

Taller de Arduino

ING1004 - Desafíos de la Ingeniería



¿Qué veremos esta semana?

- Qué es Arduino
- Señales digitales y análogas
- Cómo se programa el microcontrolador
- Comunicación serial con la placa
- Interacción con sensores y actuadores
- Ley de ohm
- Uso de librerías
- Dónde encontrar más información

Repaso del lunes

Cómo programar un Arduino

- Todo lo que está dentro de la función **setup()** corre **una vez**

Se utiliza para asignar pines, entre otros

- Todo lo que está dentro de la función **loop()** corre **infinitas veces** (de forma **muy** rápida)

Se utiliza para lo que la placa va a hacer, si nada lo detiene este código correrá por siempre mientras tenga energía.

- Se pueden declarar variables antes de la función **setup()**, estas serán **variables globales**

¿Cómo programar un Arduino?

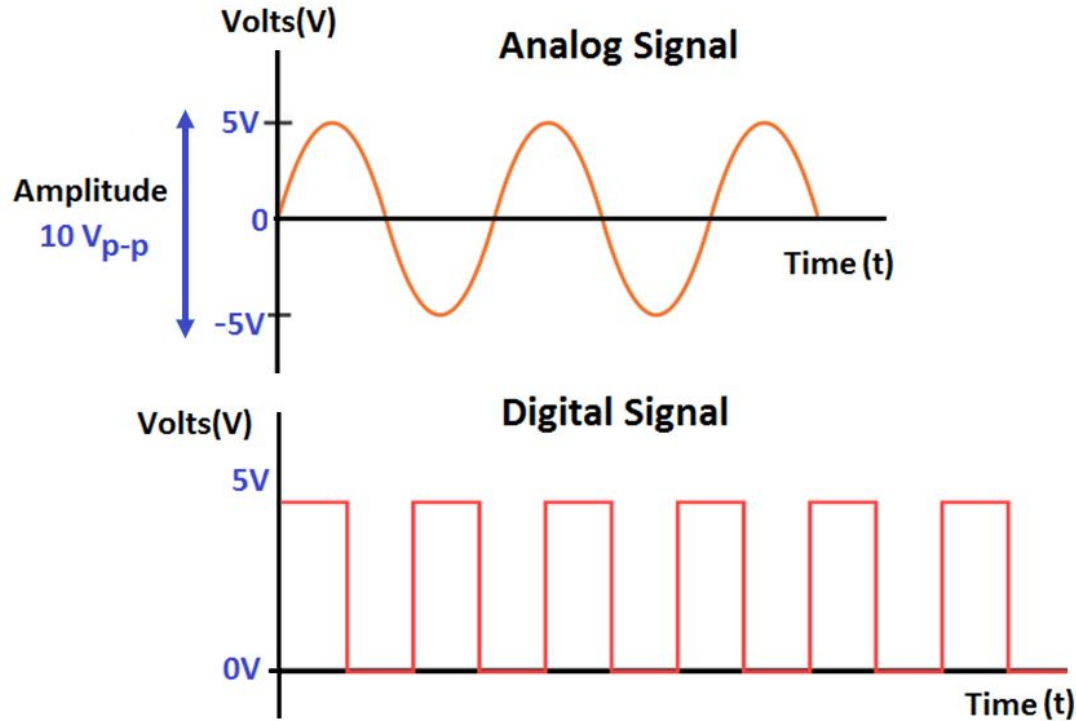
- El botón verificar **compila** el código y revisa que funcione pero no lo sube a la placa

