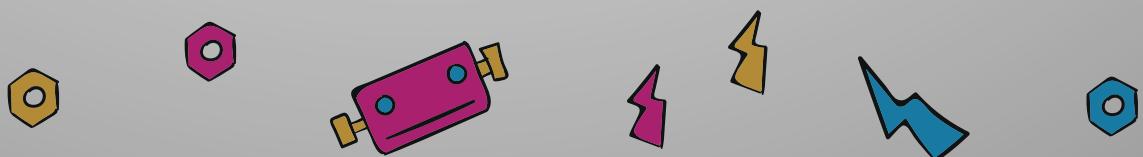




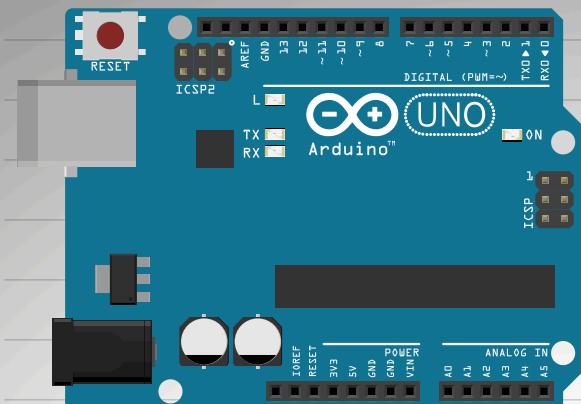
## Construcción de prototipos con Arduino



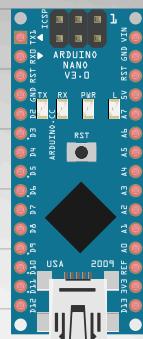
## Arduino

Arduino es una plataforma de electrónica abierta para la creación de prototipos basada en **software y hardware libres**.

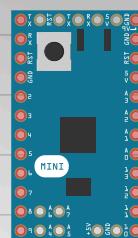
Arduino Uno



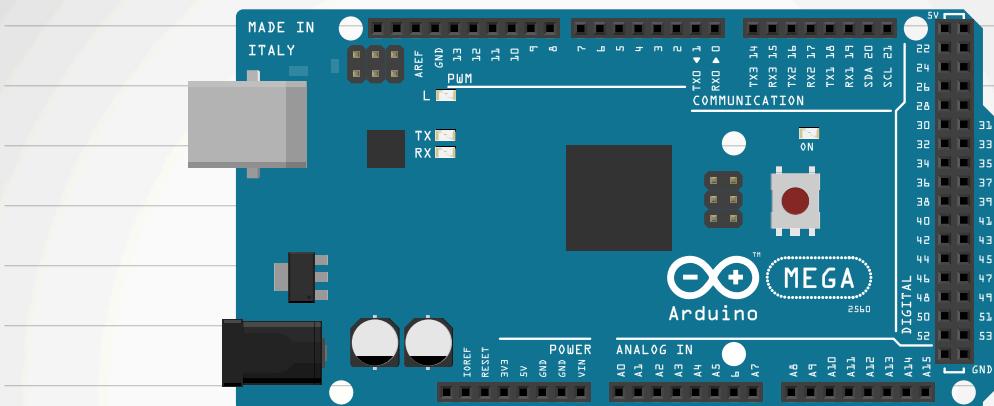
Arduino  
Nano



Arduino  
Micro



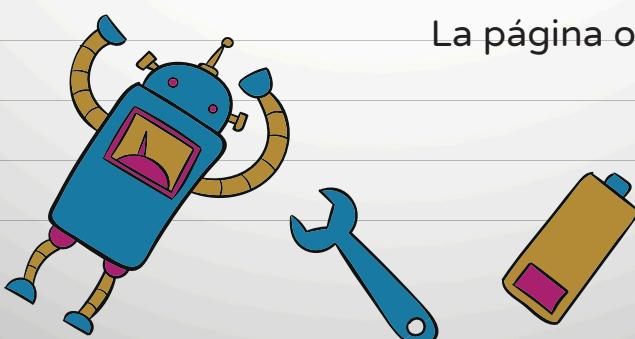
Arduino Mega



fritzing

Con Arduino podemos tomar información del entorno conectando sensores a través de sus pines de entrada y actuar controlando luces, motores y otros actuadores.

La página oficial es [www.arduino.cc](http://www.arduino.cc)

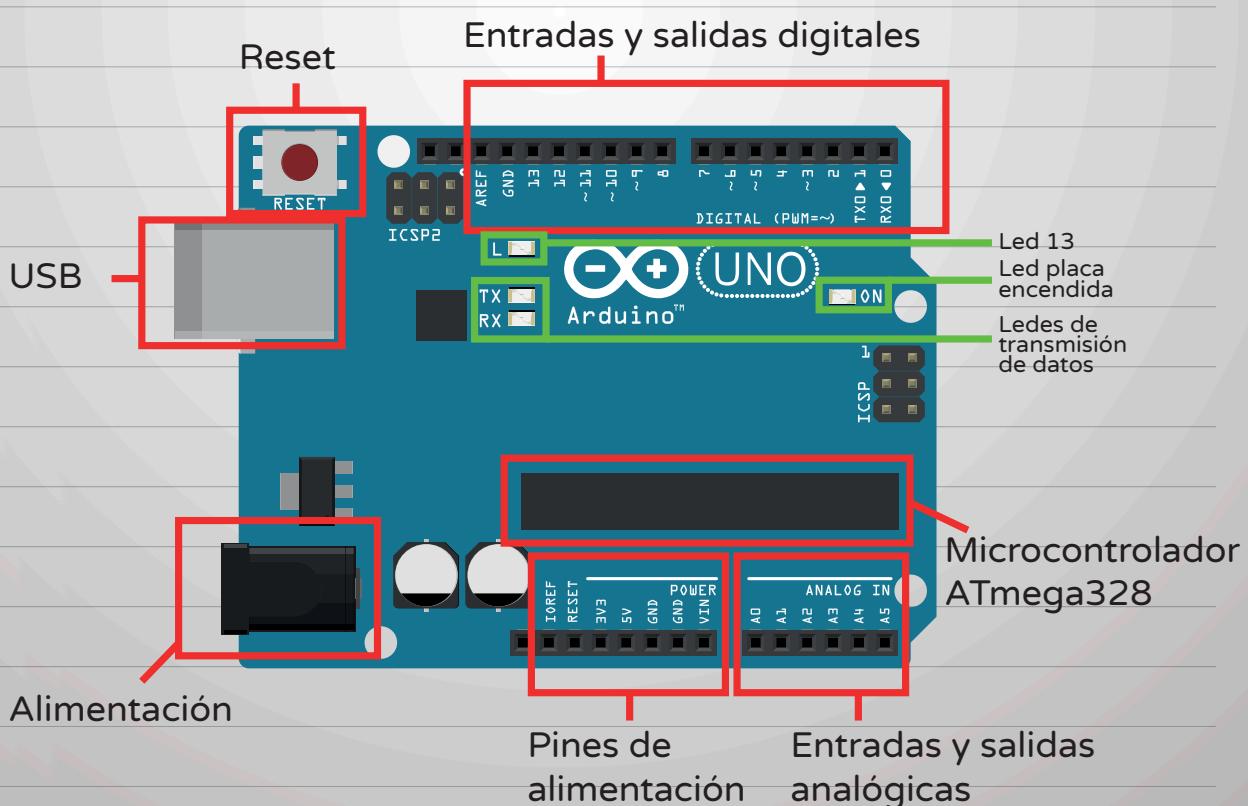


## Descripción de la placa Arduino Uno

Arduino Uno es una placa electrónica basada en el microcontrolador ATmega328.

Cuenta con 14 entradas/salidas digitales, de las cuales 6 se pueden utilizar como salidas PWM ~ (Modulación por ancho de pulsos) y otras 6 son entradas analógicas. Además, incluye un resonador cerámico de 16 MHz, un conector USB, un conector de alimentación, una cabecera ICSP es un botón de reseteo.

La placa incluye todo lo necesario para que el microcontrolador haga su trabajo, basta conectarla a un ordenador con un cable USB o a la corriente eléctrica a través de un transformador.

