SISTEMAS DE CONTROL AUTOMÁTICOS PROGRAMADOS (ROBÓTICA)

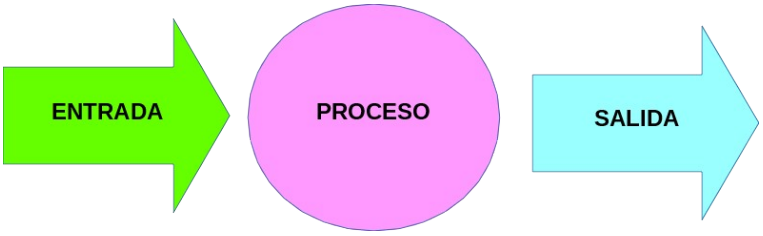
4º ESO TECNOLOGÍA

1. Introducción.

Un sistema de control automático o automatismo, es un conjunto de elementos técnicos que unidos son capaces de realizar una series de acciones para resolver un problema sin intervención humana.
Los que vamos a usar en nuestro curso son programados, que significa que el automatismo funcionará dependiendo de un programa que le haremos a medida de la función a realizar.

2. Elementos de un sistema de control.

Nuestro sistema de control va a contener básicamente tres bloques de dispositivos:

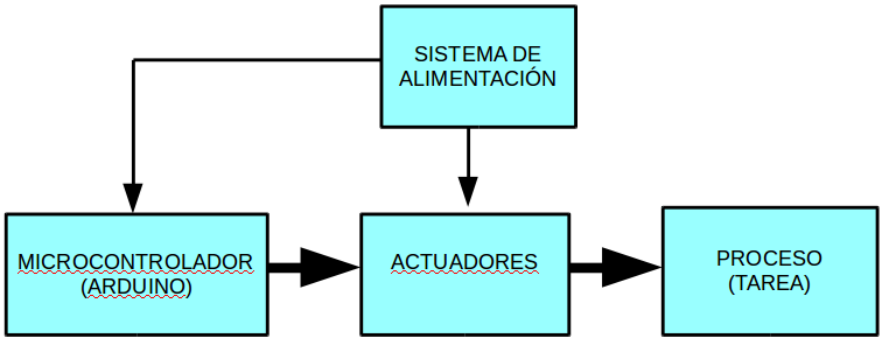


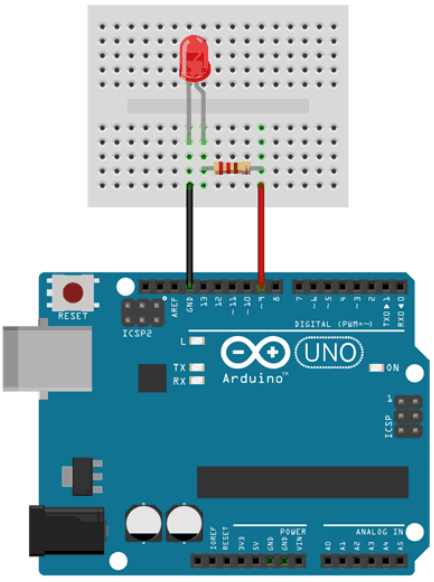
- Elementos de proceso o control: son los que se van a programar, recibirán datos de los elementos de entrada para decidir que realizar con los elementos de salida. En nuestro caso es *Arduino*.
- Elementos de entrada: serán sensores que se le pueden conectar a nuestro sistema de control, pueden ser: pulsadores, interruptores, finales de carrera, ntc, ldr, etc.
- Elementos de salida: van a ser los actuadores, los que va a decidir nuestro programa poner en funcionamiento o parar, como: diodos leds, zumbadores, motores, etc.

3. Tipos de sistemas de control

Lazo Abierto

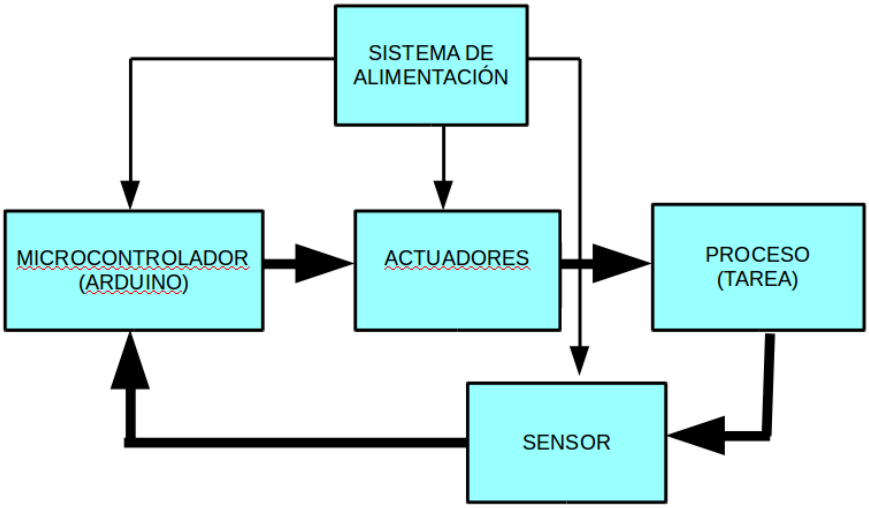
No hay una realimentación, lo que significa el controlador da una orden de puesta en marcha a los actuadores sin que se vigile en ningún momento la señal de los sensores.

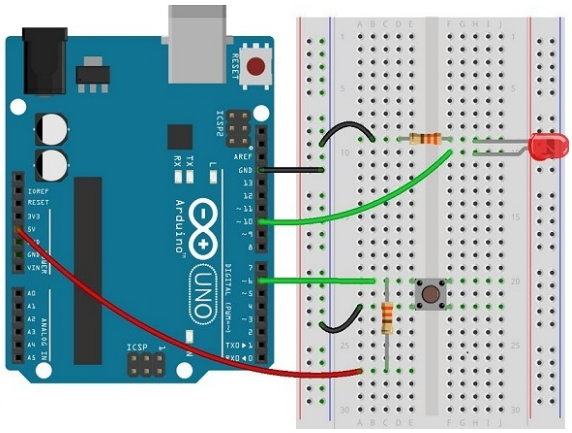




Lazo Cerrado

hay realimentación, lo que quiere decir que el sistema de control siempre está vigilando los sensores para realizar en función del estado de los mismos una acción u otra.