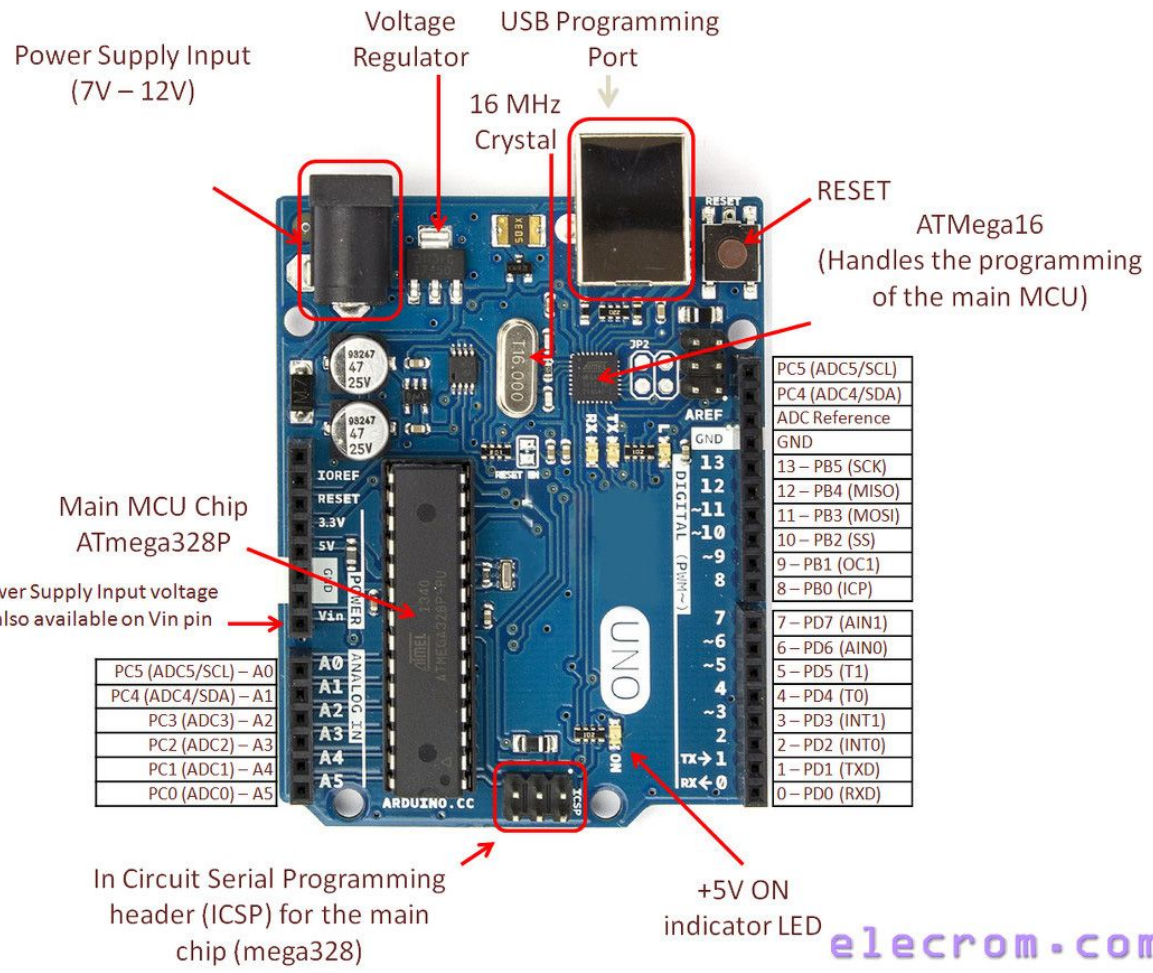


- El botón subir, **verifica** y **sube** el código a la placa



Señales digitales vs analógicas





PC5 (ADC5/SCL) – A0
PC4 (ADC4/SDA) – A1
PC3 (ADC3) – A2
PC2 (ADC2) – A3
PC1 (ADC1) – A4
PC0 (ADC0) – A5

PC5 (ADC5/SCL)	
PC4 (ADC4/SDA)	
ADC Reference	
GND	
13 – PB5 (SCK)	
12 – PB4 (MISO)	
11 – PB3 (MOSI)	
10 – PB2 (SS)	
9 – PB1 (OC1)	
8 – PB0 (ICP)	
7 – PD7 (AIN1)	
6 – PD6 (AIN0)	
5 – PD5 (T1)	
4 – PD4 (T0)	
3 – PD3 (INT1)	
2 – PD2 (INT0)	
1 – PD1 (TXD)	
0 – PD0 (RXD)	

Primer ejercicio: Hello World!



```
void setup() {  
  Serial.begin(9600);  
}
```

```
void loop() {  
  Serial.println("Hello world!");  
  delay(500);  
}
```



COM7

[illegible]

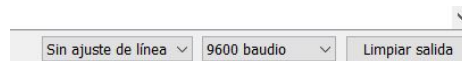
Segundo ejercicio, lectura COM serial:



```
/*
code adapted from https://www.arduino.cc/en/serial/read
and https://hetpro-store.com/TUTORIALES/arduino-serial-read/
*/
char incomingChar; // for incoming serial data
void setup() {
  Serial.begin(9600);
}

void loop() {
  // send data only when you receive data:
  if (Serial.available() > 0) {
    // read the incoming char:
    incomingChar = Serial.read();

    // say what you got:
    Serial.print("I received: ");
    Serial.println(incomingChar);
  }
}
```



- Poner monitor serial en modo sin ajuste de línea
- ¿Qué pasa si enviamos más de un caracter?



I received: a
I received: t

Tercer ejercicio: Blink LED



Blink

```
// the setup function runs once when you press reset or power the board
void setup() {
    // initialize digital pin LED_BUILTIN as an output.
    pinMode(LED_BUILTIN, OUTPUT);
}

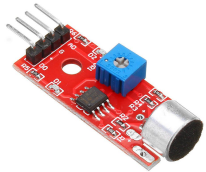
// the loop function runs over and over again forever
void loop() {
    digitalWrite(LED_BUILTIN, HIGH);    // turn the LED on (HIGH is the voltage level)
    delay(1000);                        // wait for a second
    digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
    delay(1000);                        // wait for a second
}
```

Alternativamente: cambiar `LED_BUILTIN` por 13

Módulos

Sensores

- Permiten leer datos de nuestro entorno de forma **digital o análoga**
- Considerar:
 - Precisión
 - Exactitud
 - Consumo de energía
 - Tamaño
 - Otros



Actuadores

- Transforman energía en trabajo
- Más comunes: Luces y motores
- Actúan bajo la orden del microcontrolador (generalmente desencadenada por la lectura de un sensor)



Ultrasonic module



Human body sensor module



Tilt sensor



Photosensitive sensor



Smoke sensor



Infrared barrier sensor



Vibration sensor



Sound sensor



1 path search sensor



Flame sensor



Laser Head Sensor



Clock module



Super regenerative module



Temperature and humidity sensor



Raindrop sensor



Soil Sensors

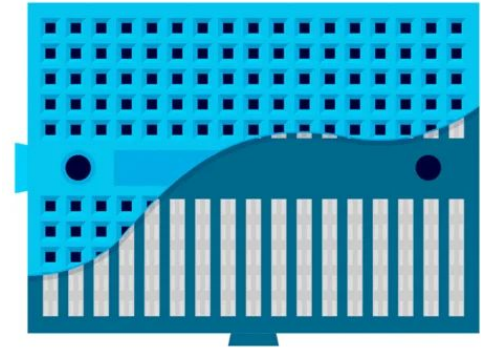
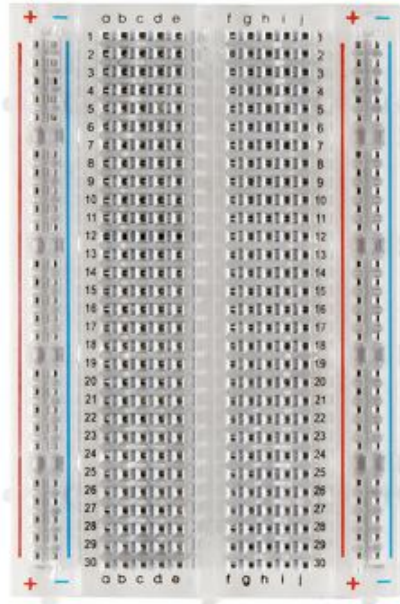


