Prácticas

Proyectos I

Ingeniería del Software

U-TAD

Guillermo casado

2021-2022

1

Práctica 1: Primeros pasos

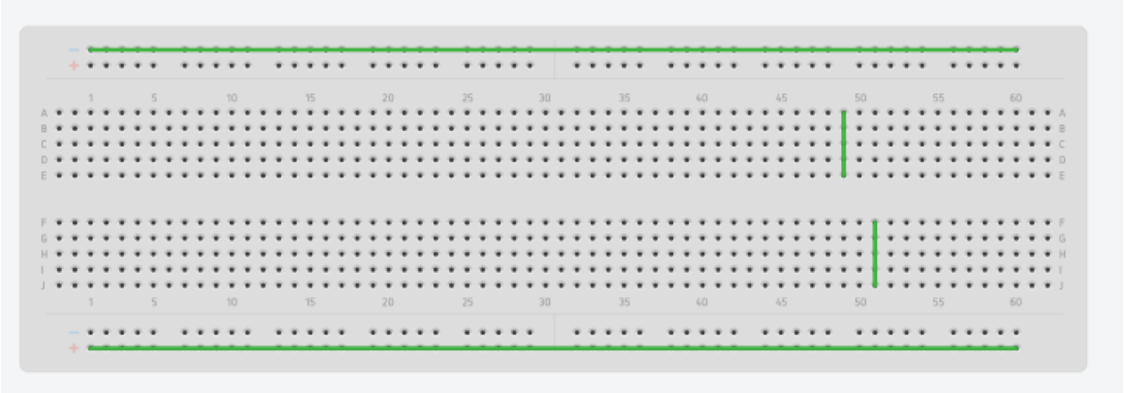
- Conocer arduino
- Arduino ide
- Programar arduino.
- Blink básico: Conectar en Arduino y luego modificar
- Usar breadboard
- Poner varios leds conectados a un mismo puerto

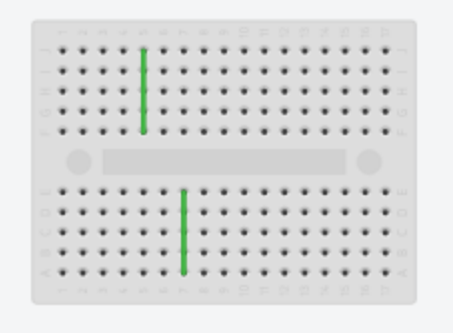
Breadboard

- Sirve para montar los circuitos de un prototipo sin necesidad de soldar o utilizar conectores para unir los elementos.

- Consiste en multitud de puntos interconectados de cierta forma que facilita la conexión directa de componentes de un circuito.

- Las líneas de las imágenes indican qué zonas están conectadas entre sí.





LED

Light Emiting Diode

TIene **polaridad**: los extremos deben estar conectados a voltajes mayor (+, **ánodo**) y menor (-, **cátodo**)

Hasta que no tiene un voltaje dado en sus bornes ($\mathbf{V_D}$) no comienza a emitir luz (curva característica)

Normalmente se pone una **resistencia** para controlar el voltaje en sus bornes y que disipa potencia extra.

Los LEDs pueden soportar una **Intensidad máxima ($\mathbf{I_D}$)**. A partir de ella se calcula la potencia que debe disipar la resistencia y su valor.

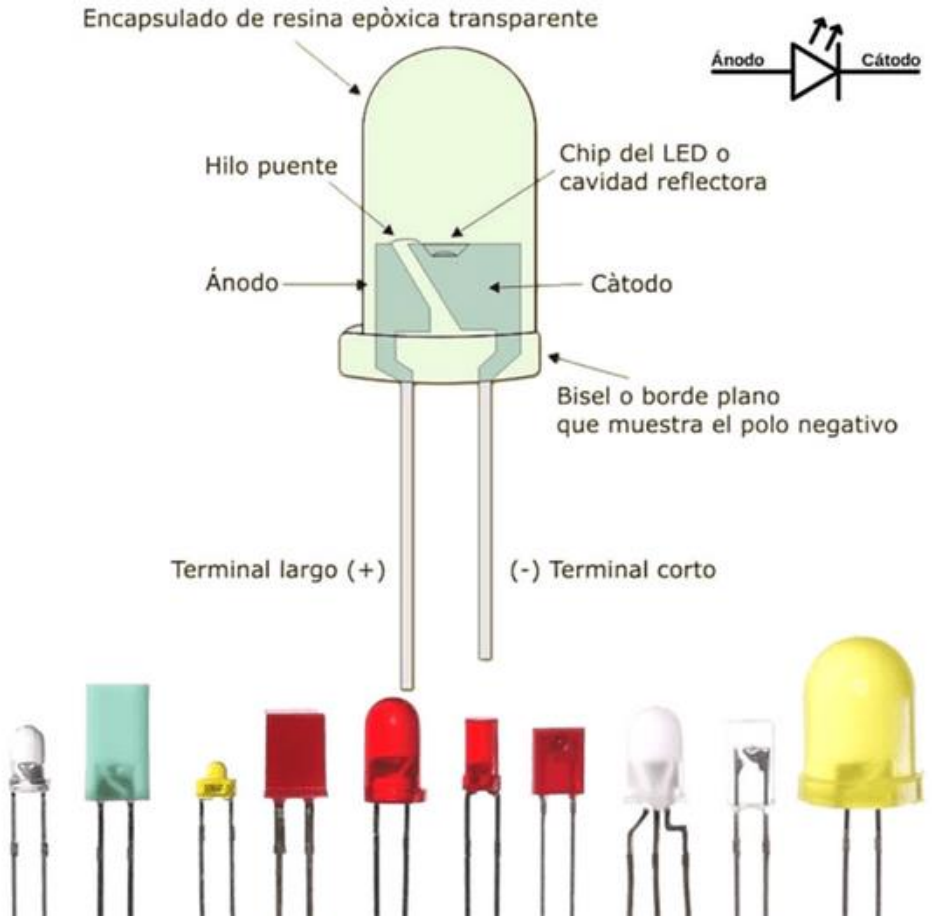
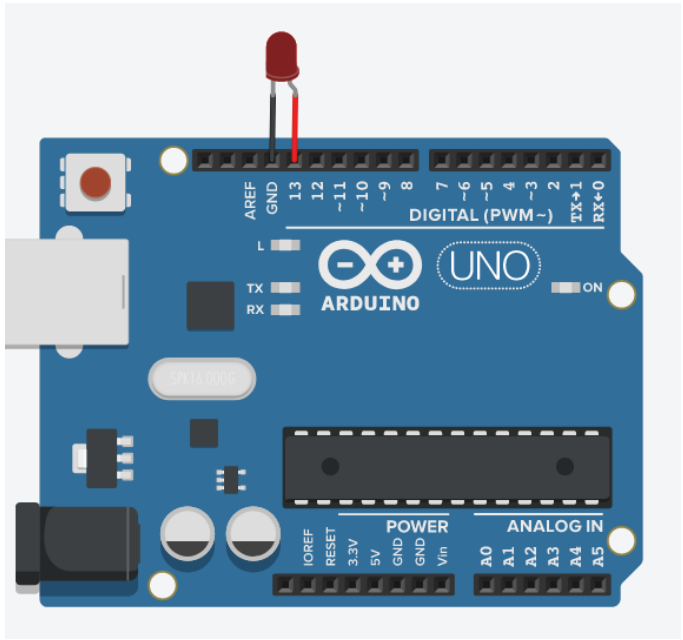
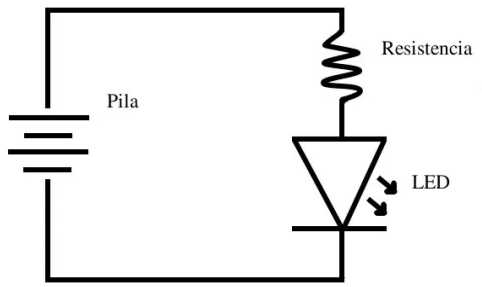
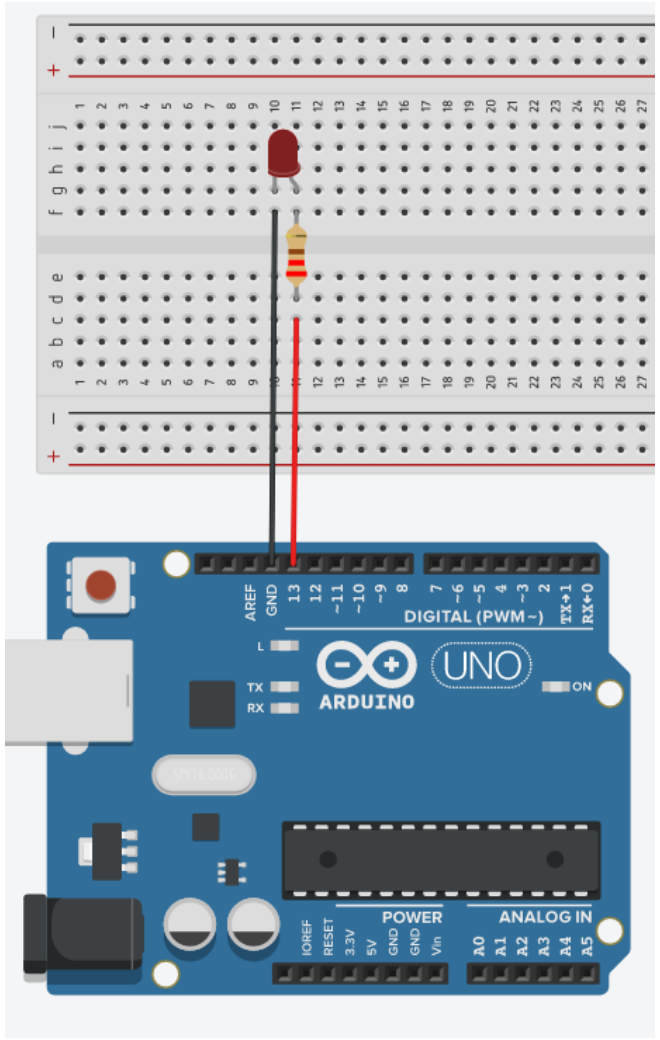