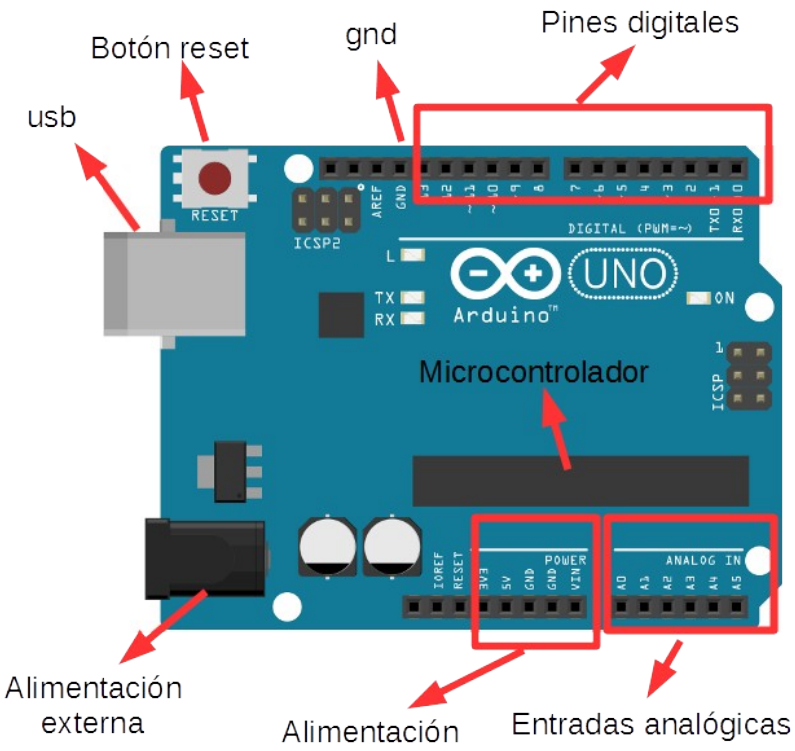


3. Elementos de control programado.

En nuestro caso vamos a utilizar una placa de control programable que se llama **Arduino**, la cual tiene un uso muy extendido por varias razones:

- Hardware libre: los diseñadores han puesto a disposición de todo el mundo los circuitos físicos de la placa para poder fabricarla.
- Software libre: el software que controla la placa también está puesto a disposición de todo el mundo para compartirlo e incluso modificarlo.
- Comunidad: hay una gran cantidad personas compartiendo, proyectos, código y soluciones por internet.
- Precio: muy económico.
- Simplicidad: se pueden realizar soluciones de una manera sencilla.

3.1. Partes de arduino.



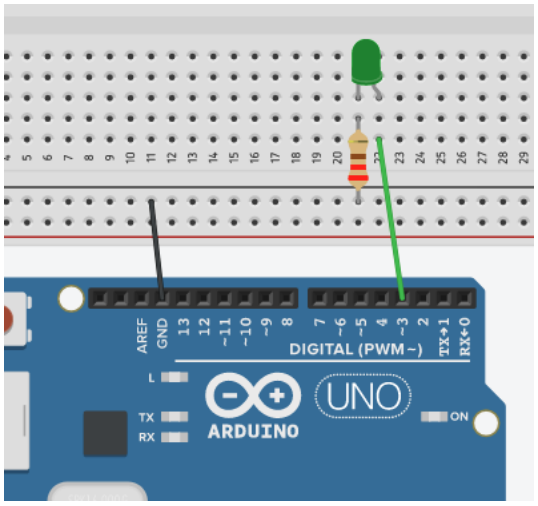
Pines digitales (0 a 13, 5V 40mA) (6 Salidas PWM 8 bits): algo es digital cuando sólo puede tomar dos valores (1 o 0, encendido o apagado). Sirven para conectar tanto salidas (actuadores) digitales (diodos led, motores, zumbadores, etc), como entradas (sensores) digitales (interruptores, pulsadores, finales de carrera, etc).
Estos pines cuando se usan como salidas podemos decirles que estén conectados o no, que tengan corriente (5V) o no (0V). Cuando se usan como entrada detectan cuando los elementos de entrada están cerrados o no. Cuando son de salida dan como máximo 5V y 40 mA.
Pines de alimentación o power: tenemos los pines gnd (ground tierra), sirven para cerrar el circuito, son el polo negativo de nuestra alimentación. Tenemos un gnd en la fila de los pines digitales y dos gnd más en la zona de alimentación (power). Además tenemos pines de alimentacón de 5V, 3,3V, y Vin.
Entradas analógicas (A0 a A5, 0-5V 10 bits): sirven para introducir en el sistema datos que no toman los valores 1 o 0 (digitales), pueden tomar muchos valores, como temperaturas (con ntc), luz (con ldr), etc.
Microcontrolador (ATMEGA 328P, 8 bit, 16Mhz, 32KB flash, 1KB EEPROM, 2KB SRAM): es el dispositivo que recibe el programa y hace que se ejecute, así como controla todos los elementos de la placa. Sólo tiene almacenado un programa en memoria que se sobreescribe cuando se le sube otro.
Usb: es un conector usb que sirve para dar alimentación a la placa y además comunicarse con el ordenador.
Alimentación externa: conector que sirve para alimentar arduino con un elemento externo y que no se alimente por el PC (cargador, pila de 9v, etc). Soporta de 7 a 12V.
Botón de reset: cuando se pulsa reinicia el programa que tiene en memoria.

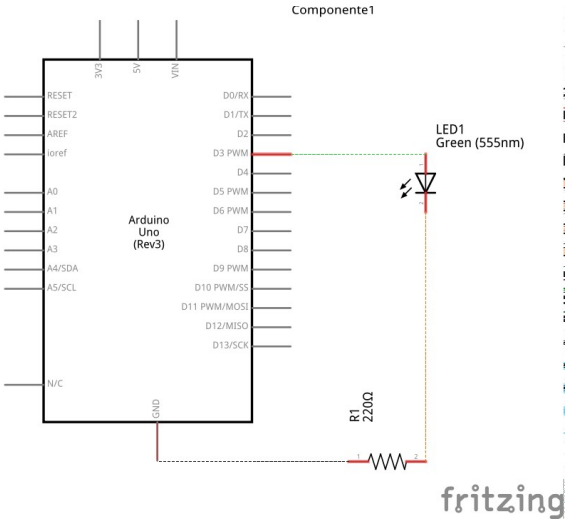
4. Estructura de programa

```
int variable;//zona de variables antes de setup

void setup() {//se pone el código de lo que quiero que se ejecute una vez por ejemplo pinMode (pin, MODO)
        instrucciones a ejecutar una sóla vez
}

void loop() {// se pone el código de lo que quiero que se ejecute indefinidamente
        instrucciones a ejecutar en bucle infinito
}
```

5. Trabajar con Salidas digitales

- Pines 0 a 13.
- Conectar LEDs a pines digitales

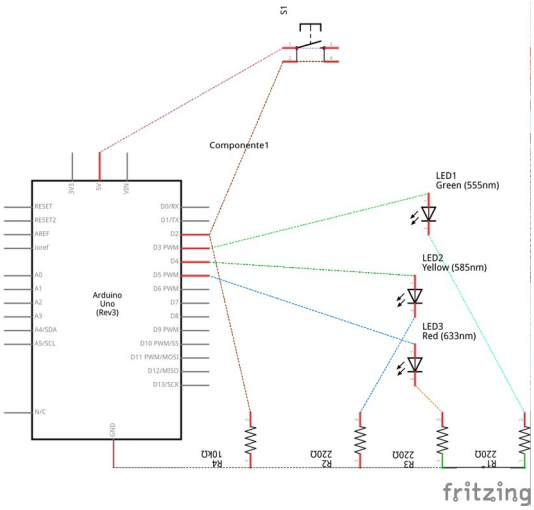




- Órdenes: pinMode (OUTPUT, pin), digitalWrite (pin,HIGH/LOW), delay (tiempo).

