

Taller de Arduino

ING1004 - Desafíos de la Ingeniería



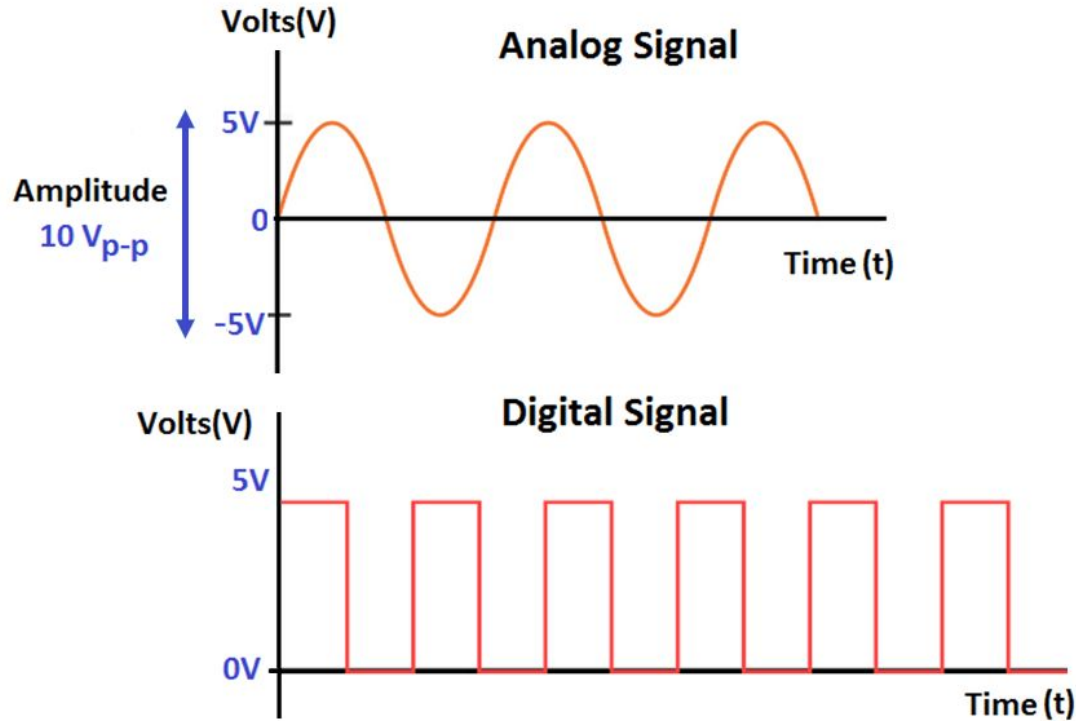
¿Qué veremos esta semana?

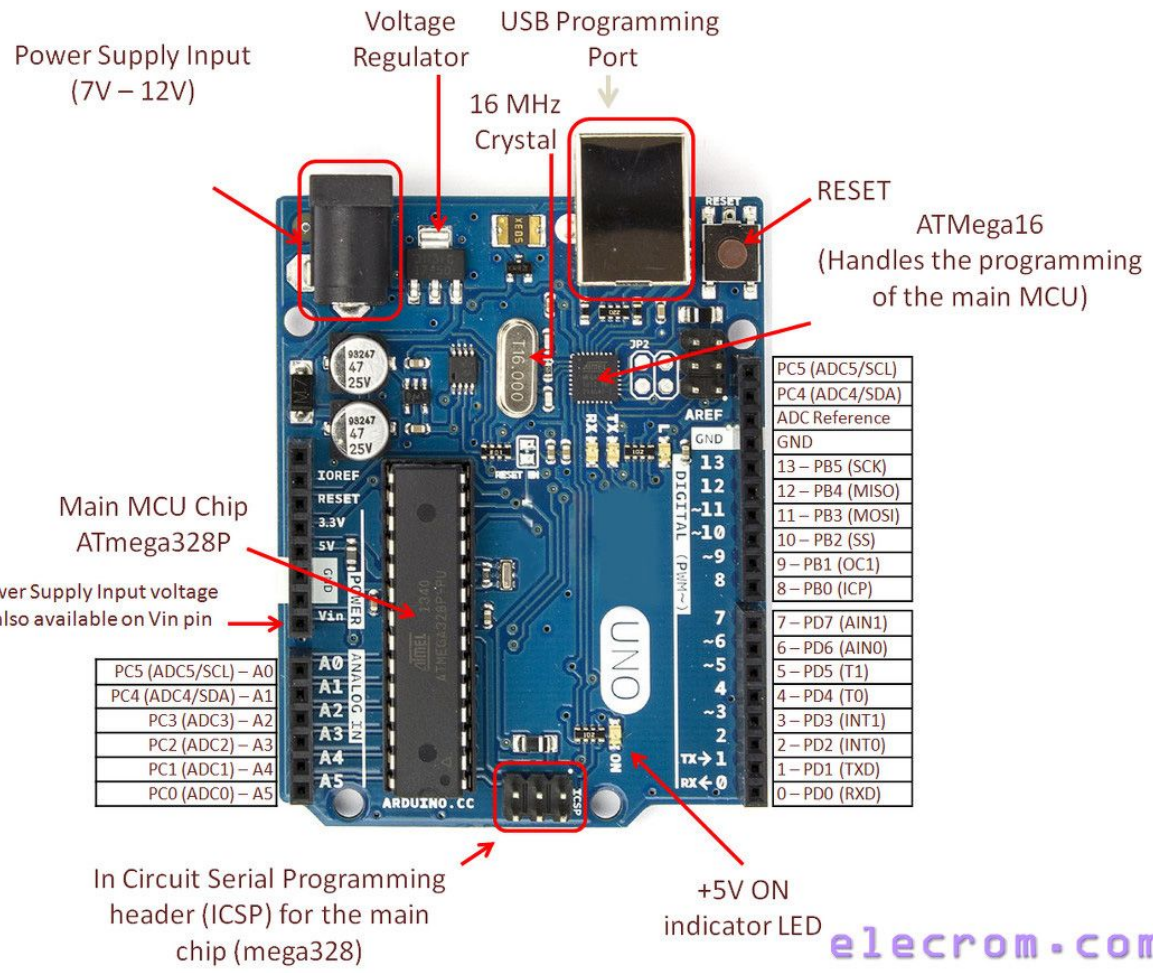
- Qué es Arduino
- Señales digitales y análogas
- Cómo se programa el microcontrolador
- Comunicación serial con la placa
- Interacción con sensores y actuadores
- Ley de ohm
- Uso de librerías
- Dónde encontrar más información

