

# **Sistemas Empotrados y de Tiempo Real**

## **Desarrollo con Arduino**

### **Entradas/Salidas Analógicas Digitales**

---

Grado en Ingeniería de Robótica Software

Teoría de la Señal y las Comunicaciones y  
Sistemas Telemáticos y Computación

Roberto Calvo Palomino  
[roberto.calvo@urjc.es](mailto:roberto.calvo@urjc.es)

# Esqueleto del sketch

- Todo sketch tendrá 2 funciones obligatorias:
  - **setup()**: Función que se utiliza para inicializar datos o puertos, y que se ejecuta 1 única vez al inicio del programa.
  - **loop()**: El contenido de esta función ejecuta repetidamente mientras la placa arduino siga encendida.



```
// the setup routine runs once when you press reset:
void setup() {

}

// the loop routine runs over and over again forever:
void loop() {

}
```

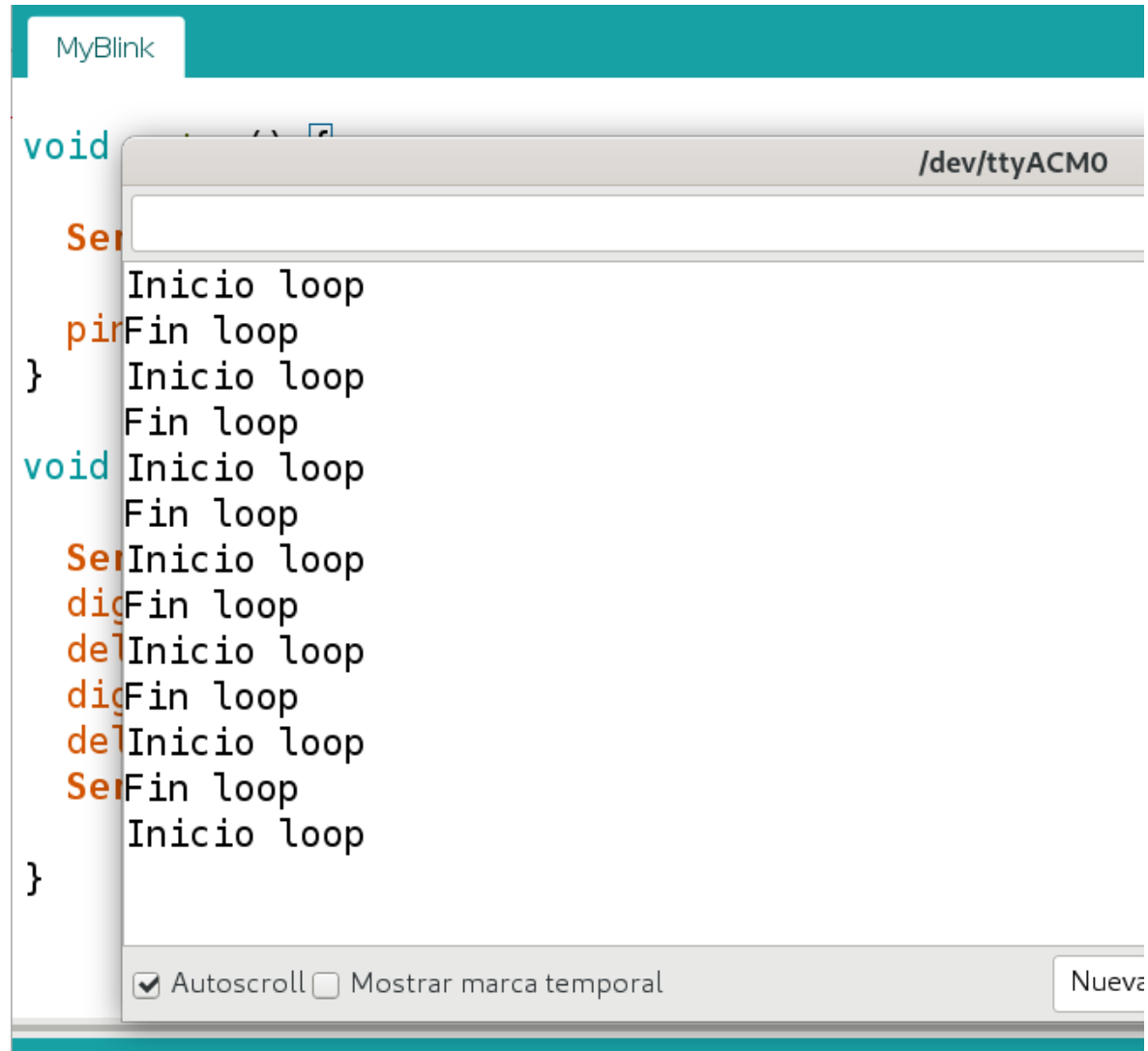
# Depuración

- A través del puerto serie (USB)
- Configurar la velocidad del puerto serie en la función `setup()`  
`Serial.begin(SPEED)`
- Imprimir mensajes con  
`Serial.println(MESSAGE)`
- Todos los mensajes son mostrados por la interfaz serial de Arduino.
- Normalmente conectada a la interfaz USB.

