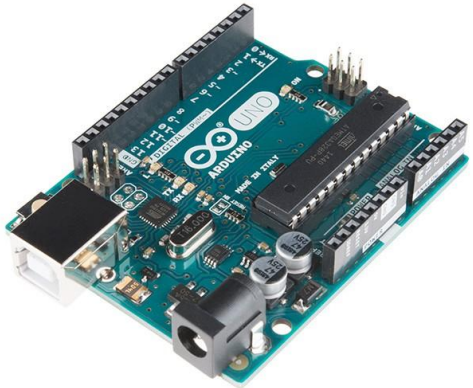
Introducción a Arduino

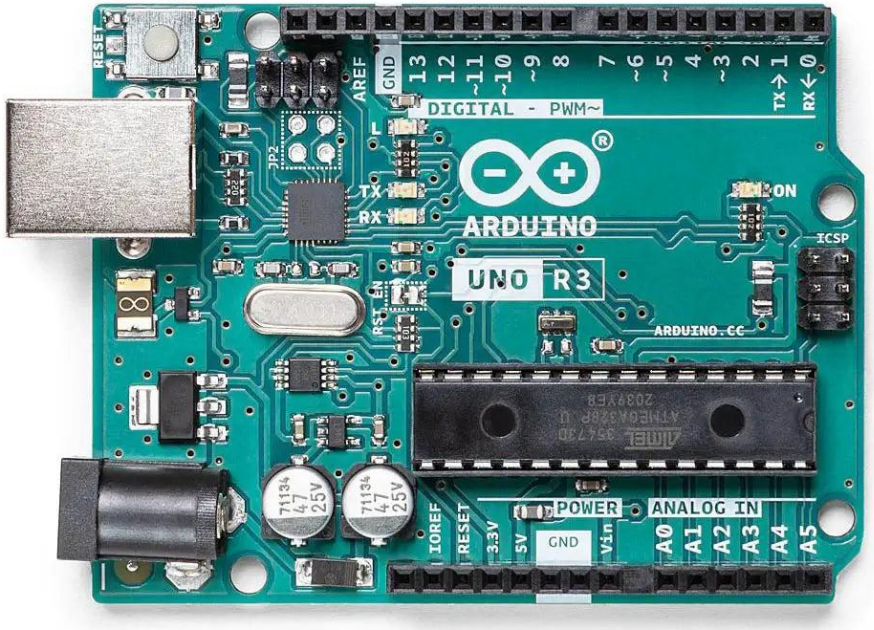


Antonio Rodríguez Nieto

Sesión 1

Qué es Arduino Uno?

- Plataforma de software libre
- Microcontrolador
- Reprogramable
- Basado en C++
- I/O (Entradas/salidas) configurables



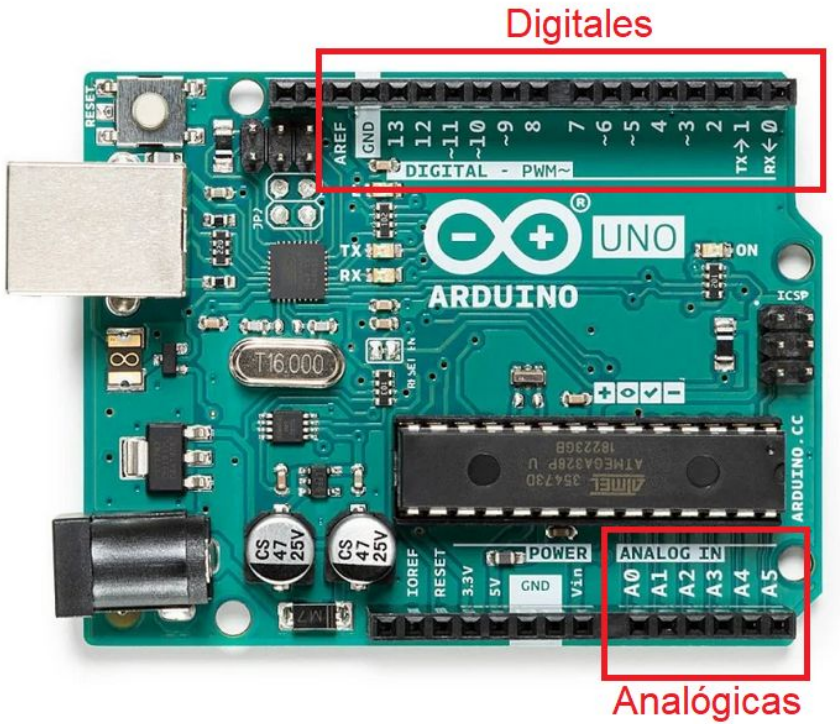
Pines Arduino Uno

Se pueden encontrar dos tipos de pines:

- Digitales (entrada y salida):

  Además, existen seis pines (~) que se pueden convertir en salidas analógicas “falsas”.

- Analógicos (entrada)

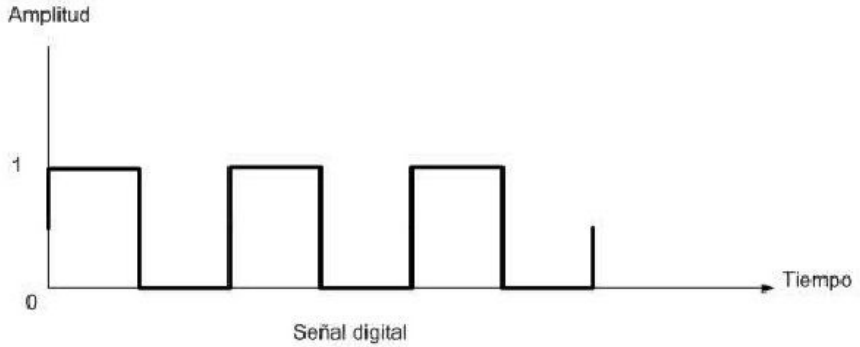


Señales digitales

Las señales digitales se caracterizan por tener únicamente dos estados: BAJO o ALTO (LOW o HIGH).

Para actuadores, su uso suele ser en aquellos que únicamente tengan dos estados, encendido o apagado (una alarma, una bombilla…).

Para sensores, sería el mismo principio (detectar una presencia, un contacto…).