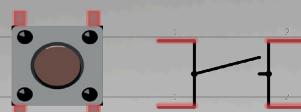


## Materiales

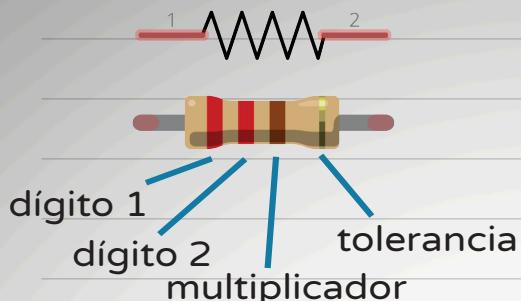
### Led



### Pulsador



### Resistencia

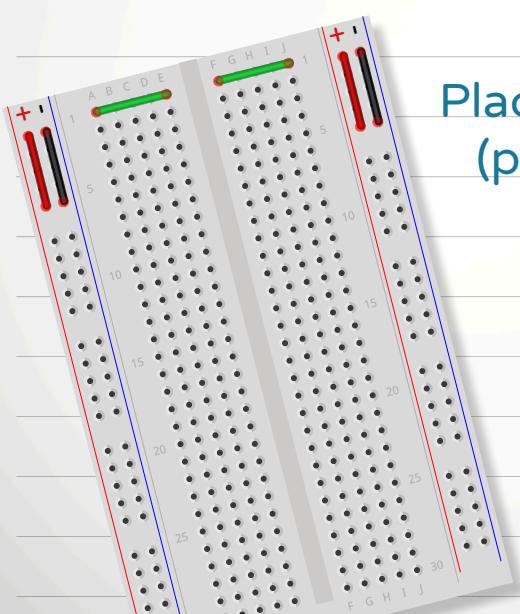
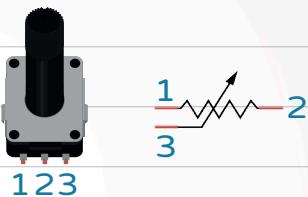


	1º Dígito	2º Dígito	Multiplicador	Tolerancia
Negro	0	0	$\times 10^0$	
Marrón	1	1	$\times 10^1$	$\pm 1\%$
Rojo	2	2	$\times 10^2$	$\pm 2\%$
Naranja	3	3	$\times 10^3$	
Amarillo	4	4	$\times 10^4$	
Verde	5	5	$\times 10^5$	$\pm 0,5\%$
Azul	6	6	$\times 10^6$	$\pm 0,25\%$
Violeta	7	7	$\times 10^7$	$\pm 0,1\%$
Gris	8	8	$\times 10^8$	$\pm 0,05\%$
Blanco	9	9	$\times 10^9$	
Dorado			$\times 0,1$	$\pm 5\%$
Plateado			$\times 0,01$	$\pm 10\%$

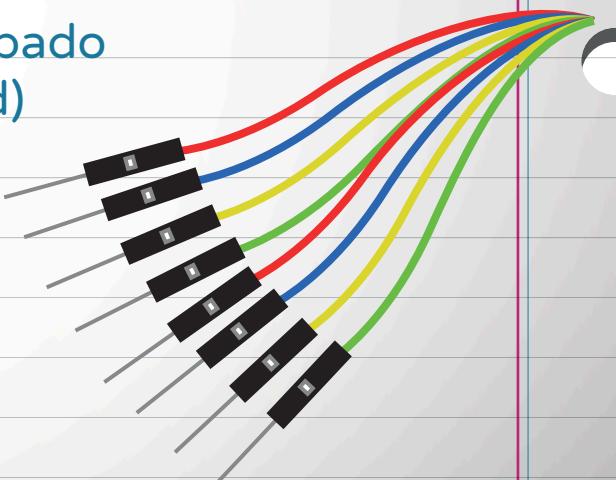
### Fotoresistencia LDR



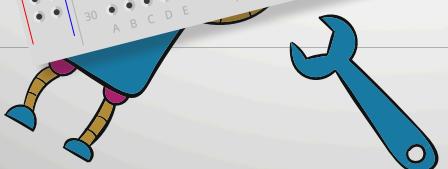
### Potenciómetro



### Placa prototipado (protoboard)



...y cables.



## Entorno Arduino

El entorno de programación Arduino recibe el nombre de IDE de Arduino, correspondiente a las siglas de Integrated Development Environment o entorno de desarrollo integrado.

Los ficheros que se manejan con el entorno de Arduino se denominan sketch y por defecto son nombrados del siguiente modo: **sketch+fecha+letra**. El fichero que contiene el código del programa tiene la extensión **\*.ino**.

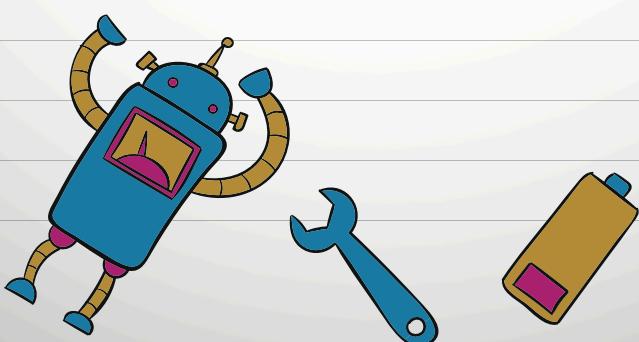
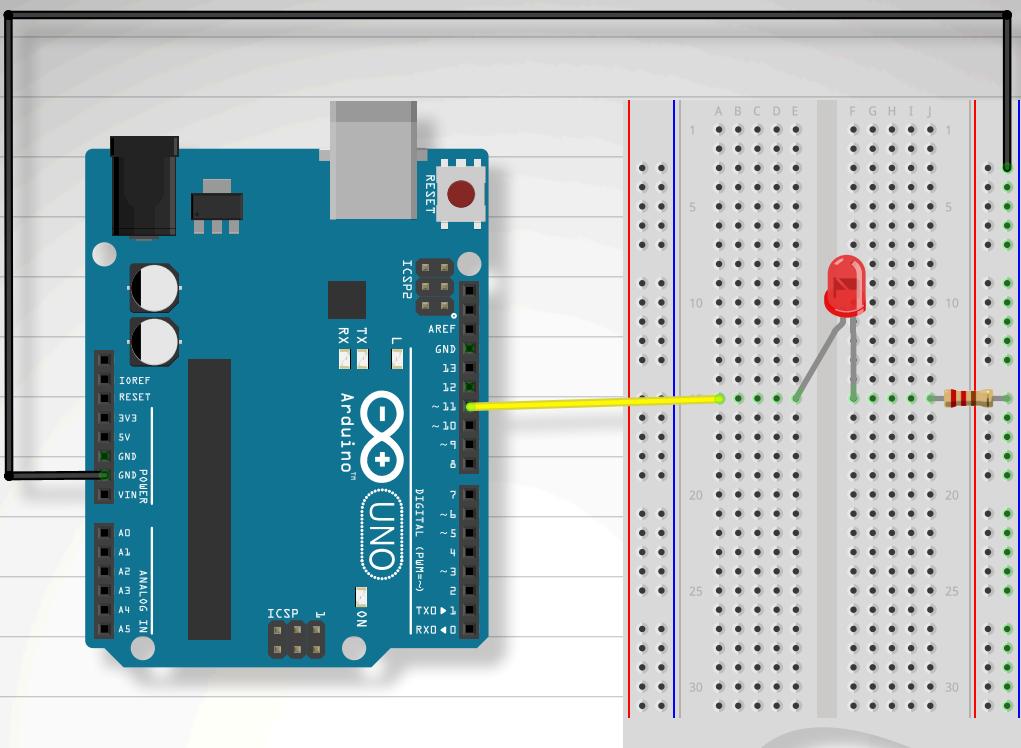