Gráfico LED: http://j-rpm.com/2017/05/led-corriente-de-trabajo/

LED

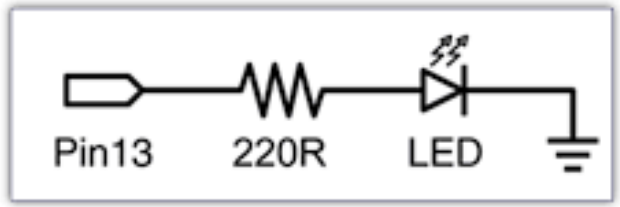






Resistencia 220 Ohm
Valdría una resistencia de 160 Ohm
Voltaje 5V
I_D se suele poner 20mA

2. Cálculo resistencia de led



V(arduino) = V(led) + V(resistencia);

- I (led): 20 mA
- V (arduino): 5 V
- V (led): 1,8 V

V = I · R; R = V / I = (5-1,8) / 0,02 = 3,2 / 0,02 = 160 Ω

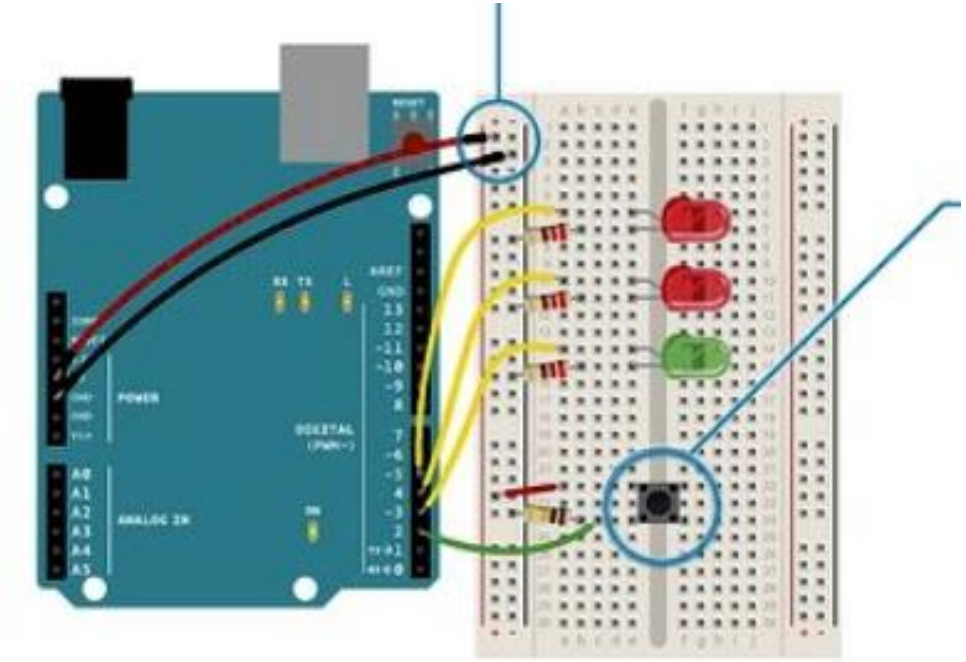
Color	Tensión umbral
Rojo	1.9v
Amarillo	1.7v a 2v
Verde	2.4v
Naranja	2.4v
Blanco	3.4v
Azul	3.4v

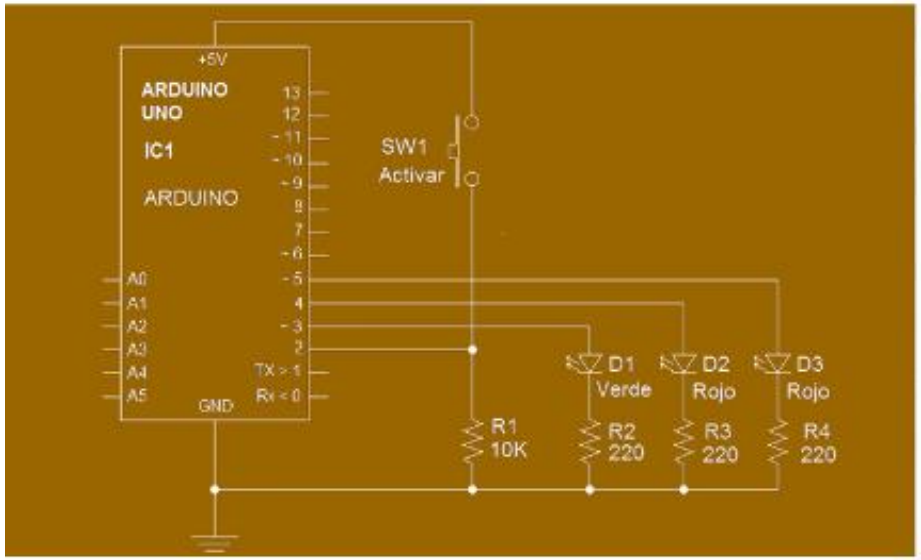
2

Práctica 2:

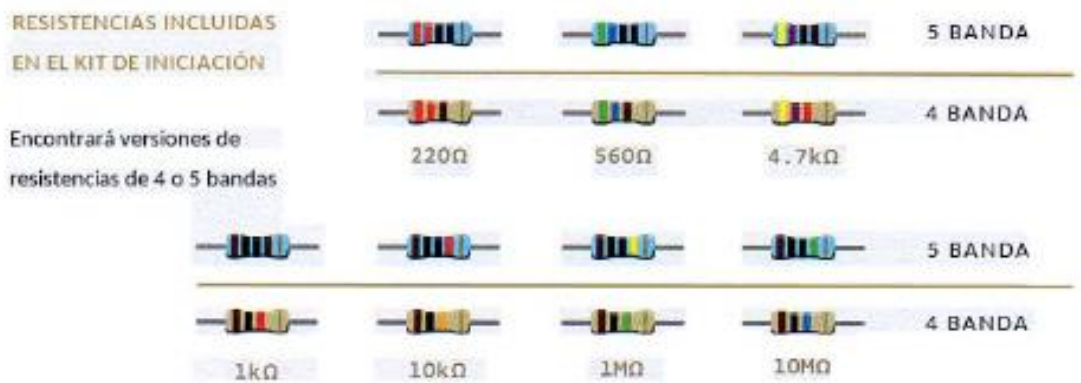
- LEDs + interruptor
- Nueva orden: `digitalRead()`
- Switch en pin 2.
- Programación:
  - Si switch off: Led Verde on, Rojos off
  - Si switch on: Led verde off, rojos on alternativamente cada 250ms

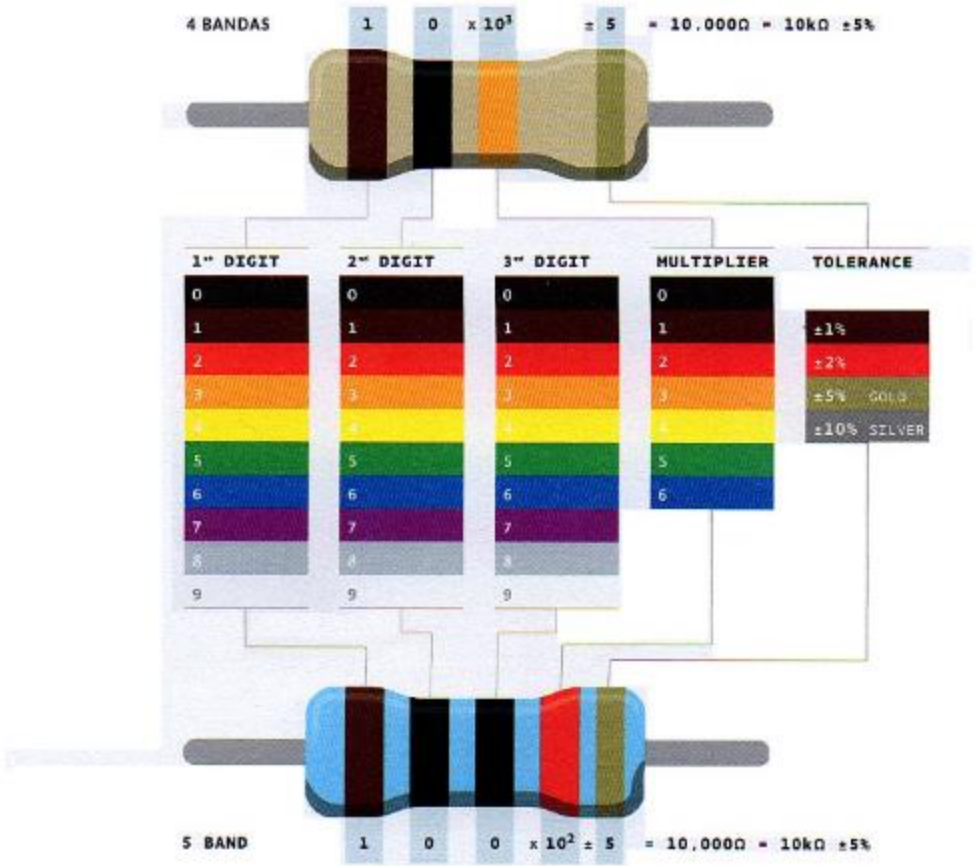
Extra: ¿Cómo cambiamos la programación para que los LEDs parpadeen sin utilizar delay()?





Resistencias





Código - Estructura general

```
// A partir de los simbolos // lo escrito son comentarios. No se procesan
// Sirven para ayudar a entender el código. MUY ÚTILES.
//
// Llamar librerías
// Declarar variables globales
int pin1 = 12;

// Inicialización del programa
void setup() {
   // ordenes de inicialización de variables y más...
}
// Bucle contínuo
void loop() {
   // acciones que se ejecutan cíclicamente
   // Leer inputs
   // lógica del programa
   // fijar outputs
}
```

Código

En setup():

Declarar un pin digital como INPUT u OUTPUT

```
pinMode(numPin, INPUT / OUTPUT);
```

En loop():

// Leer el estado de un pin:

```
int valor = digitalRead(numPin); // debe ser INPUT
```

// Fijar el estado de un pin digital OUTPUT

```
digitalWrite(numPin, HIGH / LOW);
```

// Esperar un tiempo

```
delay(1000); // Esto esperaría 1000 ms, esto es 1 segundo
```