6. Trabajar con Entradas digitales

- Pines de 0 a 13.
- Conectar pulsador a entradas digitales
- Órdenes: pinMode (INPUT, pin), digitalRead (pin)



7. Órdenes de control de flujo programa

<u>bucles:</u>

for (valor inicial de condición;condición de salida;acción sobre condición) {instrucciones;}

while (condición a cumplir) {instrucciones;}

<u>condicionales:</u>

if (condición) {instrucciones;}

else {instrucciones;}

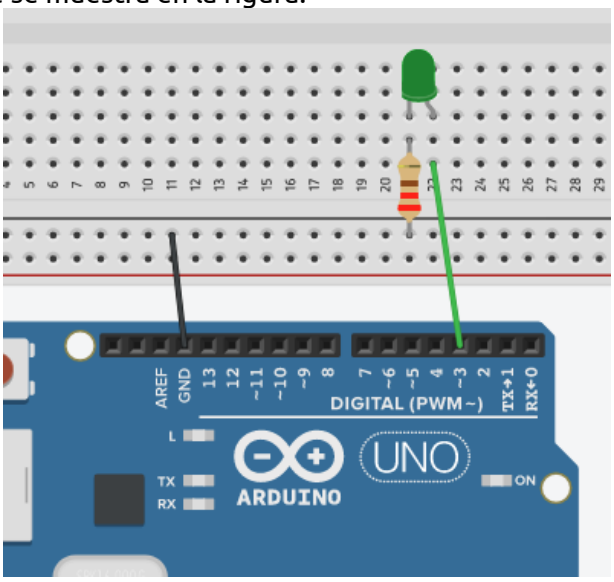
```
switch (expresion)
{
        case valor1:
                instrucciones;
                break;
        case valor2:
                instrucciones;
                break;
        default:
                instrucciones;
                break;
}
```

8. Prácticas a realizar

Reto 1: LED parpadeante
El reto consiste en iluminar de manera intermitente un led utilizando cualquier pin digital de Arduino, en nuestro caso vamos a usar el 3. El led estará encendido durante 500ms y que se apague 100ms y así de forma ininterrumpida. Si se utiliza el pin 13 no hace falta resistencia en otros casos sí.
Objetivos:

- Familiarizarse con el entorno de programación.
- Reconocer las partes de un programa de Arduino.
- Conocer órdenes como: pinMode, digitalWrite y delay.

Conexionado
La forma de conexión será la que se muestra en la figura:



Código fuente

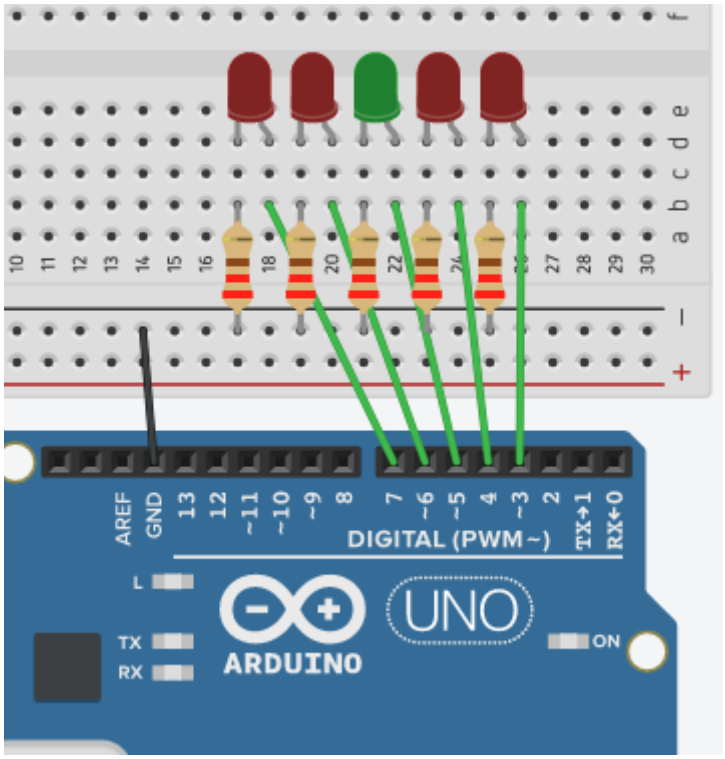
```
void setup()
{
  pinMode(3, OUTPUT);//configura el pin 3 como de salida
}


void loop()
{
  digitalWrite(3, HIGH);//enciendo el pin 3
  delay(500);//espero 500 ms
  digitalWrite (3,LOW);//apago el pin 3
  delay (100);//espero 100 ms
 }
```

Reto 2: Secuencia de leds con for
Se trata de encender y apagar 5 leds secuencialmente. Los leds deben estar conectados a los pines 3, 4, 5, 6 y 7. Tanto el tiempo de encendido y apagado será de 200 milisegundos.

Objetivos:

- Familiarizarse con el entorno de programación.
- Estructura de control for.

Conexionado:
La forma de conexión será la que se muestra en la figura:



Código fuente

```
int n;// variable entera (int) para controlar los pines
int tiempo=200;//tiempo de encendido y apagado de leds

void setup()
{
  for (n=3;n<8;n=n+1) {//for para hacer los pinMode
   pinMode(n, OUTPUT);
  }
}

void loop()
{
  for (n=3;n<8;n++) {//for para encender y apagar los pines
   digitalWrite(n, HIGH);
   delay (tiempo);
   digitalWrite(n, LOW);
   delay (tiempo);
  }
}
```

Reto 3: Secuencia de leds con while y función

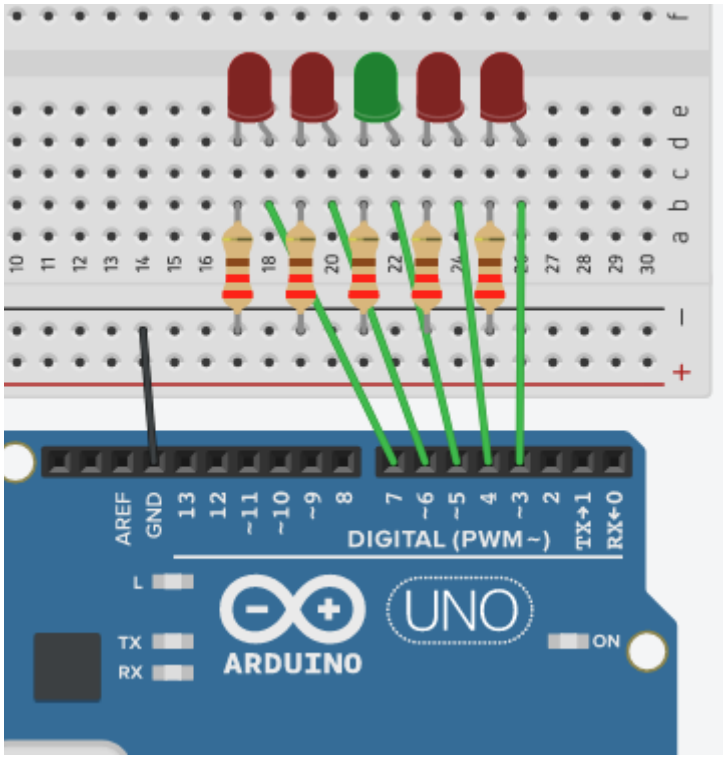
Se trata de encender y apagar 5 leds secuencialmente. Los leds deben estar conectados a los pines 3, 4, 5, 6 y 7. Tanto el tiempo de encendido y apagado será de 200 milisegundos. La secuencia principal del programa debe estar reproducida en una función a la que llamará el programa principal.

Objetivos:

- Familiarizarse con el entorno de programación.
- Estructura de control While.
- Usar funciones propias.