Repaso

- Señales digitales y análogas
- Anatomía del Arduino UNO
- Protoboards
- LEDs
- Potenciómetro
- Sensor ultrasónico

Señales digitales vs analógicas

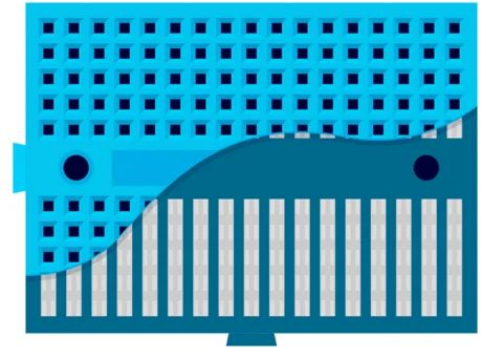
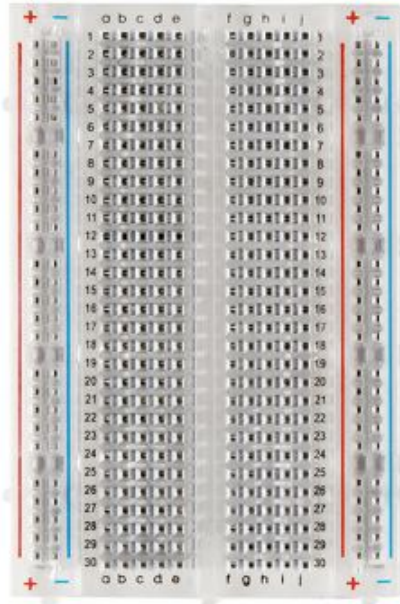




PC5 (ADC5/SCL) – A0
PC4 (ADC4/SDA) – A1
PC3 (ADC3) – A2
PC2 (ADC2) – A3
PC1 (ADC1) – A4
PC0 (ADC0) – A5

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)	

Protoboards

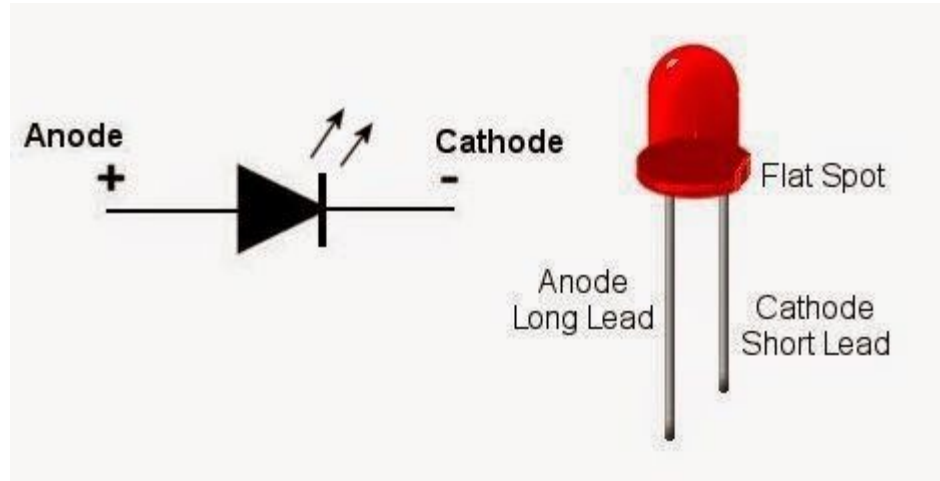


Diodos LED

Generalmente de

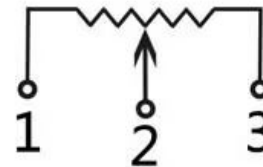
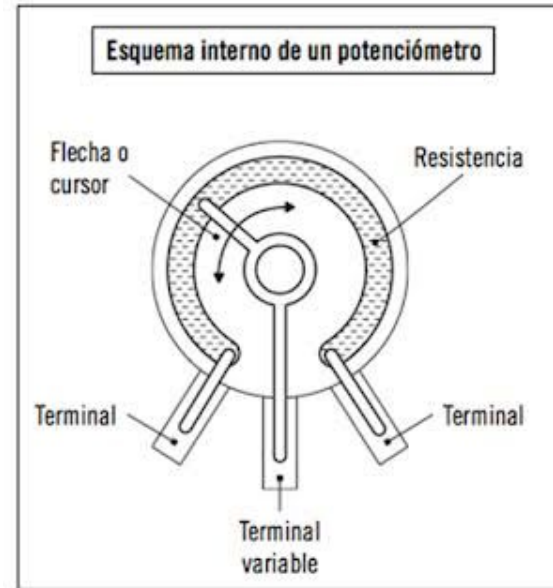
3,7V

