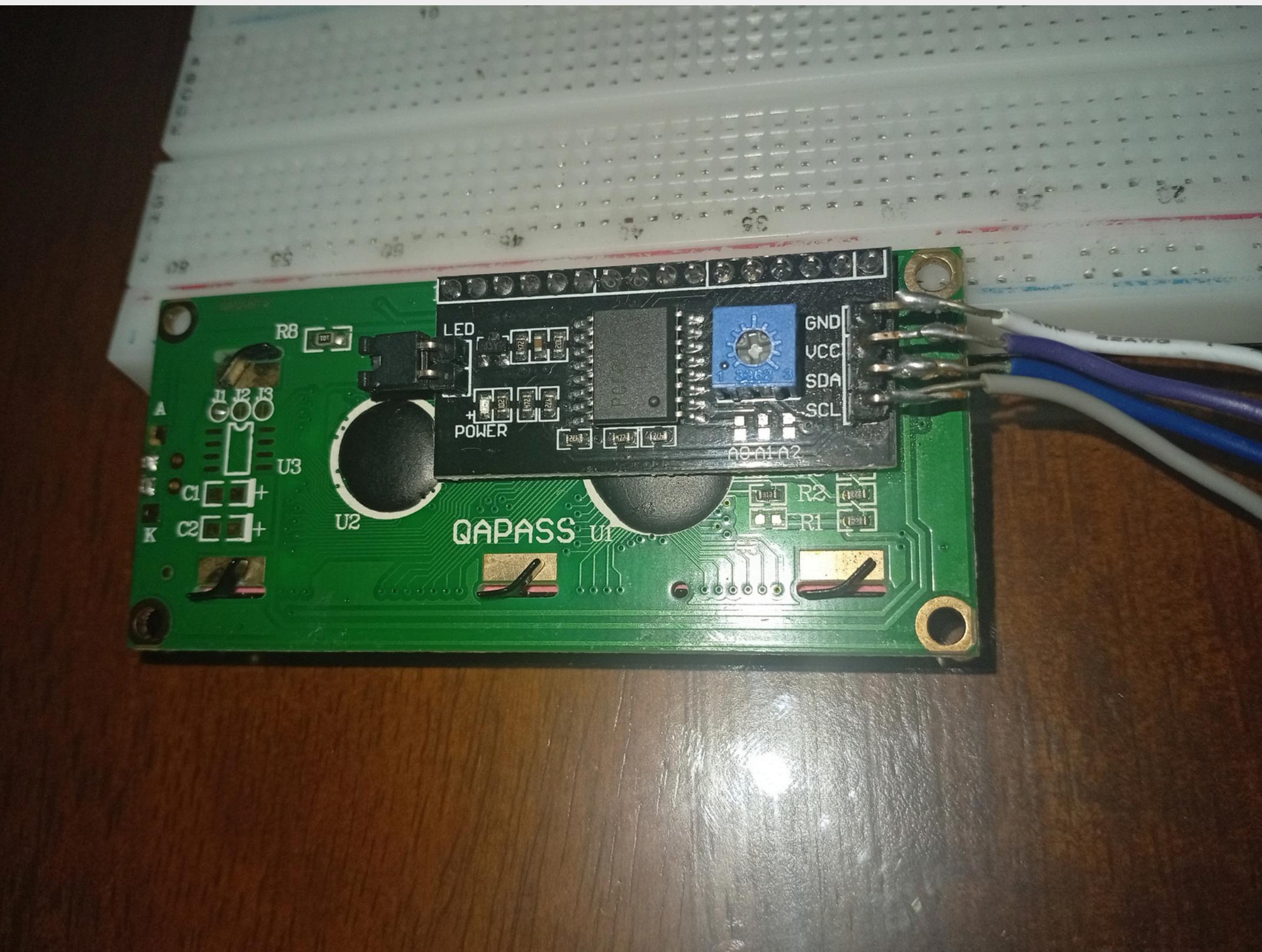
- Terminal de alimentación positiva.
- Generalmente funciona con 5V (aunque algunas placas aceptan 3.3V).

