Taller
Robótica Libre con Arduino

Desarrollo del pensamiento computacional a través de la programación y la robótica. UIMP. Julio de 2017.

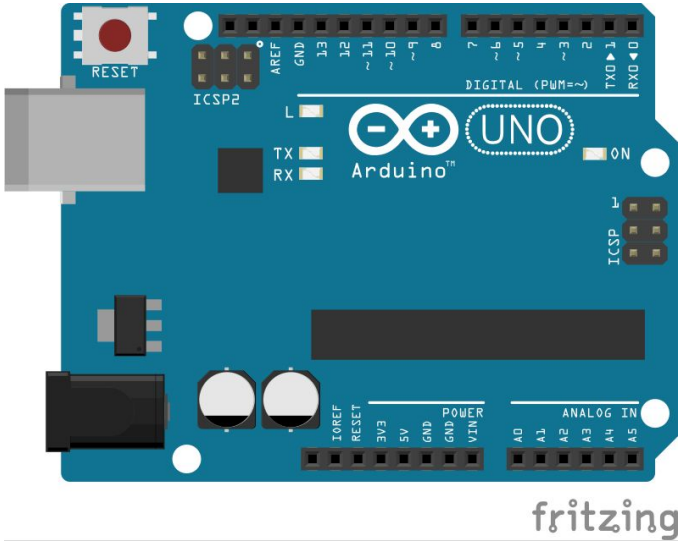
María Loureiro
@tecnoloxia
tecnoloxia.org

José Pujol
@jo_pujol
tecnopujol.wordpress.com

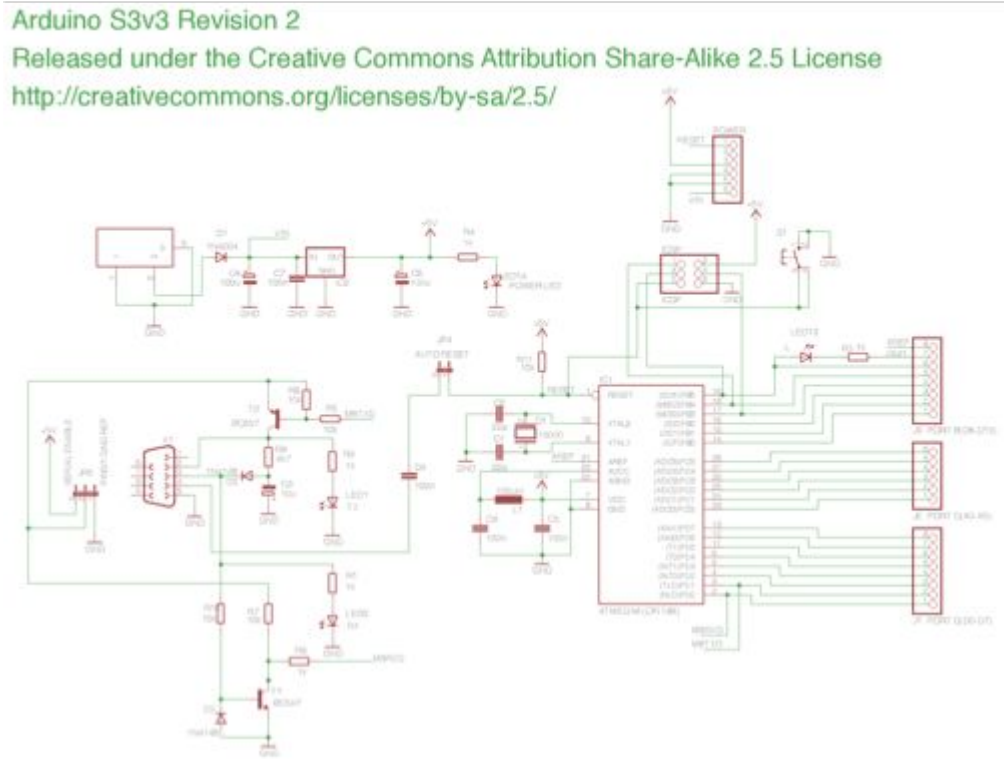
Arduino y el Open Source

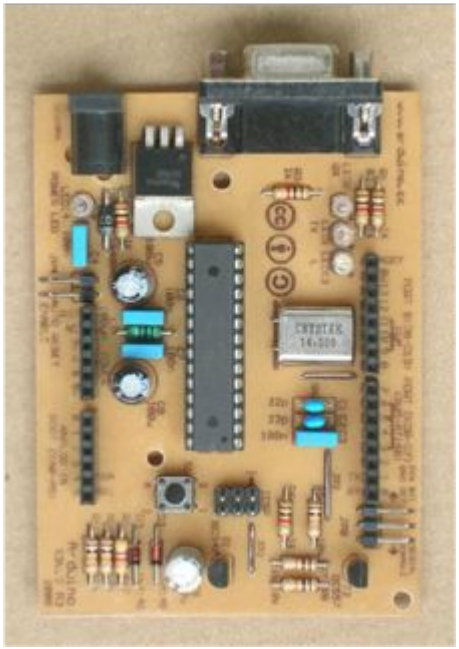
Arduino

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

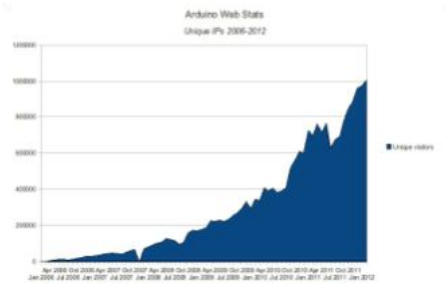


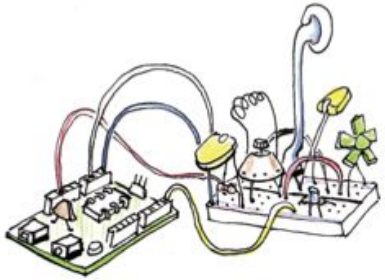
Qué es el hardware libre





Motivos para trabajar con Open Source





Introducción a Arduino

Arduino

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

Página oficial: http://arduino.cc/

[SparkFunElectronics, CC BY]

Descripción de la placa Arduino UNO

Arduino UNO Pinout (pdf)

Entornos de programación

- Visuales
- IDE

Entornos Visuales vs Entornos Escritos


```
// Declaración de variables:

int led  = 12;                  // Led  conectado en el pin 12

// 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 (12) (enciende)
  delay(500);                   // espera 500ms = 0,5s
  digitalWrite(led, LOW);       // Envía 0V (apaga)
  delay(100);                   // espera 100ms = 0,1s

}
```

Entornos Visuales

+

- Curva de aprendizaje rápida
- Permite centrarse en los algoritmos sin pensar en la sintaxis.