20mA

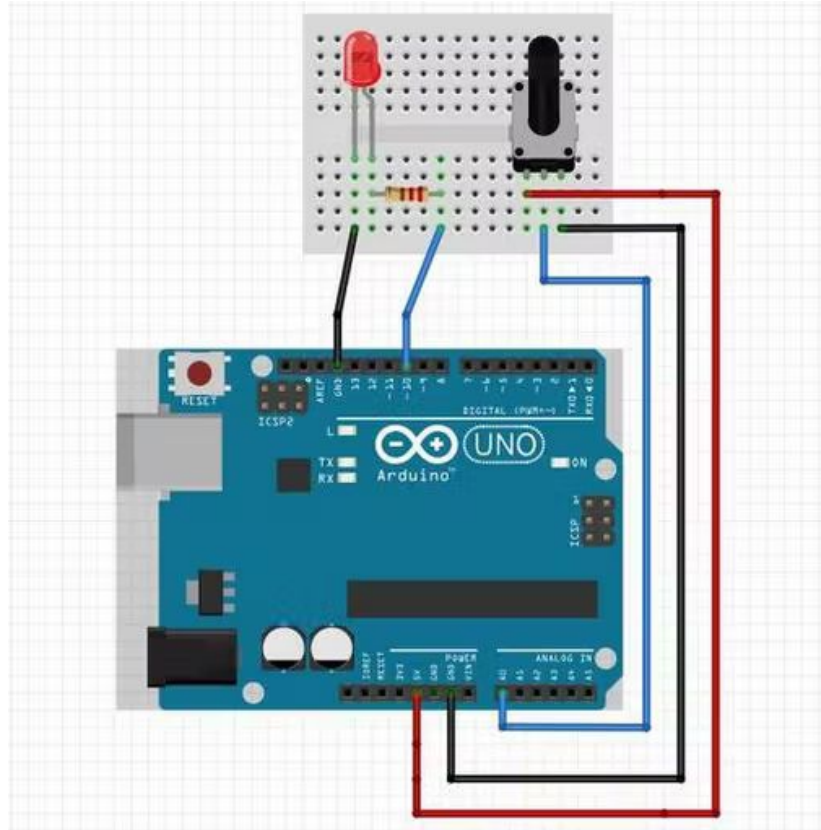


Potenciómetro


- Resistencia variable
- Lectura análoga



Ejercicio: Intensidad de LED



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);
}
```

Sensor de ultrasonido

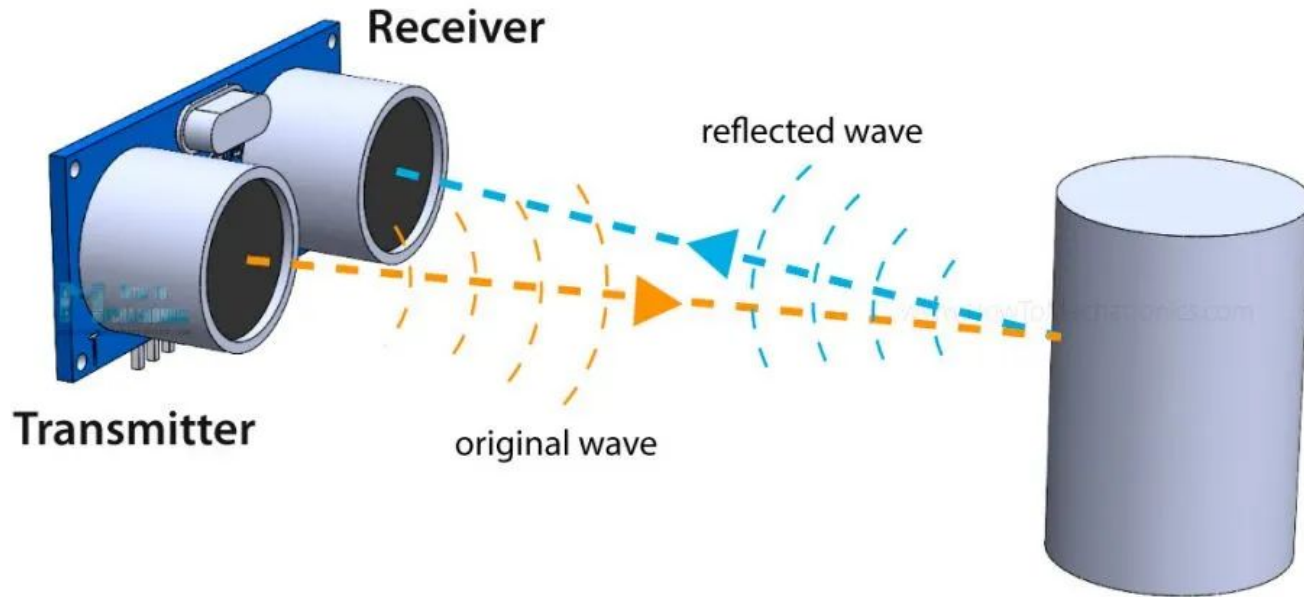
El sensor **HC-SR04** cuenta con dos parlantes ultrasónicos, uno para emitir una señal y otro para recibir la señal emitida reflejada, esta diferencia permite conocer la distancia de objetos.

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

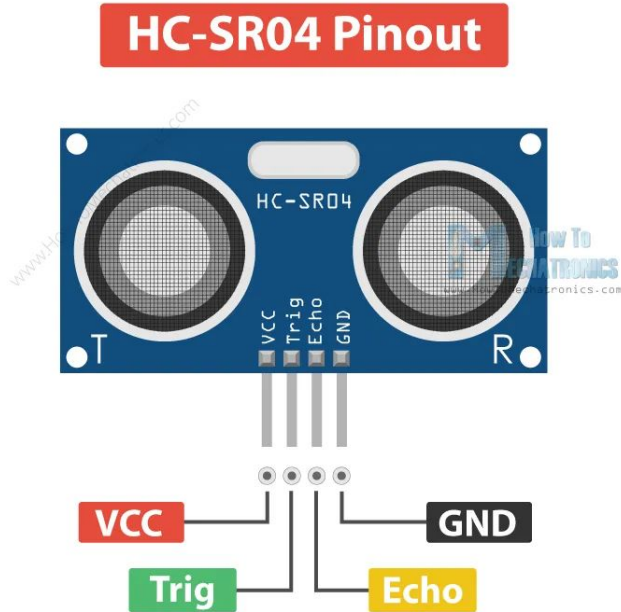
Algunas cosas interesantes que se pueden construir con él son: Radares, sistemas de detección de colisión, alarmas, etc.



Sensor de ultrasonido



Sensor de ultrasonido

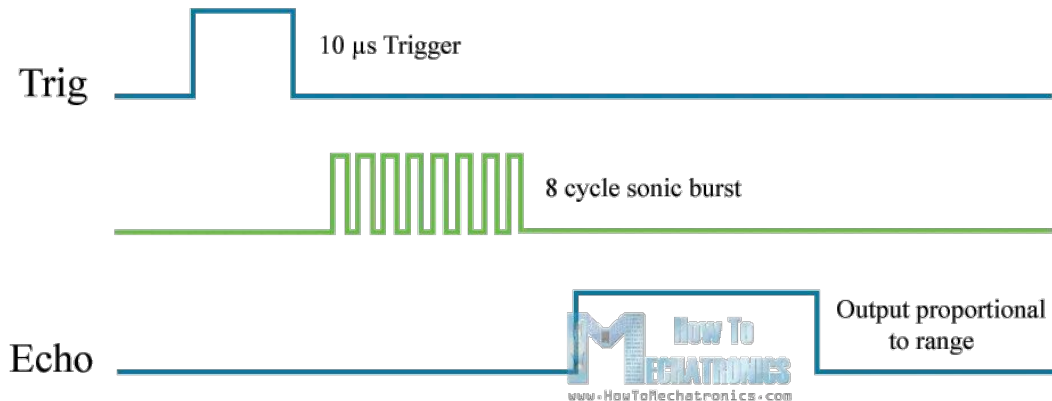


Operating Voltage	5V DC
Operating Current	15mA
Operating Frequency	40KHz
Min Range	2cm / 1 inch
Max Range	400cm / 13 feet
Accuracy	3mm
Measuring Angle	<15°
Dimension	45 x 20 x 15mm

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.

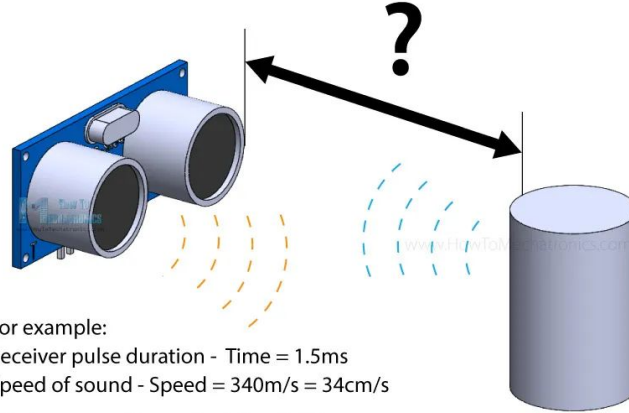