Hoy: Sensores y actuadores

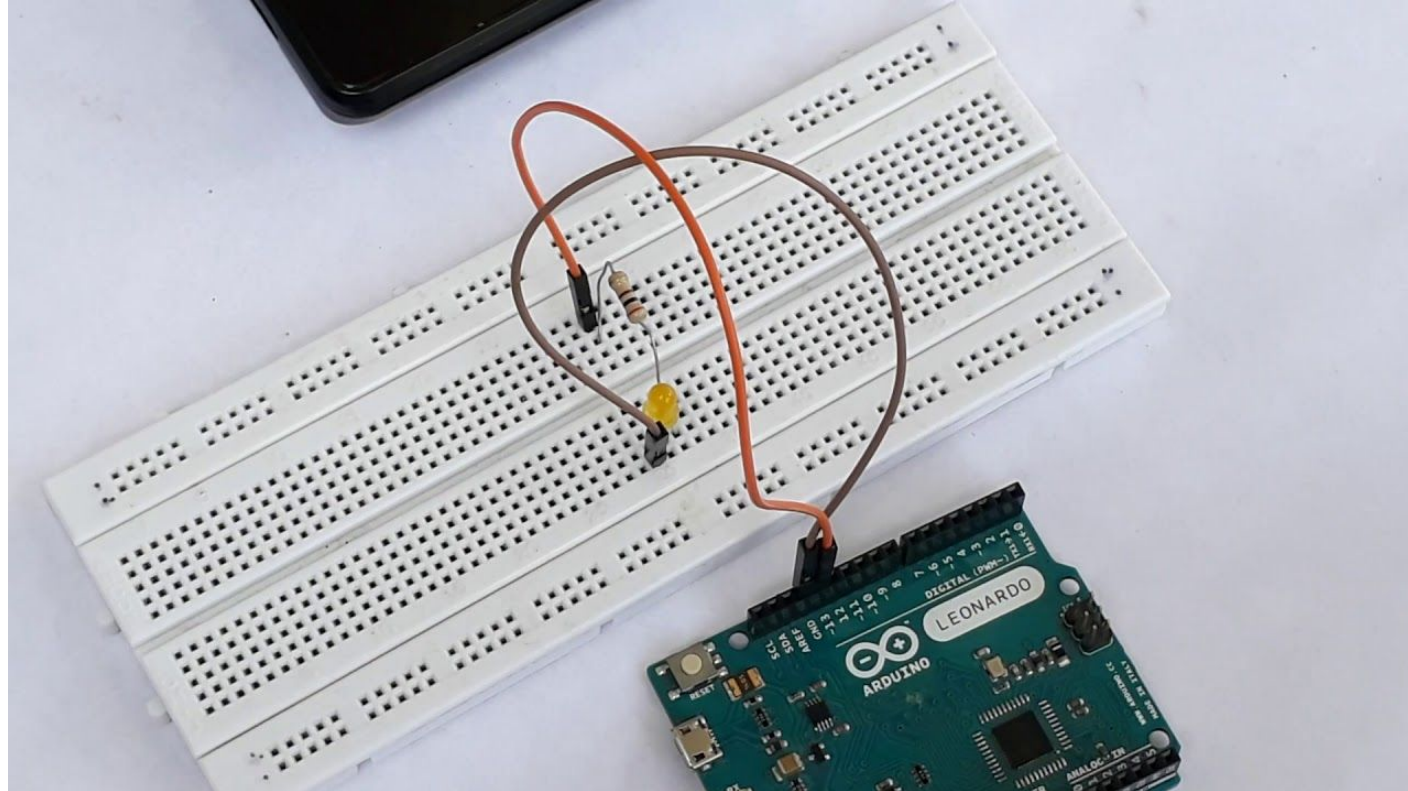
- **Protoboards**
- **Diodos LED**
- **Resistencias**
- **Ley de ohm**
- **Lectura y escritura digital**
- **Potenciómetros**
- **Lectura y escritura analógica**
- **Sensor ultrasónico**