-

- Cierta limitación a la hora de programar

Entornos Escritos

+

- Mayor libertad programación
- Potencial del uso librerías
- Portabilidad del código

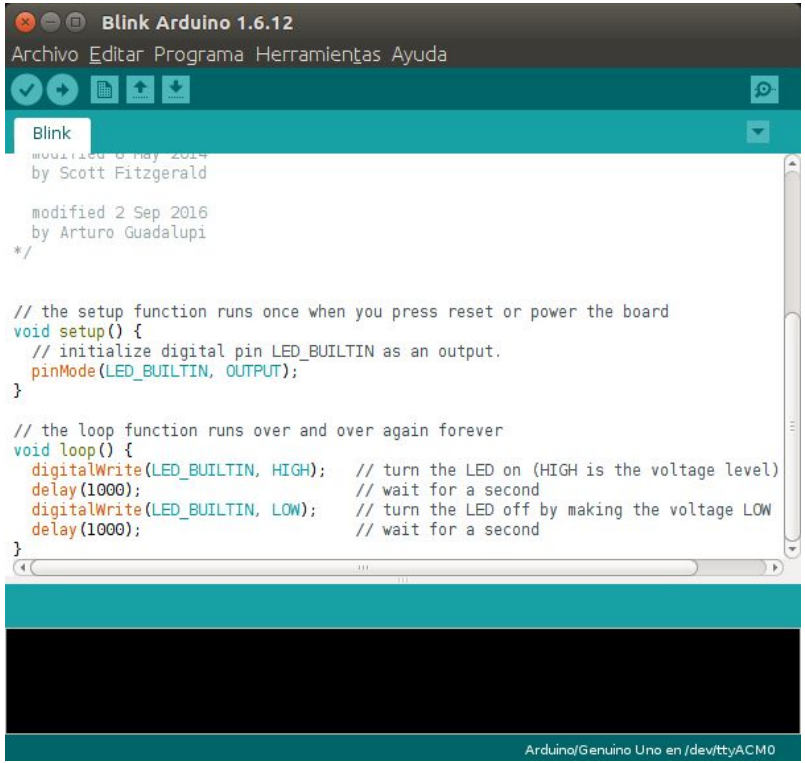
-

- Errores sintaxis del lenguaje
- Curva aprendizaje lenta

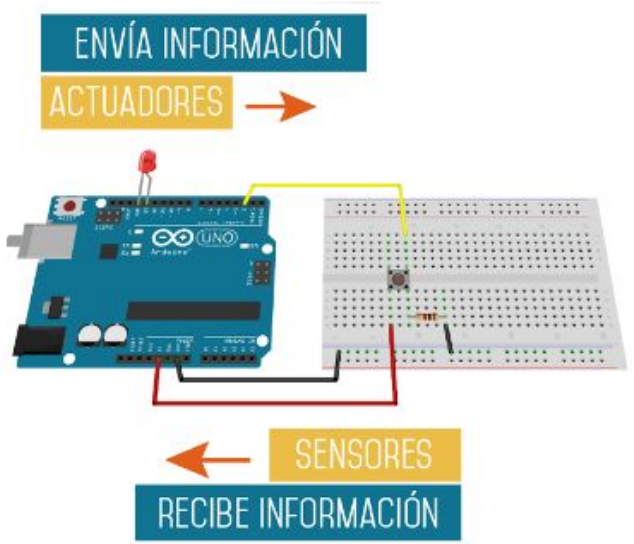
Comparativa de entornos visuales de programación

Makeblock

IDE de Arduino

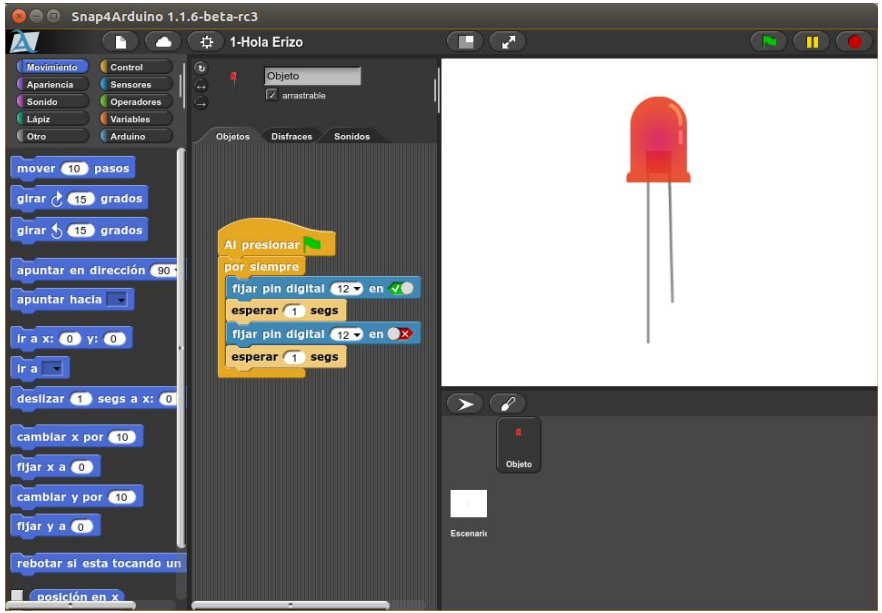


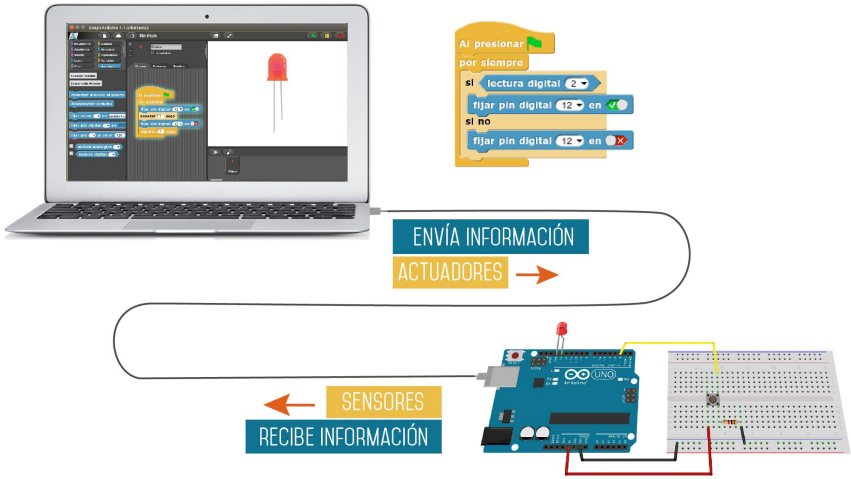
Los programas se cargan en la placa



Snap4Arduino

Los programas se ejecutan en el ordenador.

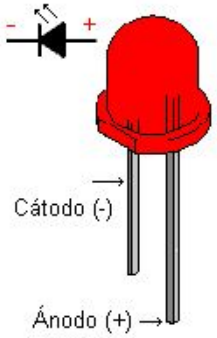




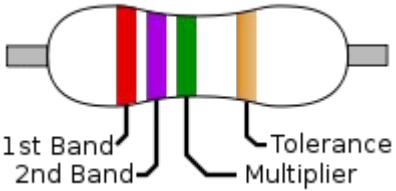
Prácticas

Material

LEDs

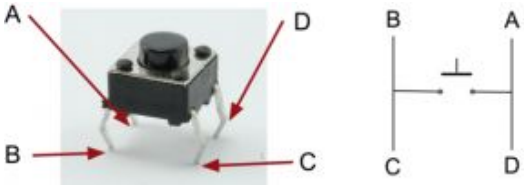


Resistencias fijas

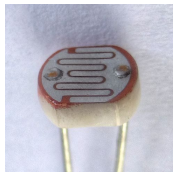


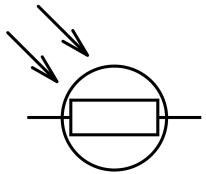
[Código de colores de las resistencias]