Conexionado:

La forma de conexión será la que se muestra en la figura:



Código fuente

```
int n;// int significa integer (entero);
int tiempo=200;

void setup()
{
  n=3;
  while (n<8) {
  pinMode(n, OUTPUT);
  n=n+1;
  }
}

void loop()
{
  secuencia();//llamada a la función secuencia
}

void secuencia() {//definición de la función secuencia
  n=3;
  while (n<8) {
   digitalWrite(n, HIGH);
   delay (tiempo);
   digitalWrite(n, LOW);
   delay (tiempo);
   n=n+1;
  }
}
```

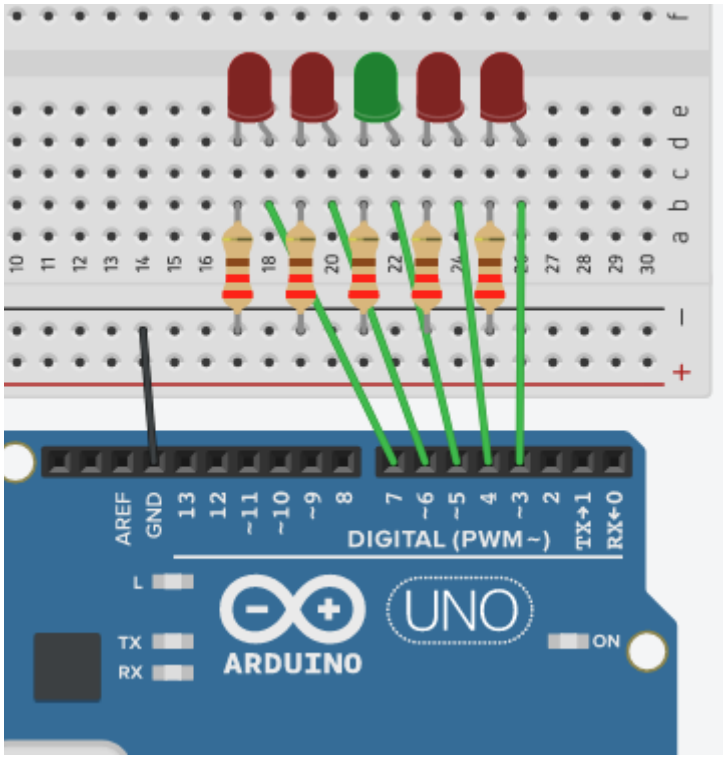
Reto 4: Coche fantástico

Se trata de encender y apagar 5 leds secuencialmente. Los leds deben estar conectados a los pines 3, 4, 5, 6 y 7. Se deben encender y apagar los leds desde el pin 3 al 7, con un tiempo de encendido y apagado de 50 ms, más tarde se deben encender y apagar los leds desde el pin 7 al 3, con un tiempo de encendido y apagado de 50 ms. La secuencia se debe repetir indefinidamente. El efecto del programa es el de las luces delanteras de nuestro querido "Coche fantástico". Podéis intentar hacer una segunda solución quitando el doble encendido de los leds de los extremos.

Objetivos:

- Familiarizarse con el entorno de programación.
- Repasar órdenes de control de programa como: for.

Conexionado



Código fuente

```
int n;// int significa integer (entero);
int tiempo=50;

void setup()
{
  for(n=3;n<8;n++) {
   pinMode (n,OUTPUT);
  }
}

void loop()
{
  for (n=3;n<8;n++){//for que enciende de 3 a 7
   digitalWrite(n,HIGH);
   delay (tiempo);
   digitalWrite(n,LOW);
   delay (tiempo);
  }
  for (n=7;n>2;n--){//for que enciende de 7 a 3
   digitalWrite(n,HIGH);
   delay (tiempo);
   digitalWrite(n,LOW);
   delay (tiempo);
  }
}
```

Reto 5: SOS.

Se trata de un led que en código morse (destellos largos/cortos) especifica una palabra, en nuestro caso SOS. Para el que no lo sepa, la S son tres destellos de corta duración y la O tres destellos de larga duración. También se puede hacer con señales acústicas y un zumbador.

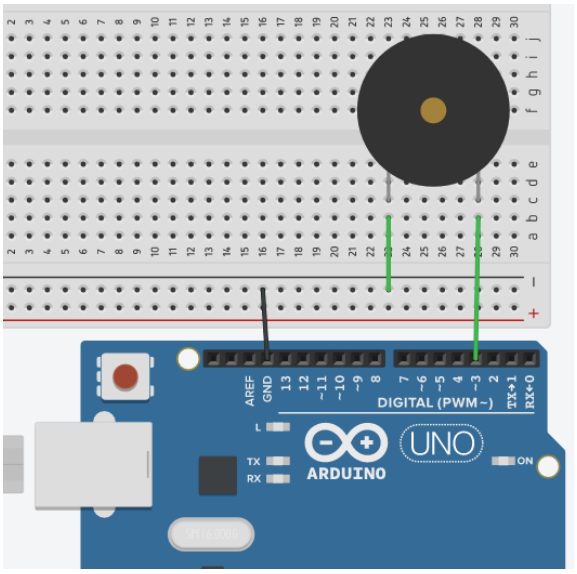
El led debe estar conectado al pin 3, aunque para ser ruidosos se puede hacer con un zumbador puesto en ese mismo pin, los destellos pitidos cortos tendrán una duración de 100 ms y los largos 300 ms. Entre letra y letra debe pasar un tiempo de 300 ms y entre SOS (entre palabras) debe haber un tiempo de 1000 ms.

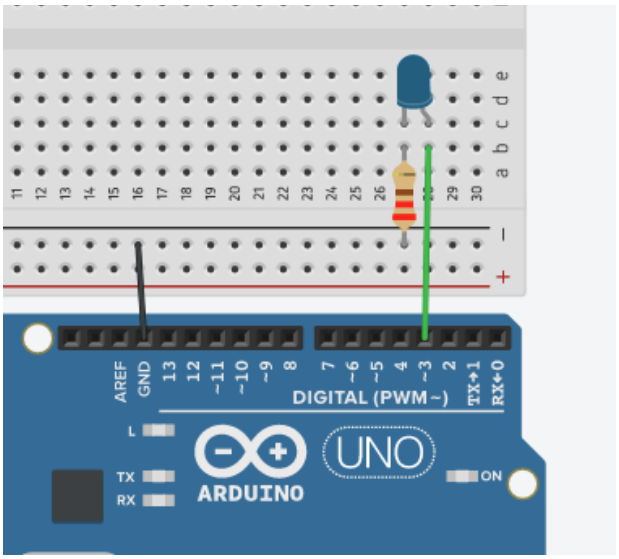
Nota: Debes usar variables para guardar los tiempos que vas a usar.

Objetivos:

- Repaso a bucle for.
- Repaso a funciones.
- Introducción del zumbador.

Conexionado





Código fuente

```
int tcorto=100;//tiempo corto para la S
int tlargo=300;//tiempo largo para la O
int tletras=500;//tiempo entre letras
int tpalabras=1000;//tiempo entre SOS
int n;

void setup()
{
  pinMode(3, OUTPUT);//configuramos el pin 3 de salida
}

void loop()
{
  s();//llamamos a la función s
  delay (tletras);
  o();//llamamos a la función o
  delay (tletras);
  s();//llamamos a la función s
  delay (tpalabras);
}

void s() {//función para hacer la s
  for (n=0;n<3;n++) {
   digitalWrite (3,HIGH);
   delay (tcorto);
   digitalWrite (3,LOW);
   delay (tcorto);
  }
}

void o() {//función para hacer la o
  for (n=0;n<3;n++) {
   digitalWrite (3,HIGH);
   delay (tlargo);
   digitalWrite (3,LOW);
   delay (tcorto);
  }
}
```

Reto 6: Secuencia de leds con pulsador

Se trata de encender y apagar 5 leds secuencialmente al accionar un pulsador. El pulsador debe estar conectado al pin 2, y los leds a los pines 3, 4 , 5, 6 y 7.