Luego de esto se prende el pin del *echo* (establecer en **HIGH**) para leer la onda reflejada, una vez reciba la onda, dejará de escuchar. Ese tiempo permite calcular una distancia.



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

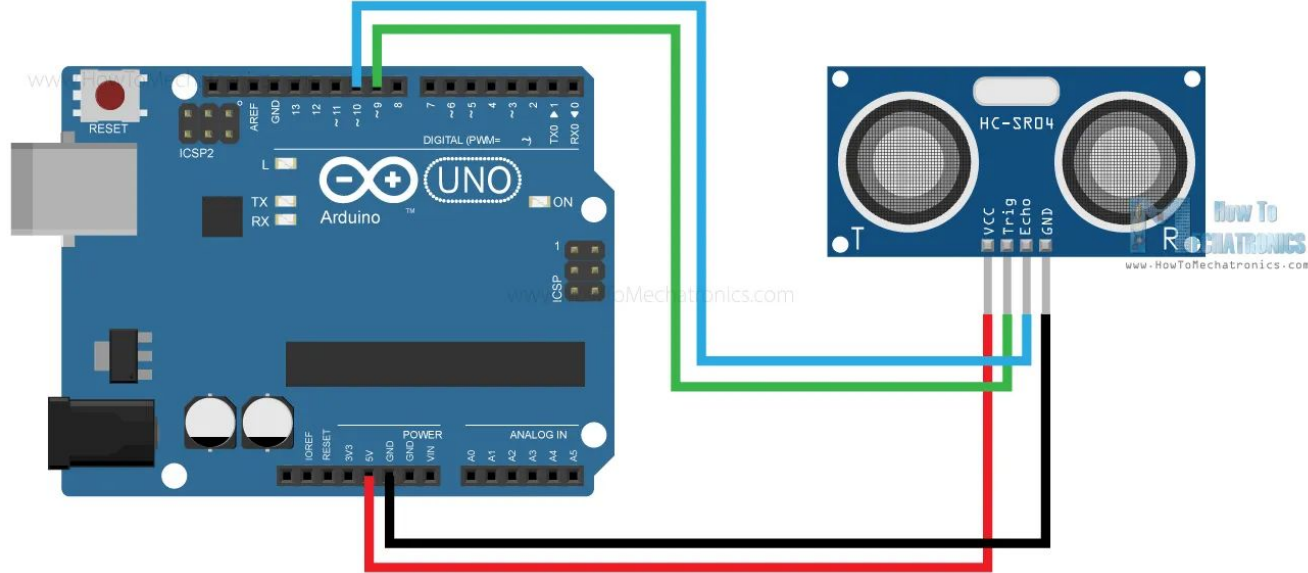
Speed of sound - Speed = 340m/s = 34cm/s

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

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

Ejercicio: Lectura ultrasónica

HC-SR04 Ultrasonic Sensor and Arduino Wiring



Ejercicio: Lectura ultrasónica



The image shows a screenshot of an IDE window with a teal header bar. The header bar contains five icons: a checkmark, a right arrow, a document, an upload arrow, and a download arrow. 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:

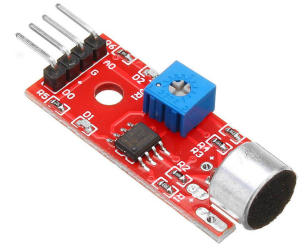
```
const int trigPin = 6;
const int echoPin = 5;
// definir variables
long duration;
int distance;
void setup() {
  pinMode(trigPin, OUTPUT); // Establecer el pin del trigger como salida
  pinMode(echoPin, INPUT); // Establecer el pin del echi como entrada
  Serial.begin(9600); // Starts the serial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

