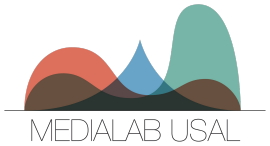
Básicos



Introducción

¿Que es Arduino?[1]

Arduino es una herramienta para hacer que los ordenadores puedan sentir y controlar el mundo físico a través de tu ordenador personal. Es una plataforma de desarrollo de computación física (physical computing) de código abierto, basada en una placa con un sencillo microcontrolador y un entorno de desarrollo para crear software (programas) para la placa.

Puedes usar Arduino para crear objetos interactivos, leyendo datos de una gran variedad de interruptores y sensores y controlar multitud de tipos de luces, motores y otros actuadores físicos. Los proyecto de Arduino pueden ser autónomos o comunicarse con un programa (software) que se ejecute en tu ordenador (ej. Flash, Processing, MaxMSP).La placa puedes montarla tu mismo o comprarla ya lista para usar, y el software de desarrollo es abierto y lo puedes descargar gratis.

El lenguaje de programación de Arduino es una implementación de Wiring, una plataforma de computación física parecida, que a su vez se basa en Processing, un entorno de programación multimedia.

¿Por qué Arduino?[2]

Hay muchos otros microcontroladores y plataformas con microcontroladores disponibles para la computación física. Parallax Basic Stamp, BX-24 de Netmedia, Phidgets, Handyboard del MIT, y muchos otros ofrecen funcionalidades similares. Todas estas herramientas organizan el complicado trabajo de programar un microcontrolador en paquetes fáciles de usar. Arduino, además de simplificar el proceso de trabajar con microcontroladores, ofrece algunas ventajas respecto a otros sistemas a profesores, estudiantes y amateurs:

Asequible

[1] http://arduino.cc/es

[2] http://arduino.cc/es

Las placas Arduino son más asequibles comparadas con otras plataformas de microcontroladores. La versión más cara de un módulo de Arduino puede ser montada a mano, e incluso ya montada cuesta bastante menos de 60€

Multi-Plataforma

El software de Arduino funciona en los sistemas operativos Windows, Macintosh OSX y Linux. La mayoría de los entornos para microcontroladores están limitados a Windows.

Entorno de programación simple y directo

El entorno de programación de Arduino es fácil de usar para principiantes y lo suficientemente flexible para los usuarios avanzados. Pensando en los profesores, Arduino está basado en el entorno de programación de Procesing con lo que el estudiante que aprenda a programar en este entorno se sentirá familiarizado con el entorno de desarrollo Arduino.

Software ampliable y de código abierto

El software Arduino esta publicado bajo una licencia libre y preparado para ser ampliado por programadores experimentados. El lenguaje puede ampliarse a través de librerías de C++, y si se está interesado en profundizar en los detalles técnicos, se puede dar el salto a la programación en el lenguaje AVR C en el que está basado. De igual modo se puede añadir directamente código en AVR C en tus programas si así lo deseas.

Hardware ampliable y de Código abierto

Arduino está basado en los microcontroladores ATMEGA168, ATMEGA328y ATMEGA1280. Los planos de los módulos están publicados bajo licencia Creative Commons, por lo que diseñadores de circuitos con experiencia pueden hacer su propia versión del módulo, ampliándolo u optimizándolo. Incluso usuarios relativamente inexpertos pueden construir la versión para placa de desarrollo para entender cómo funciona y ahorrar algo de dinero.

Componentes de la plataforma:

Software

SDK (Software Developer Kit): Escribir, compilar Sketches y cargarlos en el Hardware.

Hardware

Placa Arduino (diferentes versiones)

¿Que se hace con Arduino?:

Crear dispositivos que interactúen con el entorno.

Input → ARDUINO → Output

Desarrollo de computación física.

Experimentación con software y hardware

¿Para quién está dirigido Arduino?

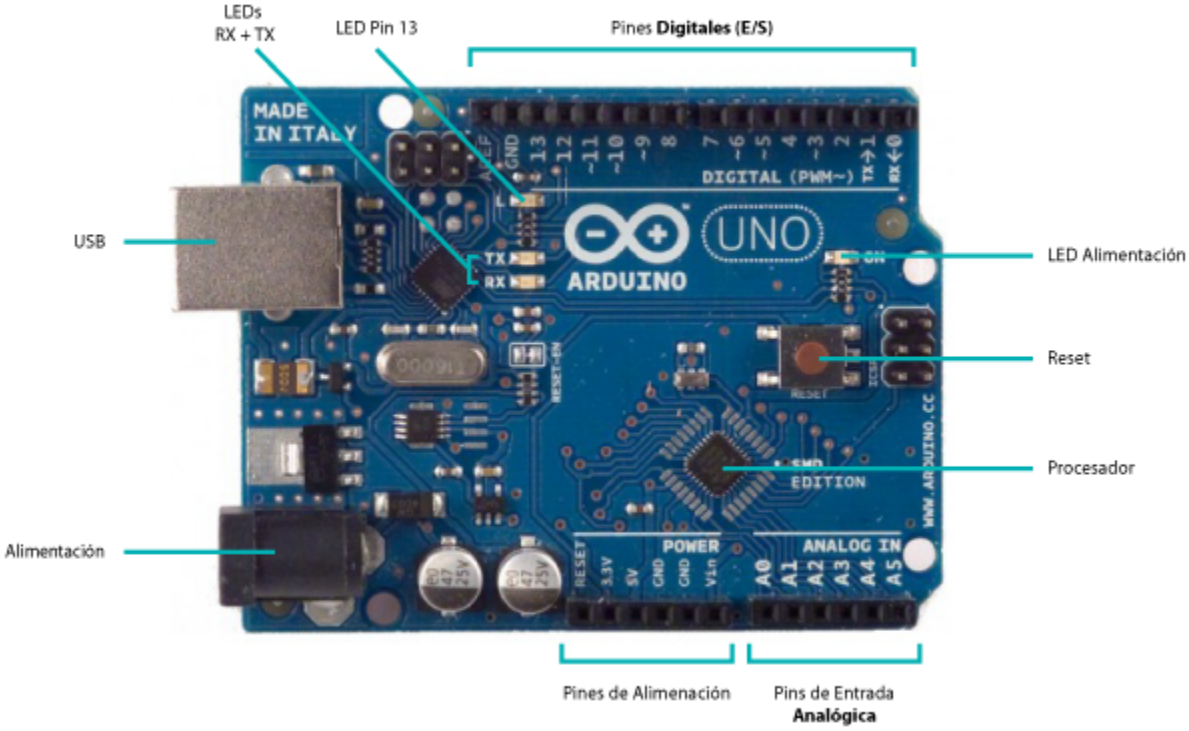
Para todos

- Aficionados - poco nivel o ninguna formación previa.
- Estudiantes - sector educación.
- Profesionales - Prototipados rápidos.

Lenguaje de programación de Arduino:

Desciende del lenguaje "wiring" (que desciende a su vez de processing)
Se basa en C/C++
Arduino = Processing

Componentes de Arduino



Primeros pasos con Arduino[3]

Instalación: instrucciones paso a paso para configurar el software de Arduino y conectarlo a un Arduino Duemilanove.

http://arduino.cc/es/Guide/HomePage

Entorno de Desarrollo para Arduino[4]

http://arduino.cc/es/Guide/Environment

Empezando a programar

Estructura

El código tiene 3 partes principales:

- La zona global
- La función **void setup()**
- La función **void loop()**

[3] http://arduino.cc/es

[4] http://arduino.cc/es

Zona **global**

Aquí será donde indicaremos a arduino los nombres de los pines y donde crearemos aquellas variables que queramos que existan en todo el programa. Aunque comprende todo lo que está fuera de las otras dos zonas, es recomendable agruparlo todo en la parte superior del código.