Cargar sketch

```
void setup() {  
    Serial.begin(9600);  
    pinMode(LED_BUILTIN, OUTPUT);  
}  
  
void loop() {  
    Serial.println("Inicio loop");  
    digitalWrite(LED_BUILTIN, HIGH);  
    delay(1000);  
    digitalWrite(LED_BUILTIN, LOW);  
    delay(1000);  
    Serial.println("Fin loop");  
}
```

# Depuración



# Funciones Comunes

- **`Serial.println(value)`**
  - Imprime el valor por el puerto serie.
- **`pinMode(pin, mode)`**
  - Configura un pin digital para leer (entrada), o escribir (salida)
- **`digitalRead(pin)`**
  - Lee un valor digital (HIGH o LOW) en un pin establecido como entrada.
- **`digitalWrite(pin, value)`**
  - Escribe un valor digital (HIGH o LOW) en un pin establecido como salida.

# Tipos de Datos

Tipos Numéricos	Bytes(*)	Rango	Uso
int	2	-32768 to 32767	Enteros negativos y positivos
unsigned int	2	0 to 65535	Solo enteros positivos
long	4	-2147483648 to 2147483647	Rango largo de enteros positivos/negativos
unsigned long	4	4294967295	Solo largo rango de enteros positivos
float	4	3.4028235E+38 to -3.4028235E+38	Representa numero coma flotante
double	4	Igual que float	En Arduino, igual que float.
bool	1	false (0) o true (1)	Valores condicionales
char	1	-128 to 127	Representa un carácter ASCII
byte	1	0 to 255	Enteros en un rango corto

(\*) Tipos de datos para placas arduino basadas en microcontroladores de 8-bit.

# Grupo de Valores

- Genera arrays de pines y asigna su modo fácilmente.

```
int ledPins[] = {10, 11, 12, 13};  
  
void setup()  
{  
  for (int index = 0; index < 4; index++)  
    pinMode(ledPins[index], OUTPUT);  
}
```

# String

- La clase String nos ayudará a trabajar con cadenas de caracteres de una manera muy fácil

```
char oldString[] = "this is a character array";
```

```
String newString = "this is a string object";
```

- Al ser una clase nos provee numerosos métodos para trabajar con strings (conversiones, comparaciones, concatenaciones, busquedas, reemplazamientos, ...)
  - <https://www.arduino.cc/reference/en/language/variables/data-types/stringobject/>



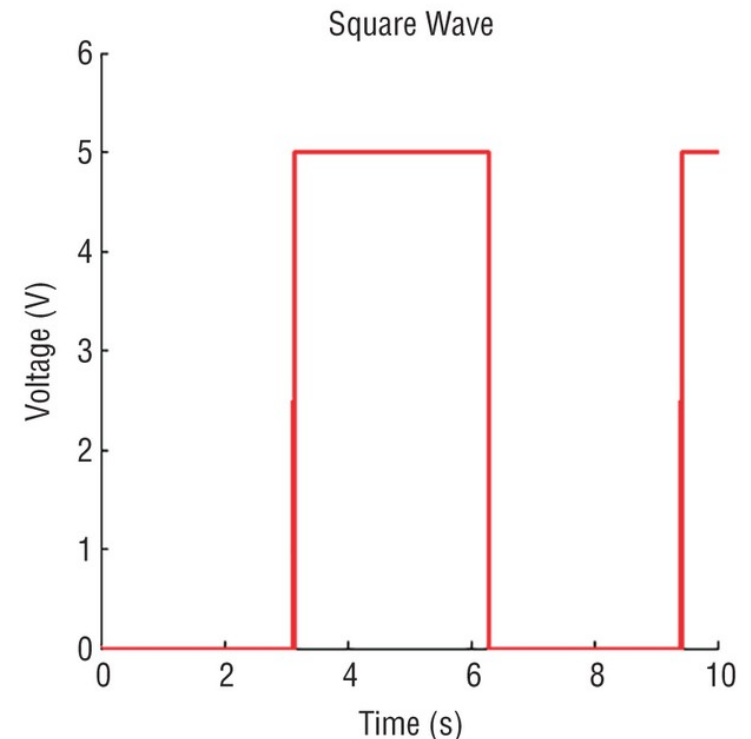
# Muy similar a C

- Structs
- Sentencias condicionales o de control
- Sentencias repetitivas
- Comparadores numéricos y lógicos