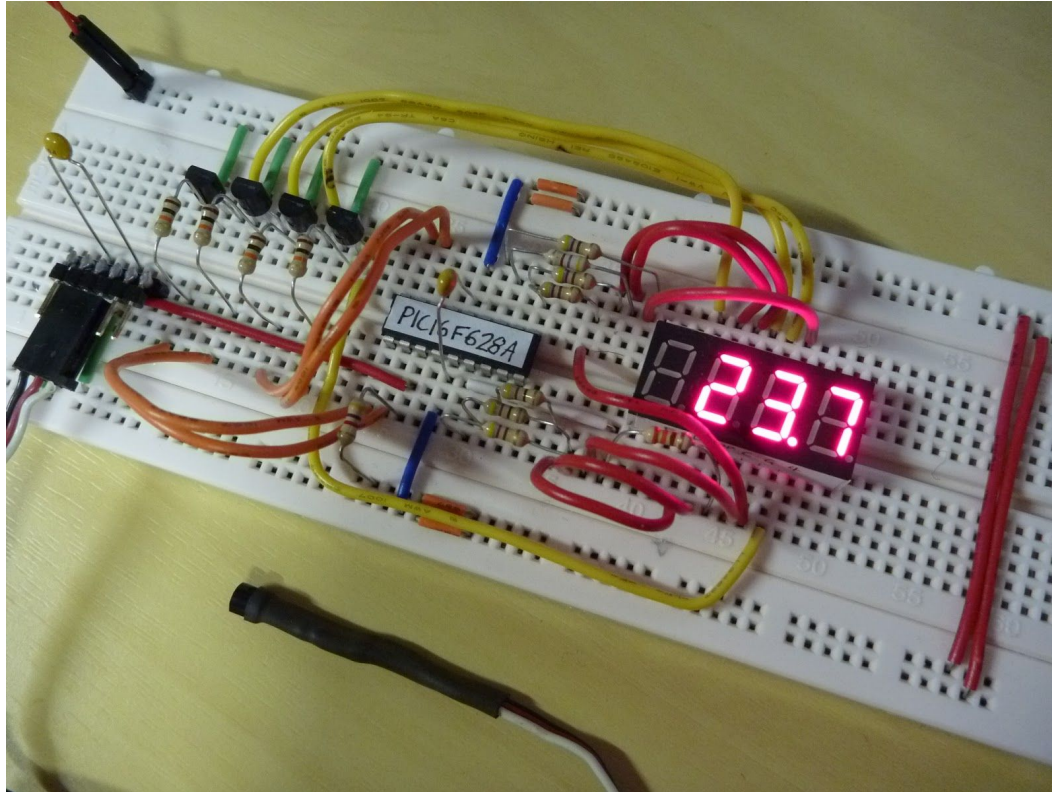
Protoboards



Protoboards



Protoboards

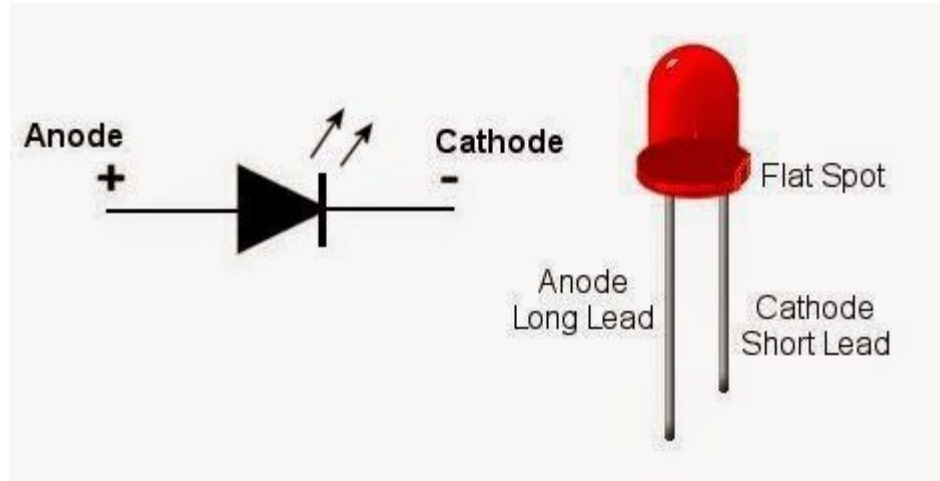


Diodos LED

Generalmente de

3,7V

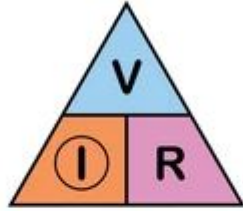
20mA



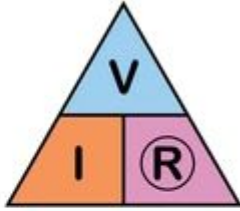
Ley de ohm



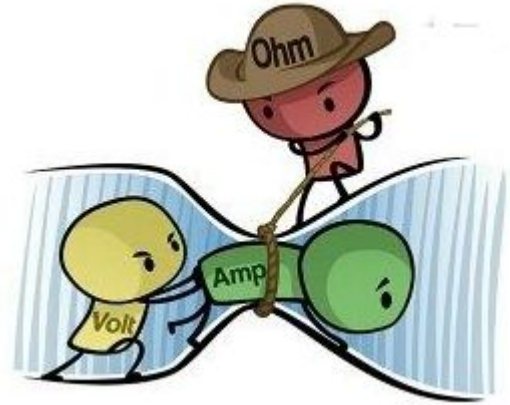
$$V = I \times R$$



$$I = \frac{V}{R}$$

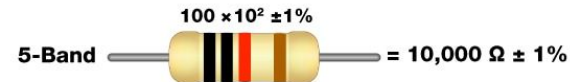
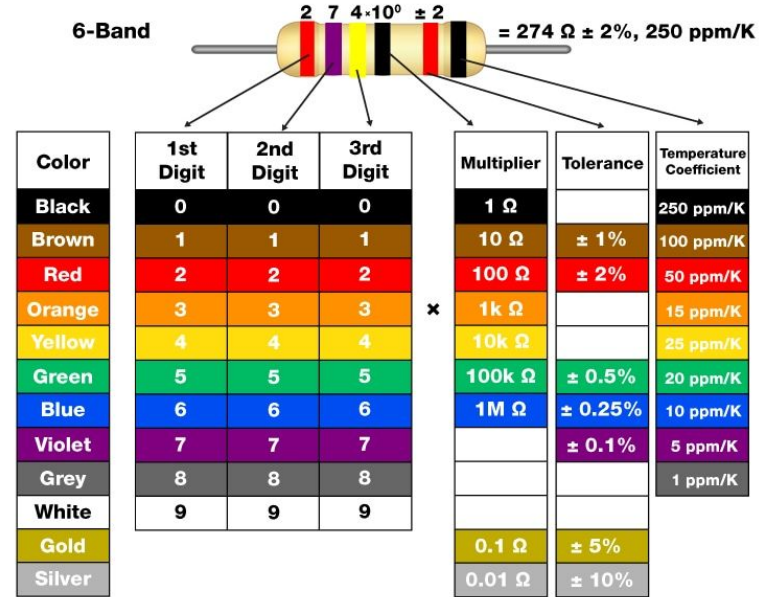


$$R = \frac{V}{I}$$



Código de colores de resistencias

How to Read Resistor Color Codes



Calculando resistencia para el LED

$$R = (V \text{ arduino} - V \text{ led}) / I \text{ led}$$

$$R = 5v - 2,7v / 0.00018 A = 65 \text{ ohm}$$

Alimentación: 5V			
tipo de led	Vled	corriente	resistencia
 azul /  blanco alta luminosidad	3,7V	20 mA	(calculado: 65 ohm)  68 ohm
 rojo alta luminosidad	1,2V	20 mA	(calculado: 190 ohm)  180 ohm
 rojo tipo indicatore	1,2V	5 mA	(calculado: 760 ohm)  680 ohm
 verde /  amarillo tipo indicatore	1,6V	5 mA	(calculado: 680 ohm)  680 ohm

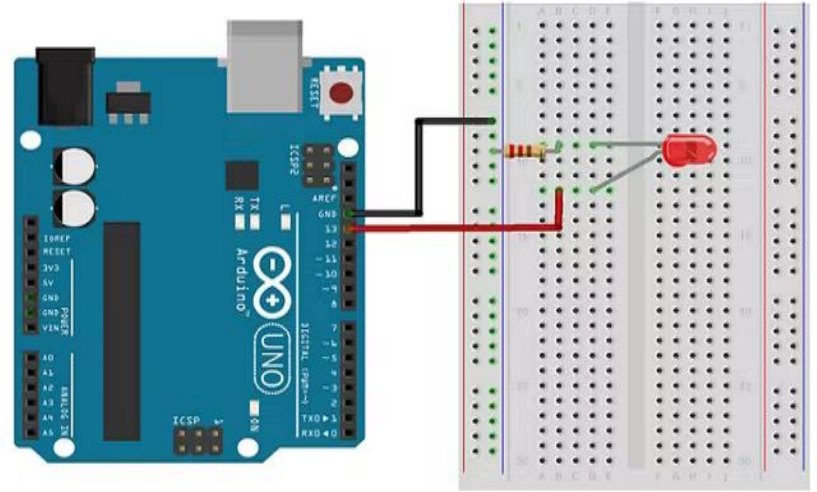
WWW.INVENTABLE.EU

Primer ejercicio: Prender un LED externo

```
led_externo

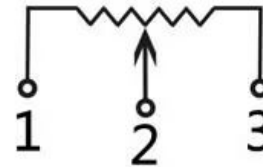
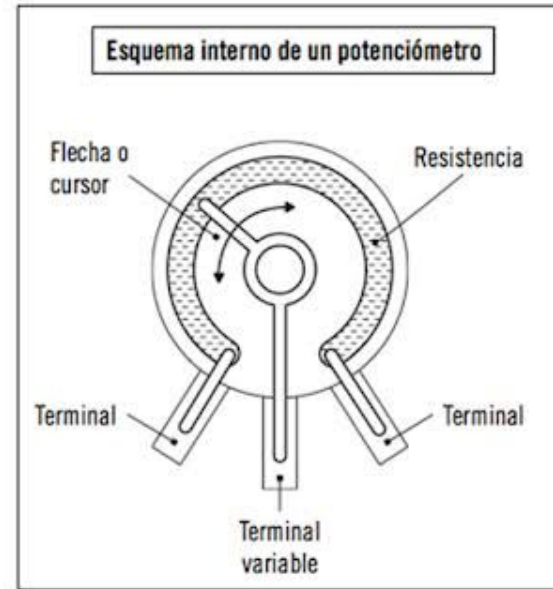
int ledPin = 13;
void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {
  digitalWrite(ledPin, HIGH);
}
```