## Ejercicio 1.- Blink

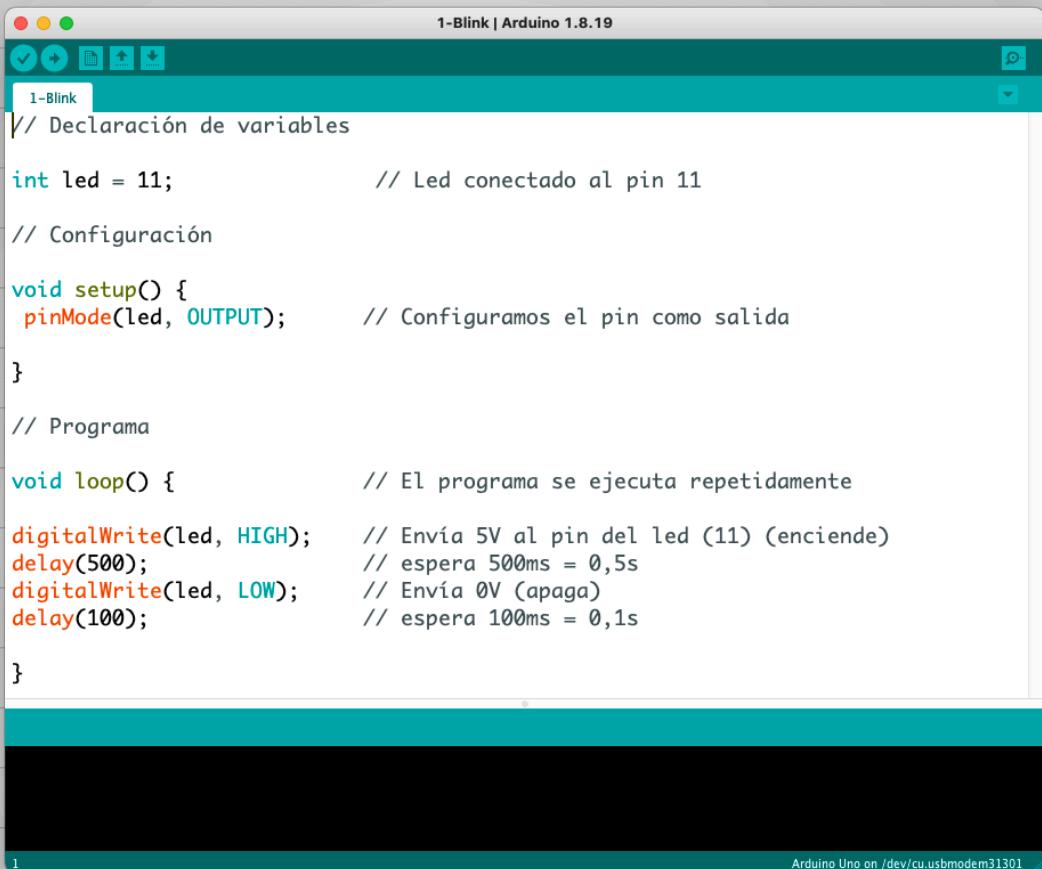
- Configurar Arduino
- Hacer parpadear un LED y variar la frecuencia de parpadeo

### Montaje

Led en el pin 11 con una resistencia de protección de 220Ω.



## Código



The screenshot shows the Arduino IDE interface with the title bar "1-Blink | Arduino 1.8.19". The code editor contains the following C-like pseudocode:

```
// Declaración de variables
int led = 11; // Led conectado al pin 11

// Configuración

void setup() {
  pinMode(led, OUTPUT); // Configuramos el pin como salida
}

// Programa

void loop() { // El programa se ejecuta repetidamente
  digitalWrite(led, HIGH); // Envía 5V al pin del led (11) (enciende)
  delay(500); // espera 500ms = 0,5s
  digitalWrite(led, LOW); // Envía 0V (apaga)
  delay(100); // espera 100ms = 0,1s
}
```

The status bar at the bottom right indicates "Arduino Uno on /dev/cu.usbmodem31301".

Y ahora tú....

1. Prueba a cambiar el tiempo de parpadeo
2. Podrías hacer que el LED parpadeara una sola vez?
3. Usa variables para definir el tiempo de parpadeo

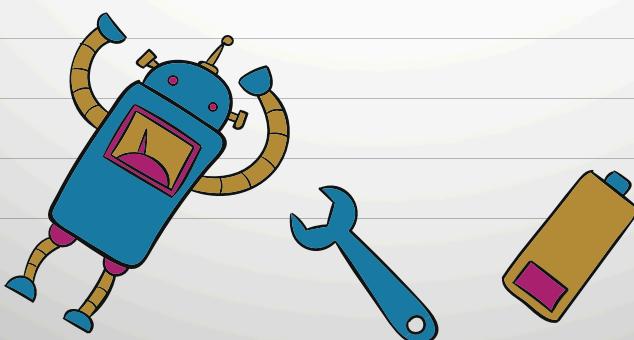
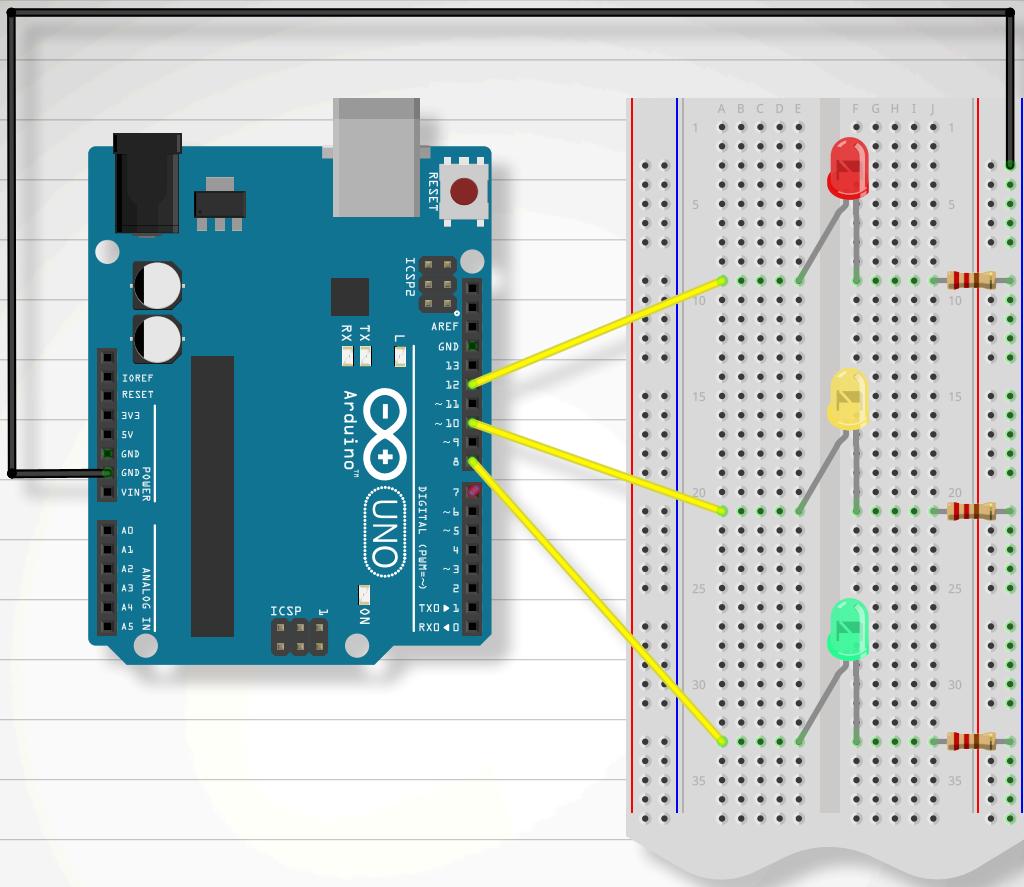


## Ejercicio 2.- Semáforo

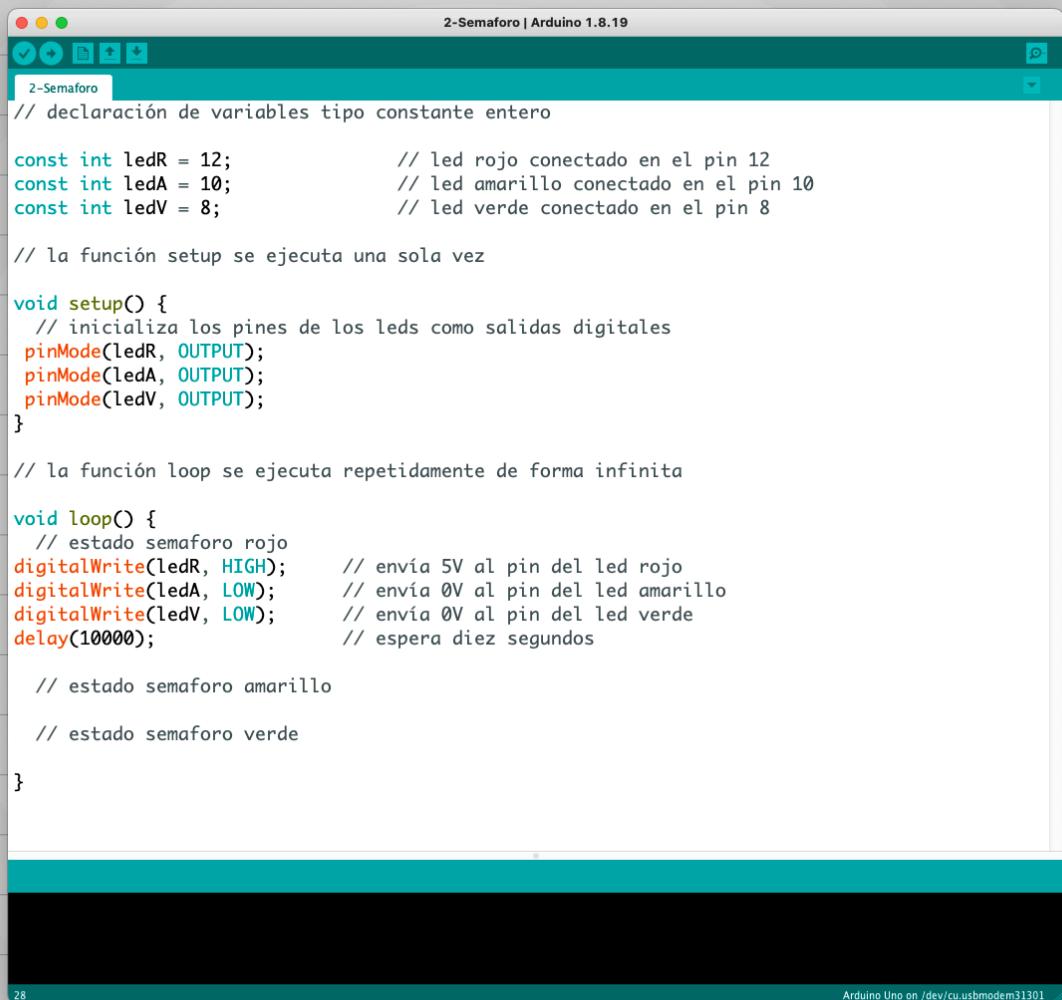
Simular un semáforo de coches (rojo, amarillo y verde) que funcione de forma cíclica.