Para mas detalle consultar la bibliografía de la asignatura

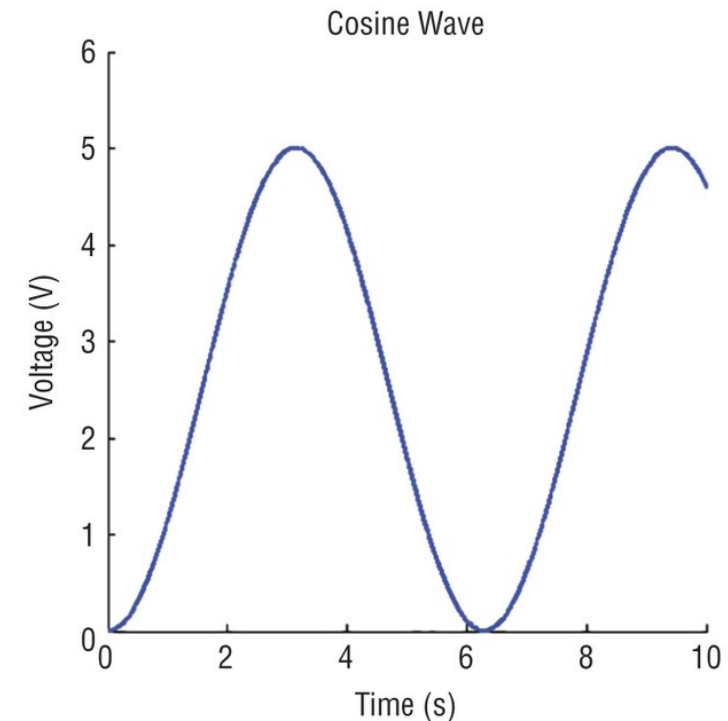
# Señales Digitales

- Una **señal digital** es un tipo de señal (normalmente generada por un fenómeno electromagnético) en que cada símbolo que codifica se puede analizar a través de valores discretos.
- Los microcontroladores usan lógica de 2 estados representados por 2 niveles de tensión eléctrica (H=High y L=Low). A alto nivel representado como 0 y 1.
- Ejemplos:
  - LED
  - Switch / Interruptor



# Señales Analógicas

- Una señal analógica es aquella que los valores de la tensión del voltaje varían constantemente durante el tiempo.
- Arduino UNO contiene un conversor (ADC) con una resolución de 10 bits, devolviendo valores en el rango de [0-1023] (valor máximo  $2^{10}$ )
- El ADC de Arduino permite configuración.
- La mayoría de los sensores son analógicos
- Ejemplos:
  - Temperatura, PIR, ultrasonido, etc.



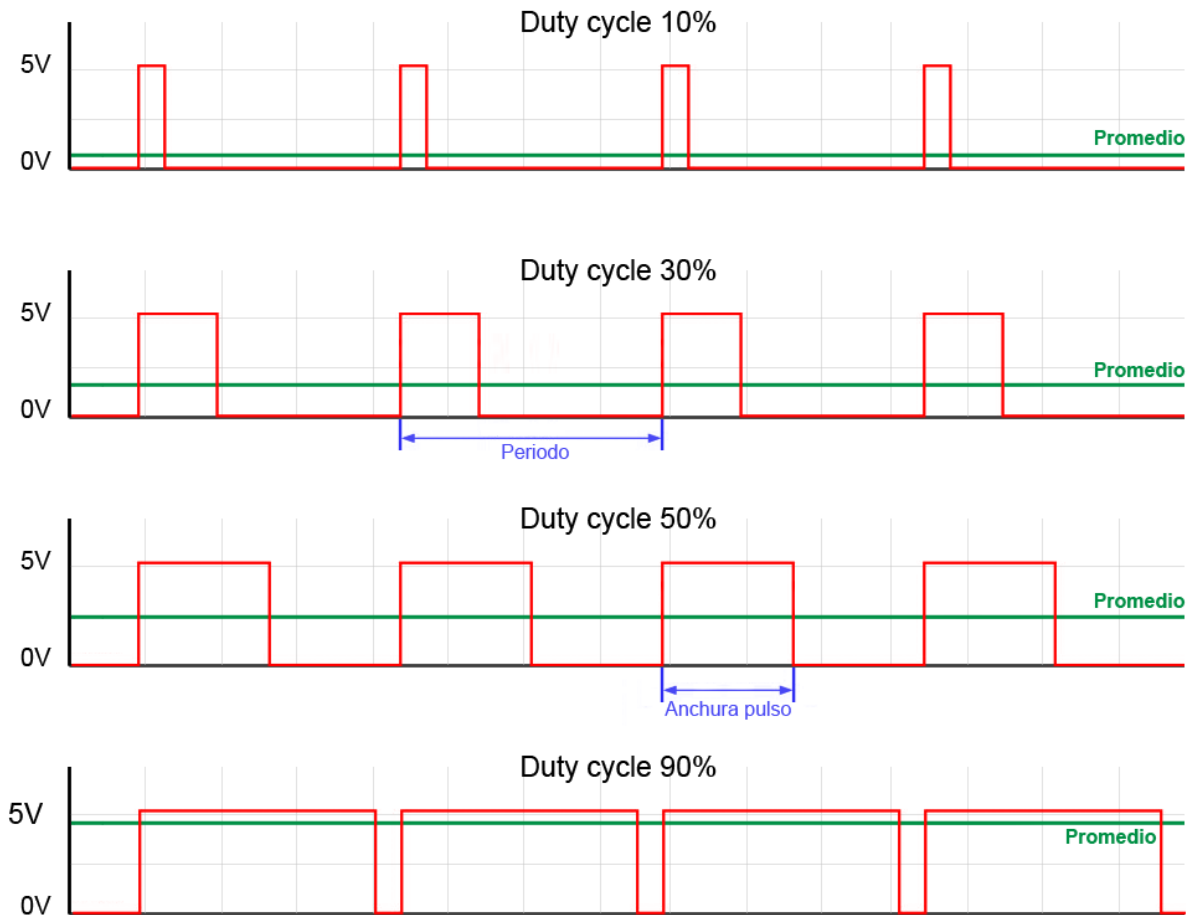
# PWM (Pulse Width Modulation)