Pulsador

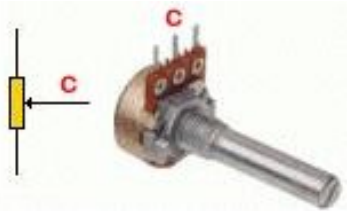


Fotorresistencia LDR

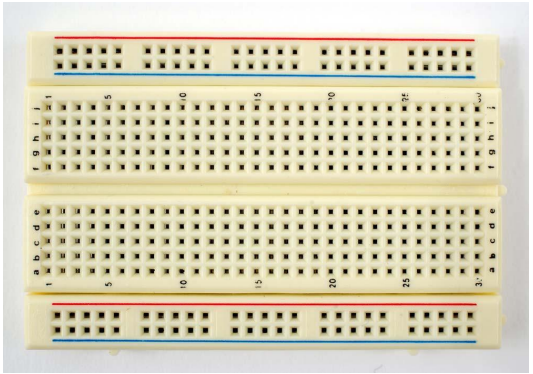


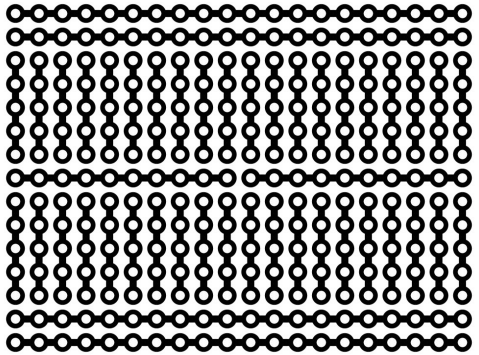


Potenciómetro



Protoboard





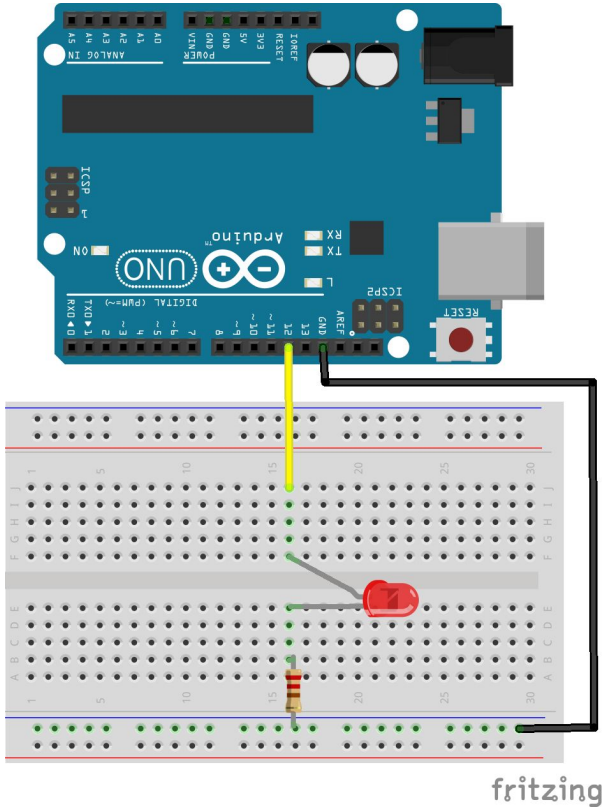
1. Blink (Hola mundo)

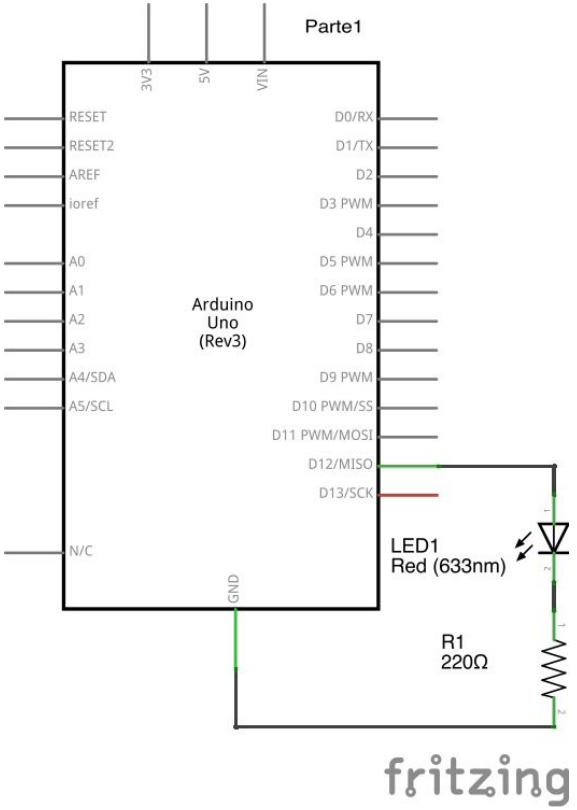
Finalidad

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

Hardware

Led en el pin 12 con resistencia de protección de 220Ω o 270Ω





1-A Blink
(Hola mundo Arduino)

Configuración

1 FILE>
EXAMPLES>
BASICS>
BLINK

2 TOOLS>
BOARD>
ARDUINO UNO

3 TOOLS>
SERIAL PORT
Windows OS COM####
Mac OS /dev/ttyusbmodem####
Linux OS /dev/ttyACM####

4 FILE>
UPLOAD

[Imagen CTC Arduino Verkstad]

Código

```
// Declaración de variables:

int led  = 12;                 // Led  conectado en el pin 12

// 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 (12) (enciende)
  delay(500);                  // espera 500ms = 0,5s
  digitalWrite(led, LOW);      // Envía 0V (apaga)
  delay(100);                  // espera 100ms = 0,1s

}
```

Propuestas

1. Prueba a cambiar el tiempo de parpadeo
2. Prueba a hacer que el LED siga el latido del corazón
3. ¿Podrías hacer que el LED parpadeara una sola vez?
4. Usa variables para definir el tiempo de parpadeo
5. Cual es el límite de la percepción humana? a partir de que tiempo de parpadeo el ojo humano deja de percibirlo

1-B. Blink
(Hola mundo Snap4Arduino)

Configuración

Snap4Arduino requiere que tengas instalado StandardFirmata en la placa Arduino. Para ello, en el IDE de Arduino abrir:

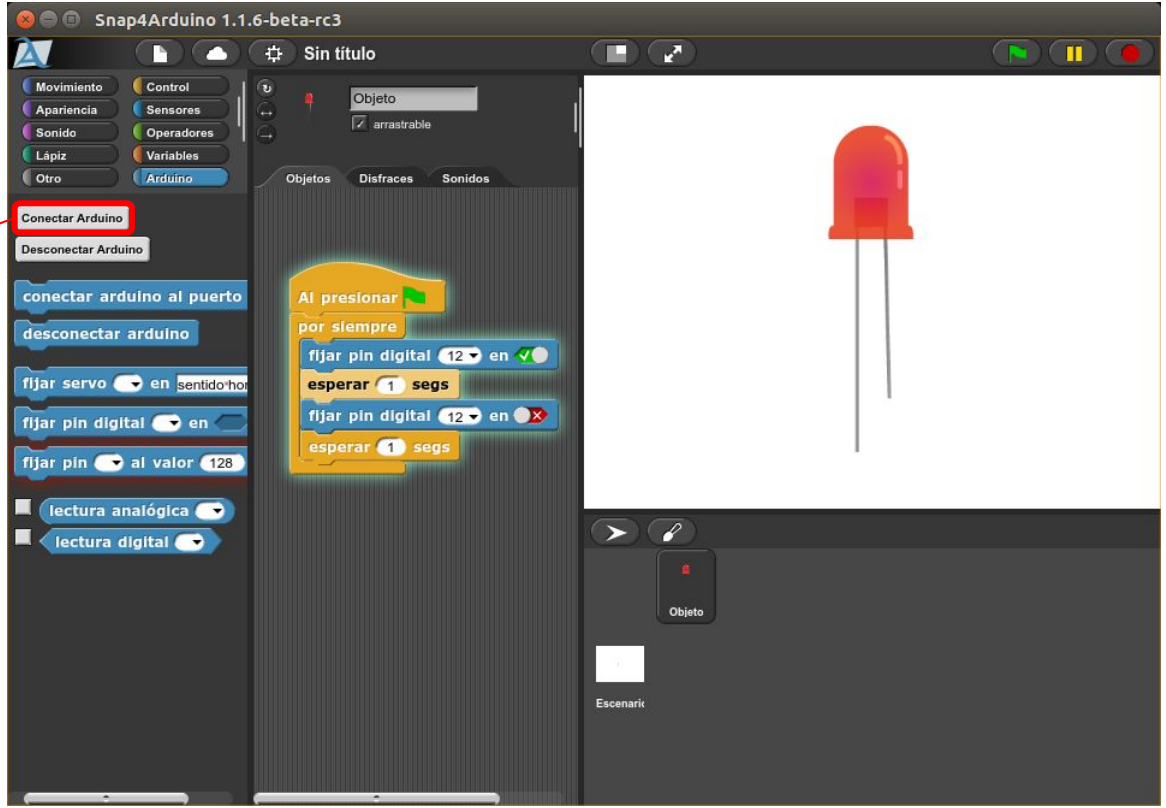
IDE Arduino

Ejemplos → Firmata → StandardFirmata

y cargar el código a la placa

Configuración

Conectar Arduino



Guardar archivos:

- Guardar online (tenemos que registrarnos)
- Guardar en local: exportar como xml y luego importar

Código Snap4Arduino

Propuestas

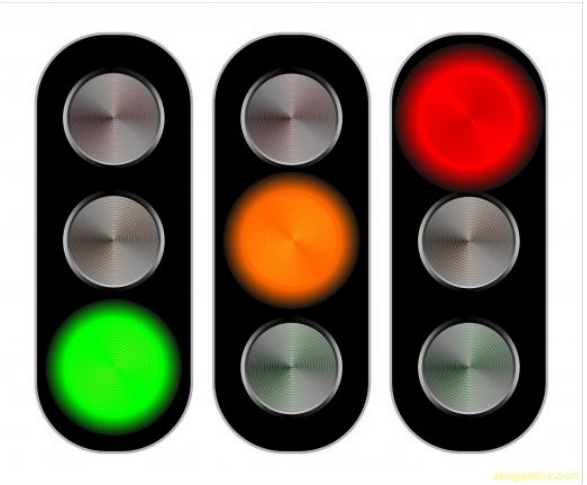
1. Cambiar el tiempo de parpadeo
2. Añadir un led virtual que parpadee al mismo ritmo

3. Controlar el encendido del LED con el teclado del ordenador

2. Semáforo

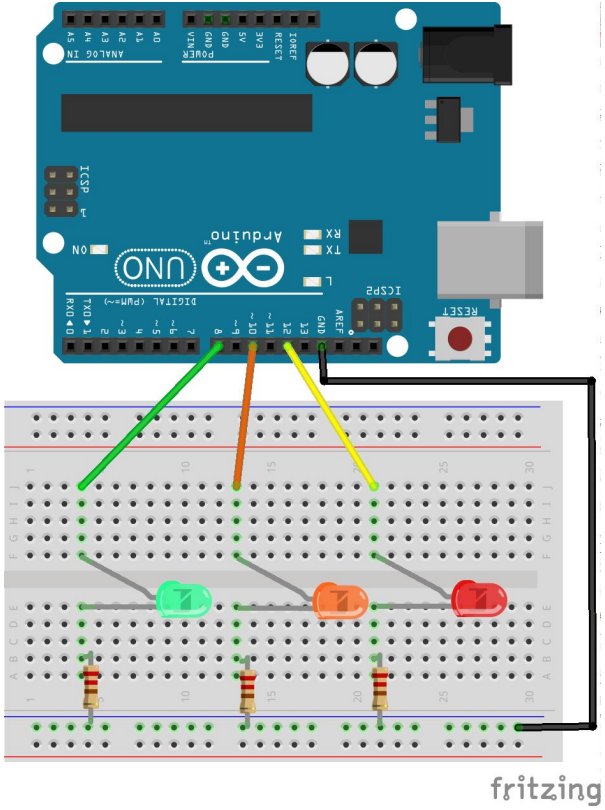
Finalidad

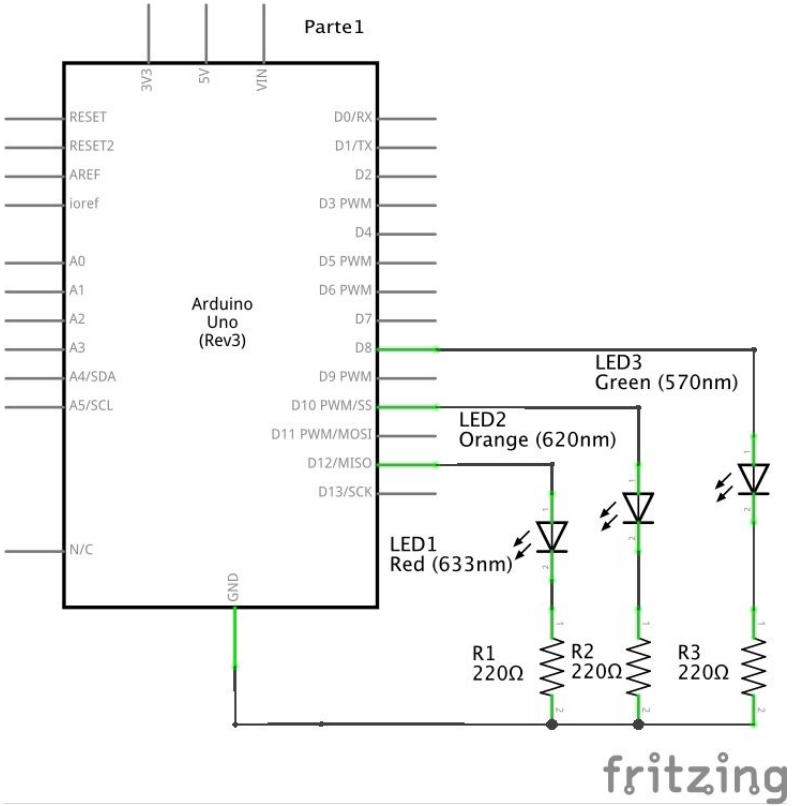
Vamos a simular un semáforo de coches (rojo, amarillo y verde) que funcione de forma cíclica.



Hardware

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





Código para completar

```
// Declaracion de variables de tipo constante entero
const int ledRPin = 12; // LED rojo asociado al pin 12
const int ledOPin = 10; // LED naranja asociado al pin 10
const int ledGPin = 8; // LED verde asociado al pin 8

// la funcion set up se ejecuta una sola vez
void setup() {
  // inicializa los pines de los leds como salidas digitales
  pinMode(ledGPin, OUTPUT);
  pinMode(ledOPin, OUTPUT);
  pinMode(ledRPin, OUTPUT);
}

// la funcion loop se ejecuta repetidamente de forma infinita
void loop() {
  // estado semaforo verde
  digitalWrite(ledGPin, HIGH); // escribe 5v en el pin del LED verde
  digitalWrite(ledOPin, LOW); // escribe 0v en el pin del LED naranja
  digitalWrite(ledRPin, LOW); // escribe 0v en el pin del LED rojo
  delay(10000);    // espera diez segundos

  // estado semaforo amarillo

  // estado semaforo rojo

}
```

En el siguiente código sólo se dan las instrucciones de encendido del led verde. Completa la secuencia.