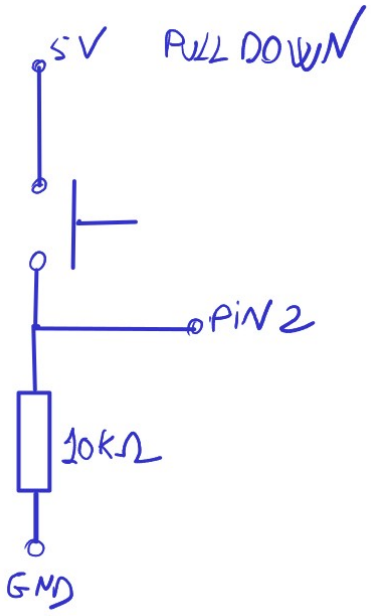
Si el pulsador no está pulsado debe estar encendido el led conectado al pin 5, y si el pulsador está pulsado se debe apagar el led conectado al pin 5 y se deben encender y posteriormente apagar los leds desde el pin 3 al 7, con un tiempo de duración de encendido y apagado de 200 milisegundos.

Nota: la secuencia principal del programa debe estar reproducida en una función a la que llamará el programa principal.

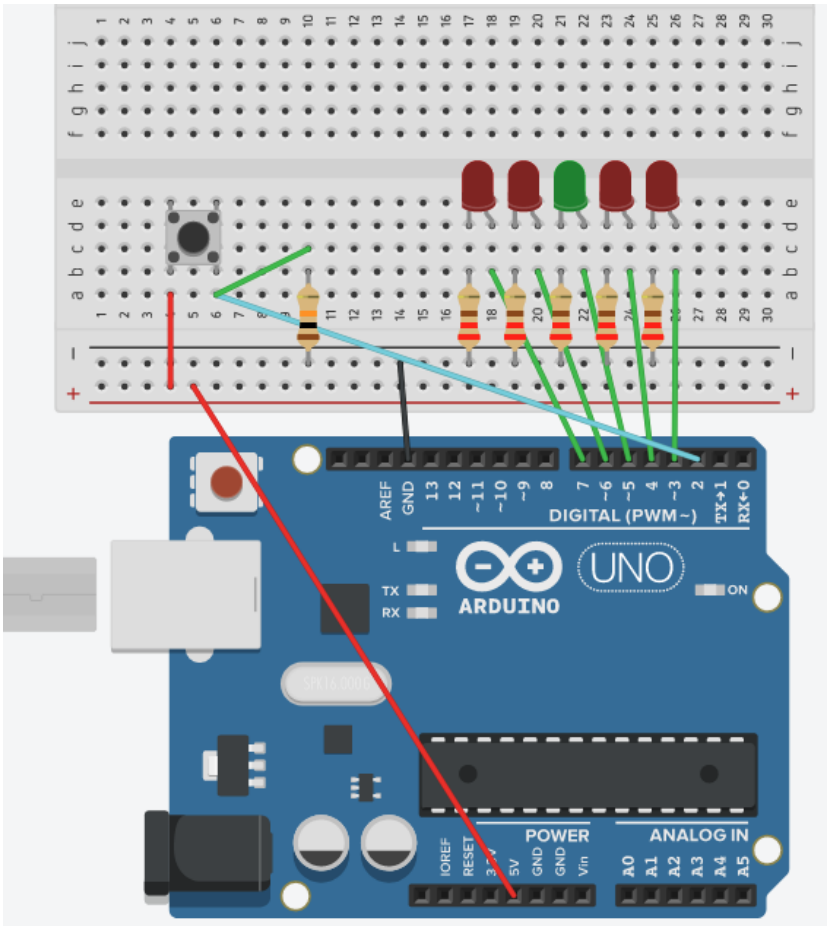
Objetivos:

- Aprender a conectar una entrada digital a arduino (pulsador). Divisor de tensión
- Conexiones Pull-down y Pull-up.
- Conocer órdenes como: digitalRead.
- Conocer órdenes de control de programa como: If y else.

Esquemas
El esquema eléctrico del pulsador será:



Conexionado
Conexionado real con pulsador:



Código fuente

```
int n;// variable entera (int) para controlar los pines
int tiempo=200;//tiempo de encendido y apagado de leds
int pulsador=2;//variable que guarda el pin del pulsador

void setup()
{
  for (n=3;n<8;n=n+1) {//for para hacer los pinMode
   pinMode(n, OUTPUT);
  }
  pinMode (pulsador,INPUT);//configuramos el pin del pulsador como de entrada
}

void loop()
{
  if (digitalRead(pulsador)==HIGH) {//comprobamos si el pin del pulsador está pulsado
    digitalWrite (5,LOW);//apagamos pin 5
    secuencia();//llamamos a función secuencia
  }
  else {//en caso contrario encendemos pin 5
     digitalWrite (5,HIGH);
    }
}

void secuencia() {//función que realiza la secuencia de leds
   for (n=3;n<8;n++) {//for para encender y apagar los pines
    digitalWrite(n, HIGH);
    delay (tiempo);
    digitalWrite(n, LOW);
    delay (tiempo);
  }
}
```

Reto 7: Ruleta de la fortuna

Se trata de cinco leds que se van encendiendo y apagando formando una secuencia, el jugador debe dar al pulsador cuando el led intermedio se enciende, si acierta parpadean todos los leds de forma rápida y la velocidad de la secuencia aumenta, si falla parpadean todos los leds de forma lenta y la velocidad de la secuencia de leds es como al principio lenta.

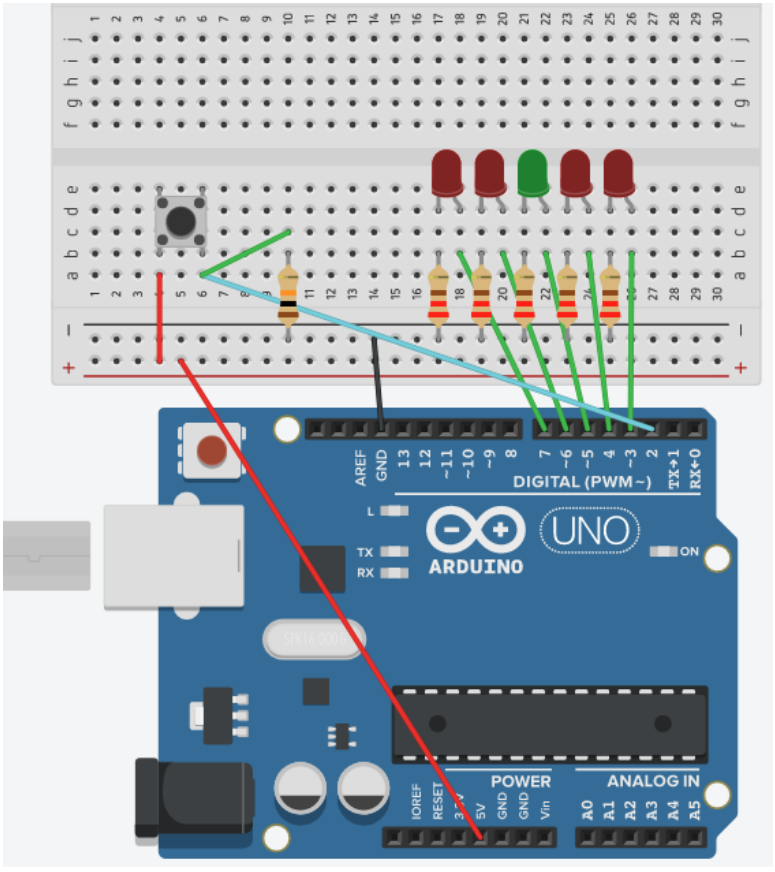
Los leds deben estar conectados de los pines 3 a 7 y el pulsador al pin 2.