- PWM nos permite simular comportamientos analógicos a través de los pines digitales de Arduino.
- Problema
  - Un led se enciende (HIGH) o se apaga (LOW)
  - Un motor está girando (HIGH) o está pagado (LOW)

¿Cómo podemos generar diferentes intensidades para el Led o diferentes velocidades para el motor?

# PWM (Pulse Width Modulation)

- La proporción de tiempo que está la señal HIGH respecto al total del ciclo, se denomina "duty cycle".



# PWM (Pulse Width Modulation)

- Arduino incorpora la función `analogWrite()` que puede ser usada para generar una señal PWM en un pin digital.
  - `analogWrite(PIN, 0)`: es una señal de ciclo de trabajo del 0%.
  - `analogWrite(PIN, 127)`: es una señal de un ciclo de trabajo del 50%.
  - `analogWrite(PIN, 255)`: es una señal de ciclo de trabajo del 100%.
- **IMPORTANTE:** `analogWrite` genera señales digitales de 5V (incluso a un duty cycle muy bajo), si el sensor solo soporta 3V puede sufrir daños.

# Trabajando con el tiempo

- Usa las siguientes funciones para obtener el contador
- `millis()`: Número de milisegundos desde que la placa empezó a ejecutar el sketch actual.
- `micros()`: Número de microsegundos desde que la placa empezó a ejecutar el sketch actual.
  - En placas de 16 MHz:
    - resolución de 4 microsegundos
  - En placas de 8 MHz:
    - resolución de 8 microsegundos

```
unsigned long time;

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

void loop() {
  Serial.print("Time: ");
  time = micros();

  Serial.println(time);
  delay(1000);
}
```

# Esperas y pausar la ejecución

- `delay(value_ms)`
  - Pausa el sketch durante la cantidad de milisegundos especificados
- `delayMicroseconds(value_micros.)`
  - delay bastante exacto para valores en el rango [3 - 16383]
  - Para valores mayores usar `delay()`

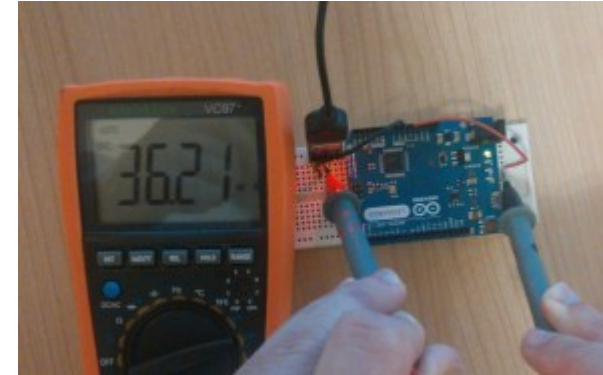
*¿Como afecta la ejecución de `delay()` en el microcontrolador?*



# Uso de delay()

```
void setup() {}  
  
void loop() {  
  delay(99999);  
}
```

```
void setup() {}  
  
void loop() {  
  while(1);  
}
```



(\*)

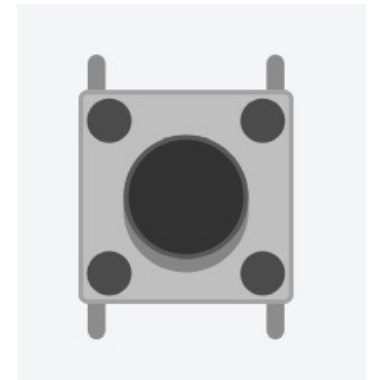
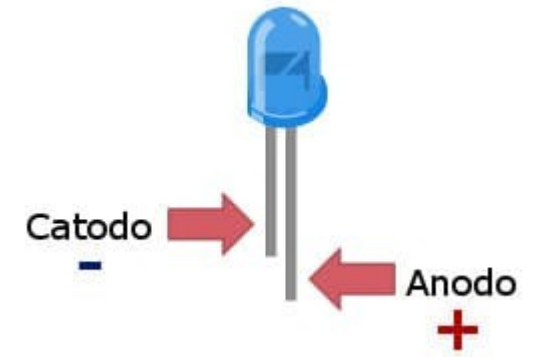


- delay() no libera la CPU del microcontrolador.
- Veremos mecanismos para mejorar este comportamiento.