Propuestas

1. Si estás usando Snap4Arduino crea un semáforo virtual
2. Añade un semáforo de peatones (leds rojo y verde) sincronizado con el de coches

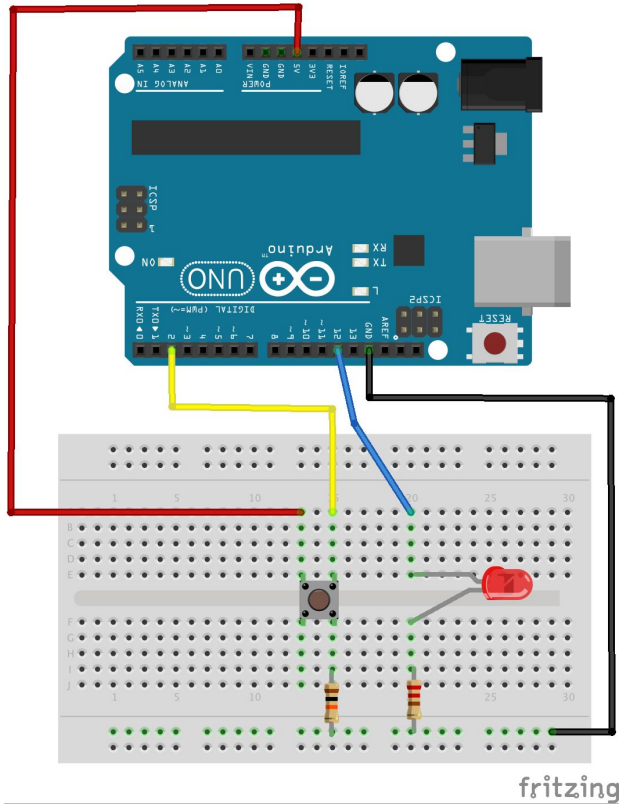
3. Pulsador y Led

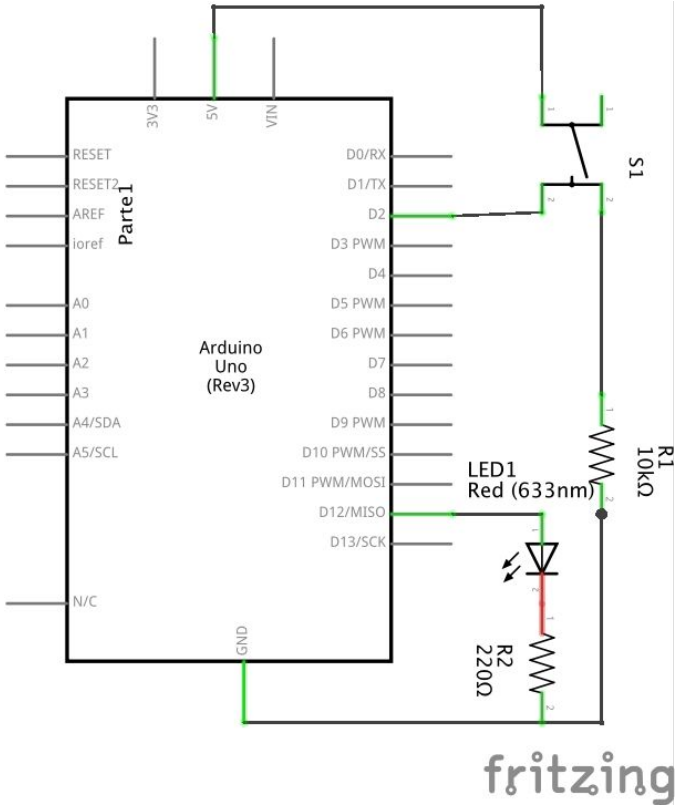
Finalidad

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

Hardware

Conectar un pulsador al pin 2 con una resistencia de 10kΩ o 4,7kΩ tal y como aparece en el esquema





Código

```
/* 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 (para las entradas no hace falta)

}

// Bucle:

void loop(){

  estado = digitalRead(pulsador); // Leemos el estado del pulsador (0 o 1) y asignamos el valor a la variable "estado"

  if (estado == 1){               // Si el pulsador está activado (estado = 1)
    digitalWrite(led, HIGH);      // Se enciende el LED
  }
  else{                           // Si no
    digitalWrite(led, LOW);       // Se apaga el LED
  }

}
```

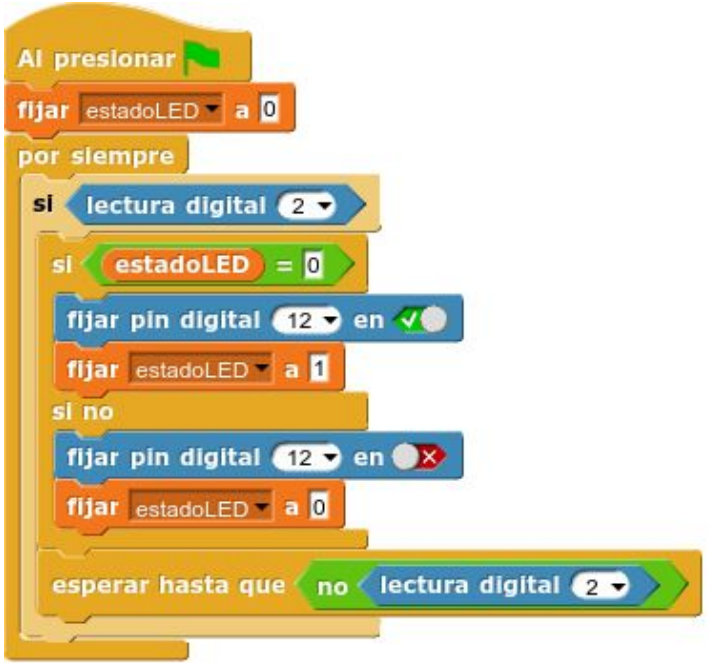
Propuestas

1. Añade otro LED y haz que se enciendan 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. Propuesta para pulsador con memoria

Vamos a añadir memoria al pulsador de forma que recuerde el estado del LED y se comporte como un interruptor. As presionar el pulsador:

- Si inicialmente el LED estaba encendido, se apaga
- Si estaba apagado se enciende