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");  
}
```

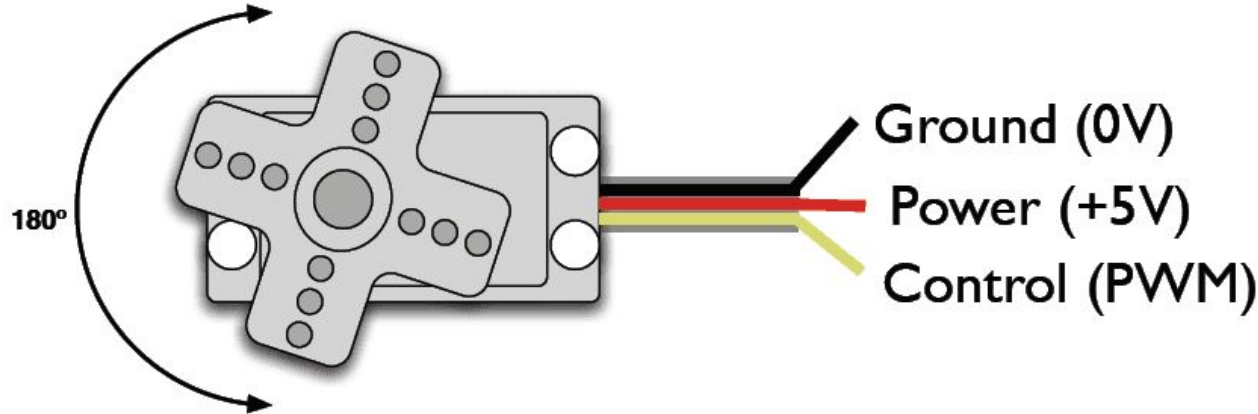
Hoy

- Servomotores
- Pulse Width Modulation (PWM)
- Librerías
- Sensor de sonido



Servomotor

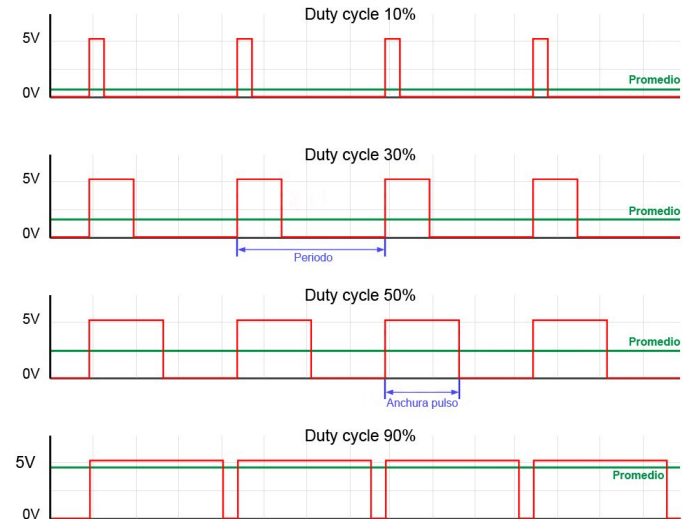
Un servomotor es un **motor con control**, es decir este dispositivo mecánico posee internamente **un controlador que posiciona** precisamente el rotor del motor en un **ángulo** especificado por la señal entrante de control.



Pulse Width Modulation (PWM)

Es la forma que tiene Arduino para entregar una salida de voltaje diferente a 5V (**simula un output análogo**), consiste en **cambiar el tiempo** de encendido y apagado de una señal digital lo suficientemente rápido para generar el efecto esperado

$$V_{medio} = (V_{cc+} - V_{cc-}) \cdot \frac{DutyCycle}{100}$$

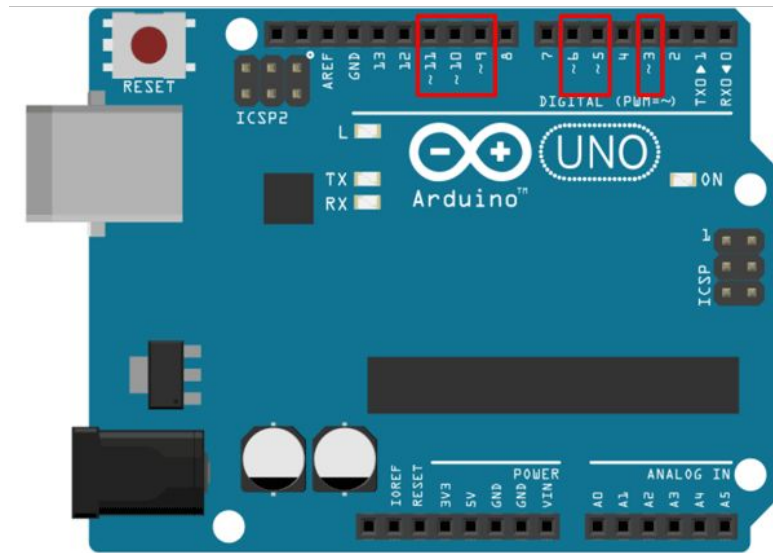


Pines PWM

Los pines digitales con ~ tienen salida PWM.

Basta con usar `analogWrite()` con un valor entre **0 y 255** para enviar el *duty cycle*.