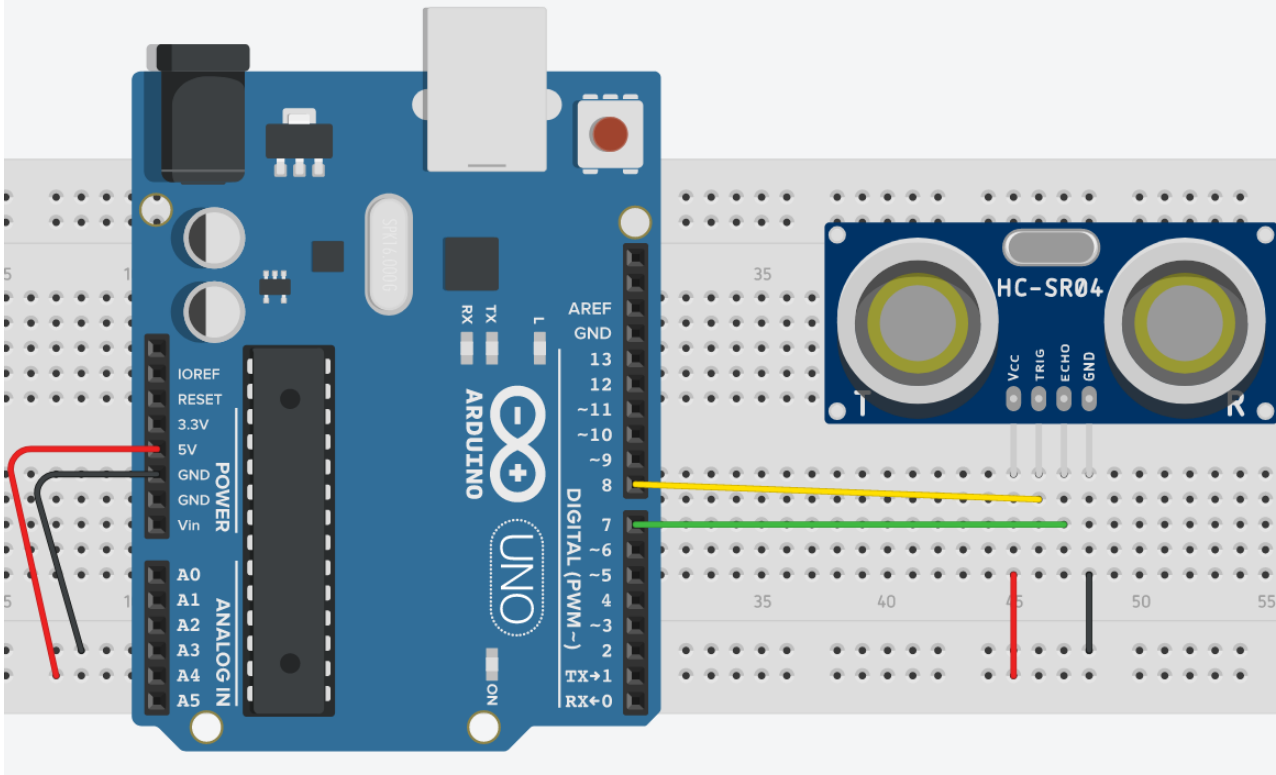
(\*) <https://todohacker.com/tutoriales/arduino-usar-delays-o-evitarlos>

# Sensores

- LED
  - Sensor Digital
  - Diodo: cátodo, ánodo.
  - 1.2 - 3.6 v.
- Botón/switch
  - Mantiene cerrado el circuito
  - Hasta que se presiona
  - ¿Cómo podemos saber si está presionado?

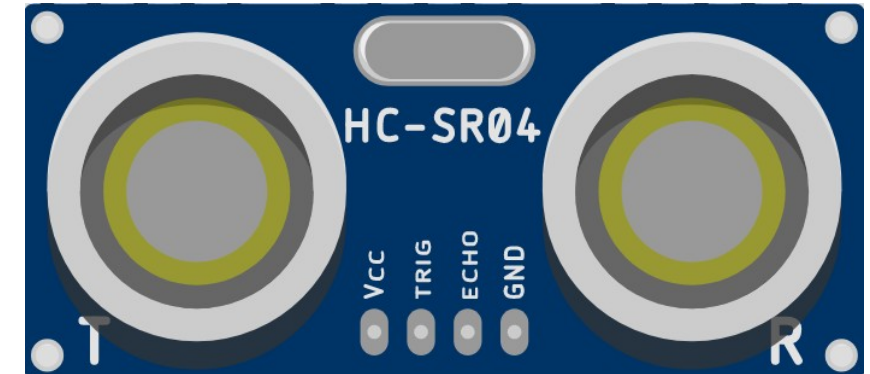


# Sensores: HC-SR04

- Ultra-Sonido (HC-SR04)
    - Rango: 2-400 cm
    - Precisión: 3mm
    - Ángulo de medición: 15°
  - VCC y GND
  - Trigger y ECHO
- 
- Mide distancias a objetos utilizando la misma técnica que un SONAR. Emite un sonido a frecuencias determinadas (~40 KHz) y mide del tiempo de vuelo que tarda en recibirse su eco (rebote sobre el objeto)