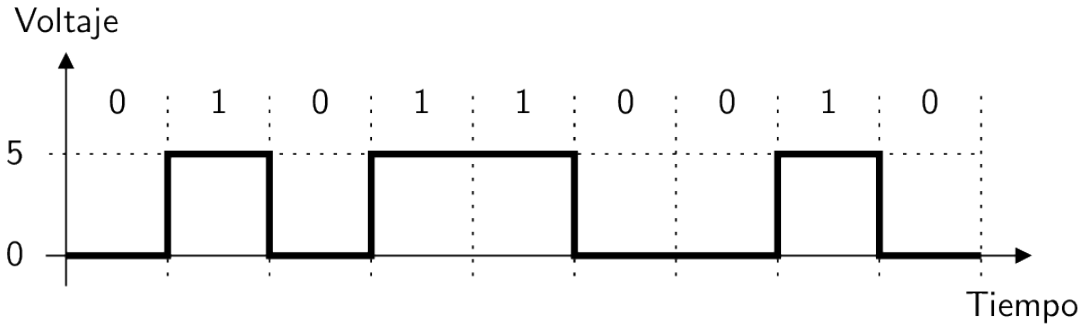


Señales digitales

En el caso de Arduino Uno, un pin digital puede ser configurado como entrada (sensor) o salida (actuador).

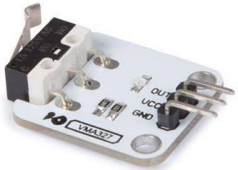
Los niveles de voltaje digitales son:

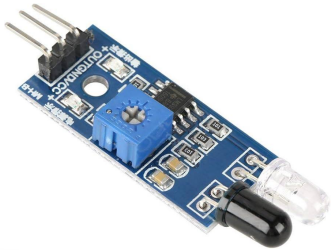
- ALTO (HIGH) = 5V. En caso de “entrada”, > 3V.
- BAJO (LOW) = 0V. En caso de “entrada”, < 3V.



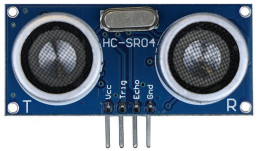
Sensores y actuadores

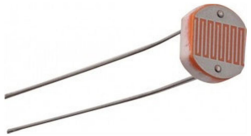
Digitales





Analógicos





Arduino IDE

El IDE (Entorno de Desarrollo Integrado) de Arduino cuenta con varias funcionalidades, siendo las más importantes:

- Creación y compilación de código
- Portar (subir) a la placa de Arduino el programa compilado
- Visualización del puerto Serial en tiempo real

Herramientas Arduino IDE

Las herramientas más importantes del IDE son las siguientes:

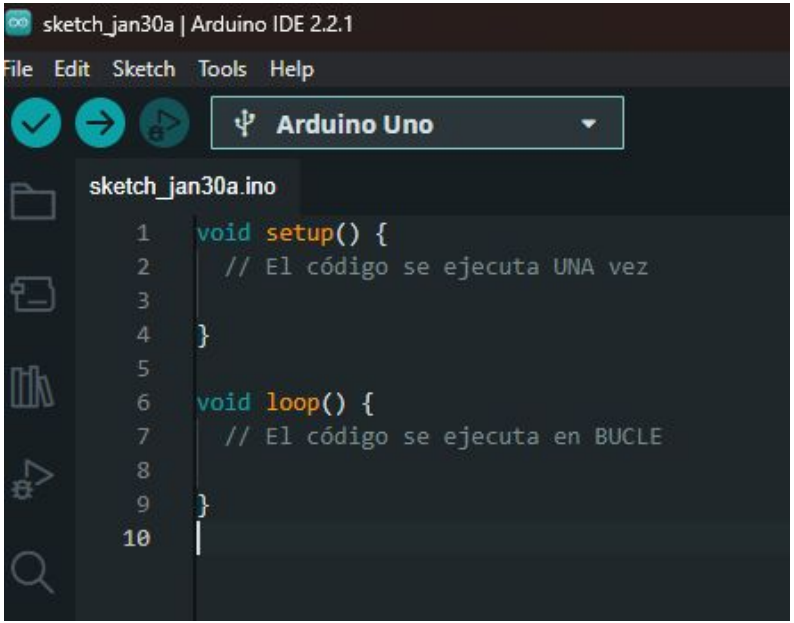
- Selección de placa (una vez conectada)
- Compilación
- Subir a la placa (si no se ha compilado, lo hace, y luego sube)

Programación en Arduino IDE

La programación en Arduino se hace por defecto en las funciones setup() y loop().

- Setup(): Se ejecuta únicamente una vez al inicio del programa. Utilizada para hacer configuración o inicializaciones.

- Loop(): Se ejecuta repetidamente una vez se ha terminado la función setup(). Es el núcleo del programa.



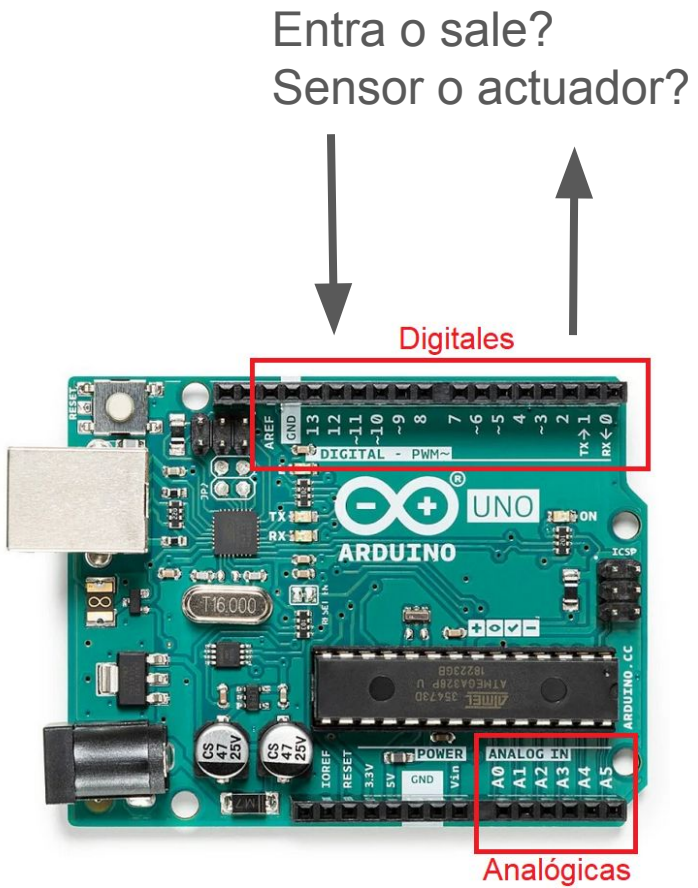
Programación en Arduino IDE

Para decirle a Arduino si un pin es de entrada o salida, se usa la función:

pinMode(pin, modo);