También se pueden crear nuestras propias funciones.

En el caso del ejemplo “parpadeo”, únicamente tenemos lo siguiente:
int led = 13;

Con esto estamos creando una variable en la que guardaremos el número del pin que utilizaremos conectado al led.

La función **void setup()**

Esta función se ejecuta cada vez que se inicia Arduino (incluyendo al pulsar RESET). Una de las operaciones que se realiza en **void setup()** es la de configurar de los pines que vamos a utilizar.

En el caso del ejemplo “parpadeo”:

```
void setup() {
  pinMode(led, OUTPUT);
}
```

Como sólo se usa un pin, llamado “led”, sólo existe una función de configuración “pinMode” en la que indicamos que lo usaremos como salida.

La función **void loop()**

Esta función es el corazón de los programas creados con arduino. Es una función que permanece en ejecución en forma de bucle infinito. Esto quiere decir que se ejecuta de comienzo a fin, de forma repetida, siempre.

Esto queda más claro viendo el contenido del ejemplo “parpadeo”:

```
void loop() {
  digitalWrite(led, HIGH);   // Enciende el LED al activar el voltage como HIGH
  delay(1000);               // Espera un segundo
  digitalWrite(led, LOW);    // Apaga el led al desactivar el voltage (LOW)
  delay(1000);               // Espera un segundo
}
```

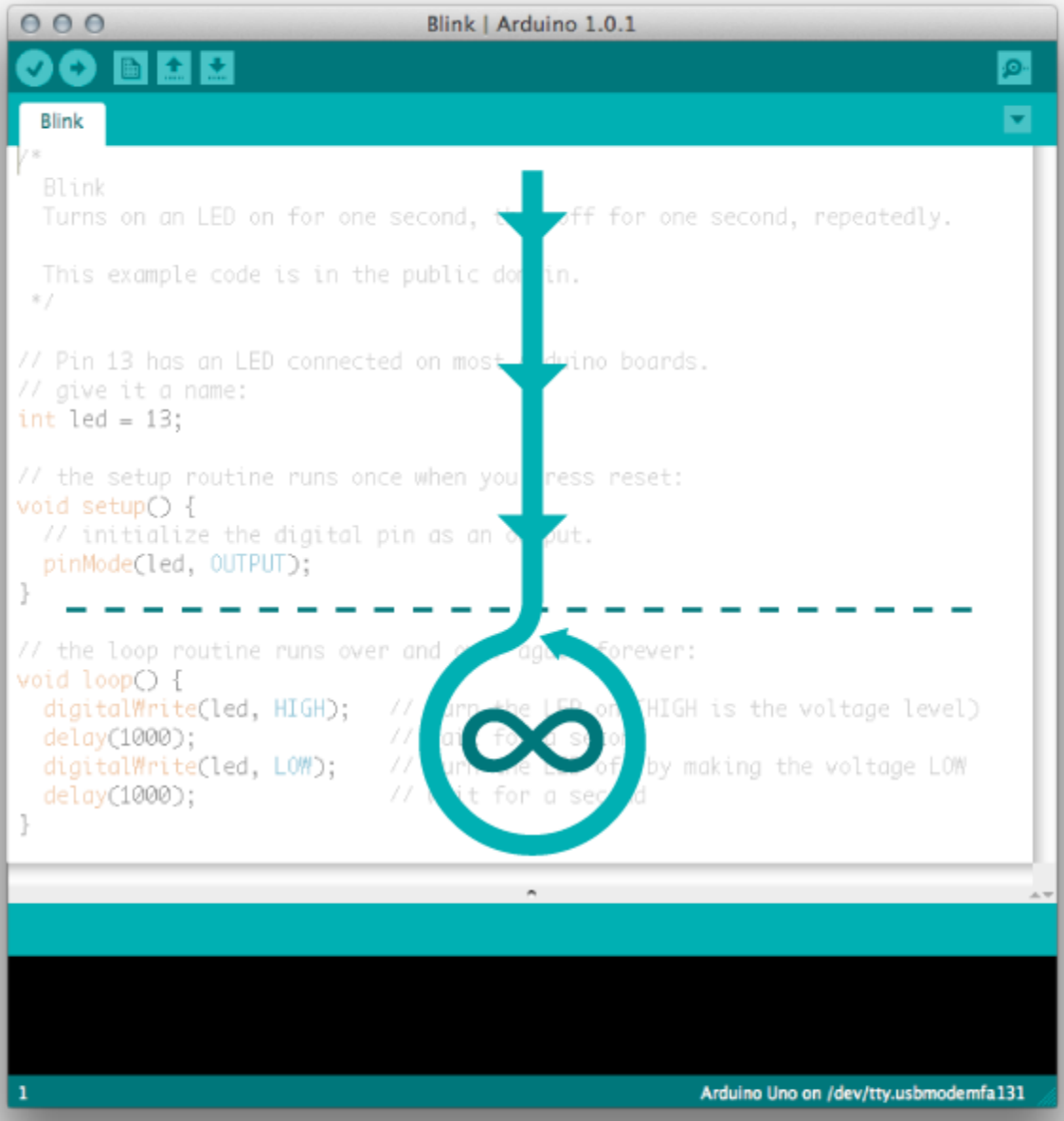
Una vez que ha esperado un segundo por segunda vez, la ejecución vuelve al principio de loop y volvería a encender el led, y así ~~hasta el infinito y más allá~~ sucesivamente.

Con lo cual el programa *Parpadeo* quedaría así:

```
/*
  Parpadeo
  Programa que hace parpadear un led con un retardo de 1 segundo.
*/

// El Pin 13 suele tener asociado un led en casi todas las placas Arduino
// Le damos un nombre al pin 13:
int led = 13;

// La funcion setup() se ejecuta cada vez que se inicia la placa (o se pulsa reset)
void setup() {
  // configuramos el pin “led” como salida
  pinMode(led, OUTPUT);
}

// la funcion loop() se mantiene ejecutandose como un bucle infinito hasta que deja de
alimentarse arduino.
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

Flujo de Programación:

Por lo tanto podemos imaginarnos, de una manera visual, que la ejecución normal de un programa de arduino es de la siguiente manera:



Lenguaje de Programación

Toda la información respecto al lenguaje de programación[5] que se emplea en Arduino se puede encontrar en:
http://arduino.cc/es/Reference

Sintaxis

- ; (punto y coma)
  Sirve para separar sentencias (llamada a función, declaración de variable...).
- {} (llaves)
  Marcan los límites de un bloque de código (inicio y fin de una función o estructura de control).
- // (comentarios en una línea)
  Inicio de un comentario en la misma linea
- /* */ (comentarios en múltiples líneas)
  Inicio y fin de un bloque de comentario

Variables

Constantes[6]

INPUT: Entrada
OUTPUT: Salida
HIGH: Encendido (5V)
LOW: Apagado (0V)
true: Verdadero
false: Falso

[5] http://arduino.cc/es
[6] http://arduino.cc/es

Tipos de Dato

***var** será el nombre que queramos dar a esa variable.
Podrá ser: valor, led, pot (potenciómetro), pin, pepito, jorgito...

- boolean (verdadero/falso)

  Ejemplo: `boolean var = true;`

- char (carácter)

  Ejemplo: `char var = 'R'`

- byte

  byte var = Bxxxxxxxx; // "B es el formateador binario (B00010010 = 18 decimal)

  Ejemplo: `byte var = B00010010;`

- int (entero, desde -32,768 hasta 32,767)

  Ejemplo: `int var = 3;`

- unsigned int (entero sin signo, desde 0 a 65,535)

  Ejemplo: `unsigned int var = 12;`

- long (entero 32bit, desde -2,147,483,648 hasta 2,147,483,647)

  Ejemplo: `long var = 8;`

- unsigned long (entero 32bit sin signo, desde 0 a 4,294,967,295)

Ejemplo:

```
unsigned long var = 18;
```

- float (es un número decimal de 32bit)

Ejemplo:

```
float var = 3.5;
```

- double (es lo mismo que float)

Ejemplo:

```
double var = 3.8;
```

- string (cadena de caracteres)

Ejemplo:

```
string var = “¡Hola Mundo!”;
```

- array (vector)

```
int vars[x]; // un vector de x elementos de tipo int
```

Ejemplo:

```
int var[14];
float var[20];
```

- void (vacío) (Sólo utilizado cuando una función no devuelve nada)

Ejemplo:

```
void funcion()
```

Funciones Principales

La plataforma Arduino tiene un enorme catálogo de bibliotecas de funciones, algunas de las cuales están directamente incluidas en el entorno de desarrollo. Las que no están incluidas en él se pueden añadir con la expresión

```
#include <biblioteca.h> .
```

Las funciones más comúnmente utilizadas son las siguientes:

*__Write__ se refiere a escritura / salida de valores de la placa al exterior.
*__Read__ se refiere a lectura / entrada de valores desde el exterior a la placa.

E/S Digitales

- pinMode: Configuración del Pin

  pinMode(número del Pin, tipo “INPUT o OUTPUT”);

  Ejemplo: `pinMode(9, OUTPUT);`

- digitalWrite: Escritura en Pin Digital

  digitalWrite(número del Pin, valor);

  Ejemplo: `digitalWrite(ledPin, HIGH);`

- digitalRead: Lectura en Pin Digital

digitalRead(número del Pin);

Ejemplo: `digitalRead(9);`

E/S Analógicas

- analogRead: Lectura de Pin analógico

analogRead(número del Pin);

Ejemplo: `analogRead(9);`

- analogWrite: Escritura Analógica - PWM (modulación por ancho de pulso)

analogWrite(número del Pin, valor);

Ejemplo: `analogWrite(9, 255);`

Tiempo

- delay: Pausa la ejecución un tiempo determinado

delay(milisegundos);

Ejemplo: `delay(1000);`

Comunicación

- Serial: Estructura que contiene todas las funciones relacionadas con la comunicación por puerto serie.

Print: Muestra en la consola los datos seleccionados

Serial.print(datos que queremos imprimir);

Ejemplos:

de texto → Serial.print("¡Hola Mundo!");
de variable → Serial.print(valPot);

Println: Hace lo mismo que *print* pero añadiendo un salto de línea

Serial.println(datos que queremos imprimir);

Ejemplos:

de texto → Serial.println("¡Hola Mundo!");
de variable → Serial.println(valPot);

Estructuras de control

if...else (comparador si... sino).

Ejemplo:

Si el led está apagado
{
enciendelo
}
sino
{
apágalo
}
	if(digitalRead(led) == LOW)
	{
	digitalWrite(led, HIGH);
	}
	else
	{
	digitalWrite(led, LOW);
	}

for (bucle con contador).

Ejemplo:

For (valor inicial; condición para que siga el bucle; forma de aumentar)

Para que (desde el valor 0; hasta el valor 100; aumenta de 1 en 1)
{
enciende el led
espera 1 segundo
apaga el led
espera un segundo
}
	int cont;
	for(cont=0; cont<100; cont=cont+1)
	{
	digitalWrite(led, HIGH);
	delay(1000);
	digitalWrite(led, LOW);
	delay(1000);
	}

while (bucle por comparación

Ejemplo:

Mientras ocurra (hay luz)
{
Apaga Led
}
	while(INPUT==HIGH)
	{
	DigitalWrite = LOW;
	}

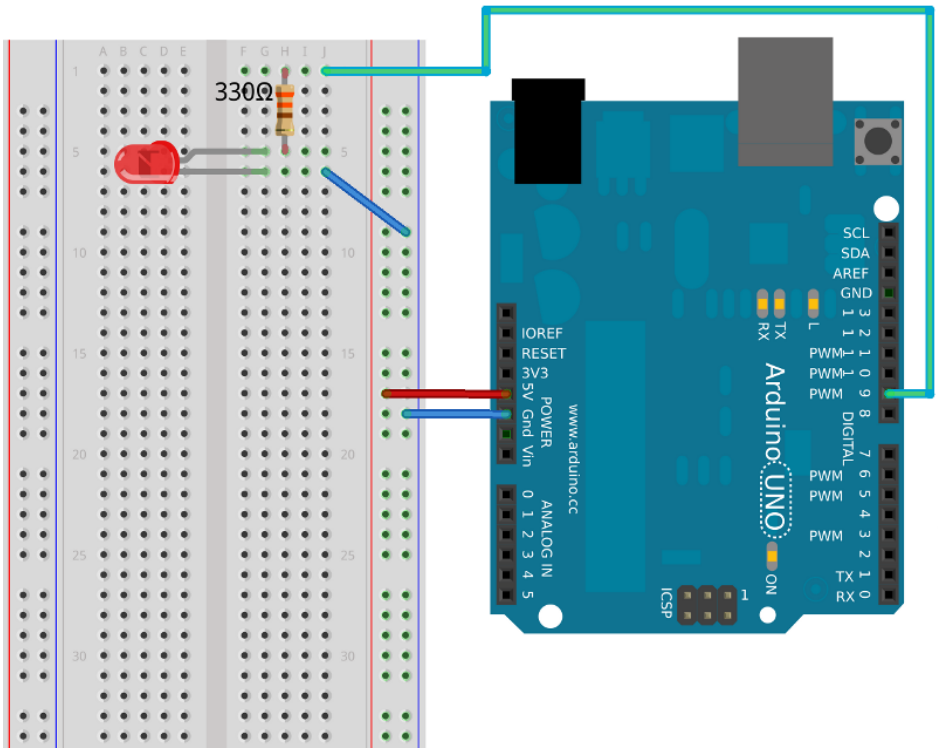
Atención.* El texto de color más suave que aparece en los códigos son comentarios y por lo tanto es **código no efectivo únicamente aclaratorio.

Ejercicio 01 - *Parpadeo*

En este primer ejemplo vamos a montar un circuito que haga parpadear un led a intervalos de 1 segundo.

Material necesario:
1x Arduino
1x Placa de Prototipos
4x Cables
1x Resistencia de 330
1x Diodo LED

Esquema del circuito:



Código:

```
/* Parpadeo
  Aplicacion que enciende y apaga un led
  a intervalos de un segundo
*/

int pinLed = 9; //llamamos pinLed al pin 9

//Funcion de configuracion de Arduino
void setup()
{
   pinMode(pinLed, OUTPUT); //Configuramos pinLed como salida
}