# Sensores: HC-SR04

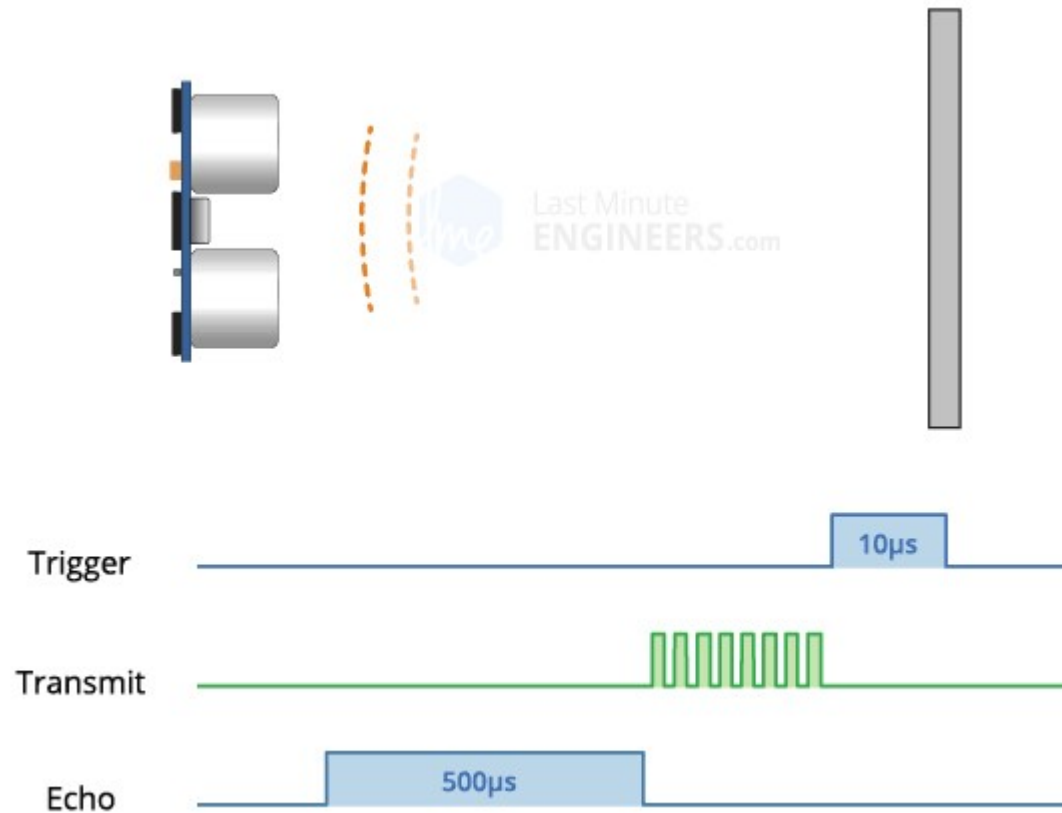
```
digitalWrite(PIN_TRIGGER, HIGH);  
delayMicroseconds(10);  
digitalWrite(PIN_TRIGGER, LOW);  
  
distancia=pulseIn(PIN_ECHO, HIGH);
```



- *pulseIn*: Devuelve el tiempo en micro-segundos hasta que aparece un pulso (HIGH o LOW).

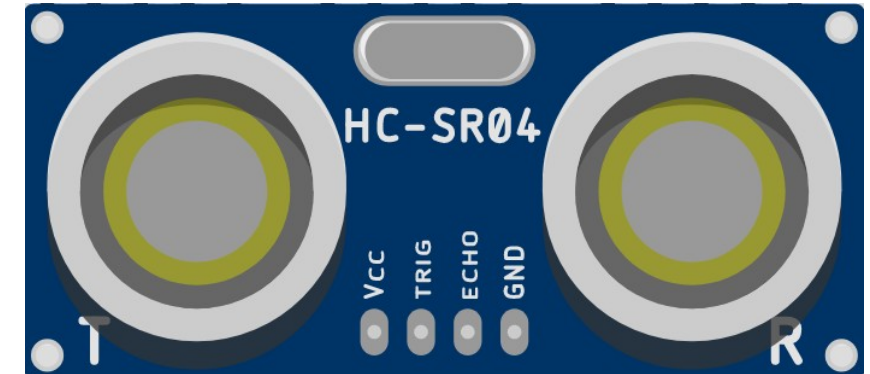
# Sensores: HC-SR04

- Ejemplo de funcionamiento: [LINK](#)