Realiza el programa con el IDE de Arduino

5. Interruptor de luz

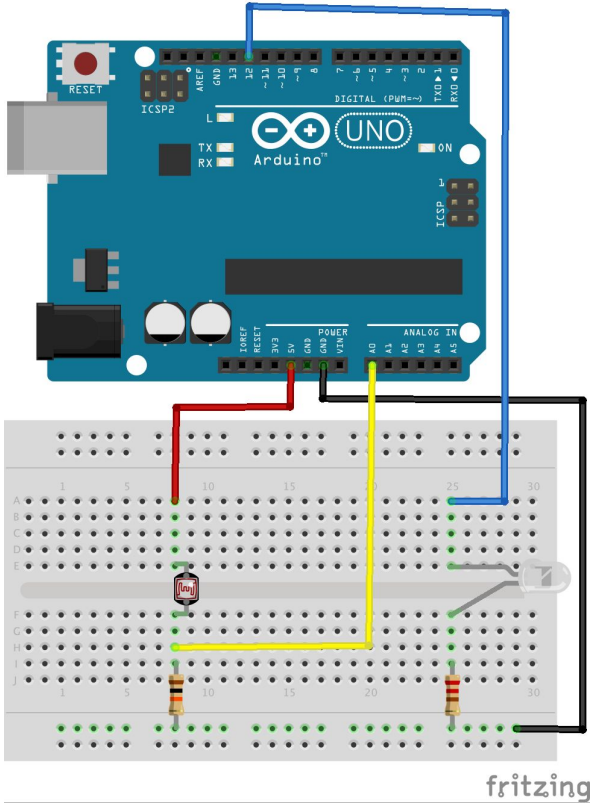
Finalidad

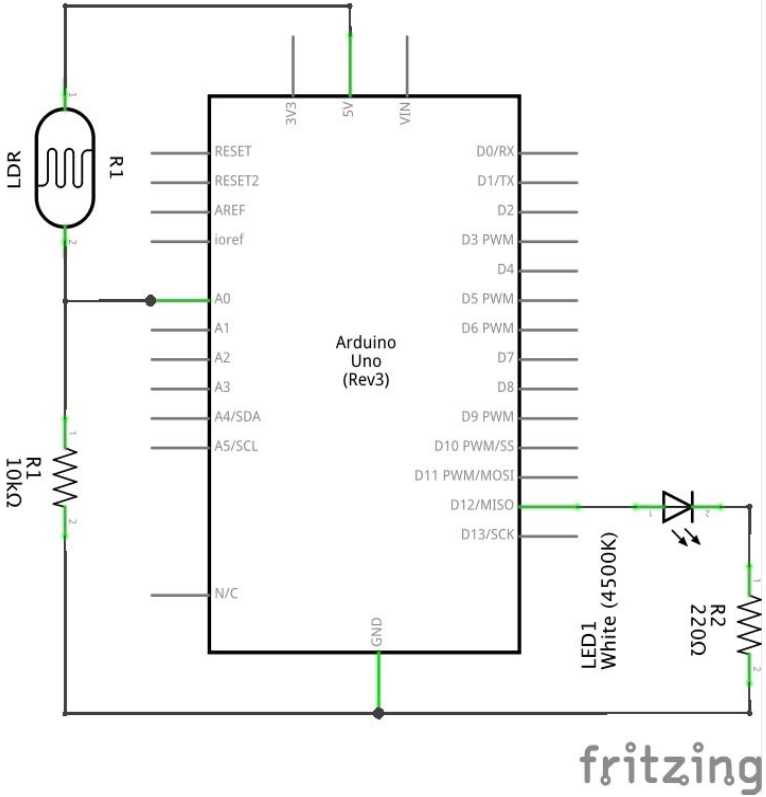
Encender un LED en función de la intensidad luminosa que recibe el sensor de luz. En oscuridad el LED está encendido y con alta iluminación apagado.

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

Hardware

Conectar una LDR al pin analógico A0 con una resistencia de 10kΩ o 4,7kΩ tal y como aparece en el esquema





Código: lectura de sensores

```
const int ldrPin = A0;     // establece el pin de la LDR
int ldrValue = 0;  // variable para almacenar el valor del

void setup() {
  // abre el puerto serie
  // y establece la velocidad de conexion en baudios
  Serial.begin(9600);
}

void loop() {
  // lee el valor del sensor
  ldrValue = analogRead(ldrPin);
  // Imprime un texto
  Serial.print("Valor LDR=");
  // Imprime el valor de la variable por el puerto serie
  Serial.println(ldrValue);
  // tiempo de espera para visibilidad
  // y para no saturar el puerto serie
  delay(1000);
}
```

Para ver el valor de la LDR hay que abrir el monitor serie

Código del interruptor de luz

```
// variables constantes para los pines
const int ldrPin = A0;    // establece el pin de la LDR
const int ledPin = 12; // establece el pin del LED

// variable para almacenar el valor del sensor
int ldrValue = 0;

void setup() {
  // establece LED como una salida
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // lee el valor del sensor
  ldrValue = analogRead(ldrPin);
  // si el valor es menor enciende los LEDs
  if (ldrValue < 200) {
    digitalWrite(ledPin, HIGH);
  }
  // sino los apaga
  else {
    digitalWrite(ledPin, LOW);
  }
}
```

Propuestas

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

```
// Si el valor ldr esta entre 200 y 400
if (ldrValue < 200 && ldrValue < 400) {
  digitalWrite(led1, HIGH);   // enciende led 1
  digitalWrite(led2, LOW);    // apaga led 2
  digitalWrite(led3, LOW);    // apaga led 3
}
```

2. En la propuesta anterior, haz que los leds parpadeen cuando la intensidad luminosa es muy baja.

6. Señales PWM

Finalidad

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

[tecnoloxia.org CC By-SA]

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

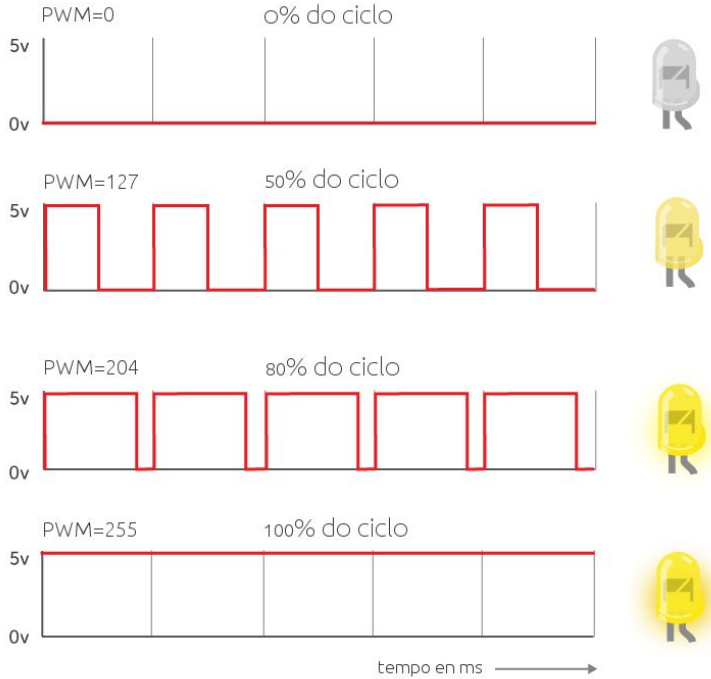
Conectaremos un **LED** con su resistencia de protección a una salida digital **PWM** (~). Le enviaremos un valor entre **0** y **255**, obteniendo diferentes niveles de brillo.