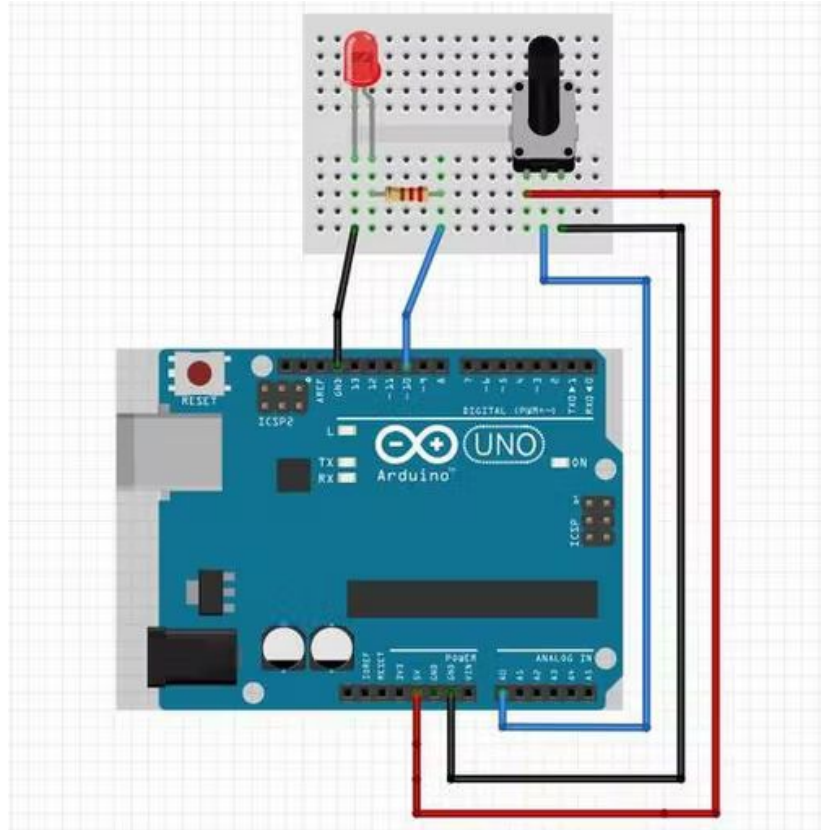


Potenciómetros

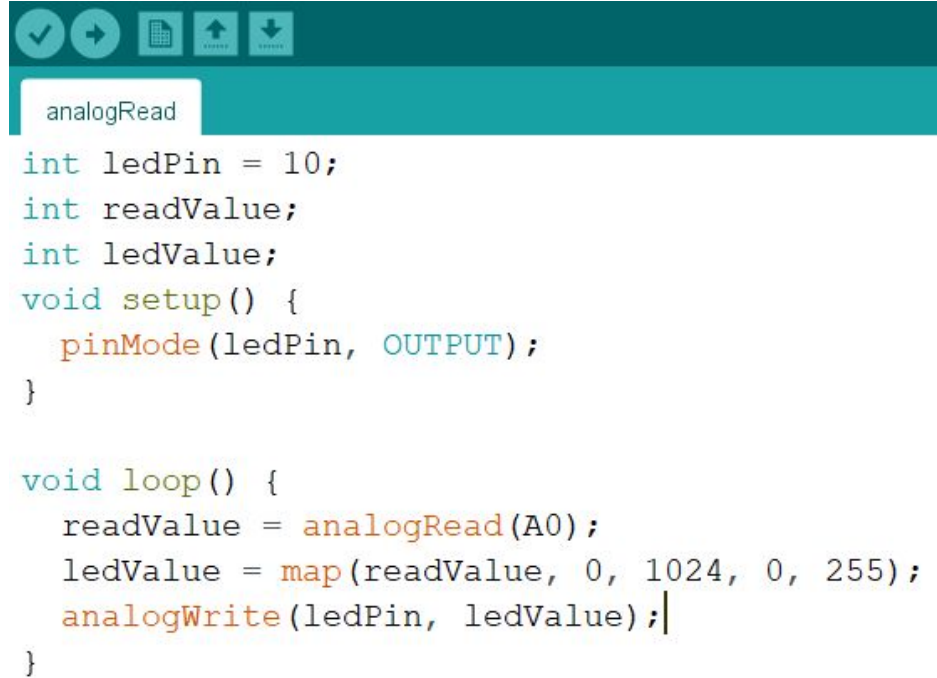
- Resistencia variable
- Lectura análoga



Segundo ejercicio: Intensidad de LED



Segundo ejercicio: Intensidad de LED



The image shows a screenshot of an IDE window with a teal header bar containing icons for a checkmark, a right arrow, a grid, and upload/download buttons. Below the header, a search bar contains the text "analogRead". The main area displays the following C++ code:

```
int ledPin = 10;
int readValue;
int ledValue;
void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {
  readValue = analogRead(A0);
  ledValue = map(readValue, 0, 1024, 0, 255);
  analogWrite(ledPin, ledValue);
}
```