3

Práctica 3: Sensor de humedad y temperatura

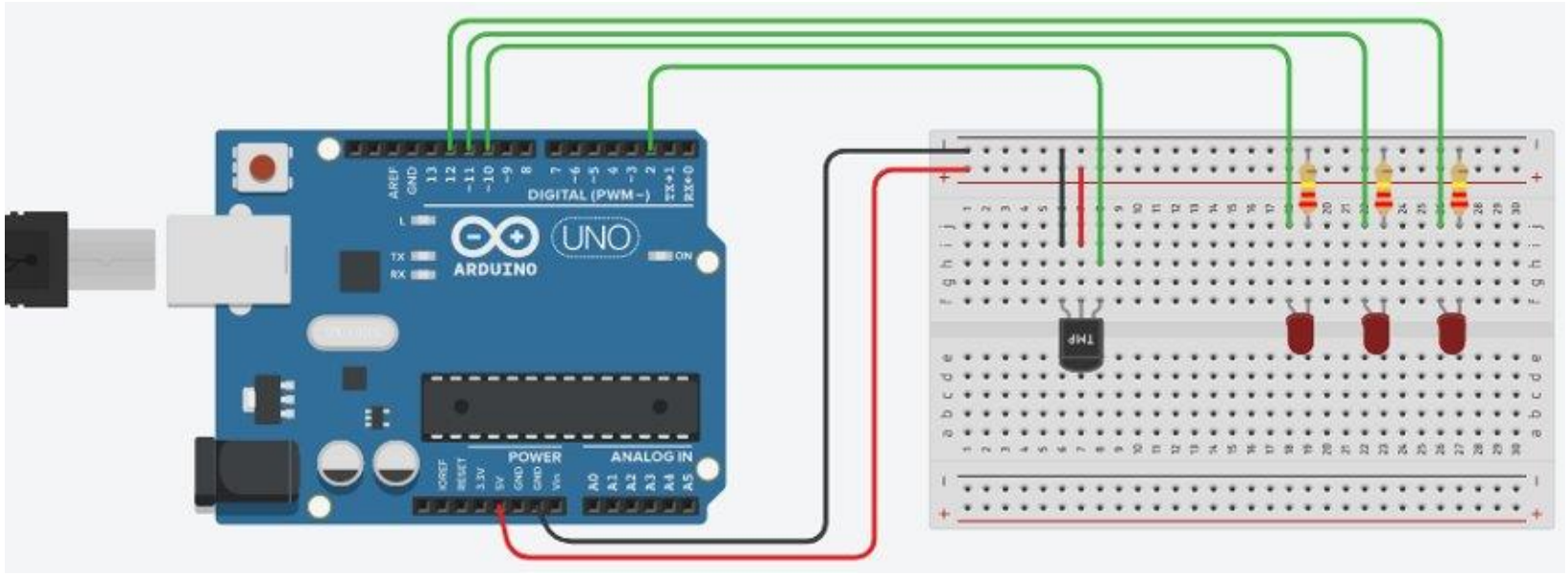
- Sensor digital DHT11: Temperatura y humedad
  En el "Libro de proyectos de Arduino" se utiliza un sensor analógico (Love-O-Meter)

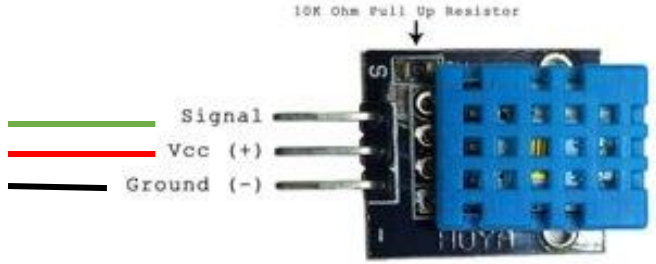
- **Librería** necesaria, una de las siguientes: **DHT11**, kit Elegoo, Adafruit... Utilizaremos DHT11.

- **Lectura de input digital**. La función se incluye en
  la biblioteca. Recoge los valores de temperatura y humedad y los guarda en variables

- Escritura de datos en **puerto serie**: Serial().

Práctica 3: montaje



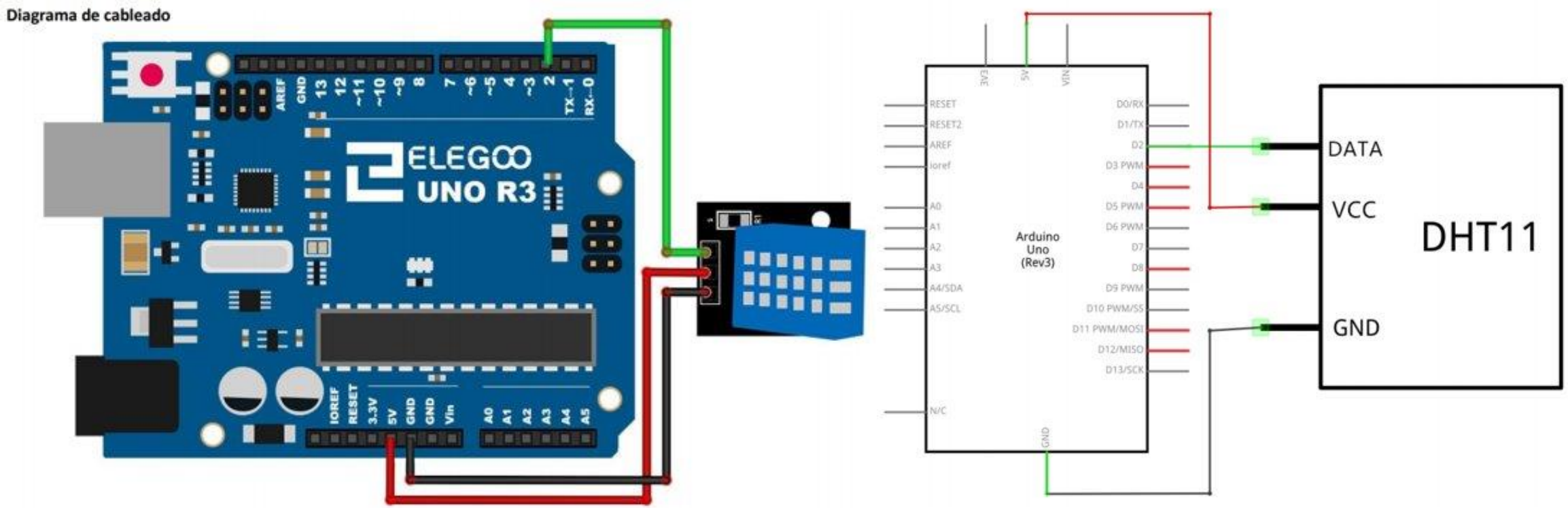


Práctica 3: código de lectura del sensor

```
// http://www.circuitbasics.com/how-to-set-up-the-dht11-humidity-sensor-on-an-arduino/

#include <dht.h>

dht DHT;

#define DHT11_PIN 2

void setup() {
  // Abrir puerto serie
  Serial.begin(9600);
  Serial.println("Hola Mundo!");
}
```

```
void loop() {
  // put your main code here, to run repeatedly:
  int chk = DHT.read11(DHT11_PIN);
  delay(1500); // importante
  Serial.print("Temperature = ");
  Serial.println(DHT.temperature);
  Serial.print("Humidity = ");
  Serial.println(DHT.humidity);
  delay(500);
}
```

Práctica 3: Sensor DHT11 – Elegoo doc



Práctica 3: Sensor DHT11

MODELO	DHT11
Alimentación	de 3,5 V a 5 V
Consumo	2,5 mA
Señal desalida	Digital
Temperatura	
Rango	de 0ºC a 50ºC
Precisión	a 25ºC ± 2ºC
Resolución	1ºC (8-bit)
Humedad	
Rango	de 20% RH a 90% RH
Precisión	entre 0ºC y 50ºC ± 5% RH
Resolución	1% RH