- `analogWrite(0)` es una señal de ciclo de trabajo del 0%.
- `analogWrite(127)` es una señal de un ciclo de trabajo del 50%.
- `analogWrite(255)` es una señal de ciclo de trabajo del 100%.



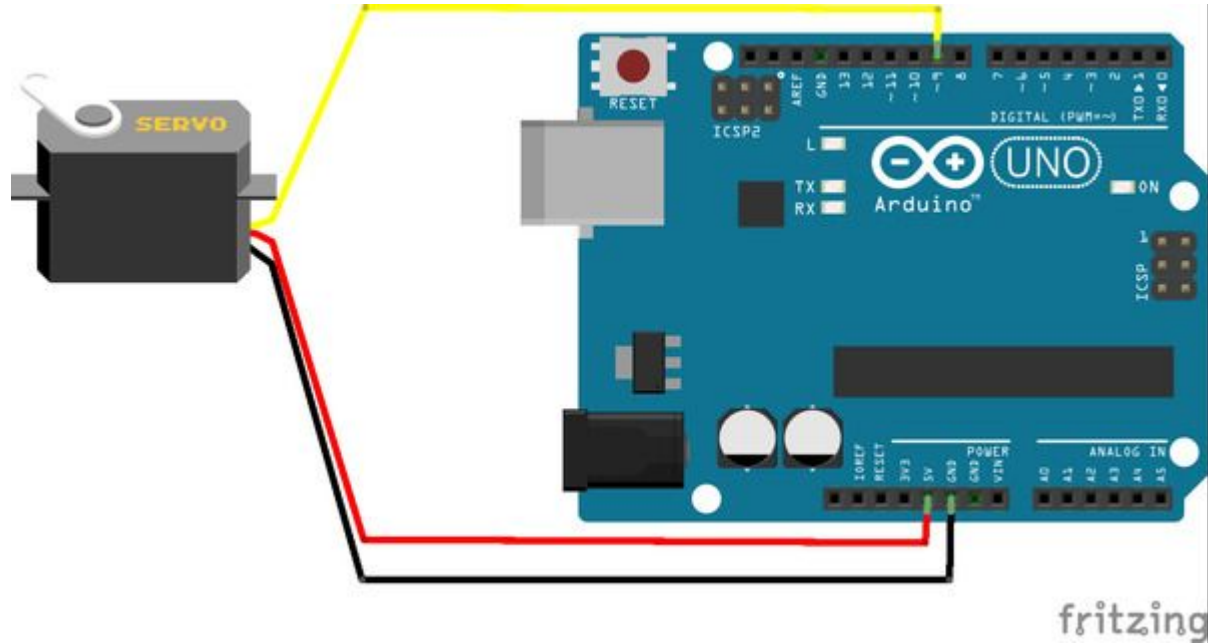
Librerías

Las librerías (o bibliotecas) son archivos escritos en C o C++ (.c, .cpp) que entregan a los programas una funcionalidad adicional (por ejemplo, la capacidad de controlar una matriz LED o leer un codificador, etc.).

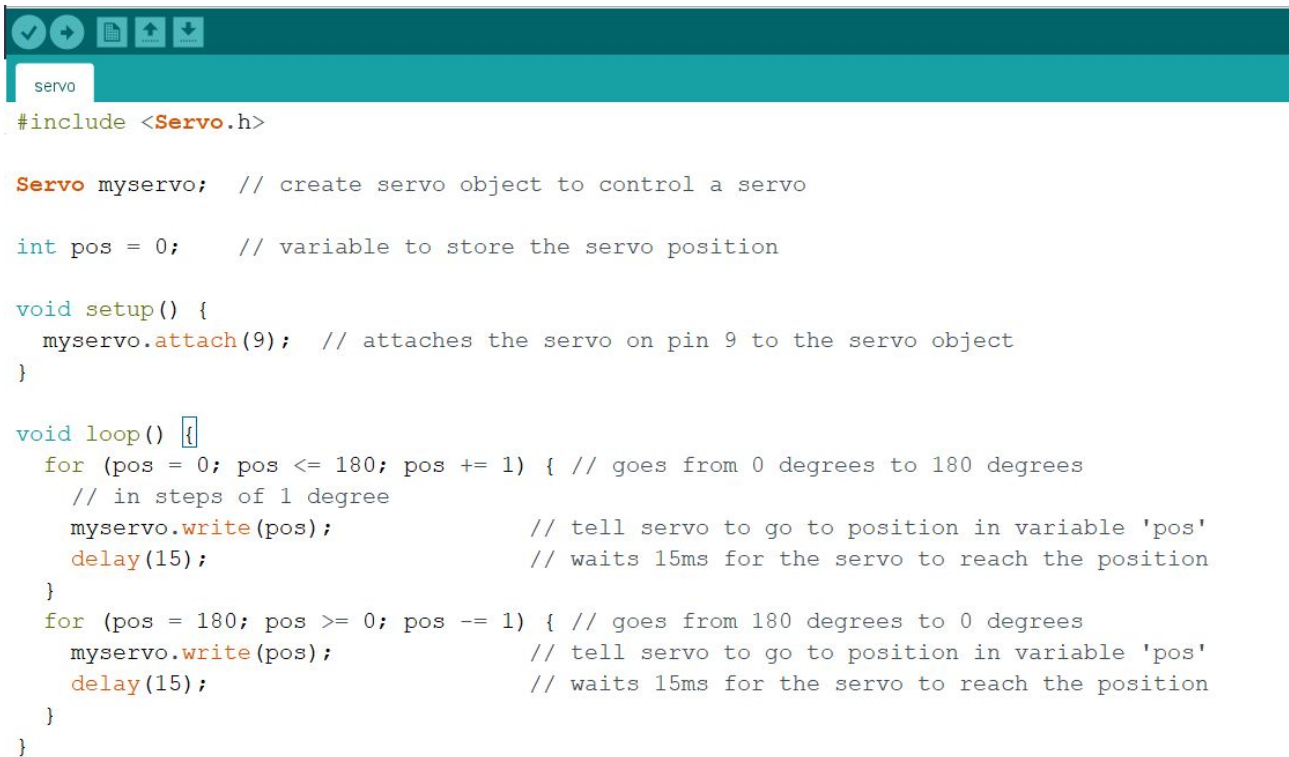
Para usar una librería, abre el menú *Programa*, selecciona *Incluir librería* y *elige* entre las librerías disponibles. Esto insertará una instrucción `#include` en la parte superior del programa para cada archivo de encabezado (.h) en la carpeta de la biblioteca.

Estas declaraciones hacen que las **funciones y constantes globales** definidas por la librería estén disponibles para el programa.

Ejercicio: Girando un servomotor



Ejercicio: Girando un servomotor