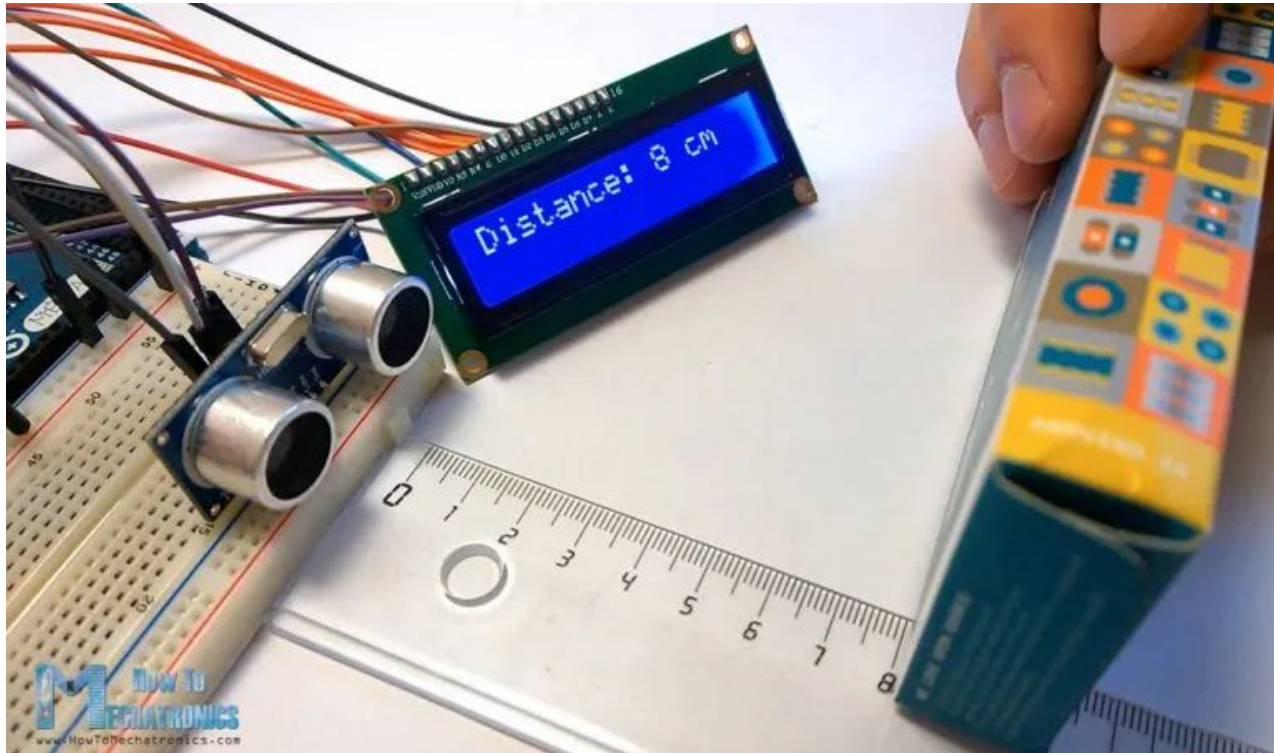
Desafío: Leer valor que se está enviando al LED

En el monitor serial

Desafío: Leer valor que se está enviando al LED

```
analogRead  
  
int ledPin = 10;  
int readValue;  
int ledValue;  
void setup() {  
    pinMode(ledPin, OUTPUT);  
    Serial.begin(9600);  
}  
  
void loop() {  
    readValue = analogRead(A0);  
    ledValue = map(readValue, 0, 1024, 0, 255);  
    analogWrite(ledPin, ledValue);  
    Serial.println(ledValue);  
}
```

Sensor de ultrasonido



Sensor de ultrasonido

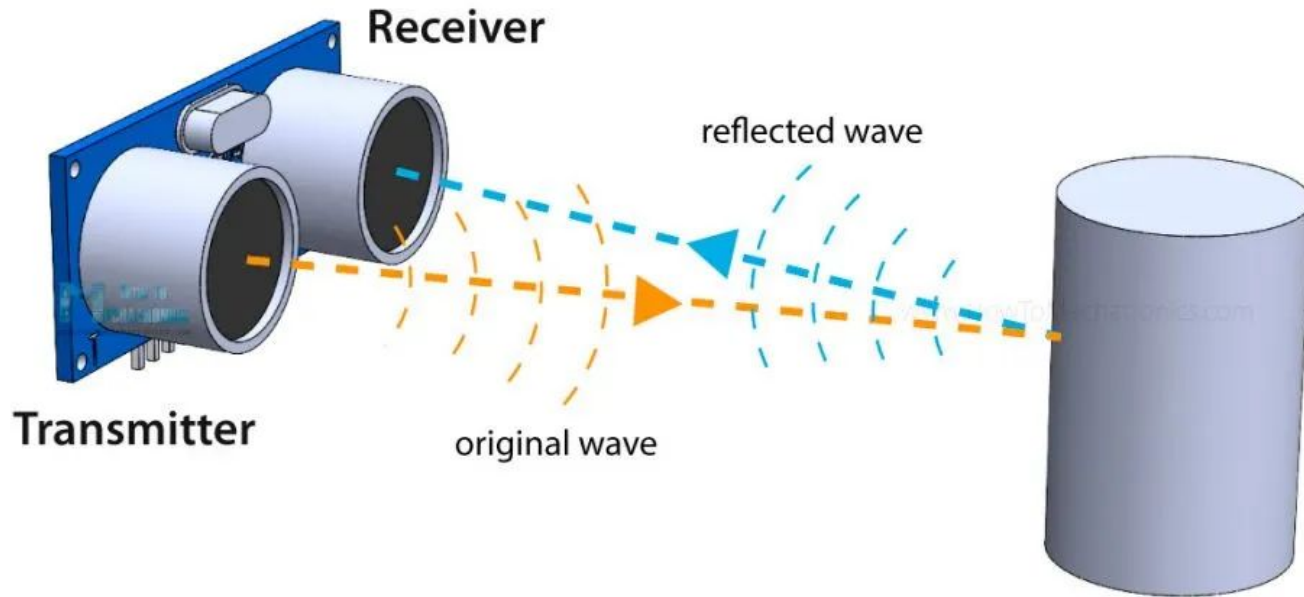
El sensor **HC-SR04** cuenta con dos parlantes ultrasónicos, uno para emitir una señal y otro para recibir la señal emitida reflejada, esta diferencia permite conocer la distancia de objetos.

Fun fact: Se parecen a un par de ojos robóticos como los de Wall-e.

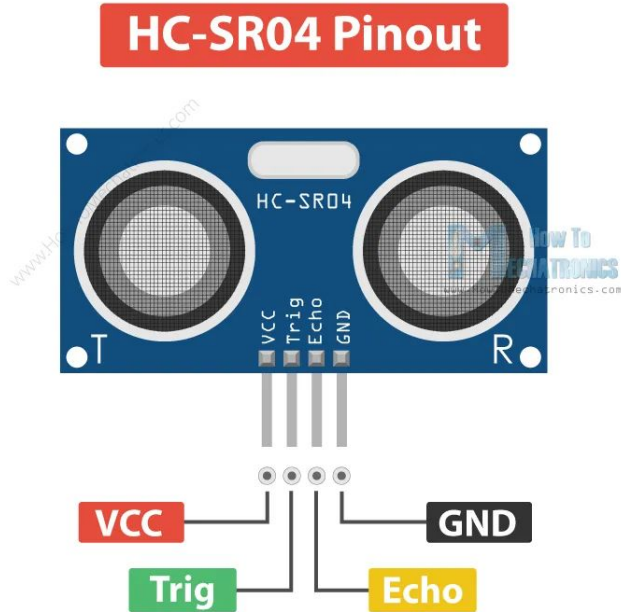
Algunas cosas interesantes que se pueden construir con él son: Radares, sistemas de detección de colisión, alarmas, etc.