### Montaje

Debemos conectar 3 leds con sus resistencias de protección a los pines 12, 10 y 8.



## Código



The screenshot shows the Arduino IDE interface with the title bar "2-Semaforo | Arduino 1.8.19". The code is written in C++ and defines three constant integer variables: ledR (pin 12), ledA (pin 10), and ledV (pin 8). The setup() function initializes the pins as outputs. The loop() function alternates between three states: red (ledR HIGH), yellow (ledA LOW), and green (ledV LOW), with a 10-second delay between each state. The status bar at the bottom indicates "Arduino Uno on /dev/cu.usbmodem31301".

```
// declaración de variables tipo constante entero

const int ledR = 12;           // led rojo conectado en el pin 12
const int ledA = 10;           // led amarillo conectado en el pin 10
const int ledV = 8;            // led verde conectado en el pin 8

// la función setup se ejecuta una sola vez

void setup() {
    // inicializa los pines de los leds como salidas digitales
    pinMode(ledR, OUTPUT);
    pinMode(ledA, OUTPUT);
    pinMode(ledV, OUTPUT);
}

// la función loop se ejecuta repetidamente de forma infinita

void loop() {
    // estado semáforo rojo
    digitalWrite(ledR, HIGH);    // envía 5V al pin del led rojo
    digitalWrite(ledA, LOW);     // envía 0V al pin del led amarillo
    digitalWrite(ledV, LOW);     // envía 0V al pin del led verde
    delay(10000);               // espera diez segundos

    // estado semáforo amarillo

    // estado semáforo verde
}
```

Y ahora tú.....

1. Completa el código con semáforo en amarillo y verde
2. Añade un semáforo de peatones (leds rojo y verde) sincronizado con el de coches

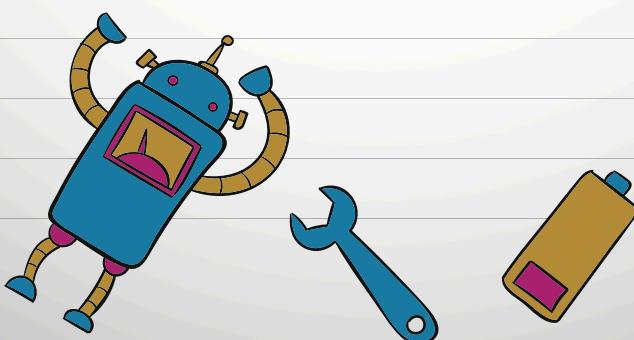
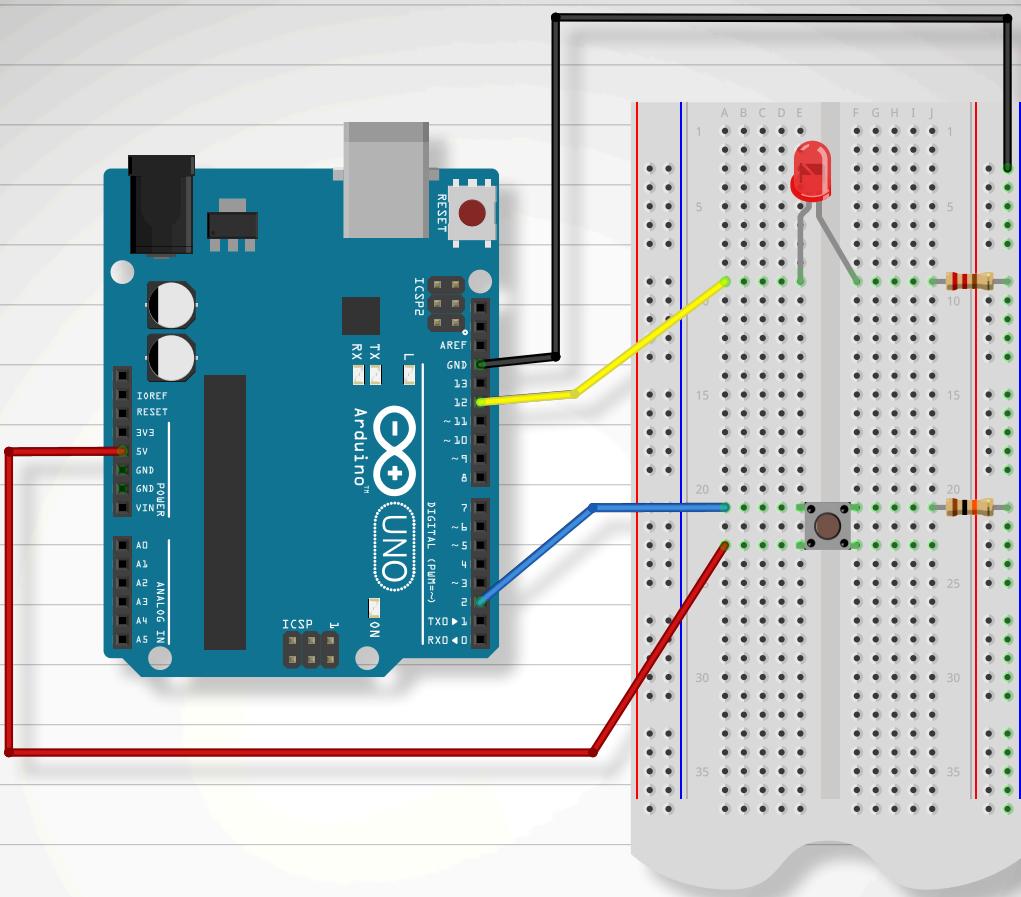