//Funcion bucle de Arduino. Corazon del programa
void loop()
{
 digitalWrite(pinLed, HIGH); //Activamos pinLed. Enciende el LED
 delay(1000); //Esperamos un segundo (1000ms)
 digitalWrite(pinLed, LOW); //Desactivamos pinLed. Apaga el LED
 delay(1000); //Esperamos un segundo
}
```

MEJORA tu Proyecto:

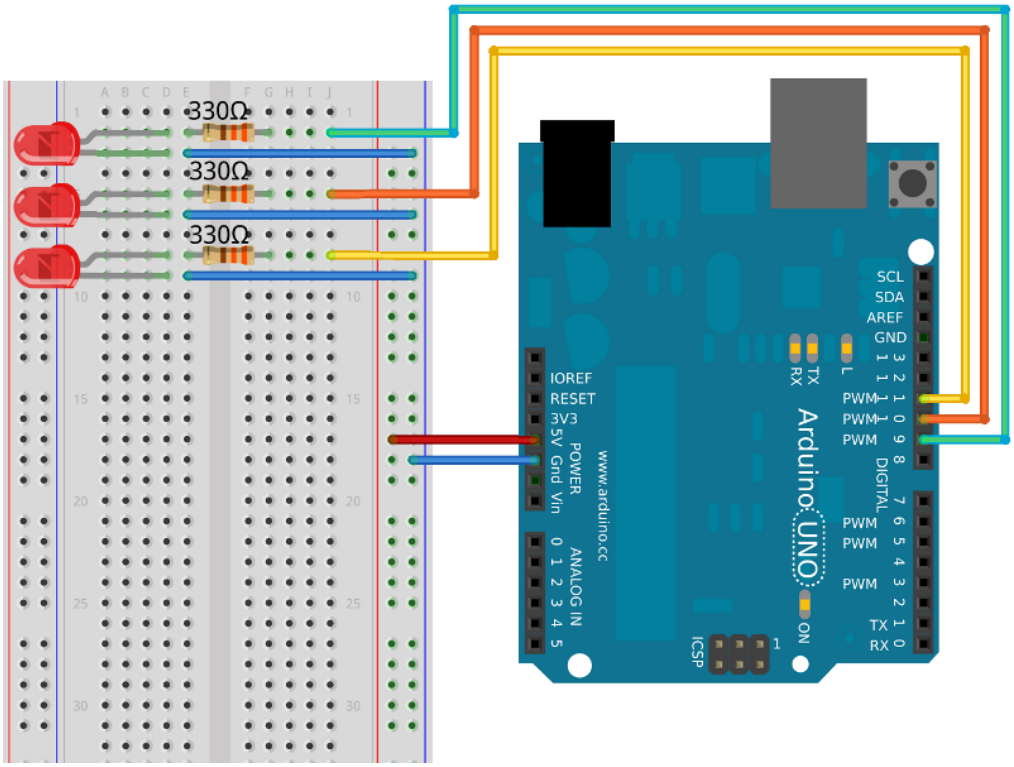
Mejora 01 // Parpadeo con múltiples LED:

Añade al circuito 2 diodos LED y haz que se enciendan y apaguen secuencialmente, de uno en uno.

Material ADICIONAL necesario:
+ 2x Diodo LED
+ 2x Resistencia 330
+ 4x Cables

Esquema del circuito:

******Atención.*** La patilla más **corta** de los LED es la patilla que va a tierra.



Código:

```
/*
ParpadeoMejora01
Aplicacion que enciende y apaga tres led secuencialmente a intervalos de medio segundo
*/

int pinLed1 = 9; //llamamos pinLed1 al pin 9
int pinLed2 = 10; //llamamos pinLed1 al pin 10
int pinLed3 = 11; //llamamos pinLed1 al pin 11

//Funcion de configuración de Arduino
void setup()
{
   pinMode(pinLed1, OUTPUT); //Configuramos pinLed1 como salida
   pinMode(pinLed2, OUTPUT); //Configuramos pinLed2 como salida
   pinMode(pinLed3, OUTPUT); //Configuramos pinLed3 como salida

}
```

```
//Función bucle de Arduino. Corazon del programa
void loop()
{
  digitalWrite(pinLed1, HIGH); //Activamos pinLed. Enciende el LED
  delay(500); //Esperamos un segundo (1000ms)
  digitalWrite(pinLed1, LOW); //Desactivamos pinLed. Apaga el LED
  delay(500); //Esperamos un segundo
  digitalWrite(pinLed2, HIGH); //Activamos pinLed. Enciende el LED
  delay(500); //Esperamos un segundo (1000ms)
  digitalWrite(pinLed2, LOW); //Desactivamos pinLed. Apaga el LED
  delay(500); //Esperamos un segundo
  digitalWrite(pinLed3, HIGH); //Activamos pinLed. Enciende el LED
  delay(500); //Esperamos un segundo (1000ms)
  digitalWrite(pinLed3, LOW); //Desactivamos pinLed. Apaga el LED
  delay(500); //Esperamos un segundo
}
```

Mejora 02 // Parpadeo con múltiples LED haciendo uso del *Bucle For*:

Modifica el código para hacer lo mismo que el circuito de la mejora 01, pero con un bucle “for”.

Código:

```
/*
ParpadeoMejora02
Aplicacion que enciende y apaga tres led secuencialmente
a intervalos de medio segundo, con bucle FOR
*/

int pinLed[3] = {9, 10, 11};

//Funcion de configuracion de Arduino
void setup()
```

```
{
   int i=0;
   for(i=0; i<3; i++)
   {
      pinMode(pinLed[i], OUTPUT); //Configuramos los pines pinLed como salida
   }
}

//Funcion bucle de Arduino. Corazon del programa
void loop()
{
 int i=0;
 for(i=0; i<3; i++)
 {
   digitalWrite(pinLed[i], HIGH); //Activamos pinLed. Enciende el LED
   delay(500); //Esperamos un segundo (1000ms)
   digitalWrite(pinLed[i], LOW); //Desactivamos pinLed. Apaga el LED
   delay(500); //Esperamos un segundo
 }
}
```

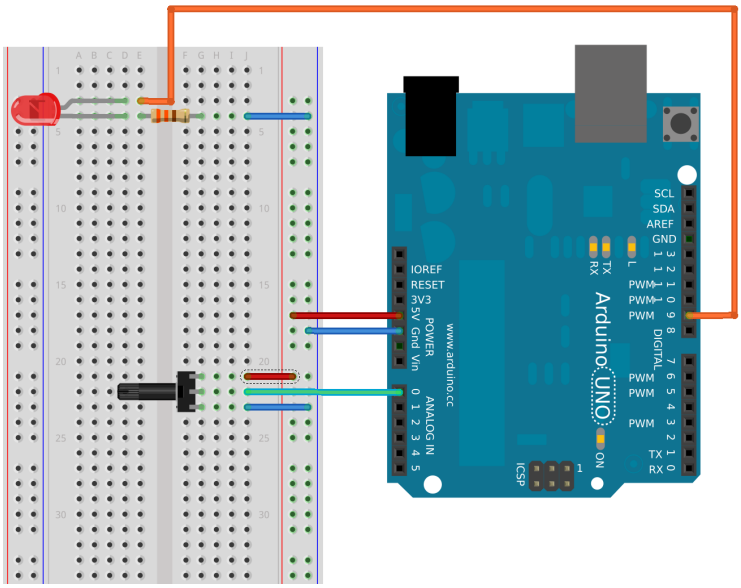
Mejora 03 // Cambio luminosidad:

En esta mejora vamos a montar un circuito que haga variar la luminosidad de un led en función del valor que introduzcamos con un potenciómetro. Además, utilizaremos una nueva ventana, la "Consola de Serial" para monitorizar (ver) el valor que llega desde el potenciómetro.

Material necesario:
1x Arduino
1x Placa de Prototipado
7x Cables
1x Diodo Led
1x Resistencia 330 (LED)

1x Potenciómetro

Esquema del circuito:



Código:

```
/*
ParpadeoMejora03
Aplicacion que haga variar la luminosidad de un led en función del valor que introduzcamos con un potenciómetro. Además, utilizaremos una nueva ventana, la “Consola de Serial” para monitorizar el valor que llega desde el potenciómetro.
*/

int pinLed = 9; //llamamos pinLed al pin 9
int pinPot = 0; //llamamos pinPot al pin 0. Mas adelante indicaremos que es el pin analogico en vez del digital

//Funcion de configuracion de Arduino
void setup()
{
  pinMode(pinLed, OUTPUT); //Configuramos pinLed1 como salida
  Serial.begin(9600); //Iniciamos el puerto Serie a 9600.
```

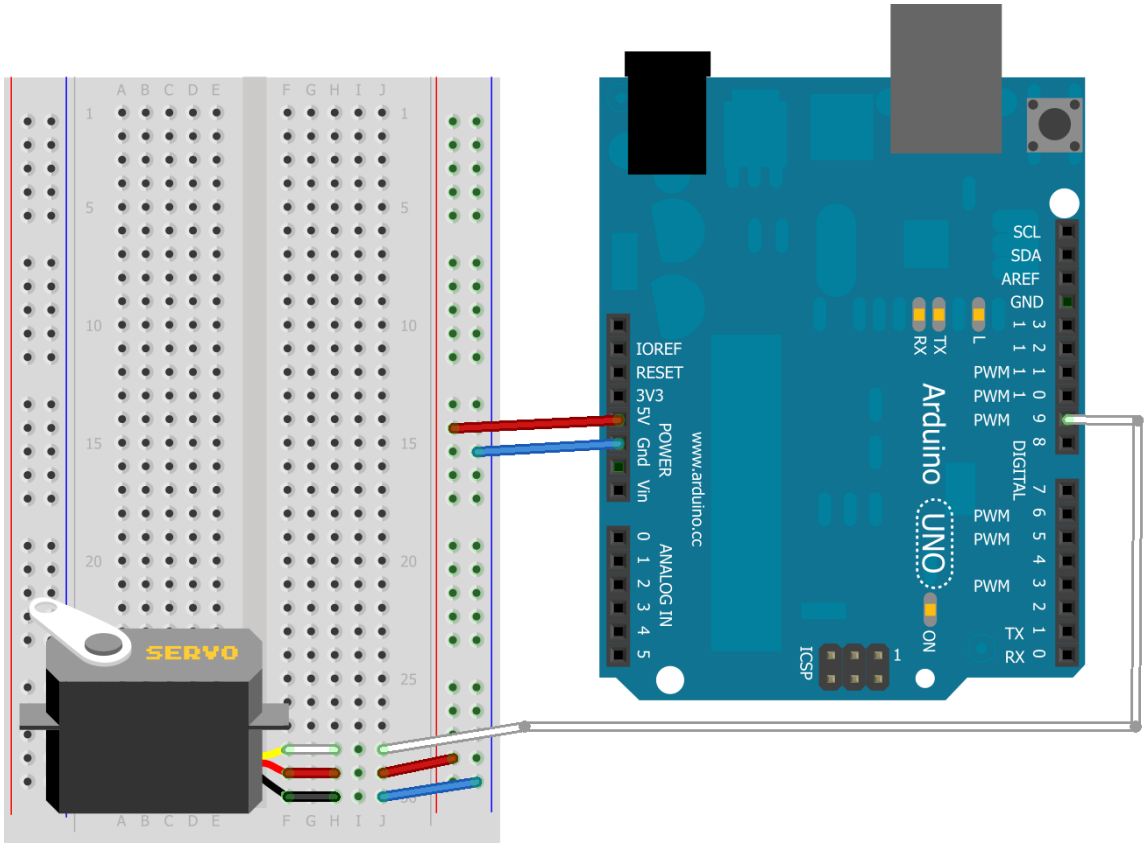
```
}

//Funcion bucle de Arduino. Corazon del programa
void loop()
{
  int valorPot = 0; //variable en la que guardaremos el valor del potenciometro
   int valorLed = 0; //variable en la que guardaremos la intensidad que queremos dar al LED
    valorPot = analogRead(pinPot); //leemos el pin analogico del potenciometro y lo guardamos en valorPot
    valorLed = map(valorPot, 0, 1024, 0, 255); //Convertimos los valores que lee el potenciometro a los que daremos al Led
 analogWrite(pinLed, valorLed); //Sacamos el valor de intensidad por el pin del Led

 //ahora vamos a imprimir por el puerto serie los valores, para comprobarlos.
 Serial.print("valorPot = ");
 Serial.print(valorPot);
 Serial.println();
 Serial.print("valorLed = ");
 Serial.print(valorLed);
 Serial.println();
}
```

Ejercicio 02 - *Servomotor*

En este segundo ejemplo vamos a montar un circuito que mueva un servomotor a distintas posiciones a intervalos de tiempo de 1 segundo

Material necesario:
1x Arduino
1x Placa de Prototipos
5x Cables
1x conector de 3 pines
1x Servomotor

Esquema del circuito:



Código:

```
/*
Servo
Aplicacion que mueve un Servo a distintas posiciones, con intervalos de 1 segundo
*/
#include <Servo.h> //incluimos una biblioteca de funciones para manejar Servomotores

int pinServo = 9; //llamamos pinServo al pin 9
Servo miServo; //Creamos un elemento “Servo” en el sistema

//Funcion de configuracion de Arduino
void setup()
{
   miServo.attach(pinServo); //Indicamos que el servo esta conectado a pinServo
}

//Funcion bucle de Arduino. Corazon del programa
void loop()
{
  miServo.write(0); //movemos el servo a la posicion 0 grados
  delay(1000); //Esperamos un segundo (1000ms)

  miServo.write(90); //movemos el servo a la posicion 90 grados
  delay(1000); //Esperamos un segundo (1000ms)

  miServo.write(180); //movemos el servo a la posicion 180 grados
  delay(1000); //Esperamos un segundo (1000ms)
}
```

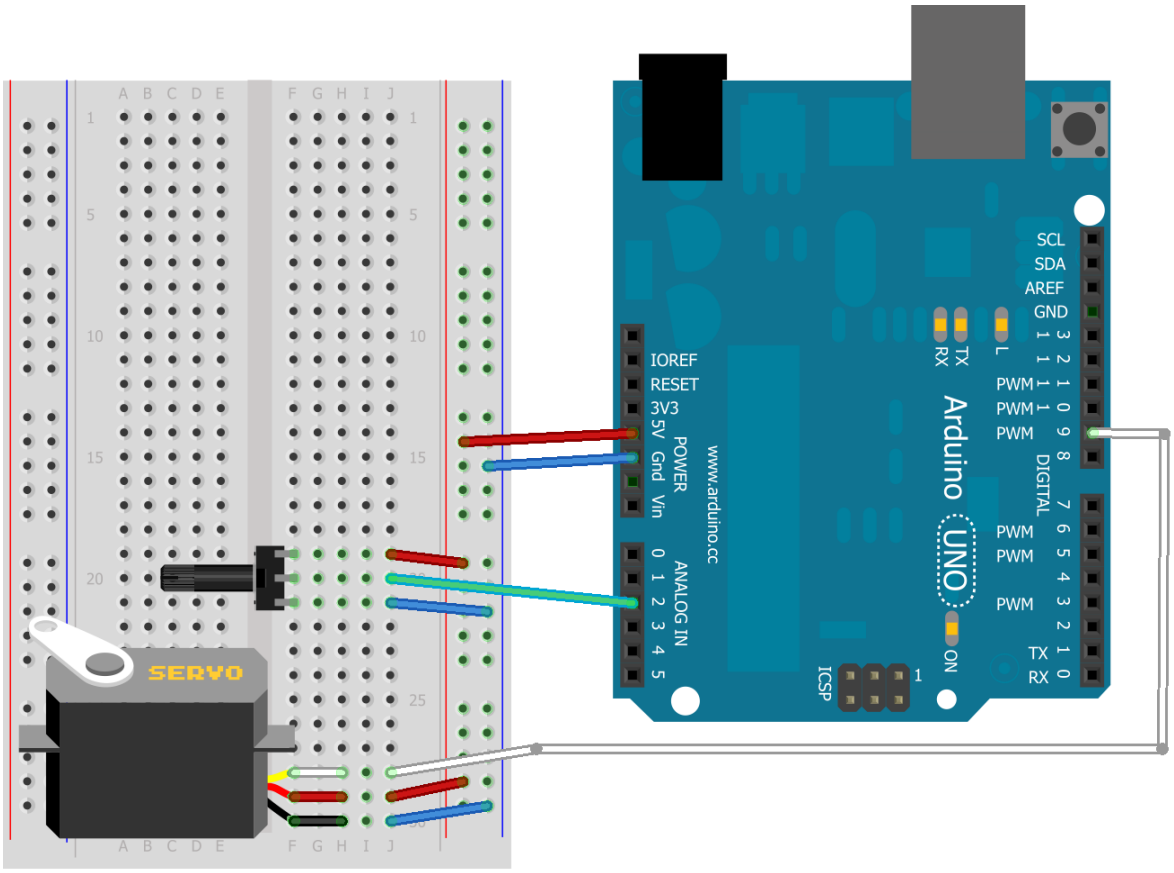
MEJORA tu Proyecto:

Mejora 01 // Controlar el movimiento con un potenciómetro:

Añade al circuito un potenciómetro que nos permita mover el servomotor a la posición deseada

Material ADICIONAL necesario:
+ 1x Potenciometro
+ 3x cables

Esquema del circuito:



Código:

```
/*
ServoPot
Aplicacion que mueve un Servo a la posicion indicada por el potenciometro
*/
#include <Servo.h> //incluimos una biblioteca de funciones para manejar Servomotores

int pinServo = 9; //llamamos pinServo al pin 9
int pinPot = 2; //llamamos pinPot al pin 2
Servo miServo; //Creamos un elemento “Servo” en el sistema

//Funcion de configuracion de Arduino
void setup()
{
   miServo.attach(pinServo); //Indicamos que el servo esta conectado a pinServo
```

```
}

//Funcion bucle de Arduino. Corazon del programa
void loop()
{
  int valorPot = 0;
  int valorServo = 0;

  valorPot = analogRead(pinPot); //leemos el valor del potenciometro
  valorServo = map(valorPot, 0, 1024, 0, 179); //convertimos ese valor a grados
  miServo.write(valorServo); //movemos el servo a la posicion que hemos calculado

}
```

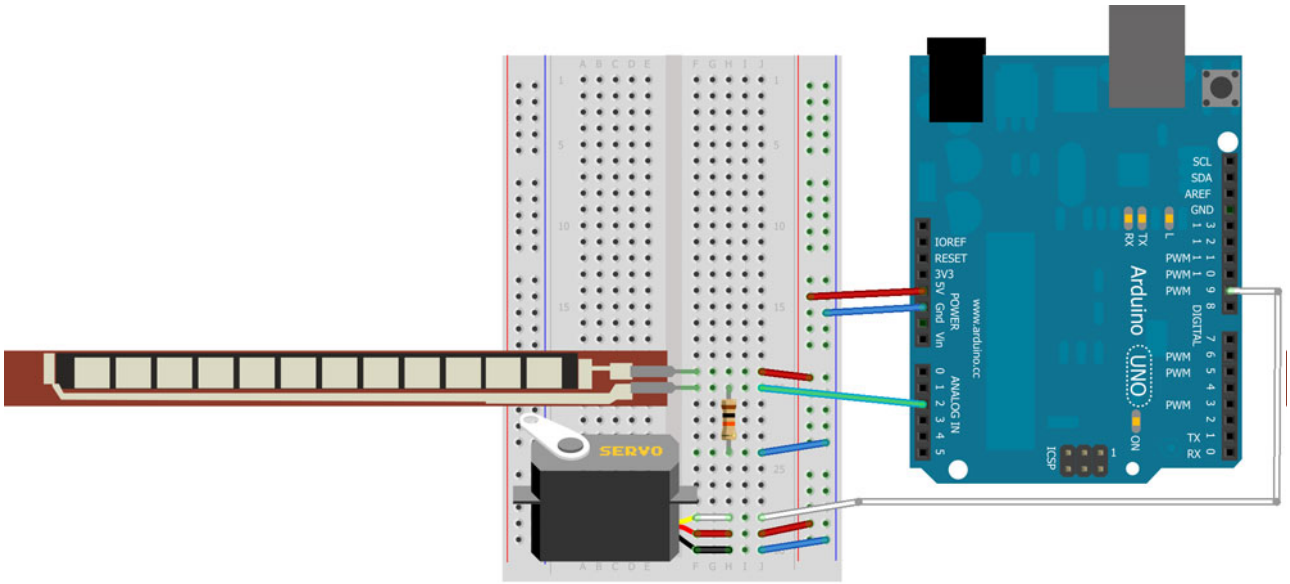
Mejora 02 // Controlar el movimiento con un flexómetro:

Añade al circuito un flexómetro que nos permita mover el servomotor a la posición deseada. El flexómetro requerirá que lo calibres, por lo tanto añade la consola Serial y muestra los valores del flexómetro. En la consola deberás ver los valores que devuelve el flexómetro cuanto esta doblado y cuando está estirado. Esos valores serán los que pongamos en la función map, cambiando el 50 por el valor más bajo y 300 por el valor más alto.

Material ADICIONAL necesario:

+ 1x flexómetro
- 1x cable
- 1x Potenciómetro

Esquema del circuito:



Código:

```
/*
ServoFlex
Aplicacion que mueve un Servo a la posicion indicada por el flexometro
*/
#include <Servo.h> //incluimos una biblioteca de funciones para manejar Servomotores

int pinServo = 9; //llamamos pinServo al pin 9
int pinFlex = 2; //llamamos pinFlex al pin 2
Servo miServo; //Creamos un elemento "Servo" en el sistema

//Funcion de configuracion de Arduino
void setup()
{
   miServo.attach(pinServo); //Indicamos que el servo esta conectado a pinServo
   Serial.begin(9600);  //Iniciamos la consola Serial
}

//Funcion bucle de Arduino. Corazon del programa
void loop()
{
  int valorFlex = 0;
  int valorServo = 0;

  valorFlex = analogRead(pinFlex); //leemos el valor del flexometro
  Serial.print("valorFlex = ");
  Serial.print(valorFlex);
  Serial.println();
  valorServo = map(valorFlex, 50, 300, 0, 179); //convertimos ese valor a grados
  miServo.write(valorServo); //movemos el servo a la posicion que hemos calculado