El tiempo inicial entre encendido y encendido de leds debe ser 200 ms, si se acierta el tiempo disminuye en 20 ms, si el tiempo entre encendidos llegase a 10 ms, se devuelve el tiempo a 200 ms.

Objetivos:

- Conocer órdenes: if, &&, ||
- Repaso a uso de funciones
- Repaso a entradas digitales

Conexionado



Código fuente

```
int n;// variable entera (int) para controlar los pines
int tiempo=200;//tiempo de encendido y apagado de leds
int pulsador=2;//variable que guarda el pin del pulsador

void setup()
{
  for (n=3;n<8;n=n+1) {//for para hacer los pinMode
   pinMode(n, OUTPUT);
  }
  pinMode (pulsador,INPUT);//configuramos el pin del pulsador como de entrada
}

void loop()
{
  for (n=3;n<8;n++) {//hacemos la secuencia
   digitalWrite (n,HIGH);
   delay (tiempo);
   compruebaacierto();//comprobamos si acertamos o fallamos
   digitalWrite (n,LOW);
   delay (tiempo);
  }
}

void parpadeo(int tiempoparpadeo) {//función que realiza el parpadeo de leds
 for (int m=3;m<8;m++) {//for para encender todos los pines
    digitalWrite(m, HIGH);
   }
 delay (tiempoparpadeo);
 for (int m=3;m<8;m++) {//for para apagar todos los pines
    digitalWrite(m, LOW);
   }
 delay (tiempoparpadeo);
}

void compruebaacierto() {//función para comprobar
  if (digitalRead(pulsador)==HIGH && n==5) {//comprobamos si acierta
   parpadeo(50);
   tiempo=tiempo-20;
    if (tiempo<10){
     tiempo=200;
    }
  }
  if (digitalRead(pulsador)==HIGH && n!=5){//comprobamos si falla
     parpadeo(1000);
    tiempo=200;
  }
}
```

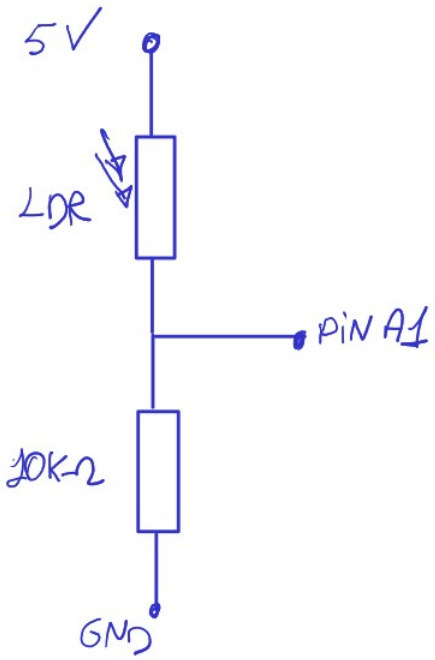
Reto 8: Detector de oscuridad

Se trata de un dispositivo que haga funcionar un led cuando la luminosidad baja de cierto valor umbral. Para ello conectaremos una ldr a la entrada analógica 1 y un led al pin 3. Cuando la luz llegue a cierto valor umbral de voltaje (entre 0 y 1023) que nosotros decidamos, se encenderá el diodo led. Además se deberá visionar el valor de voltaje en la entrada analógica (valor entre 0 y 1023) en una consola en el PC.

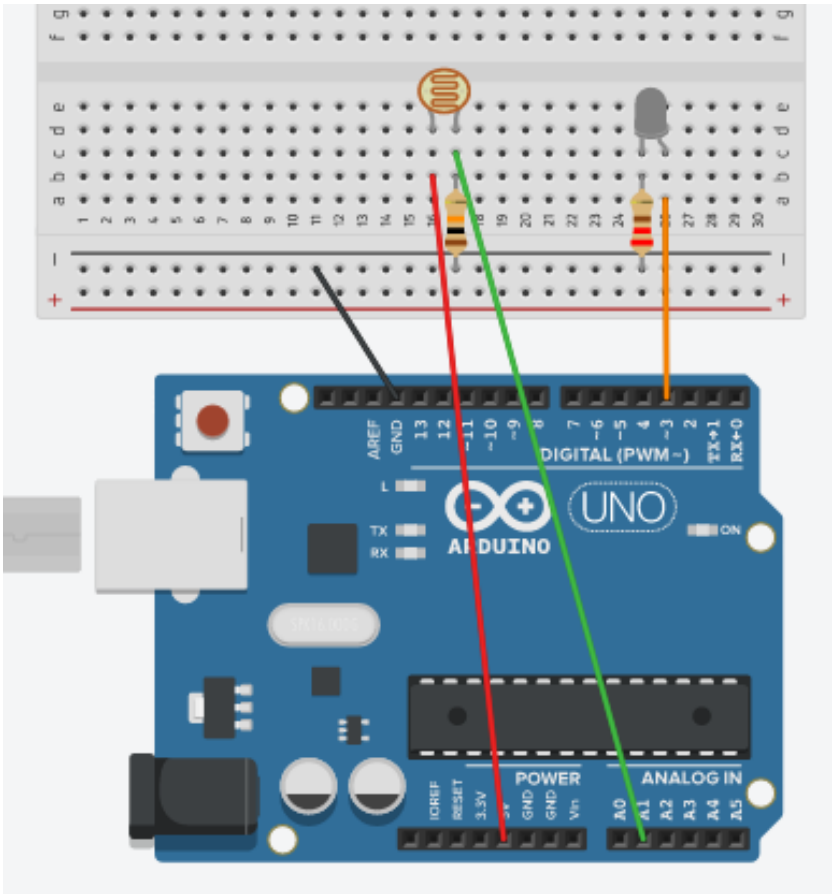
Objetivos:

- ¿Qué son las entradas analógicas?
- Conexión de entrada analógica a arduino (ldr). Repaso de divisor de tensión.
- Órdenes como: analogRead.
- Visualizar datos en consola de puerto serie, con órdenes como: Serial.begin,Serial.print.
- Órdenes de control de programa como: If else.