## Ejercicio 3.- Pulsador y Led

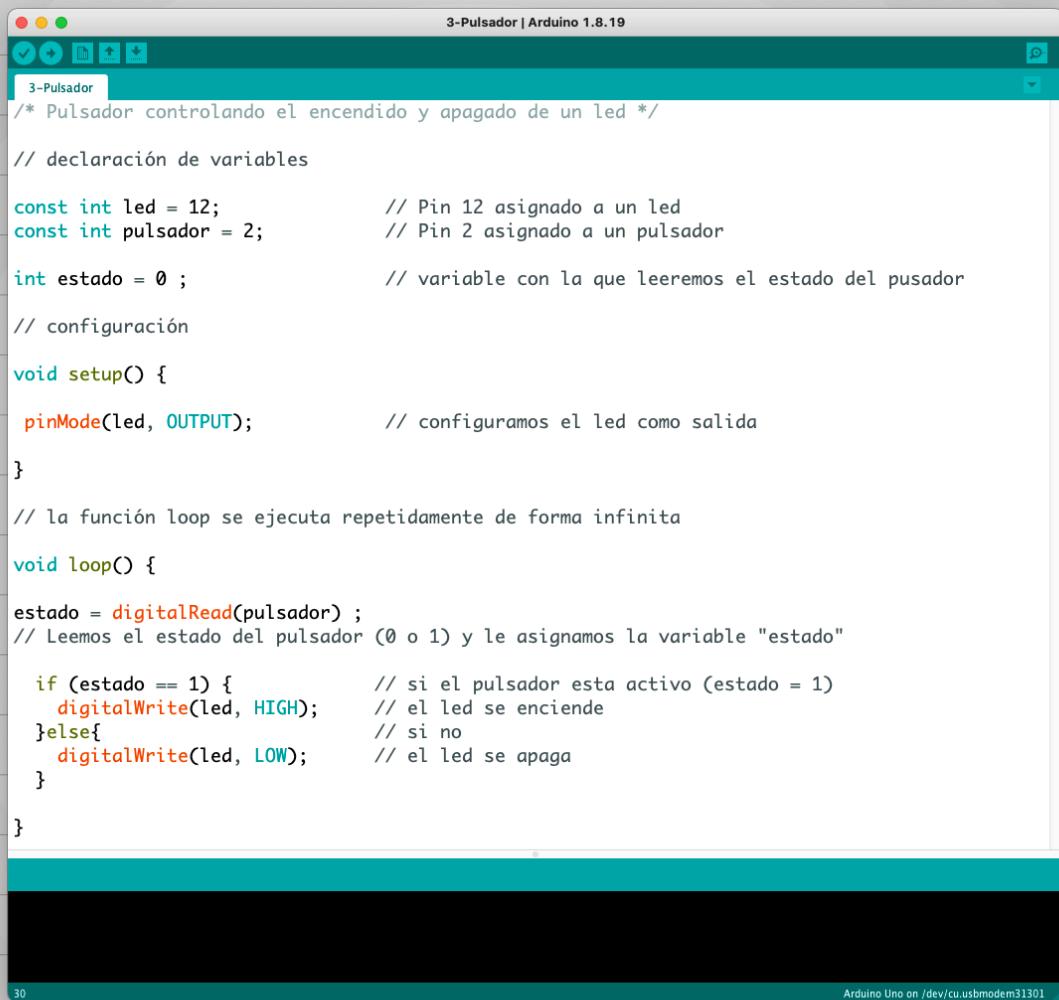
Controlar el encendido y apagado de un LED con un pulsador, de forma que se acienda o se apague cuando presionamos el botón.

### Montaje

Debemos conectar un pulsador al pin 2 con una resistencia de 10 kΩ como aparece en el esquema.



## Código



```
3-Pulsador | Arduino 1.8.19

/*
 * Pulsador controlando el encendido y apagado de un led *
 */

// declaración de variables

const int led = 12;           // Pin 12 asignado a un led
const int pulsador = 2;        // Pin 2 asignado a un pulsador

int estado = 0;               // variable con la que leeremos el estado del pulsador

// configuración

void setup() {
    pinMode(led, OUTPUT);      // configuramos el led como salida
}

// la función loop se ejecuta repetidamente de forma infinita

void loop() {
    estado = digitalRead(pulsador); // Leemos el estado del pulsador (0 o 1) y le asignamos la variable "estado"

    if (estado == 1) {          // si el pulsador esta activo (estado = 1)
        digitalWrite(led, HIGH); // el led se enciende
    }else{                      // si no
        digitalWrite(led, LOW); // el led se apaga
    }
}
```

Arduino Uno on /dev/cu.usbmodem31301

Y ahora tú.....

1. Añade otro LED y haz que se aciendan de manera alternativa al presionar y soltar el pulsador
2. Haz que cada vez que presionamos el pulsador el led se encienda y se apague dos veces.
3. Pulsador con memoria; si inicialmente el estado del led es encendido, al pulsar se apaga, y se estaba apagado se enciende.



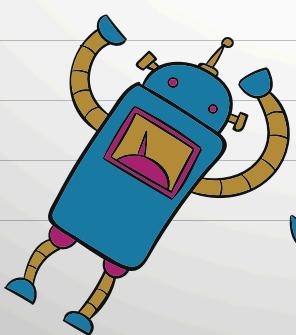
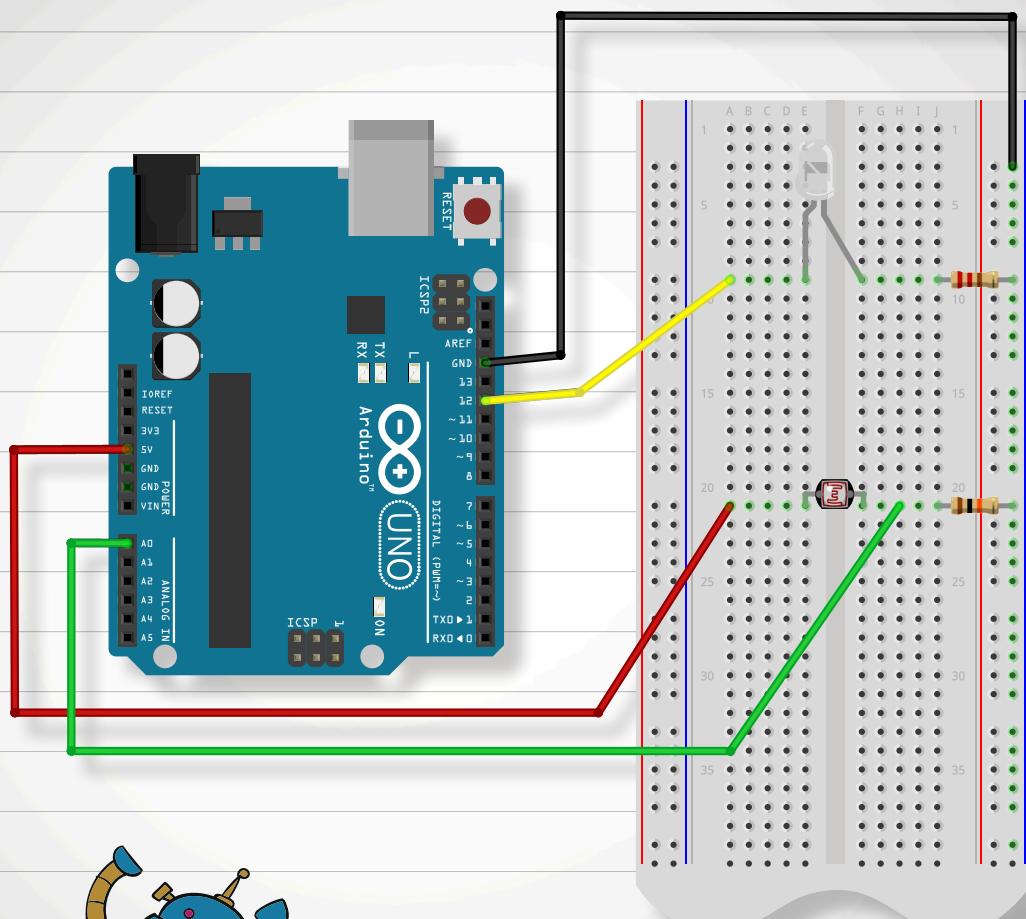
## Ejercicio 4.- Interruptor de luz

Encender un LED en función de la intensidad luminosa que recibe el sensor de luz. En oscuridad el LED está encendido y con alta iluminación apagado.