}
```

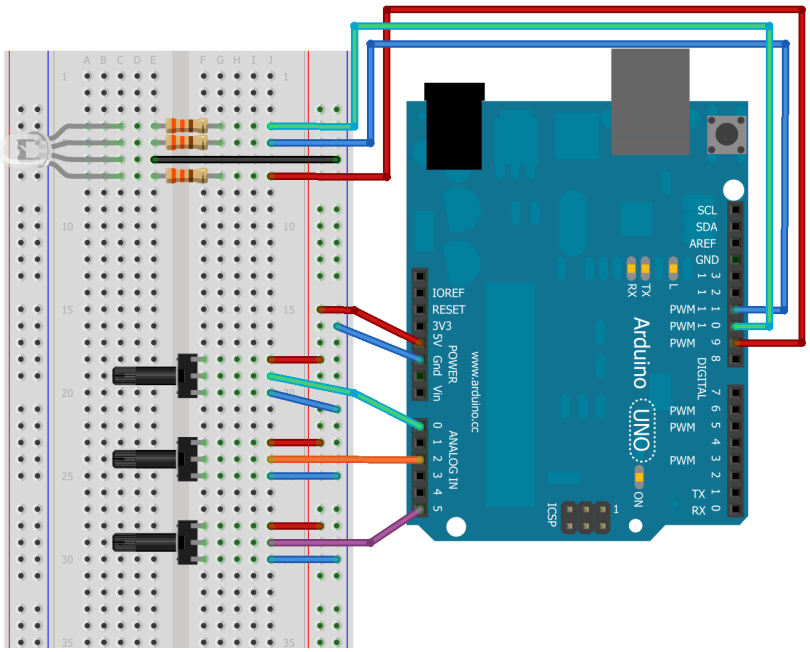
Ejercicio 03 - *LED RGB*

En este ejercicio vamos a controlar el color de un diodo LED RGB mediante el uso de 3 potenciómetros.

Material necesario:
1x Arduino
1x Placa de prototipado
1x Led RGB
3x Resistencia de 330 (LED)
3x Potenciómetro
15x Cable

Esquema del circuito:

Atención.* La patilla más **larga de los LED RGB es la patilla que va a tierra.



Código:

```
/*
PotenciometroLedRGB
Aplicacion que haga variar la luminosidad de los 3 canales de un led RGB en función del valor que introduzcamos con tres potenciómetros.
*/

int pinRojo = 9;  //llamamos pinRojo al pin 9
int pinVerde = 10;  //llamamos pinVerdel al pin 10
int pinAzul = 11; //llamamos pinAzul al pin 11

int pinPotRojo = 0;  //llamamos pinPotRojo al pin 0
int pinPotVerde = 2;  //llamamos pinPotVerde al pin 2
int pinPotAzul = 5;  //llamamos pinPotAzul al pin 5

//Funcion de configuracion de Arduino
void setup()
{
   pinMode(pinRojo, OUTPUT); //Configuramos pin Rojo (9) como salida
   pinMode(pinVerde, OUTPUT); //Configuramos pin Verde (10) como salida
   pinMode(pinAzul, OUTPUT); //Configuramos pin Azul (11) como salida
   Serial.begin(9600); //Iniciamos el puerto Serie a 9600.
}

//Funcion bucle de Arduino. Corazon del programa
void loop()
{
  int valorPot = 0; //variable en la que guardaremos el valor del potenciometro

  int valorLeds = 0; //variable en la que guardaremos la intensidad que queremos dar al LED
   valorPot = analogRead(pinPotRojo); //leemos el pin analogico del potenciometro y lo guardamos en valorPot
   valorLeds = map(valorPot, 0, 1024, 0, 255); //Convertimos los valores que lee el
```

```
potenciometro a los que daremos al Led
   analogWrite(pinRojo, valorLeds); //Sacamos el valor de intensidad por el pin del Led

  //ahora vamos a imprimir por el puerto serie los valores, para comprobarlos.
  Serial.print("valorPotRojo = ");
  Serial.print(valorPot);
  Serial.println();
  Serial.print("valorLedRojo = ");
  Serial.print(valorLeds);
  Serial.println();

  // ahora repetimos lo mismo del pinRojo para los otros dos pines
  //VERDE
  valorPot = analogRead(pinPotVerde);
  valorLeds = map(valorPot, 0, 1024, 0, 255);
  analogWrite(pinVerde, valorLeds);
  Serial.print("valorPotVerde = ");
  Serial.print(valorPot);
  Serial.println();
  Serial.print("valorLedVerde = ");
  Serial.print(valorLeds);
  Serial.println();

  //AZUL
  valorPot = analogRead(pinPotAzul);
  valorLeds = map(valorPot, 0, 1024, 0, 255);
  analogWrite(pinAzul, valorLeds);
  Serial.print("valorPotAzul = ");
  Serial.print(valorPot);
  Serial.println();
  Serial.print("valorLedAzul = ");
  Serial.print(valorLeds);
  Serial.println();

}
```

Versión del código con bucle FOR:

```
/* PotenciometroLedRGB
Aplicacion que haga variar la luminosidad de los 3 canales de un led RGB en función del valor que introduzcamos con tres potenciómetros.
En este caso basaremos la programación en el uso del bucle FOR
*/

int pinRGB[3] = {9, 10, 11};  //agrupamos los pines del LED rgb en pinRGB
int pinPOT[3] = {0, 2, 5}; //agrupamos los pines del Potenciometro en pinPOT

//Funcion de configuracion de Arduino
void setup()
{
  int i = 0;
  for(i=0; i<3; i++) //para los valores 0, 1 y 2 de la variable " i "
  {
    pinMode(pinRGB[i], OUTPUT); //Configuramos el pin " i " como salida
  }
  Serial.begin(9600); //Iniciamos el puerto Serie a 9600.
}

//Funcion bucle de Arduino. Corazon del programa
void loop()
{
  int valorPot = 0; //variable en la que guardaremos el valor del potenciometro
  int valorLeds = 0; //variable en la que guardaremos la intensidad que queremos dar al LED
  int i = 0;
  for(i=0; i<3; i++)
 {
   valorPot = analogRead(pinPOT[i]); //leemos el pin analogico del potenciometro " i " y lo guardamos en valorPOT[ i ]
   valorLeds = map(valorPot, 0, 1024, 0, 255); //Convertimos los valores que lee el potenciometro a los que daremos al Led
```

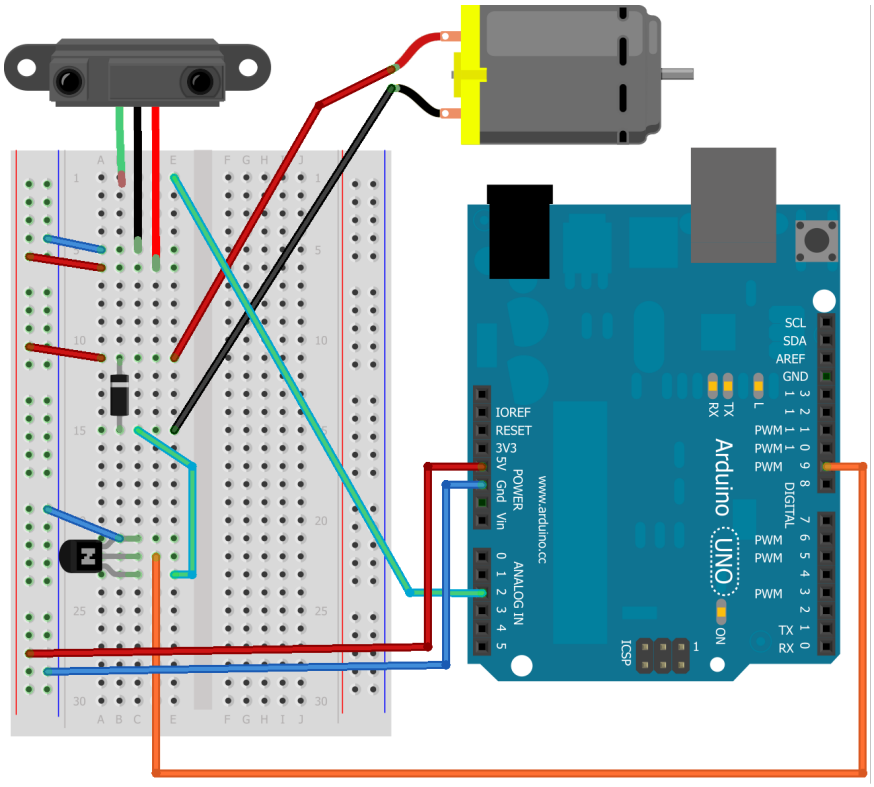
```
  analogWrite(pinRGB[i], valorLeds); //Sacamos el valor de intensidad por el pin del Led