Sensor de ultrasonido



Sensor de ultrasonido

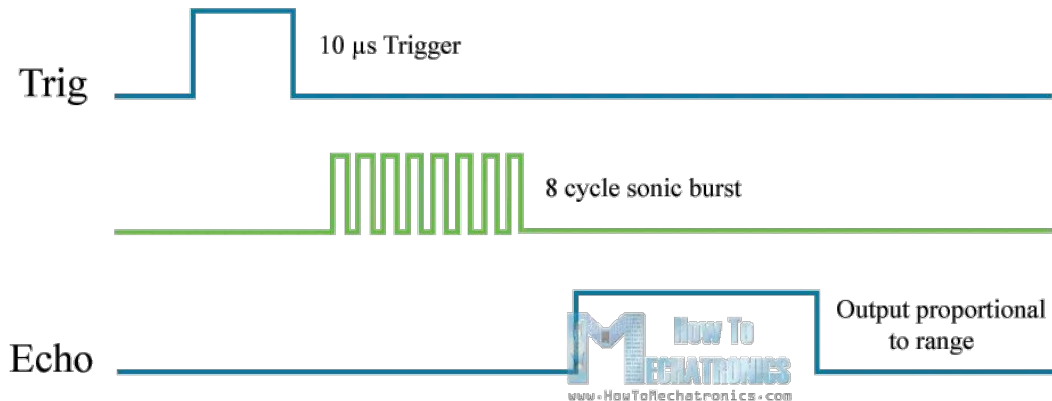


Operating Voltage	5V DC
Operating Current	15mA
Operating Frequency	40KHz
Min Range	2cm / 1 inch
Max Range	400cm / 13 feet
Accuracy	3mm
Measuring Angle	<15°
Dimension	45 x 20 x 15mm

Sensor de ultrasonido

Para generar ultrasonido se debe dar energía al pin del *trigger* por 10 microsegundos (establecer en **HIGH**). Esto enviará una ráfaga de 8 ciclos ultrasónicos que viajarán a la velocidad del sonido.

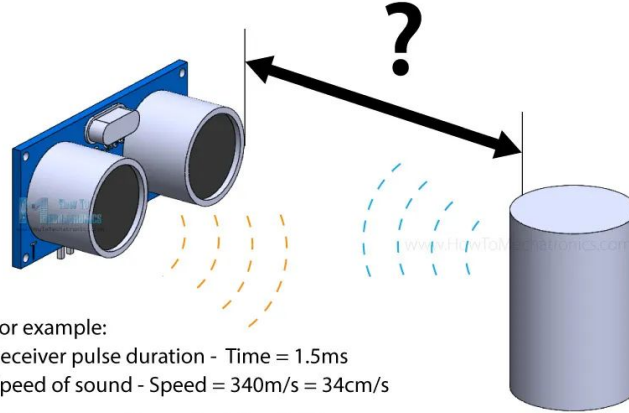
Luego de esto se prende el pin del *echo* (establecer en **HIGH**) para leer la onda reflejada, una vez reciba la onda, dejará de escuchar. Ese tiempo permite calcular una distancia.



Sensor de ultrasonido

Distancia = Velocidad * Tiempo

Sabemos la velocidad del sonido = 340m/s y la distancia que se demora, pero **ojo**: Este **tiempo** debe ser **dividido en 2** porque se considera tanto el trayecto de ida como el de vuelta



For example:

Receiver pulse duration - Time = 1.5ms

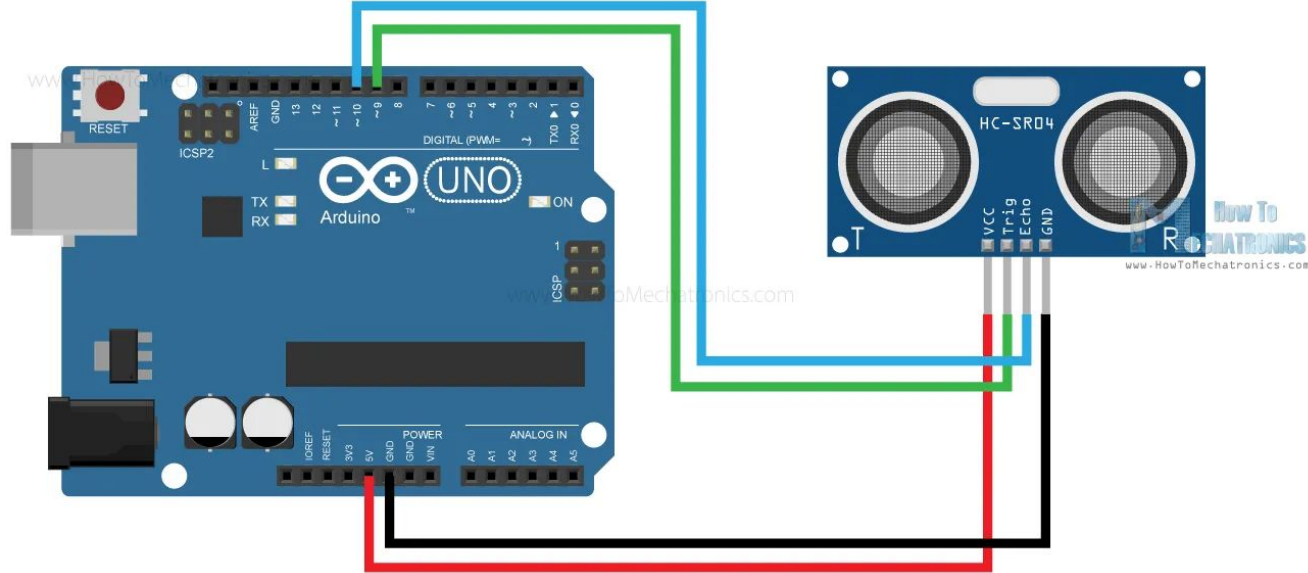
Speed of sound - Speed = 340m/s = 34cm/s