3. SDA (Serial Data)

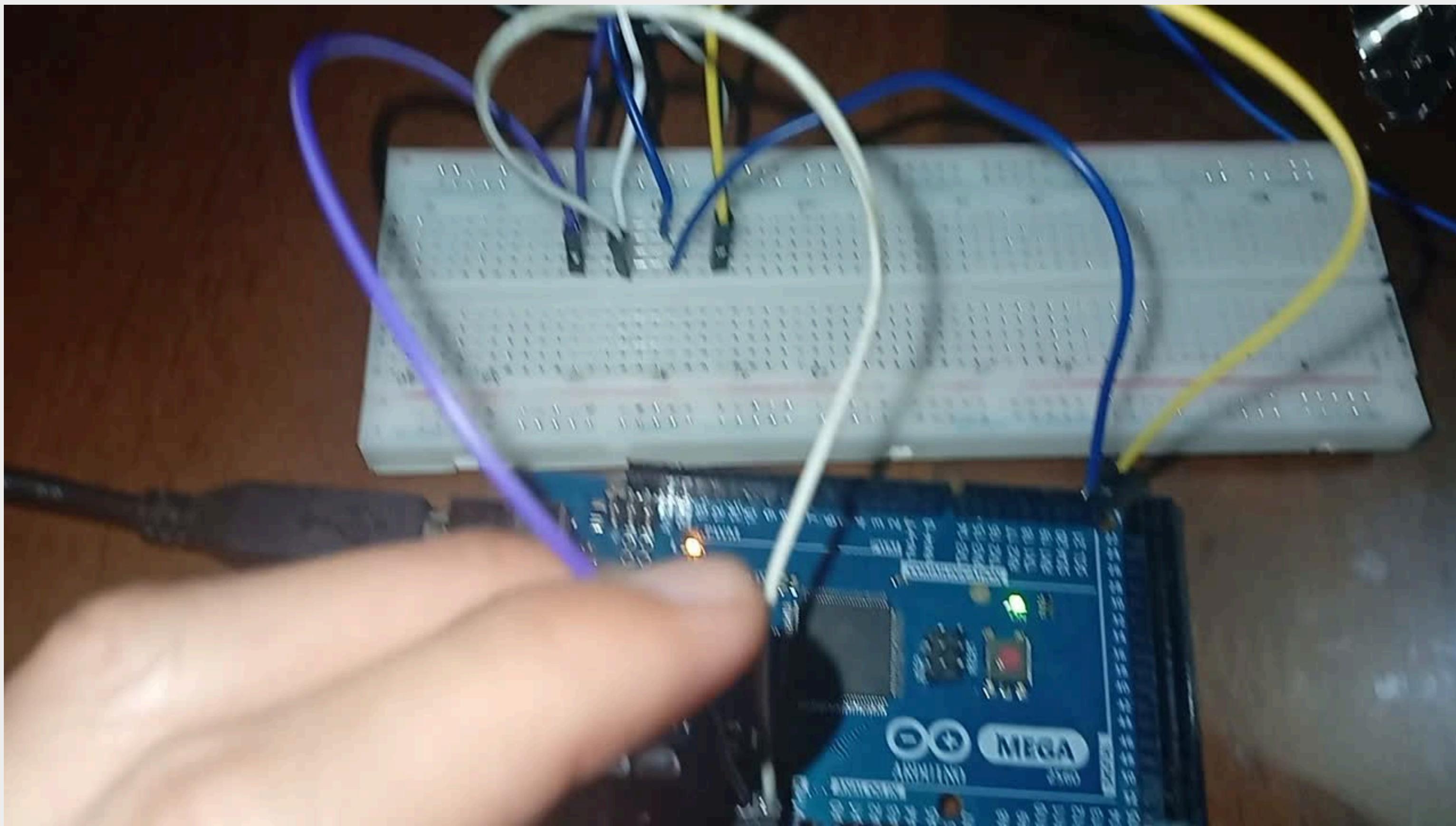
- Línea de datos en el bus I2C.
- En Arduino UNO → corresponde al pin A4.
- En Arduino Mega → corresponde al pin 20.

4. SCL (Serial Clock)

- Línea de reloj en el bus I2C, que sincroniza la comunicación.
- En Arduino UNO → corresponde al pin A5.
- En Arduino Mega → corresponde al pin 21.



LCD



Conclusiones del Proyecto

**Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Vivamus quis ultrices felis. Fusce sapien nunc, posuere at
mauris sed, sagittis luctus erat. Integer sollicitudin
pellentesque dolor ac suscipit.**



Muchas
GRACIAS

www.unsitioenial.es

RECURSOS GRÁFICOS