```
servo

#include <Servo.h>

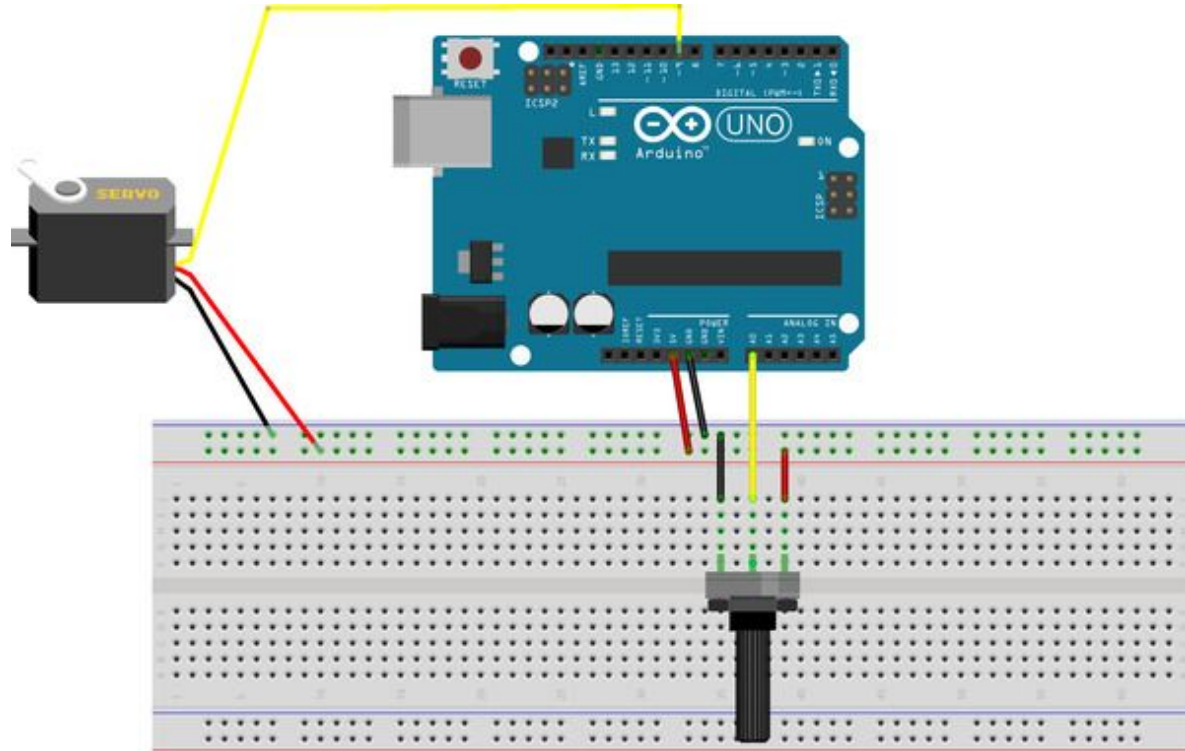
Servo myservo;  // create servo object to control a servo

int pos = 0;    // variable to store the servo position

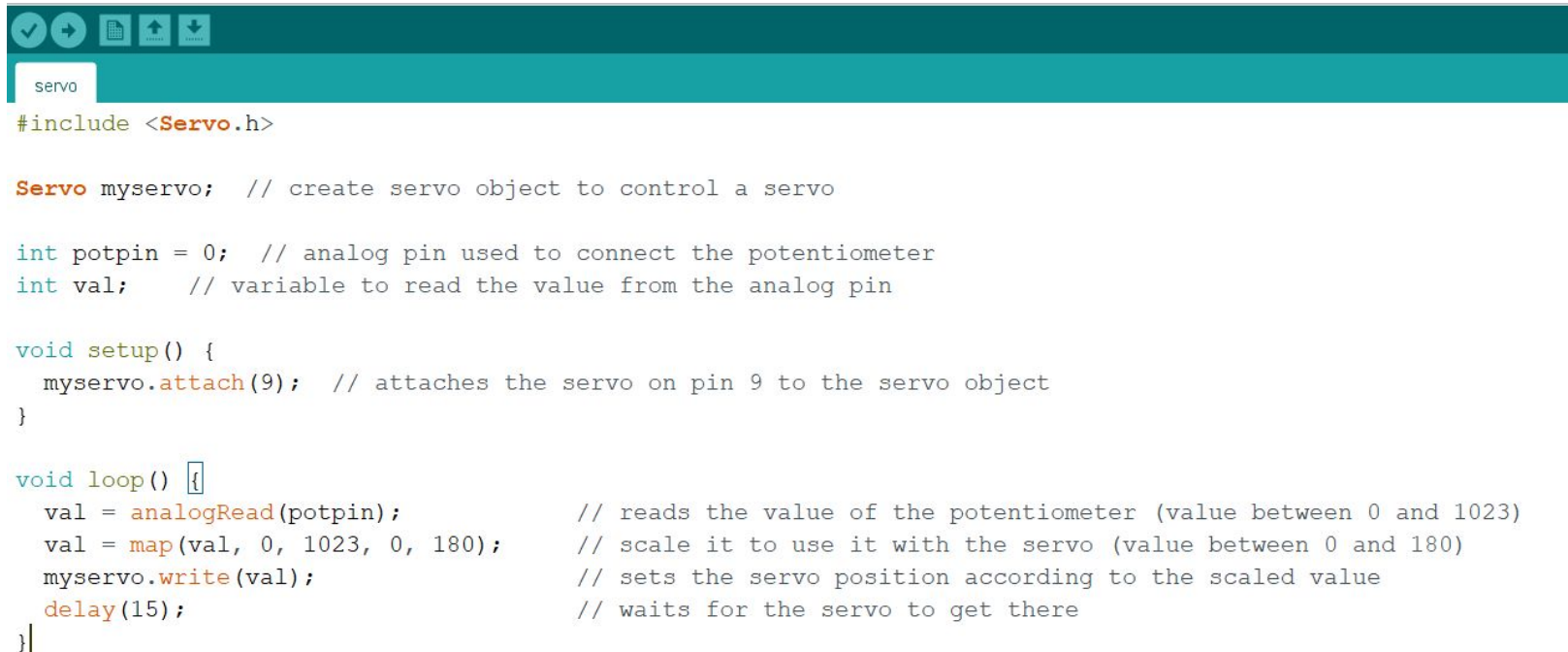
void setup() {
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
}

void loop() {
  for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
  for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position
  }
}
```

Ejercicio: Servomotor con potenciómetro



Ejercicio: Servomotor con potenciómetro