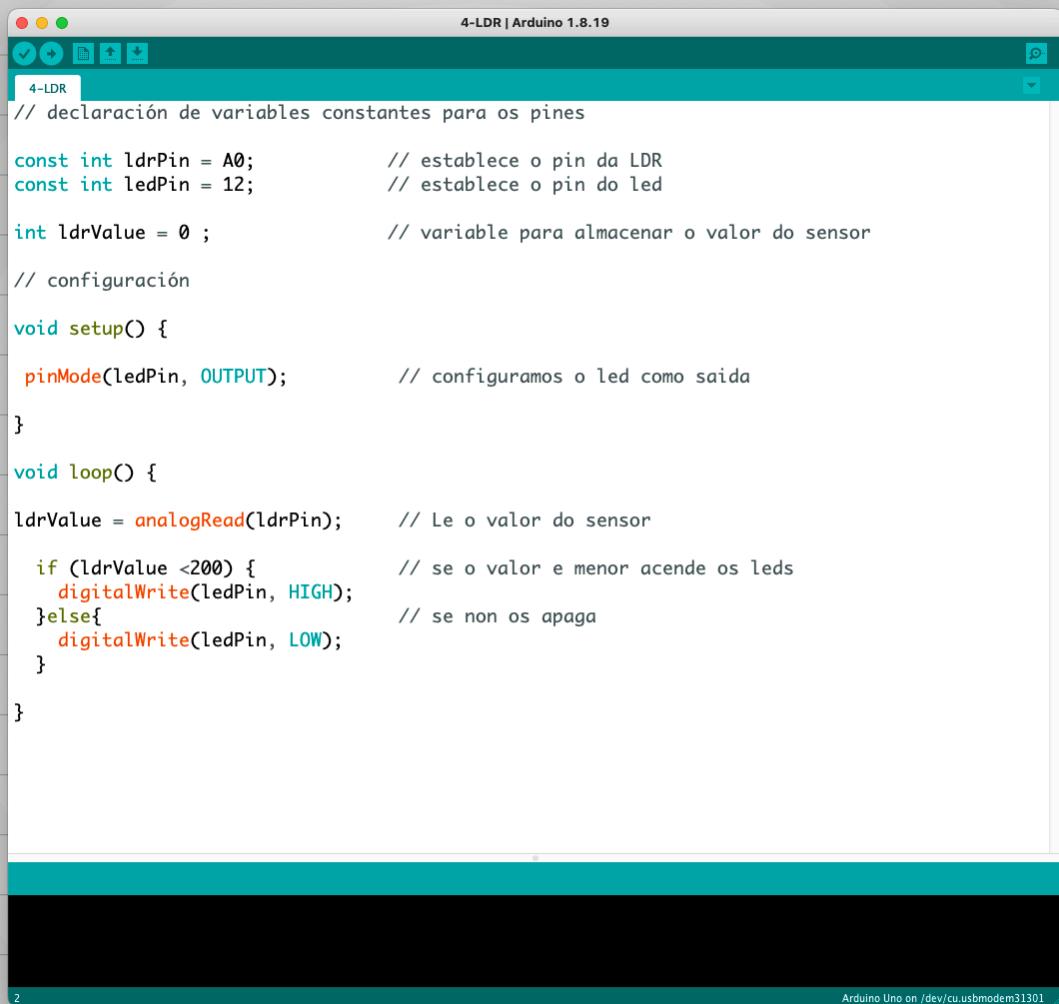
Primero debemos leer qué valores nos da el sensor en función de las diferentes condiciones de luz.

### Montaje

Conectar una LDR al pin analógico A0 con una resistencia de 10 kΩ tal y como aparece en el esquema.



## Código



The screenshot shows the Arduino IDE interface with the title bar "4-LDR | Arduino 1.8.19". The code is written in C++ and defines variables for LDR and LED pins, sets up the LED pin as an output, and reads the analog value from the LDR pin in the loop. It then checks if the value is less than 200 to turn on the LED or off if it's not.

```
// declaración de variables constantes para os pines

const int ldrPin = A0;           // establece o pin da LDR
const int ledPin = 12;          // establece o pin do led

int ldrValue = 0;               // variable para almacenar o valor do sensor

// configuración

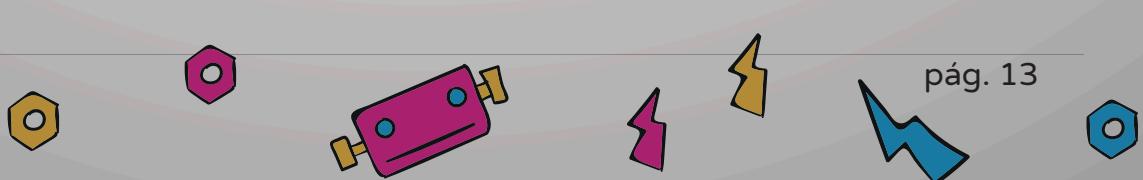
void setup() {
    pinMode(ledPin, OUTPUT);    // configuramos o led como saída
}

void loop() {
    ldrValue = analogRead(ldrPin); // Le o valor do sensor

    if (ldrValue <200) {         // se o valor é menor acende os leds
        digitalWrite(ledPin, HIGH);
    }else{                      // se non os apaga
        digitalWrite(ledPin, LOW);
    }
}
```

Y ahora tú.....

1. Añade dos leds más, de forma que crees una escala luminosa, con mucha luz pueden estar todos apagados y a medida que disminuye la intensidad luminosa se van encendiendo más leds.
2. En la propuesta anterior, haz que los leds parpadeen cuando la intensidad luminosa es muy baja.



## Ejercicio 5.- Señales PWM

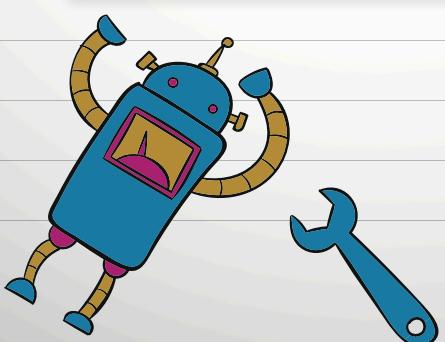
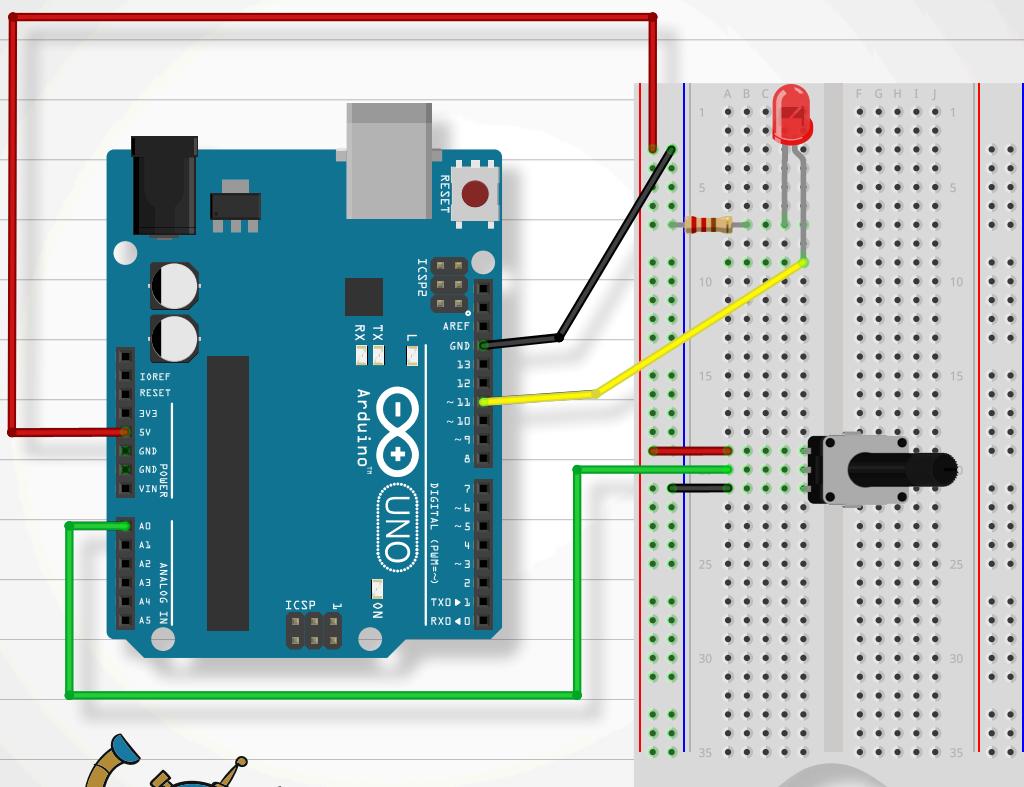
Vamos a controlar la luminosidad de un LED enviando una señal PWM que varía en función de una lectura analógica de un potenciómetro.



### Montaje

Debemos conectar un potenciómetro a una entrada analógica y realizar la lectura según la posición del cursor. Los valores leidos irán desde 0 a 1023.

Conectaremos un LED con su resistencia de protección a una salida digital PWM (~). Enviaremos un valor entre 0 y 255, obteniendo diferentes niveles de brillo.



## Código

```

5-PWM | Arduino 1.8.19
5-PWM
// Declaración de variables

int pot = 0;           // Variable que lee el valor del potenciómetro
int brillo = 0;         // Variable que enviamos a un pin pwm
int led = 11;          // Establece al pin del led

// configuración

void setup() {
  Serial.begin(9600);      // Inicia la comunicación serie
  pinMode(led, OUTPUT);    // Configuramos el led como salida
}

void loop() {
  pot = analogRead(A0);      // Lee el valor de la entrada analógica (entre 0 y 1023)
  brillo = map(pot,0,1023,0,255); // Calculamos el valor correspondiente pwm (entre 0 y 255)

  /* map() Transforma un valor comprendido entre un máximo e un mínimo en otro
   * valor comprendido entre otro máximo e otro mínimo */
  analogWrite(led,brillo);    // Enviamos el valor del brillo del led,
                             // enviando una señal de salida PWM (entre 0 y 255)

  Serial.print(pot);
  Serial.print ("---->");
  Serial.println(brillo);
  delay(200);
}

```

Arduino Uno on /dev/cu.usbmodem31301

Y ahora tú.....