- pinMode(5, INPUT);
  // pin 5 como entrada

- pinMode(10, OUTPUT);
  // pin 10 como salida



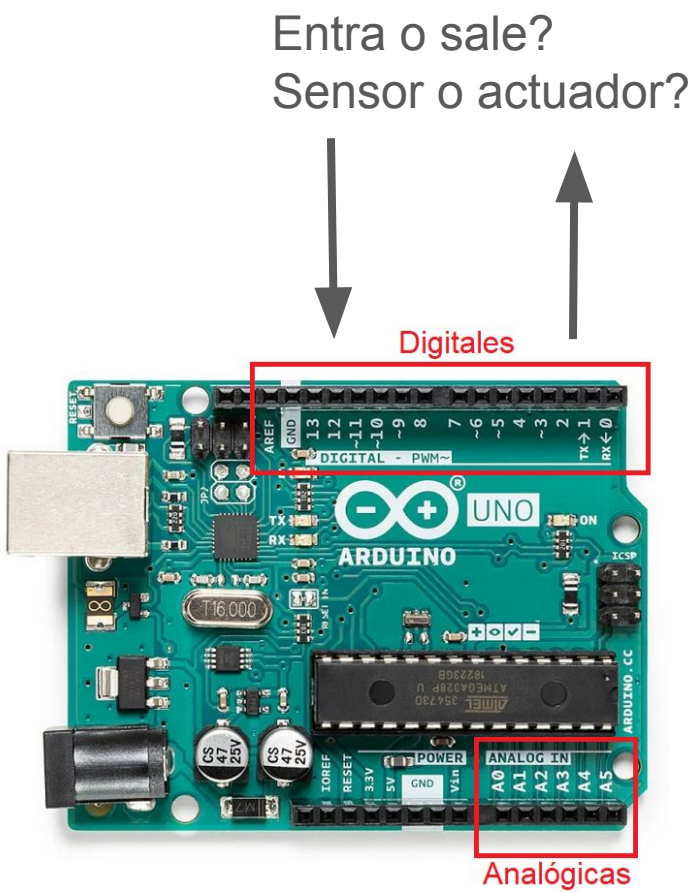
Programación en Arduino IDE

Para leer o escribir a través de los pines, se usan las siguientes funciones:

- digitalWrite(pin, estado):

  digitalWrite(10, HIGH);
  // 5V en pin 10
  digitalWrite(10, LOW);
  // 0V en pin 10

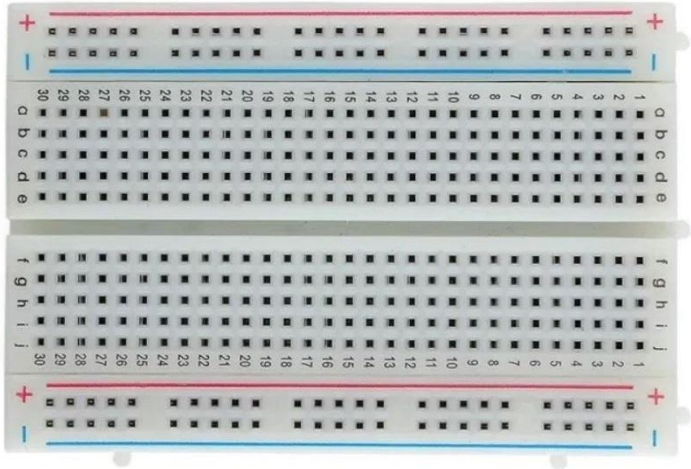
- digitalRead(pin):
  int value = digitalRead(5);
  // Lee el valor que tiene el pin 5 (HIGH 5V o LOW 0V)