Señales PWM

Para regular la potencia de salida de una señal digital utilizamos señales digitales modulados en anchura (**PWM** = Pulse Width Modulation)

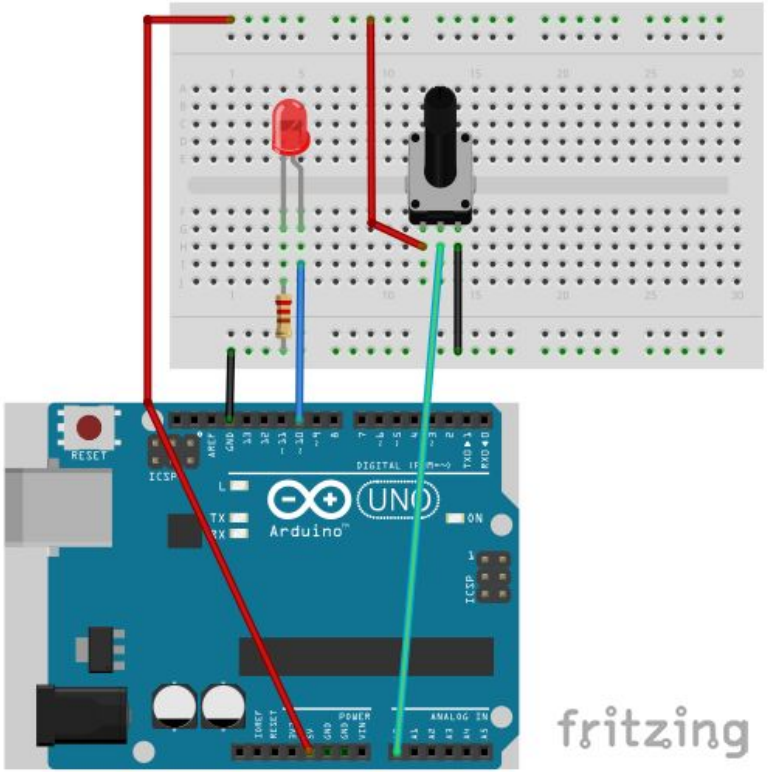
El valor PWM varía entre **0** y **255**.

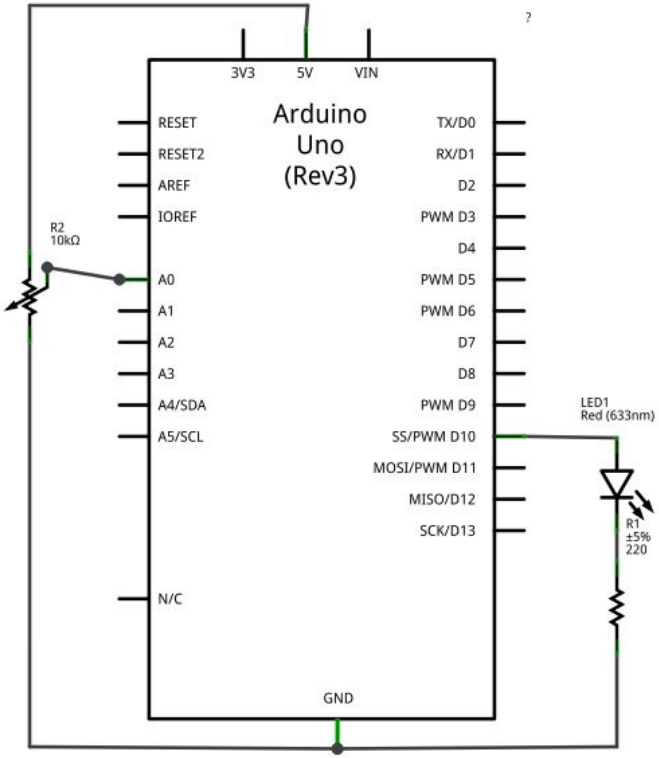
En la tarjeta Arduino UNO se pueden usar como salida PWM los pines marcados con el símbolo **~** (3, 5, 6, 9, 10, 11)



Hardware

Conectamos un LED y un potenciómetro como se muestra en el siguiente esquema





Funciones

Lectura analógica: Lee el valor de una entrada analógica. Puede tomar valores entre 0 e 1023. *analogRead(pin);*

Ejemplos: *pot= analogRead(A0);*

Escritura PWM: Envía una señal de salida PWM al pin indicado, con un valor comprendido entre 0 y 255. *analogWrite(pin, valor)*

Ejemplos: *analogWrite(led, 127);* *analogWrite(led, brillo);*

Función map: Transforma un valor comprendido entre un máximo y un mínimo en otro valor comprendido entre otro máximo y otro mínimo *map(valor,min,max,Nmin,Nmax);*

Ejemplos: *brillo=map(pot,0,1023,0,255);*

Código

```
// 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=10;     // LED conectado al pin pwm ~10

// Configuración

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

// Programa

void loop() {

  pot = analogRead(A0);             // Asignamos a la variable "pot" el valor leido en A0 (entre 0 y 1023)
  brillo = map(pot,0,1023,0,255);   // Calculamos el valor correspondente pwm (entre 0 y 255)

  analogWrite(led,brillo);          // Enviamos el valor brillo al LED

  Serial.print(pot);                // Imprimimos los valores en la consola
  Serial.print(" ----> ");
  Serial.println(brillo);
  delay(200);

}
```

Para importar la función map en Snap4Arduino descargar map.zip, descomprimir e importar. Aparecerá en el menú de operadores.

Propuestas

- 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.

  Por ejemplo: *brillo = random(20,255);*

- 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.

7. Bucles

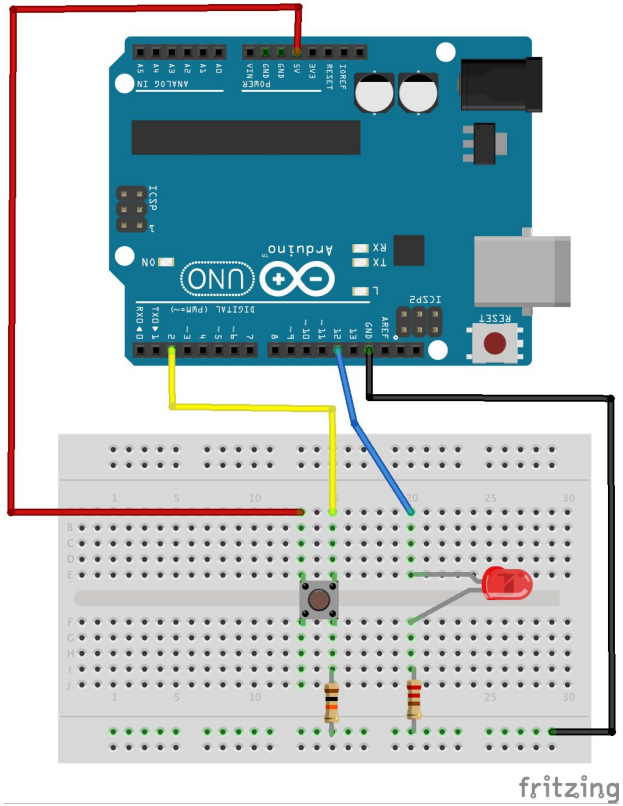
Finalidad

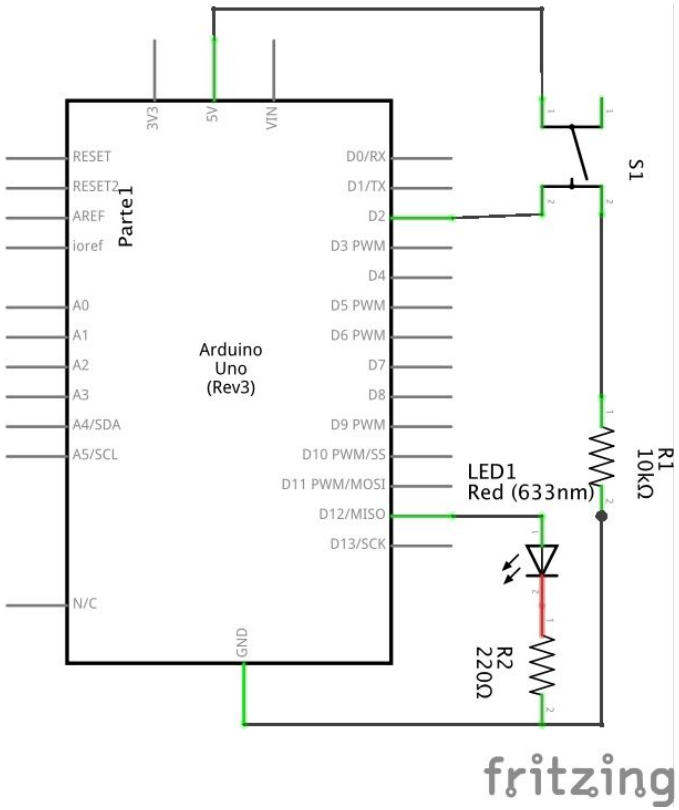
Haremos que unas instrucciones se repitan un número determinado de veces.