1. Simula mediante un LED el efecto de fuego. Puedes generar un brillo aleatorio y un tiempo de espera aleatorio para un LED. Puedes utilizar el operador **random**, tanto para el brillo como para el tiempo. **brillo = random(20,255);**
2. Haz un programa en el que un LED parpadee a diferente frecuencia en función de la posición del potenciómetro. Por ejemplo, para un valor 0 en el potenciómetro el parpadeo se realiza con 100 ms de intervalo y para un valor de 1023 del potenciómetro el intervalo es de 1000 ms. Utiliza la función **map** para transformar el intervalo analógico en el intervalo de tiempo.



## Ejercicio 6.- Bucles

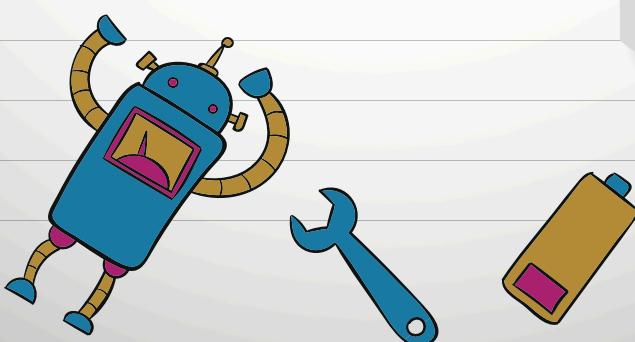
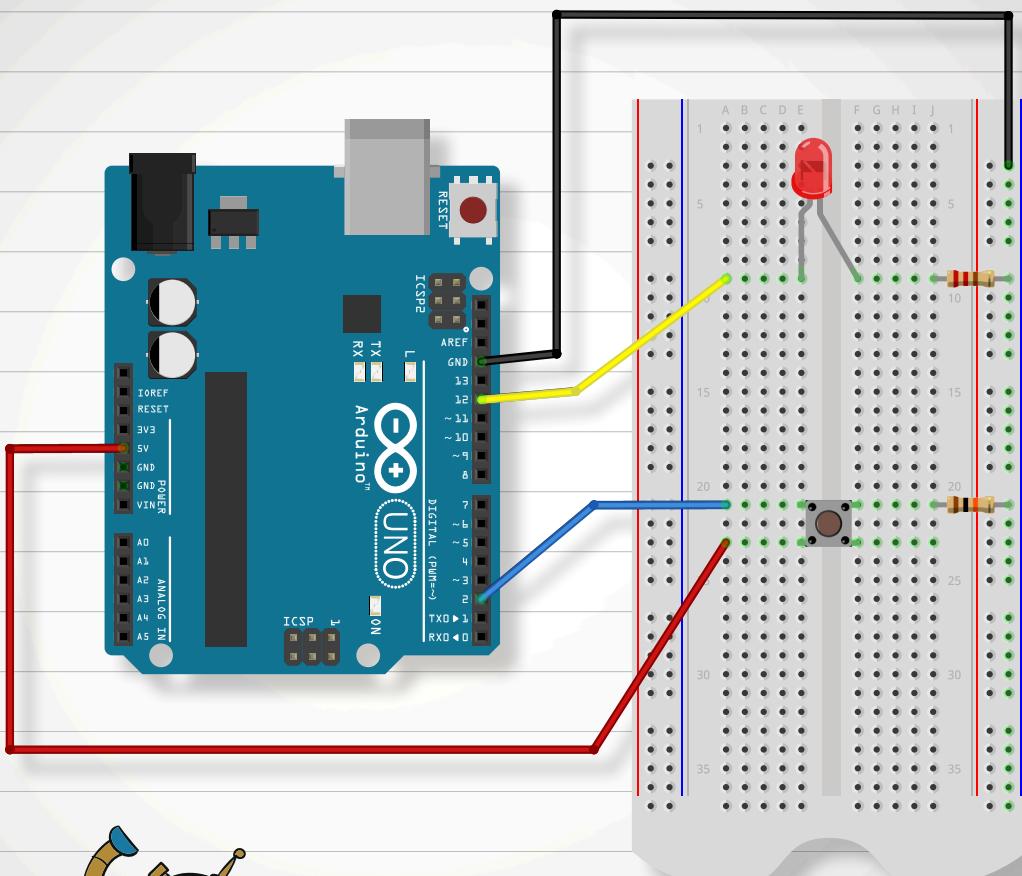
En este ejemplo haremos que unas instrucciones se repitan un número determinado de veces.

Por ejemplo, haremos que un LED se encienda y apague varias veces, esperamos un tiempo y repetimos el proceso.

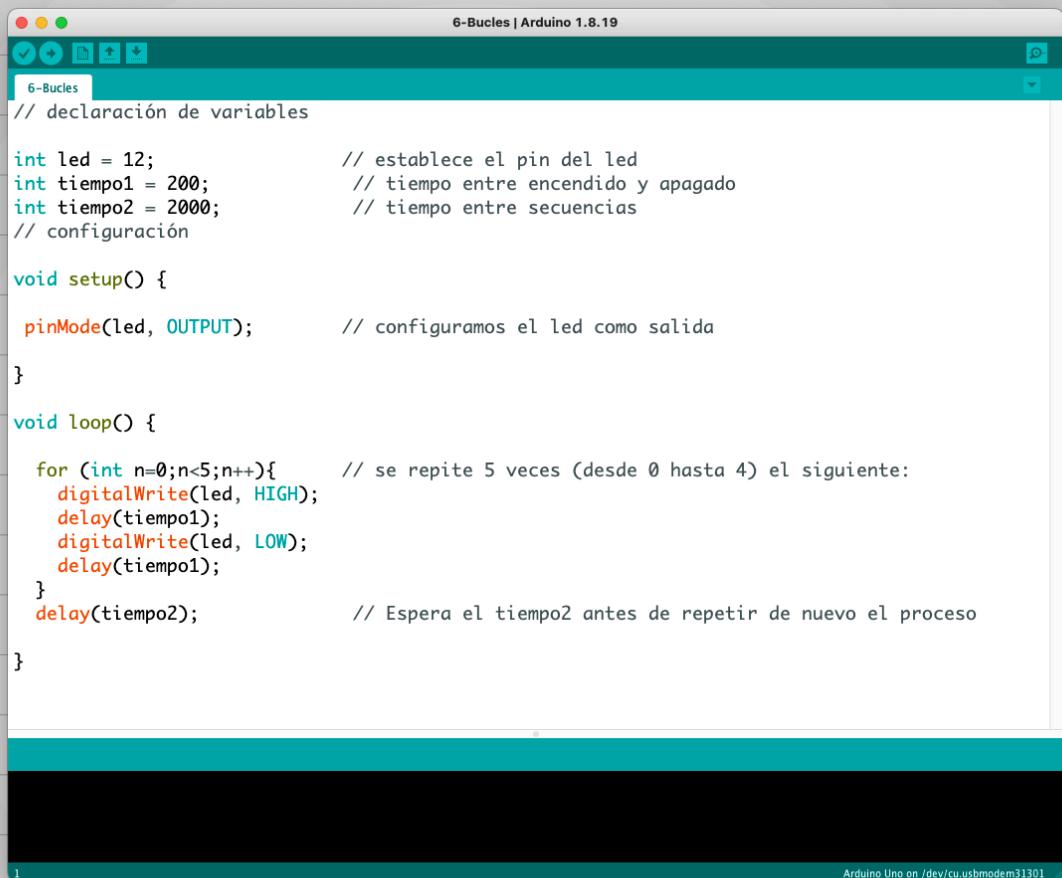
### Montaje

Necesitamos un LED y un pulsador con sus resistencias de protección. Para este caso utilizamos la función:

```
for (inicio,condición, modificador) { proceso }
```



## Código



```
6-Bucles | Arduino 1.8.19

// declaración de variables

int led = 12;           // establece el pin del led
int tiempo1 = 200;       // tiempo entre encendido y apagado
int tiempo2 = 2000;      // tiempo entre secuencias
// configuración

void setup() {
    pinMode(led, OUTPUT); // configuramos el led como salida
}

void loop() {
    for (int n=0;n<5;n++){ // se repite 5 veces (desde 0 hasta 4) el siguiente:
        digitalWrite(led, HIGH);
        delay(tiempo1);
        digitalWrite(led, LOW);
        delay(tiempo1);
    }
    delay(tiempo2); // Espera el tiempo2 antes de repetir de nuevo el proceso
}
```

Arduino Uno on /dev/cu.usbmodem31301

Y ahora tú.....