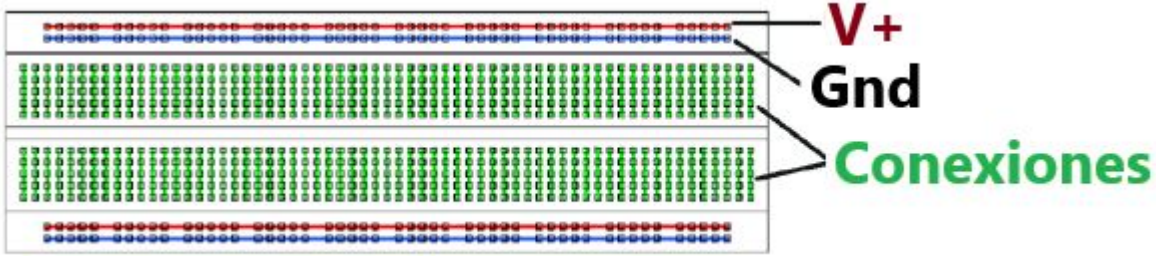


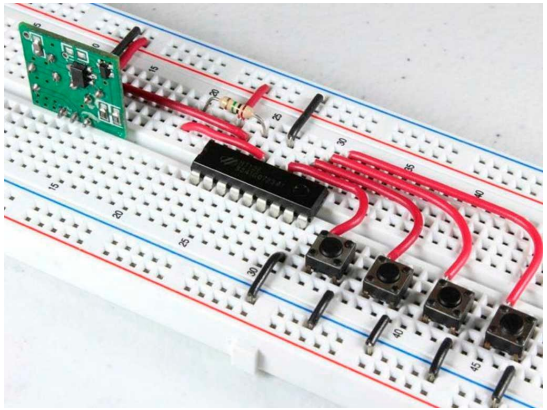
Placa de prototipos

Utilizada para conectar dispositivos

Compuesta por vías interconectadas

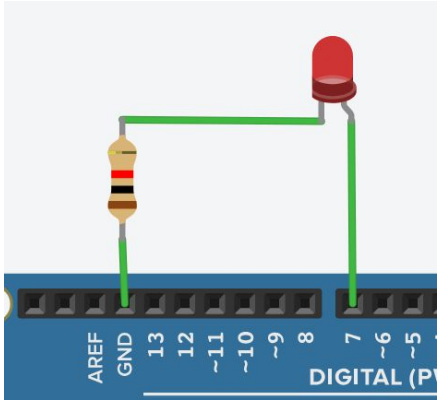


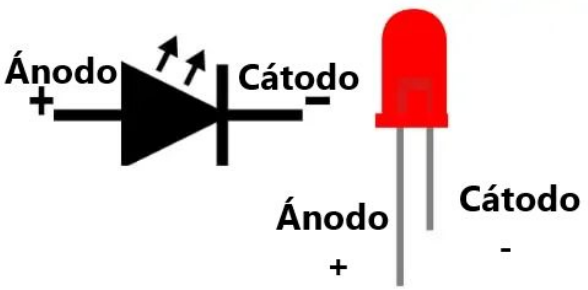


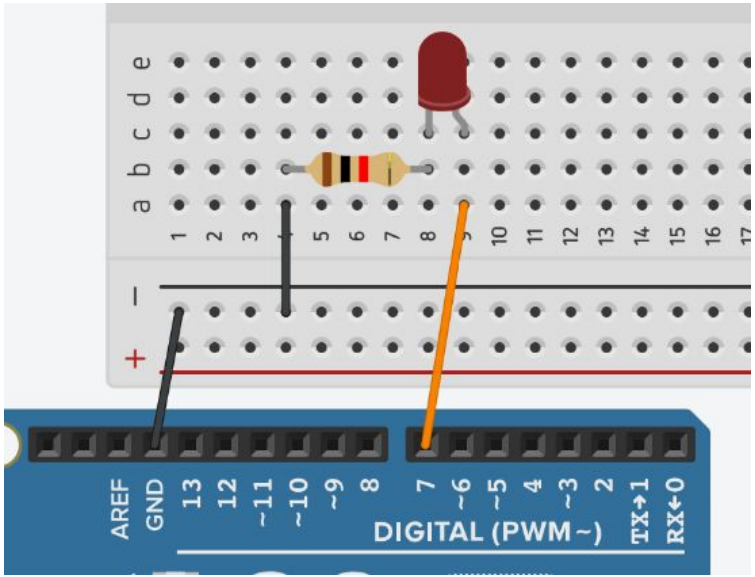


Actividades

1. Encender LED
2. Parpadear LED
3. Semáforo LED
4. Semáforo LED al pulsar un botón







Sesión 2

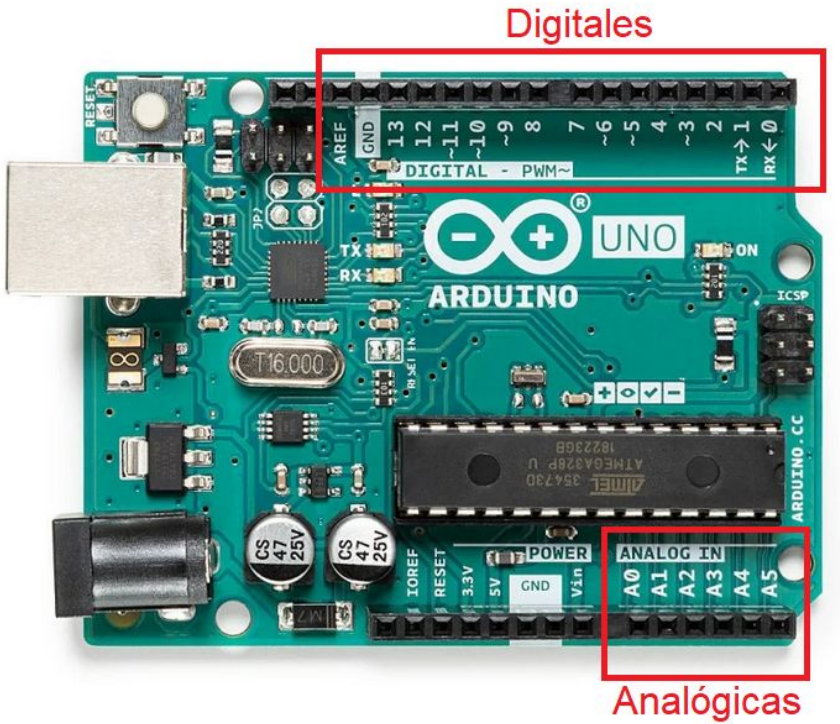
Pines Arduino Uno

Se pueden encontrar dos tipos de pines:

- Digitales (entrada y salida):

  Además, existen seis pines (~) que se pueden convertir en salidas analógicas “falsas”.

- Analógicos (entrada)

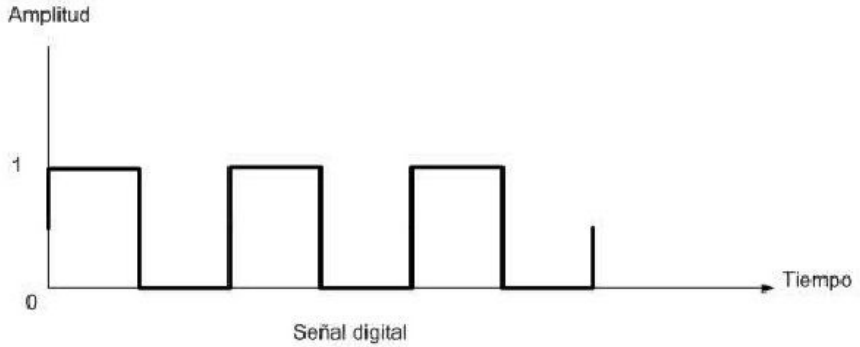


Señales digitales

Las señales digitales se caracterizan por tener únicamente dos estados: BAJO o ALTO (LOW o HIGH).

Para actuadores, su uso suele ser en aquellos que únicamente tengan dos estados, encendido o apagado (una alarma, una bombilla…).

Para sensores, sería el mismo principio (detectar una presencia, un contacto…).

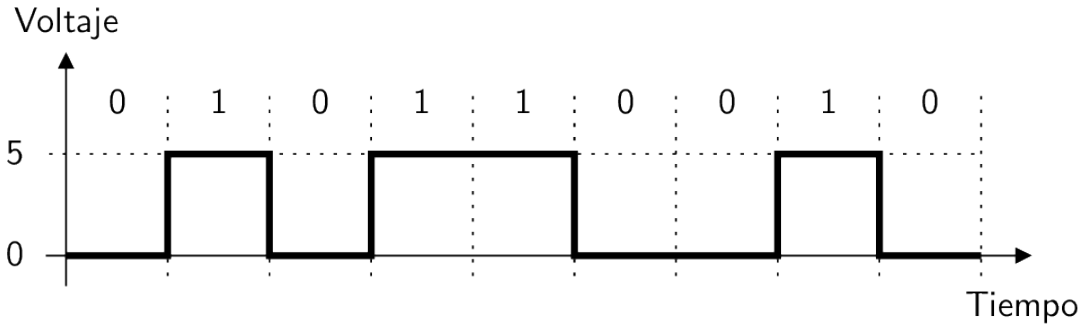


Señales digitales

En el caso de Arduino Uno, un pin digital puede ser configurado como entrada (sensor) o salida (actuador).