```
servo

#include <Servo.h>

Servo myservo;  // create servo object to control a servo

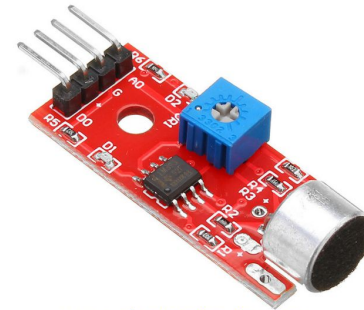
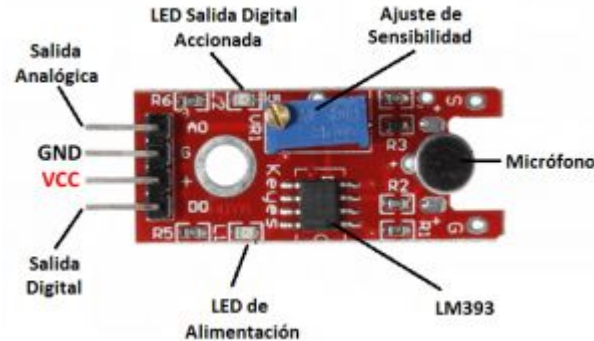
int potpin = 0;  // analog pin used to connect the potentiometer
int val;         // variable to read the value from the analog pin

void setup() {
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
}

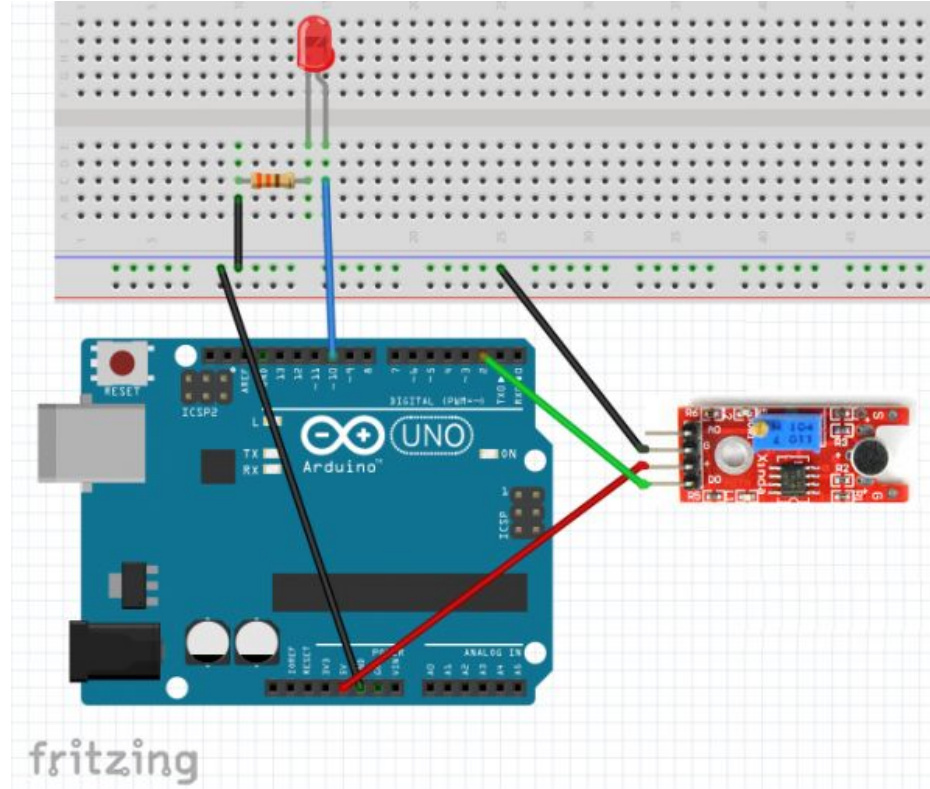
void loop() {
  val = analogRead(potpin);  // reads the value of the potentiometer (value between 0 and 1023)
  val = map(val, 0, 1023, 0, 180);  // scale it to use it with the servo (value between 0 and 180)
  myservo.write(val);  // sets the servo position according to the scaled value
  delay(15);  // waits for the servo to get there
}
```

Sensores de sonido KY-037

Este módulo está compuesto por: un Micrófono sensible KY-037, dos salidas: una llamada “A0”, que es una **salida analógica**, señal de tensión de salida en tiempo real de micrófono y otra llamada “D0”, que es una **señal digital** que se activa cuando la intensidad del sonido a alcanzado un cierto **umbral** previamente configurado, dicha salida de alta o baja se puede configurar mediante el ajuste del umbral de sensibilidad que puede configurar a través del **potenciómetro**.



Ejercicio: Encender luz con sonido



Ejercicio: Encender luz con sonido



```
servo

int LED = 10 ;
int sensor = 2 ;
bool estado = false ;

void setup()
{
  pinMode( LED, OUTPUT) ;
  pinMode( sensor , INPUT_PULLUP) ;
  digitalWrite(LED , LOW) ; // Apagamos el LED al empezar
}

void loop()
{
  bool valor = digitalRead(sensor) ; //leemos el estado del sensor
  if ( valor == true ) //Si está activada la salida D0
  {
    estado = ! estado ;           // cambiamos el estado del LED
    digitalWrite(LED, estado) ;   // escribimos el nuevo valor
    delay (1000);
  }
}
```