# Sensores: HC-SR04

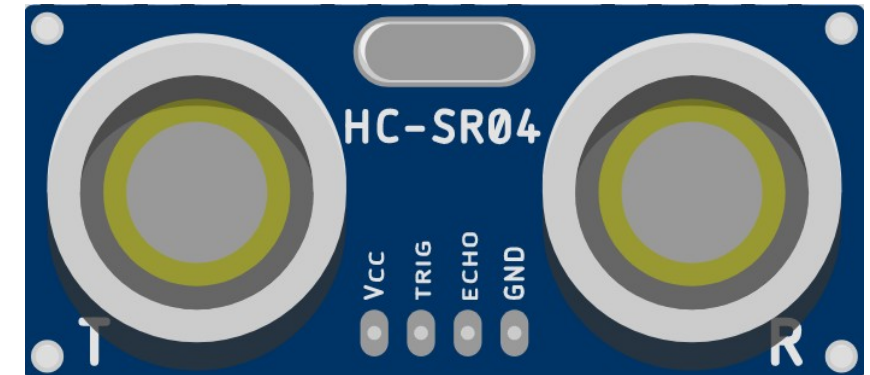
```
digitalWrite(PIN_TRIGGER, HIGH);  
delayMicroseconds(10);  
digitalWrite(PIN_TRIGGER, LOW);  
  
distancia=pulseIn(PIN_ECHO, HIGH);
```



- *pulseIn*: Devuelve el tiempo en microsegundos hasta que aparece un pulso (HIGH o LOW).
- Velocidad del sonido: 343 m/s

# Sensores: HC-SR04

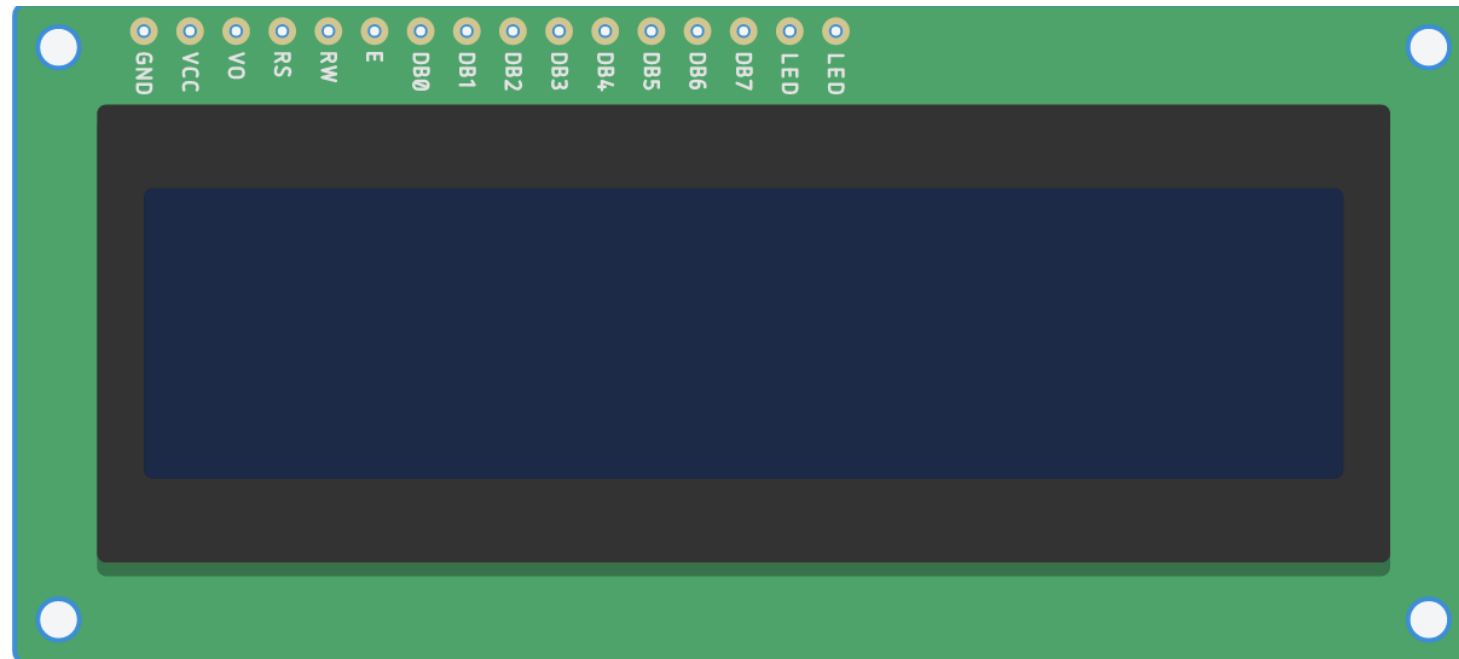
```
digitalWrite(PIN_TRIGGER, HIGH);  
delayMicroseconds(10);  
digitalWrite(PIN_TRIGGER, LOW);  
  
distancia=pulseIn(PIN_ECHO, HIGH);
```



- *pulseIn*: Devuelve el tiempo en microsegundos hasta que aparece un pulso (HIGH o LOW).
- Velocidad del sonido: 343 m/s
  - $343 * 1e^2 / 1e^6 / 2 = 0.01715$  cm/microsegundo.