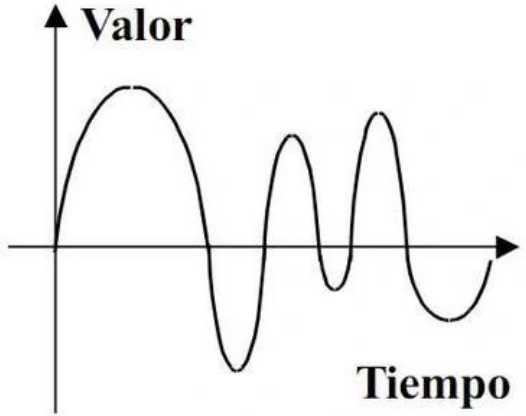
Los niveles de voltaje digitales son:

- ALTO (HIGH) = 5V. En caso de “entrada”, > 3V.
- BAJO (LOW) = 0V. En caso de “entrada”, < 3V.



Señales analógicas

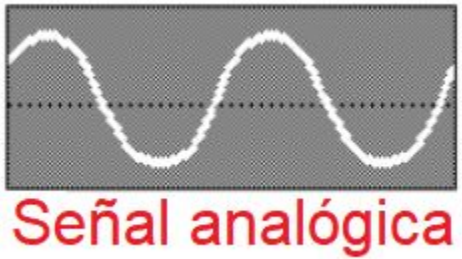
Estas señales, a diferencia de las digitales, tienen un rango de valores infinitos. Para su lectura, se convierte la señal analógica a digital.



Señales analógicas

En Arduino uno se pueden leer en los pines analógicos hasta 5V.

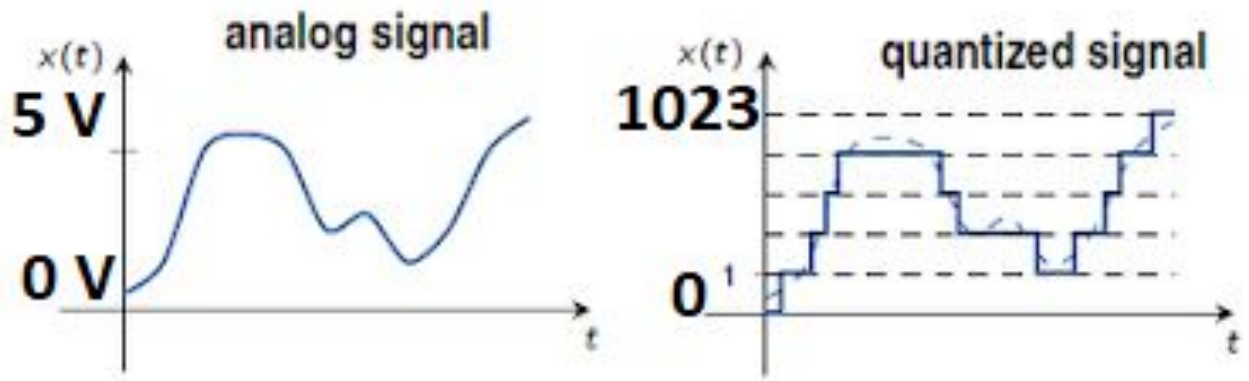
Como tenemos 10 bits de resolución, tenemos 2^{10}=1024 posibles niveles. Esto supone una precisión de ± 2,44 mV (5 V/1024 bits).



Señales analógicas

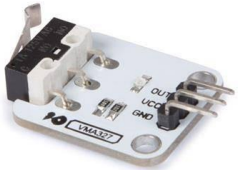
En Arduino uno se pueden leer en los pines analógicos hasta 5V.

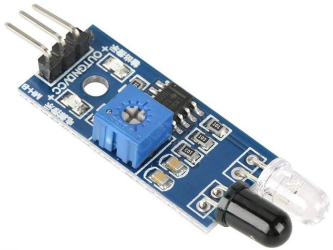
Como tenemos 10 bits de resolución, tenemos 2^{10}=1024 posibles niveles. Esto supone una precisión de ± 2,44 mV (5 V/1024 bits).