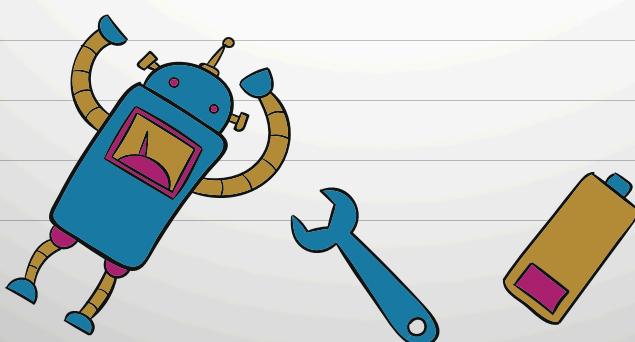
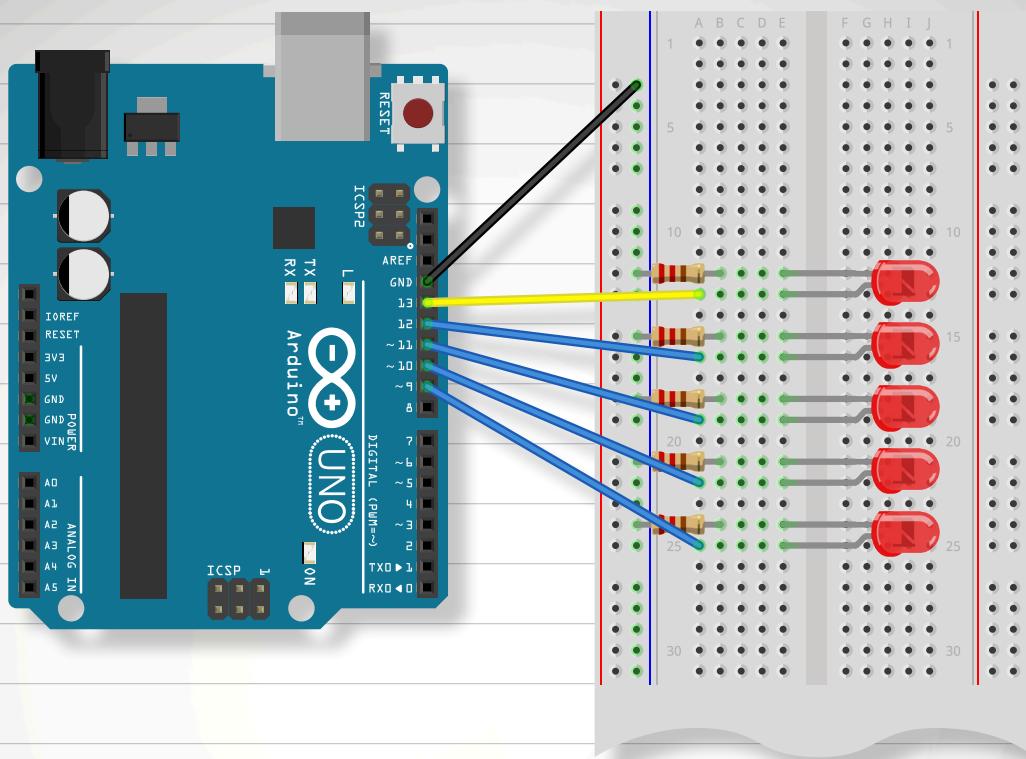
1. Conecta un segundo LED y haz que en cada ciclo el primer LED parpadee 6 veces y el segundo LED tres veces.
2. Haz que cada vez que presionamos un pulsador un LED se encienda e apague tres veces. Cuando el pulsador no está activado el LED permanece apagado.

## Ejercicio 7.- El coche fantástico

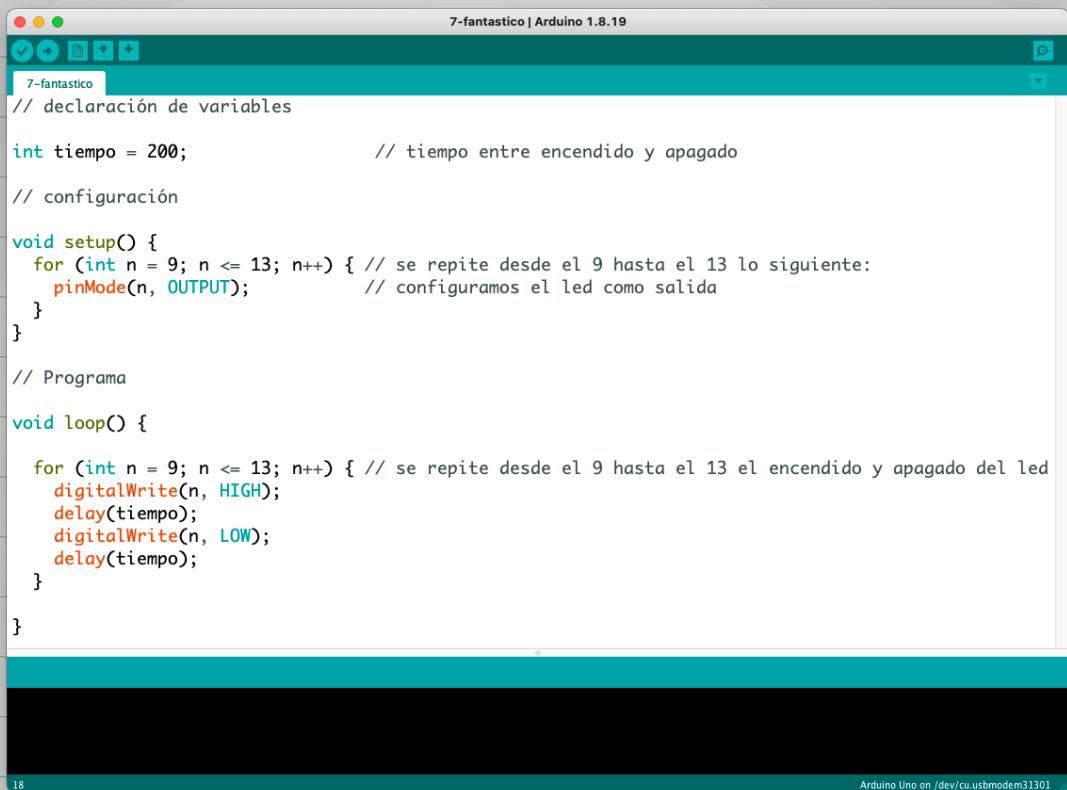
Simularemos el coche fantástico ordenando que se enciendan y se apaguen una serie de LEDs uno a uno de forma consecutiva.

### Montaje

Debemos conectar 5 LEDs con sus resistencias de protección en los pines del 9 al 13.



## Código



```
7-fantastico | Arduino 1.8.19
7-fantastico
// declaración de variables

int tiempo = 200; // tiempo entre encendido y apagado

// configuración

void setup() {
  for (int n = 9; n <= 13; n++) { // se repite desde el 9 hasta el 13 lo siguiente:
    pinMode(n, OUTPUT); // configuramos el led como salida
  }
}

// Programa

void loop() {

  for (int n = 9; n <= 13; n++) { // se repite desde el 9 hasta el 13 el encendido y apagado del led
    digitalWrite(n, HIGH);
    delay(tiempo);
    digitalWrite(n, LOW);
    delay(tiempo);
  }
}

18
Arduino Uno on /dev/cu.usbmodem31301
```

Y ahora tú.....

1. Añade un bucle descendiente al programa anterior de forma que la secuencia se produzca primero en un sentido y despues en el otro.
2. Haz que los LEDs se vayan encendiendo consecutivamente sin apagarse y, despues, que se vayan apagando de uno en uno, en el mismo sentido o en el contrario.
3. Podemos controlar la velocidad de la secuencia con un potenciómetro conectado a una entrada analógica. Utiliza la función `map` para transformar el rango analógico (de 0 a 1023) a un rango de tiempos (por ejemplo de 50ms a 1000ms).



Patrocinado por:



Desarrollado por:



Colaboran



[www.amiguslabs.org](http://www.amiguslabs.org)