Práctica 3: Serial

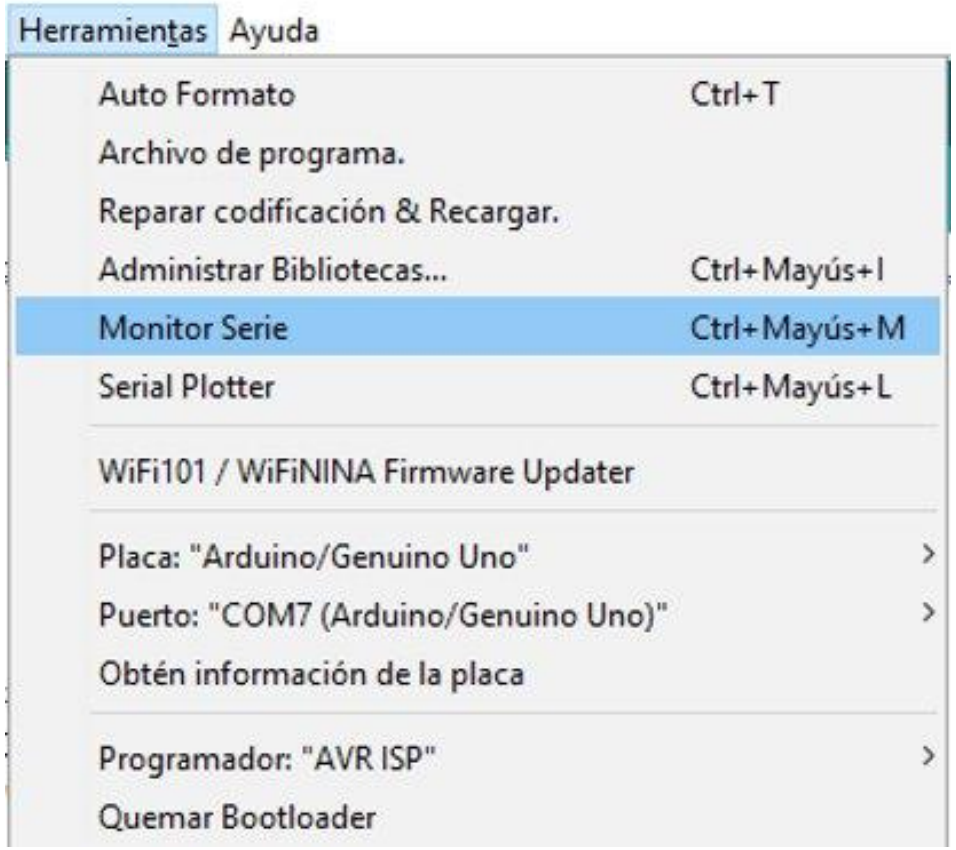
- Abrir puerto:

// en setup():

```
Serial.begin(9600); // baudios (bits por segundo)
```

- Para escribir en el Serial:

```
Serial.print("Hola qué tal ");
Serial.print(variable);
Serial.println("Al final
  de escribir con println se salta de línea");
```



Práctica 3: montaje y objetivos

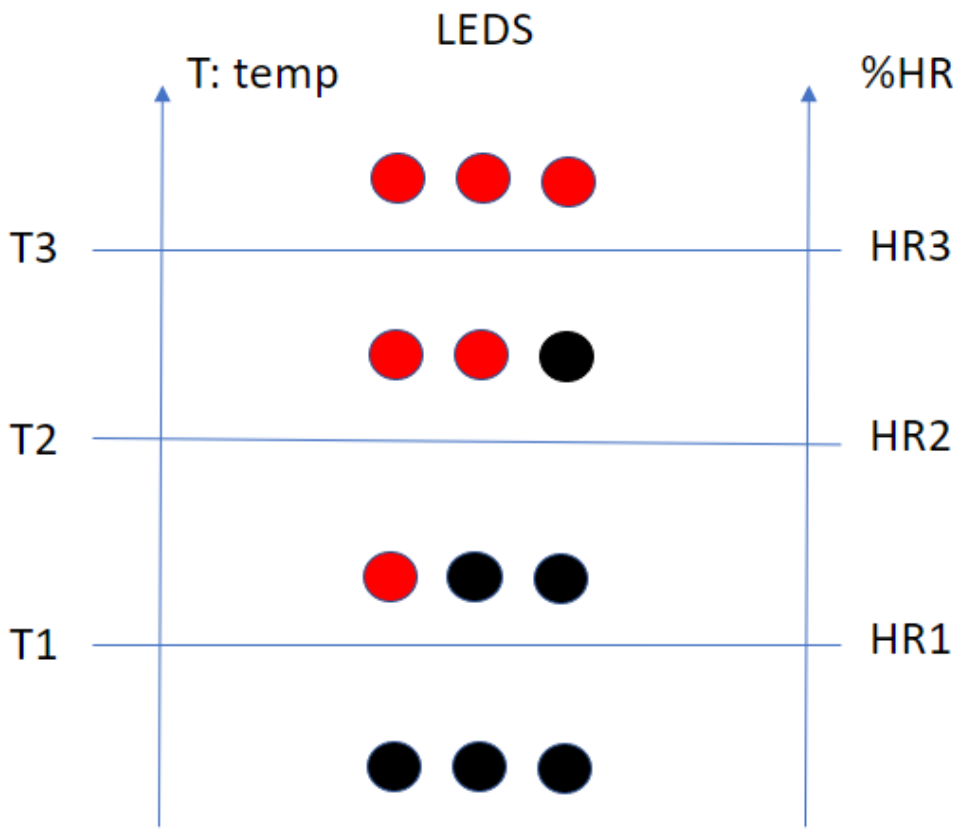
- Preparar el montaje de la figura. En la primera parte solo se va a utilizar el sensor.

- Comprobar qué se ve con el **monitor Serial**
- Comprobar qué se ve con el **Serial Plotter**
- Probar a escribir los datos de temp/humedad en una sola linea en el Serial. Comprobar y comparar qué se ve en monitor Serial y en Serial Plotter

- **Montar el sensor más un sistema de 3 LEDs** que debe indicar si la temperatura está entre ciertos rangos de temperaturas (ver página siguiente)
- Reprogramar el sistema para que haga lo mismo pero con la humedad relativa.

- **Extra**: Cómo hacemos para incorporar un pulsador al sistema de forma que si se pulsa el sistema de LEDs informa de la humedad relativa en un rango dado de %HR, y si no esta pulsado se informa de la temperatura.

Práctica 3: Ejemplo LEDs



Hay que fijar 3 temperaturas de referencia y, según la temperatura del sensor, encender 0, 1, 2, o 3 LEDs dependiendo del rango en el que esté la temperatura del sensor (DHT.temperature)

Intentar lo mismo pero con la humedad relativa.

Extra: Incorporar al sistema un pulsador para que muestre la valoración de la temperatura o la de la HR según esté accionado o no.