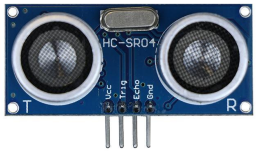
Sensores y actuadores

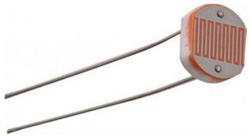
Digitales





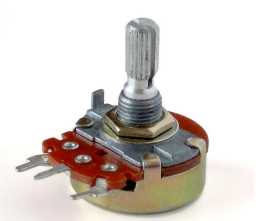
Analógicos

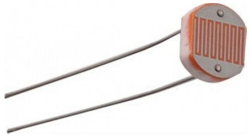


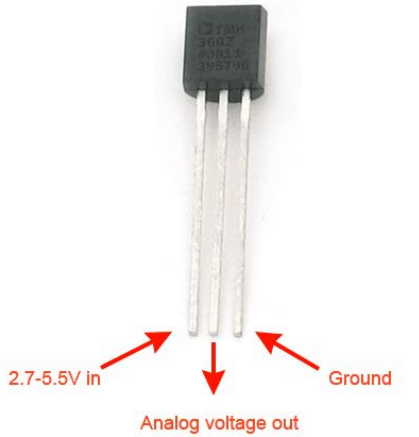


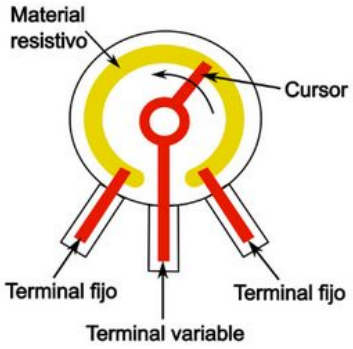
Sensores

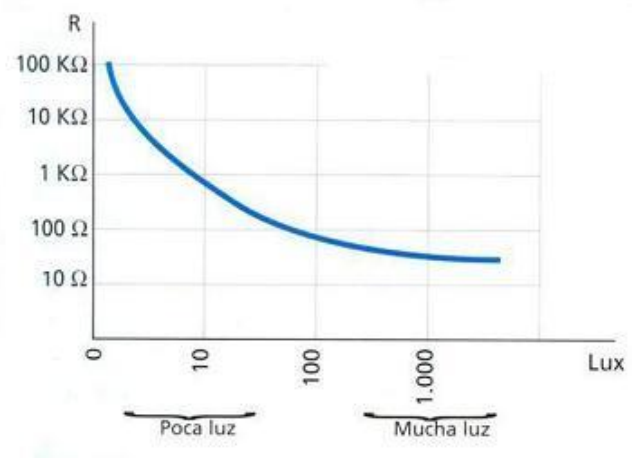
Analógicos

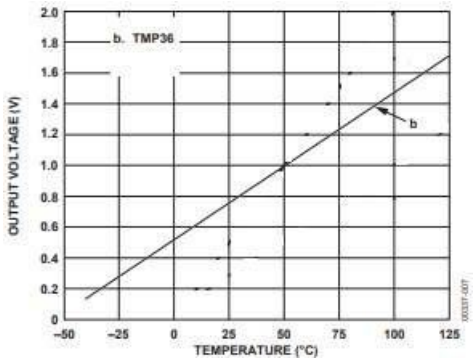






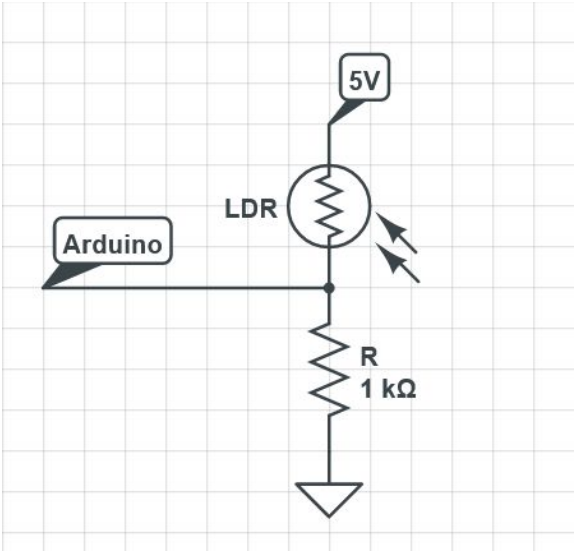






Lectura de sensores

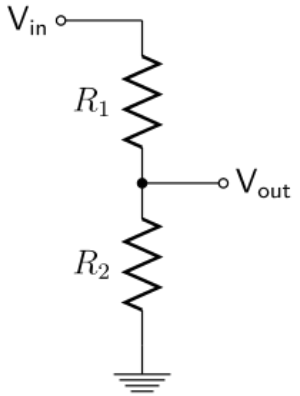
La forma más sencilla de medir sensores de resistencia variable son mediante los divisores de voltaje:



Divisor de voltaje

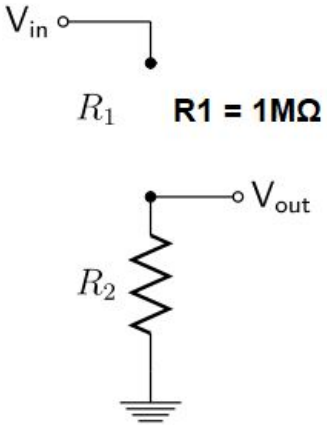
Un divisor de voltaje funciona tal que :

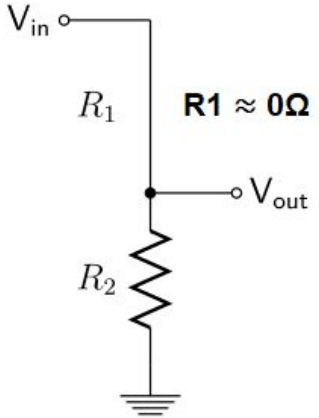
$$Vout = Vin * \frac{R2}{R1 + R2}$$



Para un Vin = 5V:

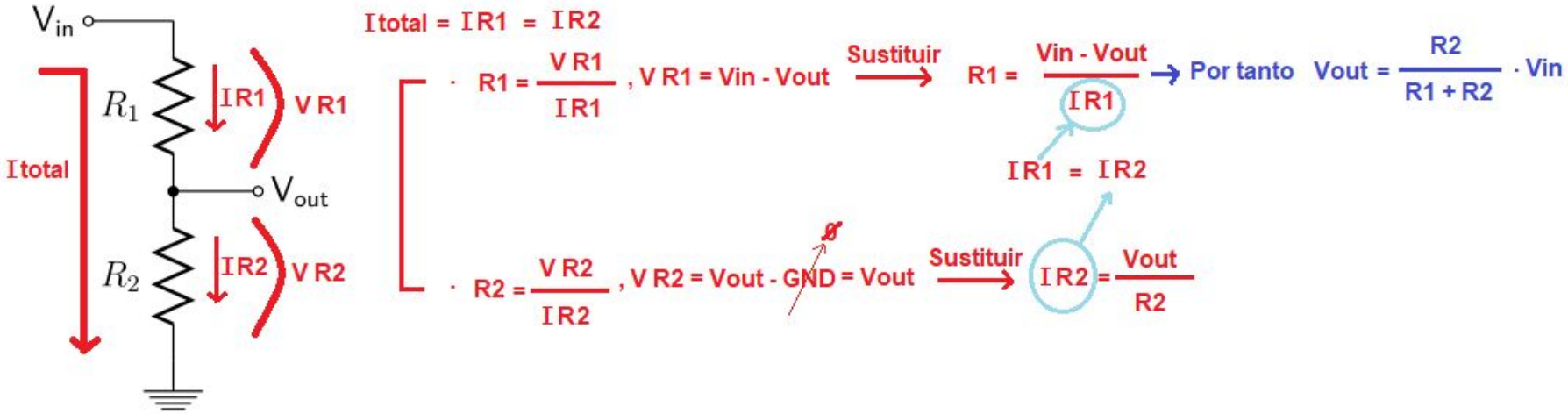
- Si R1 = R2 = 1 kΩ, Vout será la mitad, **2.5 V**.
- Si R1 = **1 MΩ**, R2 = 1 kΩ, Vout será **≈ 0 V.**
- Si R1 = **0 Ω**, R2 = 1 kΩ, Vout será **5 V.**