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

Hardware

Necesitamos un LED y un pulsador con sus resistencias de protección.





Bucles

Ya conocemos un bucle, que es el representado por la función ***loop { proceso }***. Esta función hace que las instrucciones que se encuentran entre los paréntesis se ejecuten indefinidamente, mientras la placa está conectada.

En ocasiones necesitamos que se repitan una serie de instrucciones un número determinado de veces. Para este caso utilizamos la función **for (inicio,condición,modificador) { proceso }**

Por ejemplo, si queremos que un proceso se repita 5 veces, escribimos *for(int a=0; a<5; a++) { // Proceso a repetir }*

Código

```
// Declaración de variables

int led = 13;
int tiempo1 = 200;          // tiempo entre acendido y apagado
int tiempo2 = 2000;         // tiempo entre secuencias

// Configuración

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

// Programa

void loop() {

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

}
```

El LED parpadea 5 veces, espera un tiempo y vuelve a repetir el proceso.

Propuestas

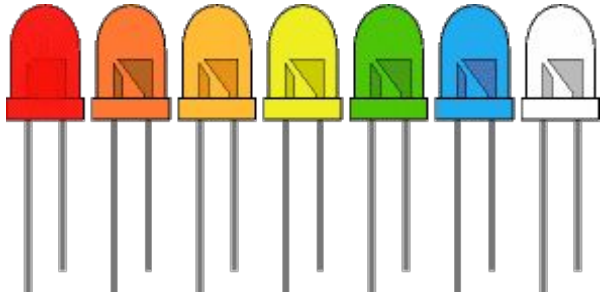
- Conecta un segundo LED y haz que en cada ciclo el primer LED parpadee 6 veces y el segundo LED tres veces.
- Haz que cada vez que presionamos un pulsador un LED se encienda y se apague tres veces. Cuando el pulsador no está activado el LED permanece apagado.

8. El coche fantástico

Finalidad

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

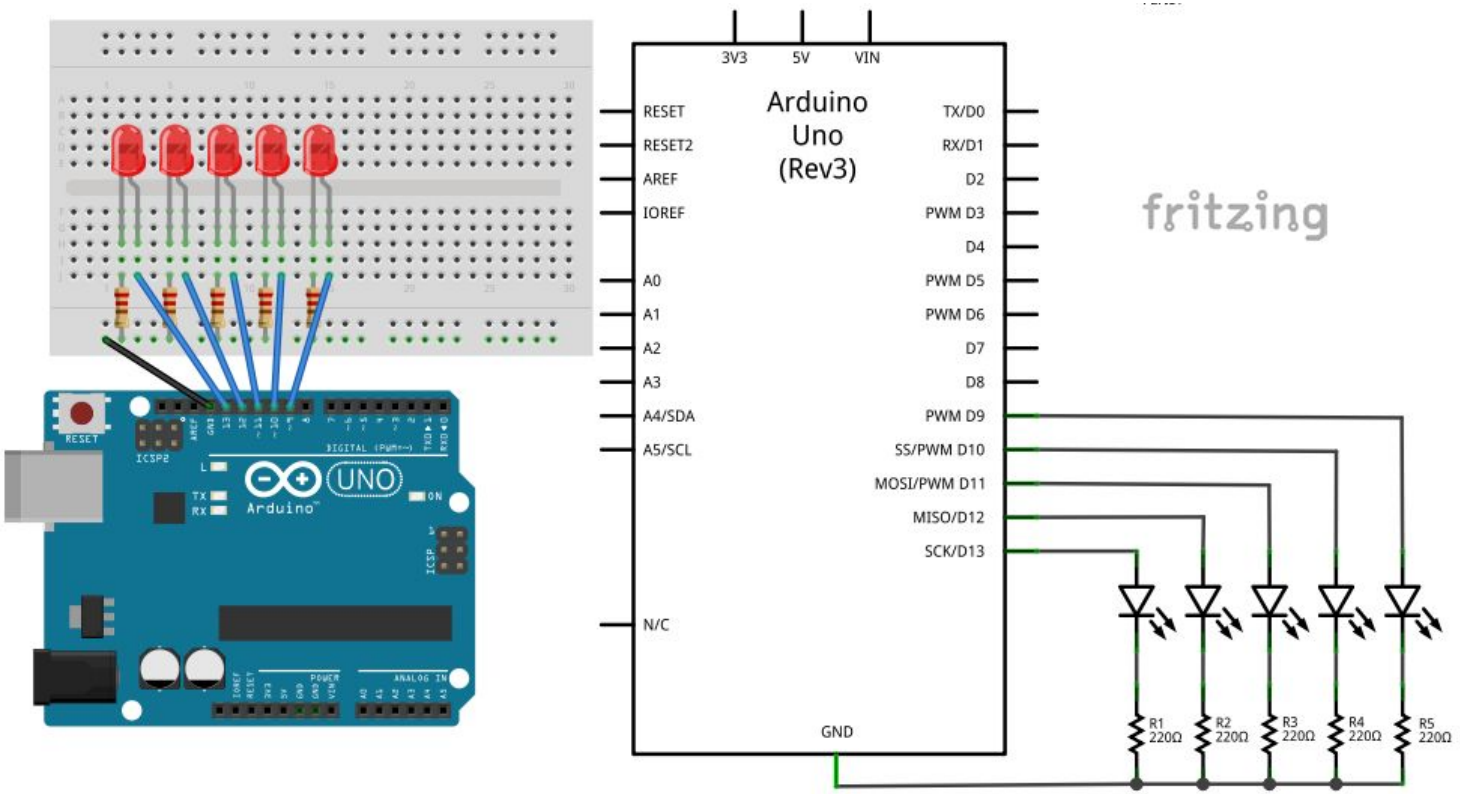
[by Fajar Abriono CC By-SA]



[WikiFisica2013, CC BySA]

Montaje

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



Código

```
// Declaración de variables:

int tiempo = 200;         // Definimos la variable tiempo con un valor de 200ms

// Configuración:

void setup() {
  for (int n=9;n<=13;n++) {  // Se repite desde o 9 hasta el 13 lo siguiente:
    pinMode(n, OUTPUT);      // vamos configurando los pines como salida
  }
}

// Programa

void loop() {

  for (int n=9;n<=13;n++) {  // Se repite desde 9 hasta el 13 el encendido y apagado de cada LED
    digitalWrite(n, HIGH);
    delay(tiempo);
    digitalWrite(n, LOW);
    delay(tiempo);
  }

}
```

Propuestas

- Añade un bucle descendiente al programa anterior de forma que la secuencia se produzca primero en un sentido y después en el otro.
- Ha que los LEDs se vayan encendiendo consecutivamente sin apagarse y, después, que se vayan apagando uno a uno, en el mismo sentido o en el contrario.
- 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 50 ms a 1000 ms)

Créditos

José Pujol y María Loureiro
CC By-SA