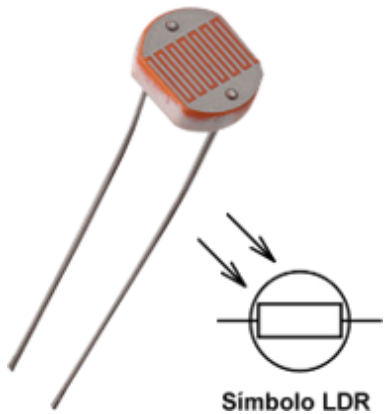
Referencias web: sensores DHTxx

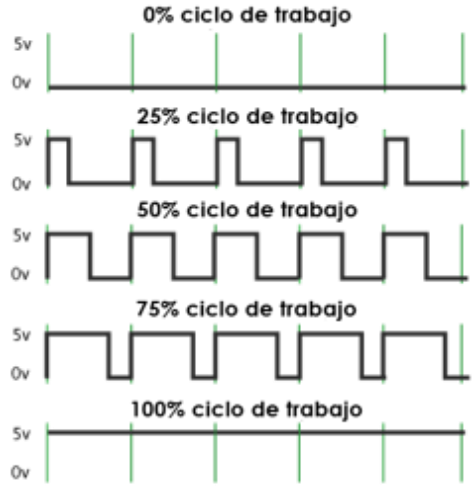
- http://www.circuitbasics.com/how-to-set-up-the-dht11-humidity-sensor-on-an-arduino/
- https://programarfacil.com/blog/arduino-blog/sensor-dht11-temperatura-humedad-arduino/
- https://learn.adafruit.com/dht/overview

4

Práctica 4: Lámpara de colores

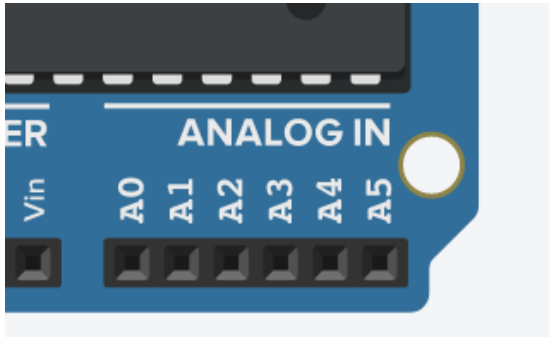
- **Fotorresistor**: (LDR) Resistencia variable con la luz. **De 500 Ω a 50 kΩ**. Es un semiconductor.
- **LED RGB**: contiene 3 LEDs (rojo, verde, azul) independientes
- **PWM**: **Pulse-Width Modulation**. Permite establecer niveles de intensidad en un LED. **Puertos digitales con ~** (ALT 126)



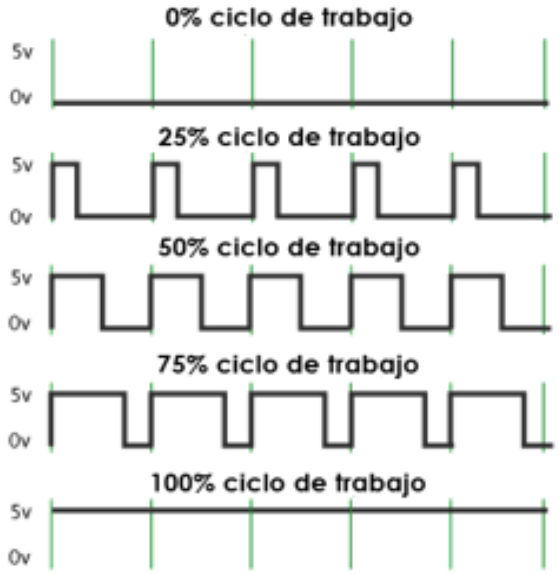


Práctica 4 - Comandos

- **`analogRead(numPIN)`**:
  Lee datos de una **entrada analógica** (A0, A1, A2...)
  Da valores entre 0 y 1023, que corresponden a valores de voltaje medido entre 0 y 5V. El voltaje de referencia es la tierra (ground) de Arduino.



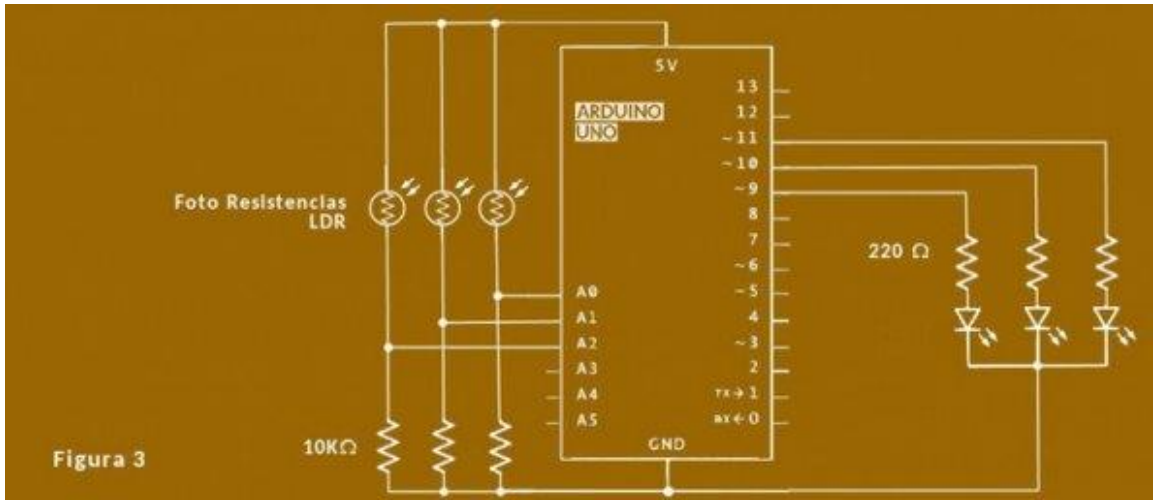
- **`analogWrite(numPin, valor)`**:
  Para poner un valor diferente de 0 (LOW) o 5V (HIGH) en una salida digital.
  Se consigue on latécnica PWM: lanzando pulsos de 5V de mayor o menor duración en el tiempo, de forma que en promedio resulta mayor o menor brillo.
  A la función se le pasan valores de 0 a 255