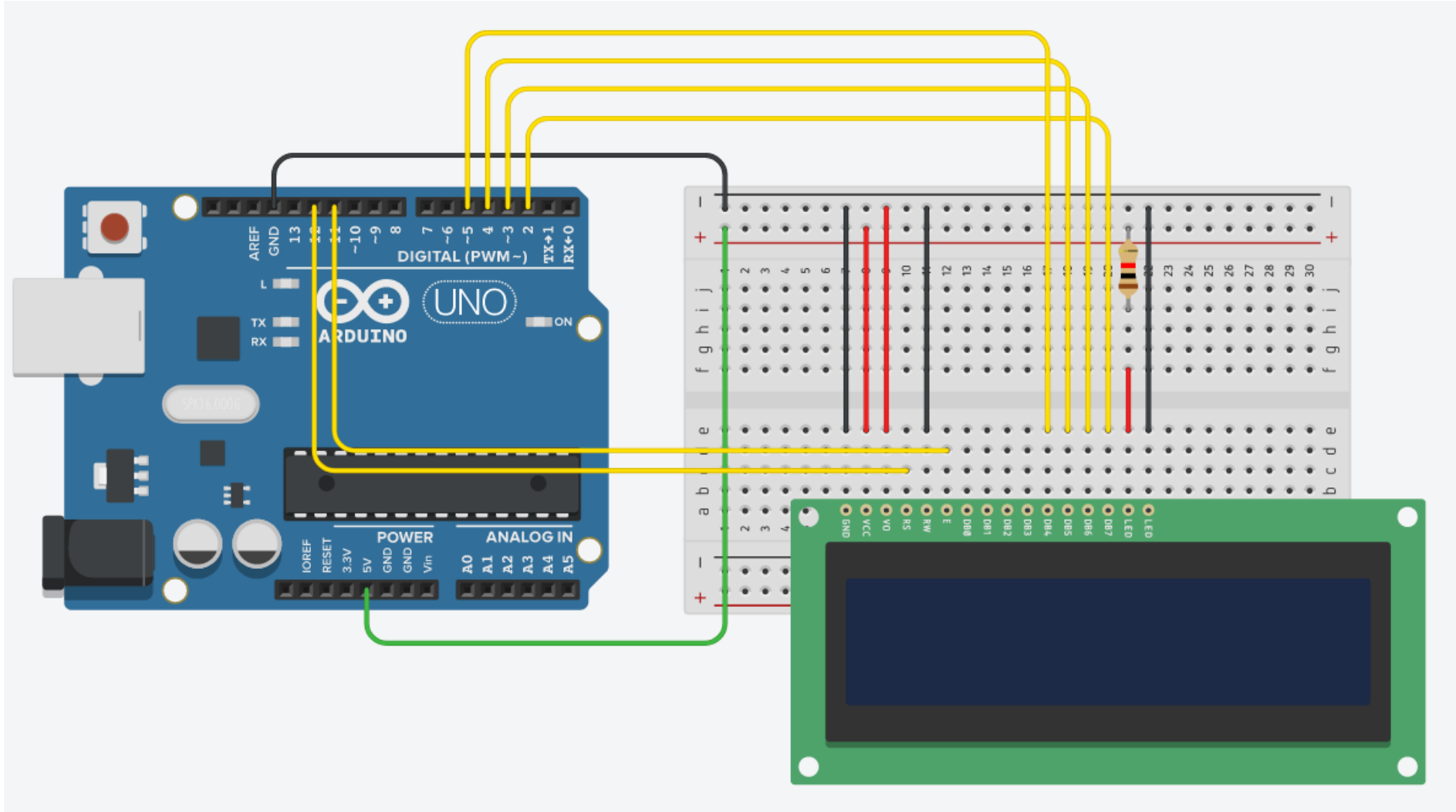
# Sensores: LED 16x2

- LED 16x2
- Bus de 8 líneas
- Manejo de contraste
- <https://www.arduino.cc/en/Reference/LiquidCrystal>





# Sensores:LED 16x2



¿Como calcularías la latencia del planificador en Arduino?

# Bibliografía

- [Libro] Arduino Cookbook, 3rd Edition, Abril 2020
  - Michael Margolis, Brian Jepson, Nicholas Robert Weldon
- [Libro] Exploring Arduino, 2nd Edition, Septiembre 2019
  - Jeremy Blum