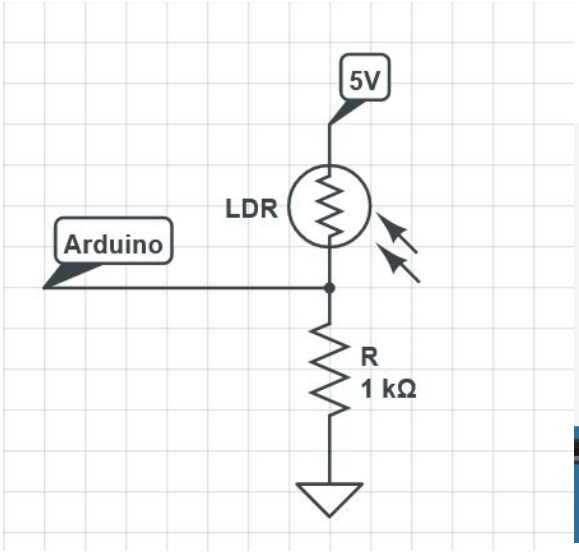
Fórmula divisor de voltaje

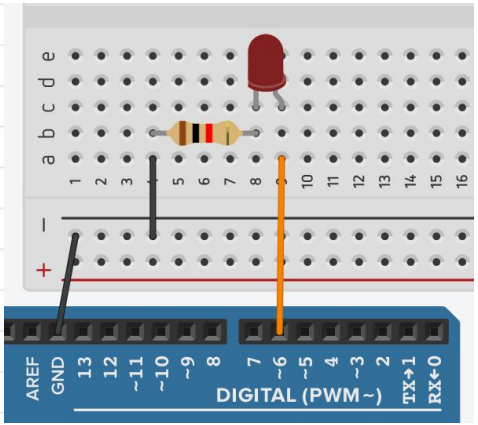
La fórmula del divisor se halla de la siguiente forma:

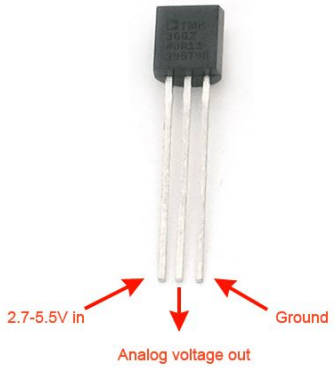


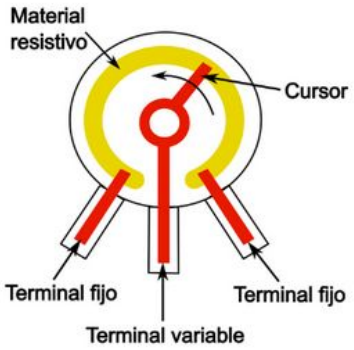
Actividades

1. Aumentar luminosidad LED
2. Controlar LED con potenciómetro
3. Leer temperatura y mostrarlo por el Serial monitor
4. Leer luminiscencia y mostrarlo por el Serial monitor
5. Modificar frecuencia sonora de zumbador con potenciómetro









Sesión 3

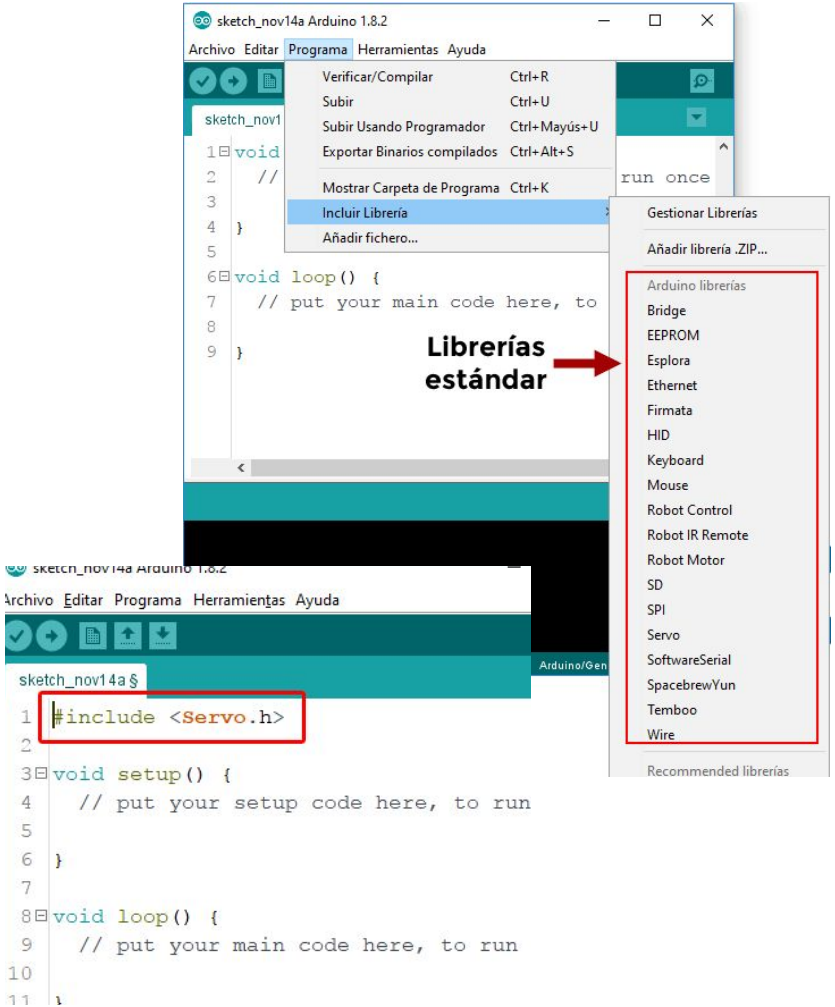
Librerías

Conjuntos de código ya escrito. Organizado en funciones o variables.

- Estándar: Librerías ya incluidas en Arduino
- Terceros: Librerías creada por gente

Para su uso, se ha de incluir con #include <nombre_de_librería>

Facilita la reutilización de código ya creado

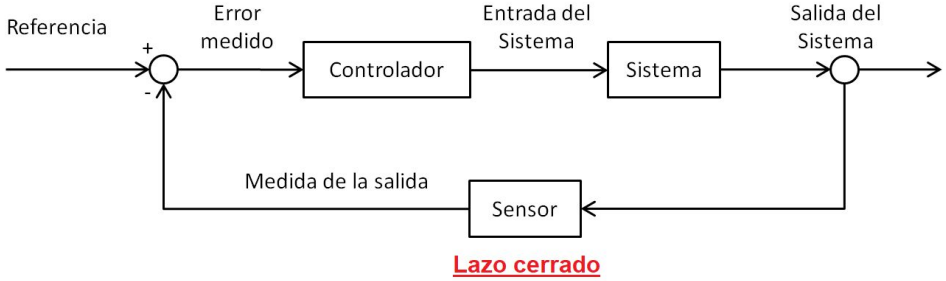


Sistema de control

Para hacer control sobre dispositivos, se pueden diferenciar dos tipos de control:

- Lazo abierto: Las operaciones son fijas
  P.ej: Ventilador

- Lazo cerrado: Las operaciones tienen en cuenta la diferencia entre la salida deseada, y la salida actual.
  P.ej: Aire acondicionado

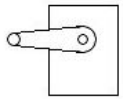


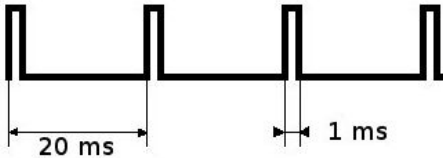
Servo

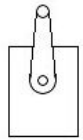
Su control se basa en modulación de ancho de pulso, pero:

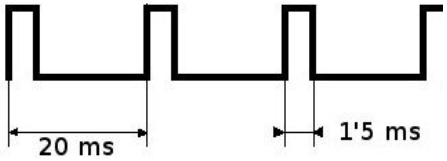
Con la ayuda de la librería Servo, se nos facilita el control de los mismos:

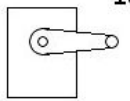
```
#include <Servo.h>

Servo miServo;

miServo.attach(pin);

miServo.write(90);

. . .
```

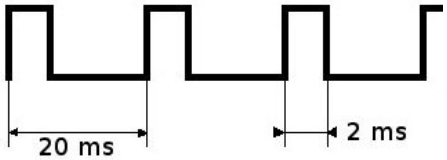






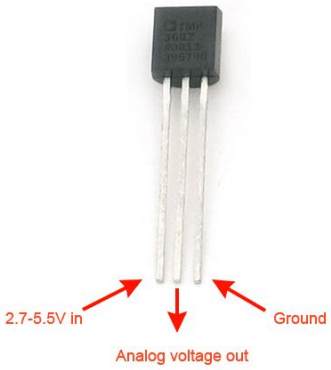


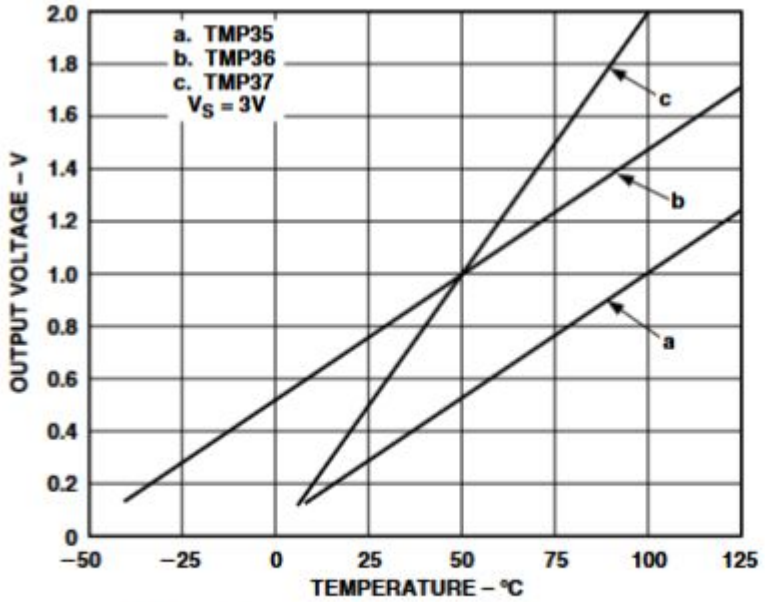




TMP36

$$TempC = \frac{5}{1024} * lectura * 100 - 50$$





TPC 1. Output Voltage vs. Temperature

HC-SR04

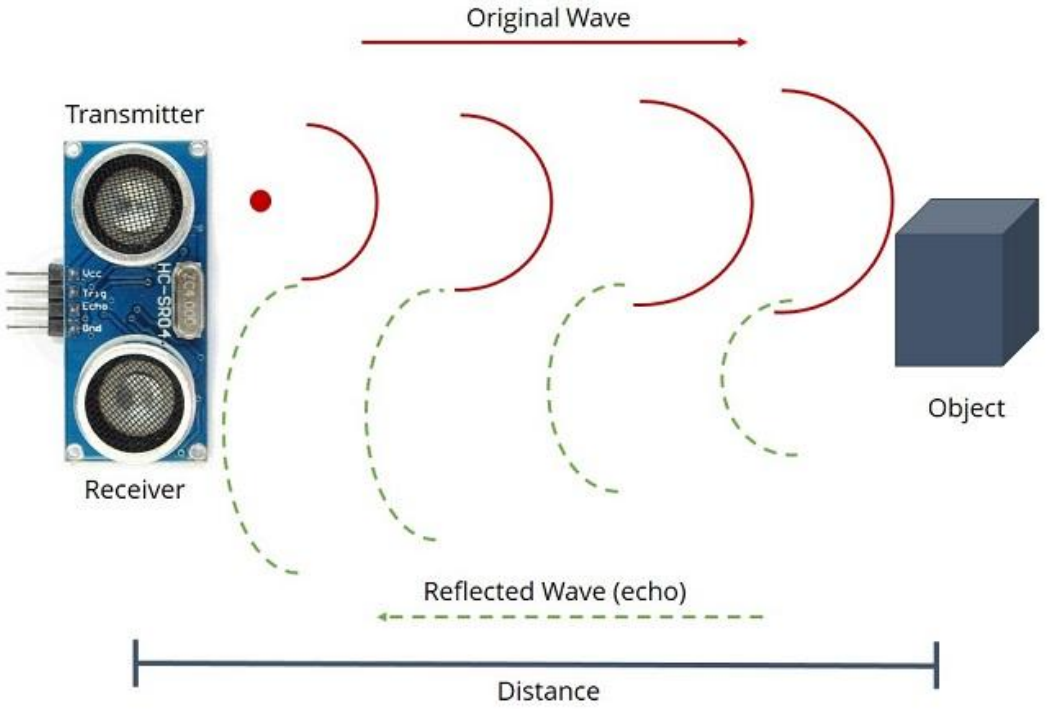
Sensor ultrasónico

Se calcula la distancia en función de lo que tarda la señal emitida en volver rebotada.

Trig -> Pulso emitido durante 10us

Echo -> Pulso rebotado. Cuanta más duración tenga, más distancia ha recorrido.

Vel. sonido = 340 m/s



Actividades

1. Observar temperatura ambiente
2. Mover un servo
3. Con el ultrasónico HC-SR04 observar valores en serial monitor
4. Hacer sensor de proximidad con HC-SR04 y zumbador

$$TempC = \frac{5}{1024} * lectura * 100 - 50$$