$Distance = (Speed \times Time) / 2$

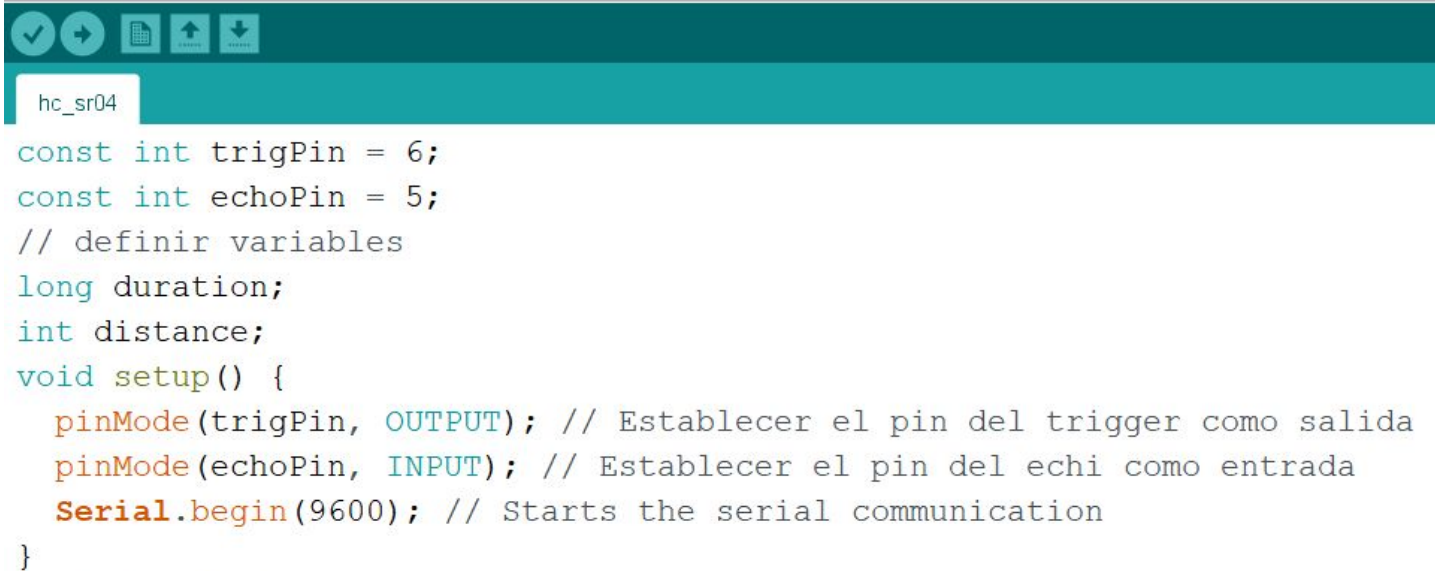
$Distance = (34cm/ms \times 1.5ms) / 2 = 25.5cm$

Tercer ejercicio: Lectura ultrasónica

HC-SR04 Ultrasonic Sensor and Arduino Wiring



Tercer ejercicio: Lectura ultrasónica



The image shows a screenshot of an IDE window with a teal header bar. The header bar contains four icons: a checkmark, a right arrow, a document icon, and two arrows (one up, one down). Below the header bar, the file name 'hc_sr04' is displayed in a small white box. The main area of the IDE contains the following C++ code:

```
const int trigPin = 6;
const int echoPin = 5;
// definir variables
long duration;
int distance;
void setup() {
  pinMode(trigPin, OUTPUT); // Establecer el pin del trigger como salida
  pinMode(echoPin, INPUT); // Establecer el pin del echi como entrada
  Serial.begin(9600); // Starts the serial communication
}
```

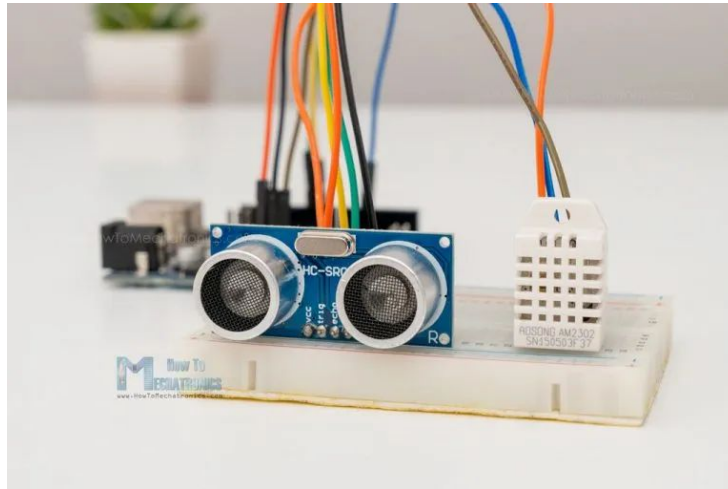
Tercer ejercicio: Lectura ultrasónica

```
void loop() {  
  // Limpiar el pin del trigger  
  digitalWrite(trigPin, LOW);  
  delayMicroseconds(2);  
  // Establecer el pin del trigger en HIGH por 10 microsegundos  
  digitalWrite(trigPin, HIGH);  
  delayMicroseconds(10);  
  digitalWrite(trigPin, LOW);  
  // Leer el pin del echo  
  // (Devuelve el viaje de la onda de sonido en microsegundos)  
  duration = pulseIn(echoPin, HIGH);  
  // Calcular la distancia  
  distance = duration * 0.034 / 2;  
  // Mostrar la distancia en el monitor serial  
  Serial.print("Distance: ");  
  Serial.print(distance);  
  Serial.println(" cm");  
}
```

Profundizando: Mejorando la lectura

La velocidad del sonido cambia considerablemente con la temperatura, por ejemplo a 20°C es 340m/s pero a -20°C es 315m/s. Para mejorar la lectura se puede implementar una compensación por temperatura con un sensor DHT22 (o similar). Usando la fórmula.

$$\text{Velocidad} = 331.4 + 0.6 * \text{Temperatura} + 0.0124 * \text{Humedad relativa}$$



Desafío: Alerta ultrasónica

Encender el LED integrado del Arduino (u otro) cuando se esté a menos de 15 cm del objeto y apagarlo cuando se esté más lejos.



Desafío: Alerta ultrasónica

```
const int trigPin = 6;
const int echoPin = 5;
// definir variables
long duration;
int distance;
void setup() {
    pinMode(trigPin, OUTPUT); // Establecer el pin del trigger como salida
    pinMode(echoPin, INPUT); // Establecer el pin del echi como entrada
    pinMode(13, OUTPUT);
    Serial.begin(9600); // Starts the serial communication
}
void loop() {
```

Desafío:

Alerta ultrasónica

```
void loop() {  
  // Limpiar el pin del trigger  
  digitalWrite(trigPin, LOW);  
  delayMicroseconds(2);  
  // Establecer el pin del trigger en HIGH por 10 microsegundos  
  digitalWrite(trigPin, HIGH);  
  delayMicroseconds(10);  
  digitalWrite(trigPin, LOW);  
  // Leer el pin del echo  
  // (Devuelve el viaje de la onda de sonido en microsegundos)  
  duration = pulseIn(echoPin, HIGH);  
  // Calcular la distancia  
  distance = duration * 0.034 / 2;  
  // Mostrar la distancia en el monitor serial  
  Serial.print("Distance: ");  
  Serial.print(distance);  
  Serial.println(" cm");  
  if (distance < 15){  
    digitalWrite(13, HIGH);  
  }  
  else{  
    digitalWrite(13, LOW);  
  }  
}
```