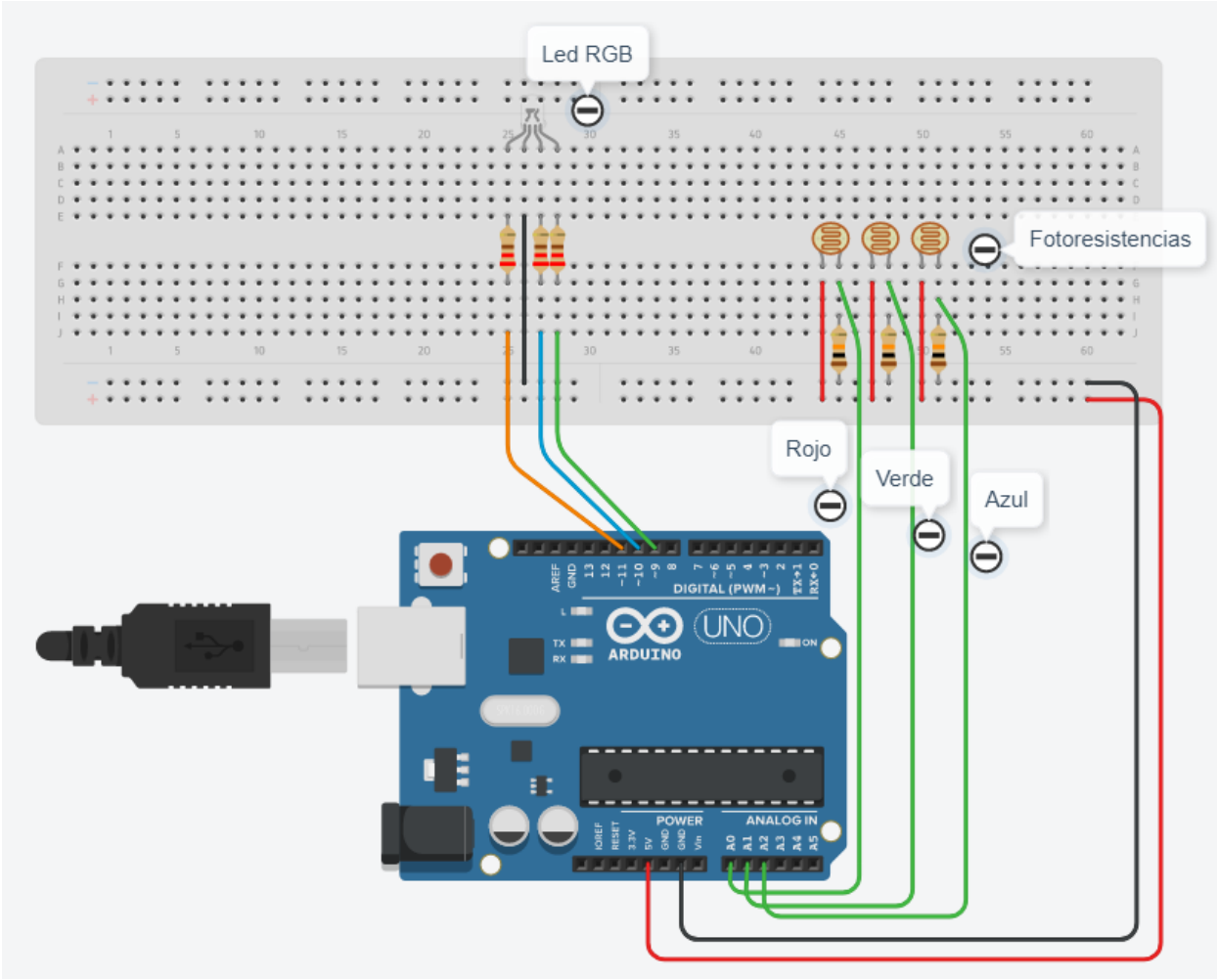


Práctica 4: circuito

Con cada fotoresistencia (LDR) se controla la intensidad de un canal RGB

El montaje de clase se puede hacer con menos LDR's y variar menos canales RGB





Práctica 4: código

- Setup:
  - Preparar los pines digitales como OUTPUT

- Loop:
  - Leer el valor de los pines analógicos.
    Esperar 5ms después de leer cada uno.
  - Convertir el valor leído del rango 0-1023 al rango 0-255. Función **`map()`**
  - Pasar el valor leído a alguno de los canales digitales mediante **`analogWrite()`**

El código se puede encontrar en el libro de proyectos de Arduino, capítulo 4, pero es recomendable desarrollarlo personalmente.

Práctica 4: Función map()

- Tenemos que pasar un valor leído de un puerto analógico (rango de 0 a 1023) a otro de puerto digital PWM (rango 0-255)

```
vAlog = analogRead(A0); // 0-1023
```

- 3 opciones

1) Podemos dividir entre 4 como aproximación: `vDig = vAlog/4`

2) Podemos hacer una regla de 3: `vDig = vAlog*255/1023;`

3) Utilizar la función **map()** es lo más cómodo. Transfoma una variable entre un rango de v alores a otro rango diferente.

```
vDig = map(vAlog,    0,1023,   0,255);
map(valor_a_transformar,      rango1_i, rango1_f,    rango2_I, rango2_f);
```

4) Es habitual utilizar después la función **constrain()** para limitar los valores de la variable a un rango definido:

```
vDig = constrain(vDig, 0,255);  // Lo ajustamos a los valores finales
vDig = constrain(vDig, vMin,Max);  // Lo ajustamos a los valores finales
```

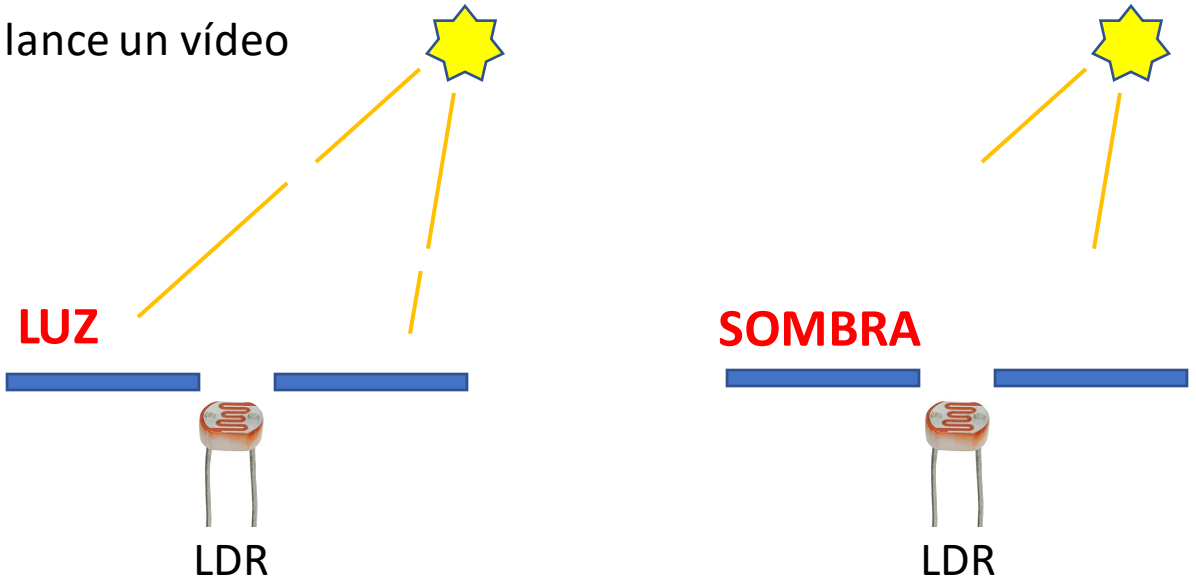
Práctica 4: Calibrar rangos de trabajo

- Se comprobará experimentalmente que el valor leído en A0 cuando la LDR **no recibe nada de luz (v0)** y cuando recibe **mucha luz (v1)** es muy diferente para cada montaje.
- Por ejemplo:
  - **`v0 (oscuridad) = 220;`**
  - **`v1 (mucha luz) = 840;`**
- Por lo tanto el rango de trabajo es entre 220 y 840 más que entre 0 y 1023.
- Se puede regular cambiando los límites 0-255 en la función map. Por ejemplo, para los valores de arriba:
- **`vDig = map(vAlog, 220,840, 0,255);`**

```
map(valor_a_transformar, v0_oscuro, v1_luz, rango2_I, rango2_f);
```