  //ahora vamos a imprimir por el puerto serie los valores, para comprobarlos.
  Serial.print("valorPot[ ");
  Serial.print(i);
  Serial.print(“ ] = “);
  Serial.print(valorPot);
  Serial.println();
  Serial.print(“valorLed[ “);
  Serial.print(i);
  Serial.print(“ ] = “);
  Serial.print(valorLeds);
  Serial.println();
  }
}
```

Ejercicio 04 - *Motor continua controlado por IR*

En este ejercicio vamos a controlar la velocidad a la que gira un motor de continua, según la distancia que haya entre un objeto y el sensor de distancia infrarojo.

Material necesario:
1x Arduino
1x Placa de prototipado
1x Sensor IR
1x Motor Continua
1x Transistor NPN
1x diodo
9x Cable

Esquema del circuito:



Código:

```
/* motorIR
Aplicacion que haga variar la velocidad de giro de un motor de continua segun la distancia que exista entre un objeto y un sensor infrarojo.
*/

int motorPin = 9;  //llamamos motorPin al pin 9
int irPin = 2; //llamamos irPin al pin 2.

//Funcion de configuracion de Arduino
void setup()
{
  pinMode(motorPin, OUTPUT);
  Serial.begin(9600); //Iniciamos el puerto Serie a 9600.
}

//Funcion bucle de Arduino. Corazon del programa
```

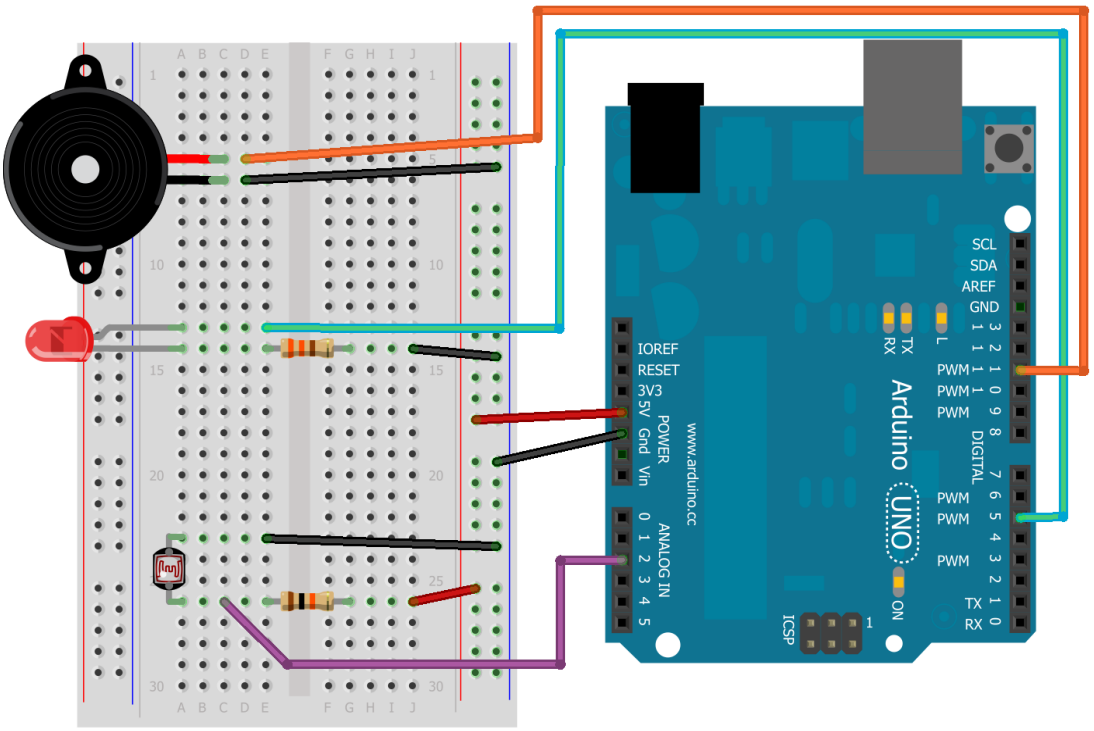
```
void loop()
{
  int valorIR = 0; //variable en la que guardaremos el valor del sensor IR
  int valorMotor = 0; //variable en la que guardaremos el valor para el motor

  valorPot = analogRead(irPin); //leemos el pin analogico del sensor IR
  Serial.print("valorInfrarojo = ");
  Serial.print(valorIR);
  Serial.println();
 valorMotor = map (valor, 400, 60, 0, 255); //Convertimos la lectura del IR en valores para la salida del motor.
  Serial.print("valorMotor = ");
  Serial.print(valorMotor);
  Serial.println();
  analogWrite(motorPin, valorMotor); //Enviamos la señal al motor.
}
```

Ejercicio 05 - *Piezo y LED controlados por Fotómetro*

En este ejercicio vamos a controlar el brillo de un diodo LED y el tono de un piezo, según la intensidad de la luz que alcance a un fotosensor.

Material necesario:
1x Arduino
1x Placa de prototipado
1x fotoSensor
1x Piezo
1x Led
1x resistencia de 330 (LED)
1x resistencia de 10K (fotosensor)
9x Cable

Esquema del circuito:



Código:

```
/* piezoLedYfotosensor
Aplicacion que haga variar el brillo de un diodo LED y el tono de un piezo, según la intensidad de la luz que alcance a un fotosensor.
*/

int pinPiezo = 11;  //llamamos pinPiezo al pin 9
int pinLed = 5; //llamamos pinLed al pin 5.
int pinFoto = 2; //llamamos pinFoto al pin 2.

//Funcion de configuracion de Arduino
void setup()
{
  pinMode(pinPiezo, OUTPUT);
  pinMode(pinLed, OUTPUT);
  Serial.begin(9600); //Iniciamos el puerto Serie a 9600.
}

//Funcion bucle de Arduino. Corazon del programa
```

```
void loop()
{
  int valorFoto = 0; //variable en la que guardaremos el valor del fotosensor
  int valorLed = 0; //variable en la que guardaremos el valor para el Led
  int valorPiezo = 0; //variable en la que guardaremos el valor para el piezo
  int tono = 0;

  valorFoto = analogRead(pinFoto); //leemos el valor del fotosensor
  valorLed = map(valorFoto, 200, 900, 0, 255); //Lo convertimos en un valor de salida para el led
  analogWrite(pinLed, valorLed); //Sacamos el valor de intensidad para el led por el pin

   tono = map(valorFoto, 200, 900, 1900, 950); //Convertimos el valor del fotosensor en la mitad del periodo de una nota musical. El DO bajo se corresponde a unos 1900 y el DO alto a unos 3800, por lo tanto, la mitad de sus periodos 950 y 1900.
  tono = tono/100; //Con esto vamos a limpiar un poco el valor del fotosensor
  tono = tono*100; // eliminando los valores de las decinas y las unidades

  int i=0;
  for(i=0; i<4; i++)
  {
  digitalWrite(pinPiezo, HIGH);
  delayMicroseconds(tono);
  digitalWrite(pinPiezo, LOW);
  delayMicroseconds(tono);
  }
}
```