Esquema conexión LDR



Conexionado



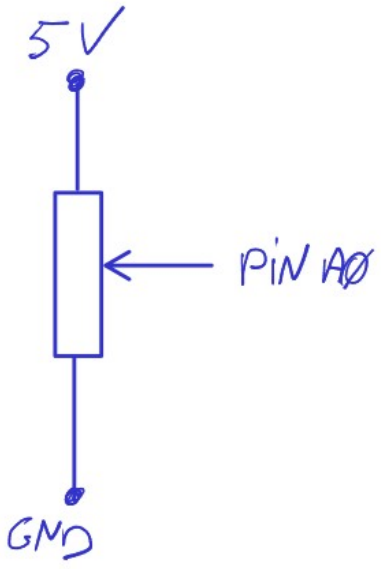
Código fuente

```
int led = 3;
int ldr = 1;
int medida = 0;
int nivel = 900; //variable que guarda el límite de luz al que se activa el led

void setup() {
  pinMode(led, OUTPUT);
  Serial.begin(9600);
}
void monitoriza() { //procedimiento que envía al puerto serie, para ser leído en el monitor,
  Serial.print("La medida es ...");
  Serial.println(medida);
  delay(1000); //para evitar saturar el puerto serie
}
void loop() {
  medida = analogRead(ldr);
  monitoriza();
  if (medida < nivel) { //si la señal del sensor supera el nivel marcado:
    digitalWrite(led, HIGH); //se enciende un aviso luminoso
  }
  else { // si la señal está por debajo del nivel marcado
    digitalWrite(led, LOW);
  }
}
```

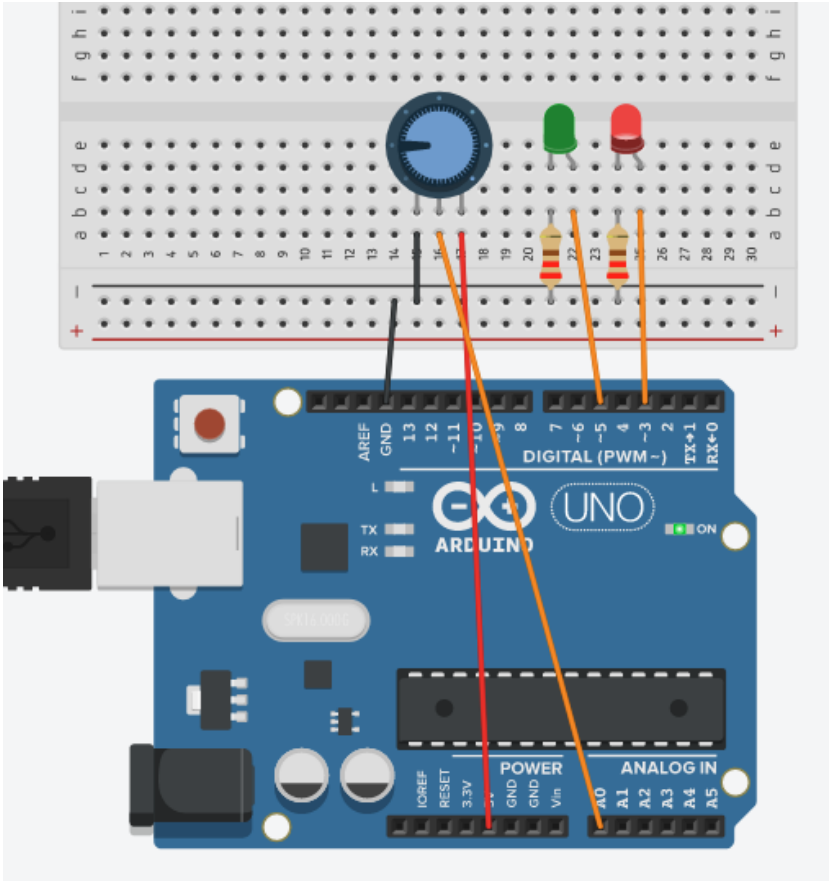
Reto 9: Potenciómetro que enciende un led rojo o uno verde en función de un valor de referencia
El reto consiste en iluminar un led rojo o un led verde en función de un valor de referencia que obtendremos de un divisor de tensión formado por un potenciómetro conectado a una entrada analógica. El led rojo estará conectado al pin 5 y el verde al pin 3. El divisor de tensión de conectará a la entrada analógica A0.
Objetivos:

- Repaso a entradas analógicas.
- Representación de valores utilizando la comunicación serie
- Repaso a órdenes de control If, else.

Esquema de conexión de potenciómetro.



Conexionado

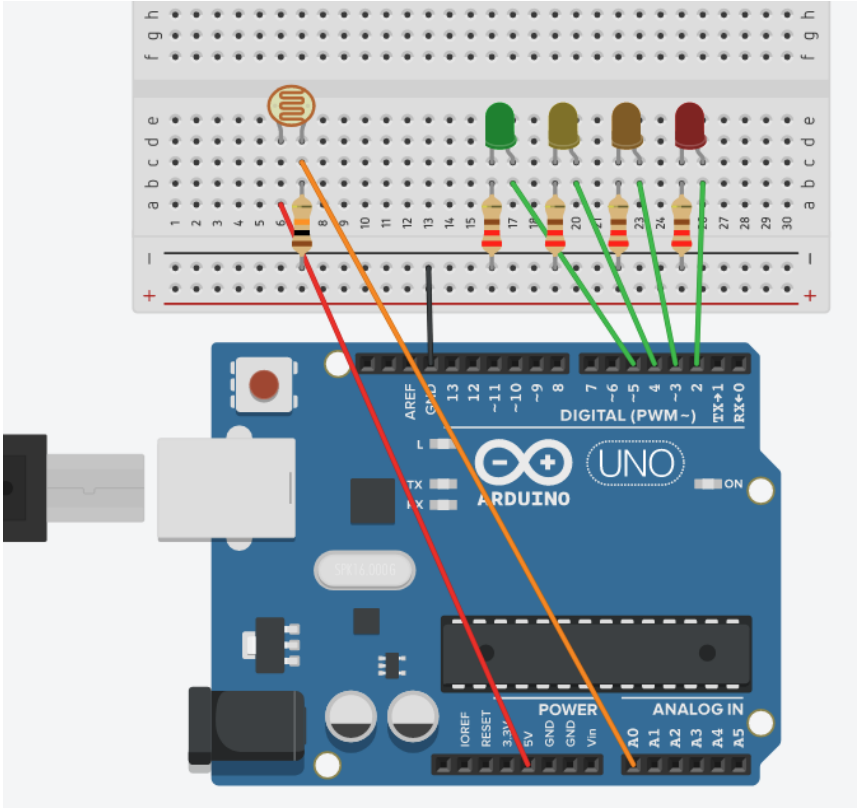


Código fuente

```
//Pines
int ledRojo = 5;
int ledVerde = 3;
int analogPin = 0;

int valor = 0;
int referencia = 512;

void setup() {
  pinMode(ledRojo, OUTPUT);
  pinMode(ledVerde, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  valor = analogRead(analogPin);// lee el pin entrada
  Serial.println(valor); //muestra el valor

  if (valor <= referencia) {
    digitalWrite(ledVerde, HIGH);
    digitalWrite(ledRojo, LOW);
  }
  else {
    digitalWrite(ledRojo, HIGH);
    digitalWrite(ledVerde, LOW);
  }
}
```

Reto 10: Varios leds se encienden en función de la luz.
El reto consiste en que se enciendan o apaguen 4 leds en función de la cantidad de luz que haya. Ya sabéis que los valores de luz se sacan a partir de la conexión de una LDR, estos valores van a variar de 0 a 1023. Divide la lectura de luz en 4 tramos (1024 /5), si la luz está en el tramo más bajo enciende todos los leds, si está en el siguiente tres leds, y así sucesivamente. Tendrás que jugar con las condicionales (if, else) y tener en cuenta que se producen varias condiciones que se tienen que cumplir (&&).

Conexionado



Código fuente 1

```
int luz;
int n;

void setup()
{
  for (n=2;n<6;n++) {
   pinMode (n,OUTPUT);
  }
  Serial.begin (9600);
}

void loop()
{
  luz=analogRead (0);
  Serial.println (luz);

  if (luz<256) {
   for (n=2;n<6;n++) {
      digitalWrite (n,HIGH);
     }
   }
  else if (luz>=256 && luz <512) {
     for (n=2;n<5;n++) {
       digitalWrite (n,HIGH);
      }
   }
  else if (luz>=512 && luz <768) {
    for (n=2;n<4;n++) {
      digitalWrite (n,HIGH);
    }
   }

   else {
     digitalWrite (2,HIGH);
       }

  delay (1000);
  apagaleds();

}

void apagaleds () {
  for (n=2;n<6;n++) {
   digitalWrite (n,LOW);
  }

}
```

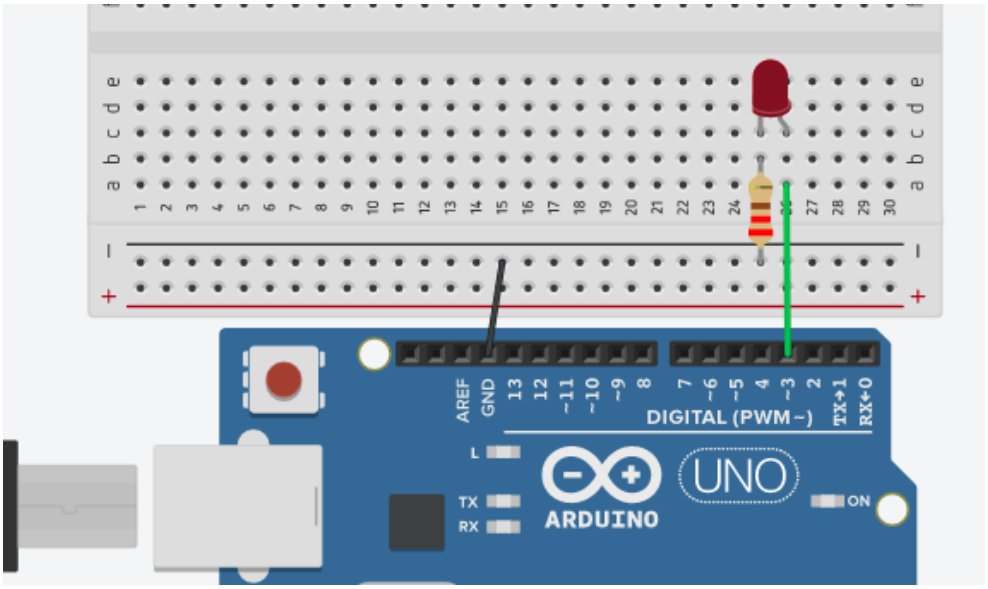
Código fuente 2

```
int luz;
int n;
int numeroleds;

void setup()
{
  for (n=2;n<6;n++) {
   pinMode (n,OUTPUT);
  }
  Serial.begin (9600);
}

void loop()
{
  luz=analogRead (0);
  numeroleds=map(luz,1023,0,1,5);
  Serial.println (numeroleds);
  Serial.println (luz);
  for (n=2;n<numeroleds+2;n++) {
   digitalWrite (n,HIGH);
  }
  delay (1000);
  apagaleds();

}

void apagaleds () {
  for (n=2;n<6;n++) {
   digitalWrite (n,LOW);
  }

}
```

Reto 11: Aumentar y disminuir intensidad luminosa de led (fading).
Se trata aumentar y disminuir la luminosidad de un led usando la capacidad de ofrecer una tensión variable que da una salida analógica. Para ello se conecta un led al pin 3 y se provoca que su luminosidad pase de mínima a máxima, para luego ir de máxima a mínima. Los valores de salidas analógicas van del mínimo 0, al máximo 255.
Objetivos:

- Conexionado de salidas analógicas (power with module pwm).
- Conocer órdenes como analogWrite.

Conexionado



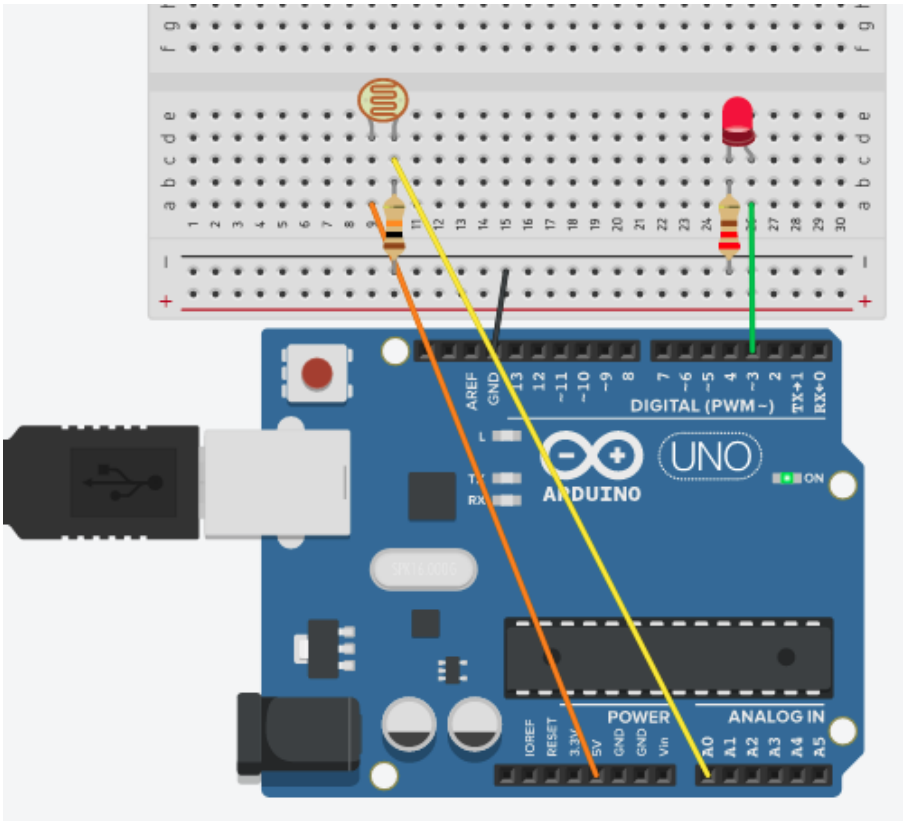
Código Fuente

```
int led=3;
int luminosidad=0;

void setup()
{
  //pinMode(3, OUTPUT);
}

void loop()
{
  for (luminosidad=0;luminosidad<=255; luminosidad=luminosidad+3){
    analogWrite (led,luminosidad);
    delay (30);
  }

  for (luminosidad=255;luminosidad>=0; luminosidad=luminosidad-3){
    analogWrite (led,luminosidad);
    delay (30);
  }

}
```

Reto 12: Linterna automática.
Se trata de aumentar o disminuir la luminosidad de un led en función de la luz captada por una ldr. Cuanta más luz menos debe lucir el led y cuanta menos luz más debe lucir el led.
Objetivos:

- Conexionado de salidas analógicas (power with module pwm).
- Conocer órdenes como analogWrite.
- Conocer órdenes como map.

Conexionado



Código Fuente

```
int led=3;
int luzLed=0;
int luzLdr=0;

void setup()
{

}

void loop()
{
  luzLdr=analogRead (0);
  luzLed=map(luzLdr,1023,0,0,255);
  analogWrite (led,luzLed);
  delay (500);
}
```

Versión 23/05/2021
Por: *Pedro Ruiz Fernández*