Práctica 4: Uso como botón

- Con una LDR podemos hacer un botón gestual touchless.
- Basta con iluminar la LDR con un foco, y detectar cuando se tapa la luz.
- En ese momento se puede desencadenar algún otro proceso:
  - activar mecanismo,
  - lanzar aviso sonoro,
  - enviar una señal a un ordenador para que lance un vídeo
  - …



5

Práctica 5: Theremin Óptico

https://www.youtube.com/watch?v=ajM4vYCZMZk

Práctica 5: Theremin Óptico

- Zumbador pasivo
  o piezoeléctrico: vibra
  cuando se le aplica un voltaje

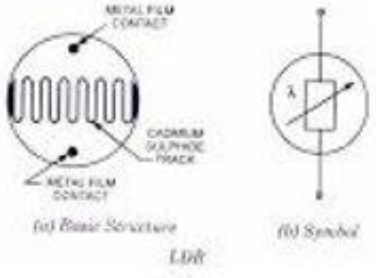
SI

Zumbador pasivo

NO

Zumbador
activo (con circuito)

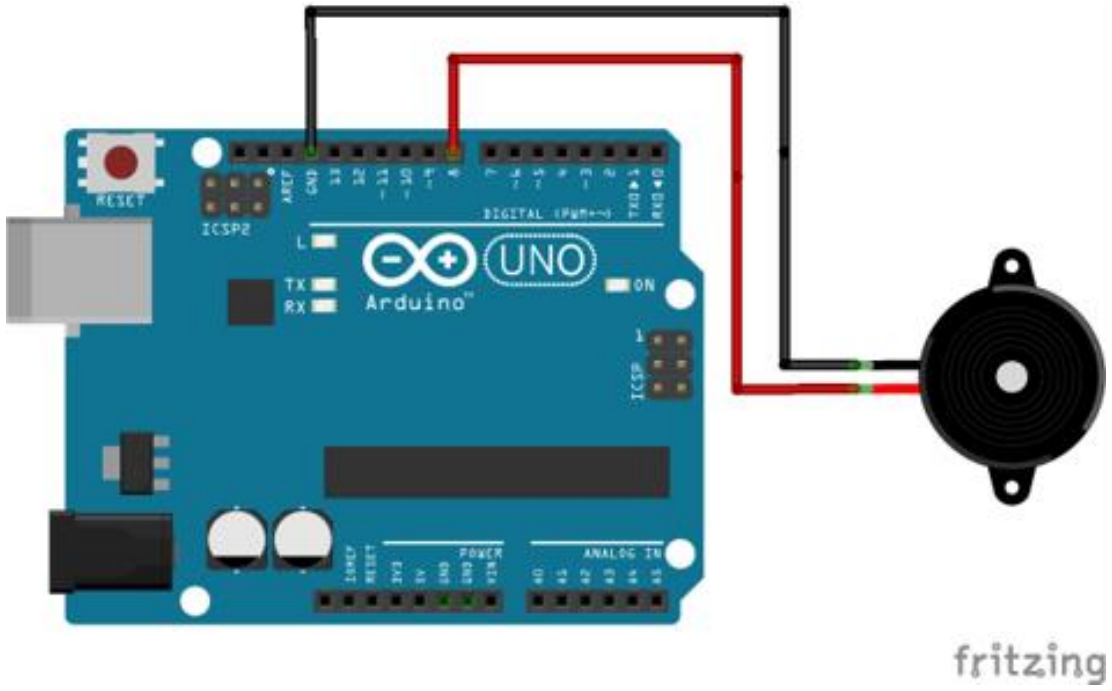
- Fotorresistor: (LDR)
  Resistencia variable con la luz.
  De 500 Ohm a 50 kOhm



Práctica 5: (1) Zumbador

- Se conecta directamente a un PIN digital. No es necesario que sea un pin con PWM.
La otra pata se conecta a Ground

- Conectarlo en breadboard.



Práctica 5: (1) Zumbador-código

- Se utiliza `tone()` para hacer que el piezo vibre a una cierta frecuencia.
- Referencias:
- manual Arduino - tone()
- tutorial tone()

```
tone(pin, frequency)
tone(pin, frequency, duration)
```

pin: the Arduino pin on which to generate the tone.
frequency: the frequency of the tone in hertz. Allowed data types: `unsigned int`.
duration: the duration of the tone in milliseconds (optional). Allowed data types: `unsigned long`.

Práctica 5: (1) Zumbador-código

- El código es simplemente:
- setup():
  - Poner el pin del zumbador como OUTPUT
- loop():
  - Utilizar la funcion tone() y un delay equivalente a la duración del tone()

```
int pitch = 100; // entre 50 y 4000
tone(pinZumbador, pitch, 20); // duracion 20 ms.
delay(20); // probar a poner diferentes valores comaprados con la duracion
           // del tono
```

  - Probad diferentes valores para pitch para la duración y el delay;

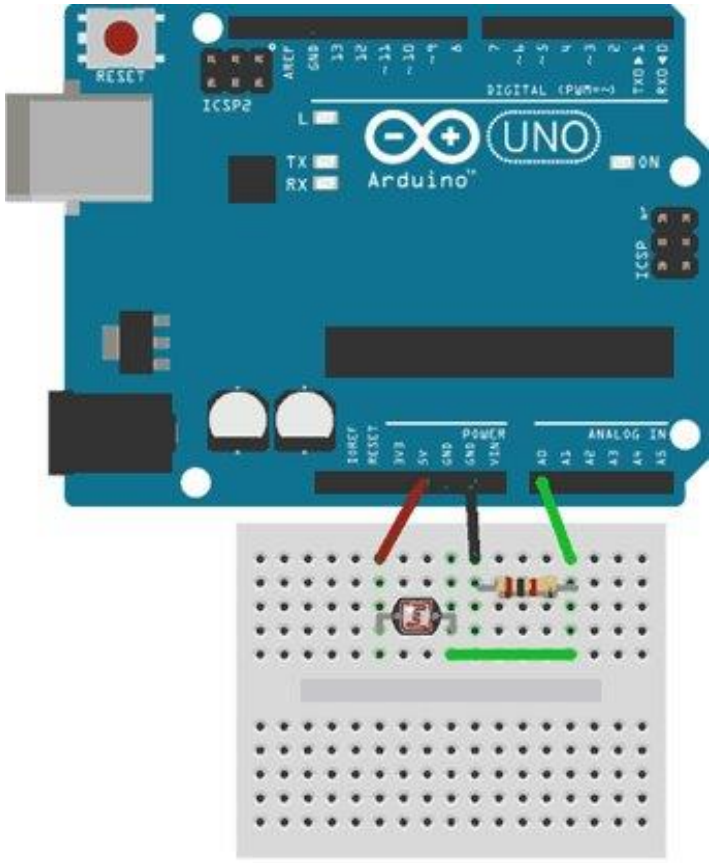
Práctica 5: (2) LDR

- Se monta el circuito que vimos en la práctica 4 (divisor de tensión), con una resistencia de 10k

- El código es el que utilizamos para leer el valor en el pin analógico A0.

- **Calibración**
- Conviene encontrar experimentalmente el valor minimo y el máximo que devuelve la LDR y guardarlos en dos variables:

- `int sensorLow = (valor medido)`
- `int sensorHigh = (valor medido)`



Práctica 5: (3) Código Theremin

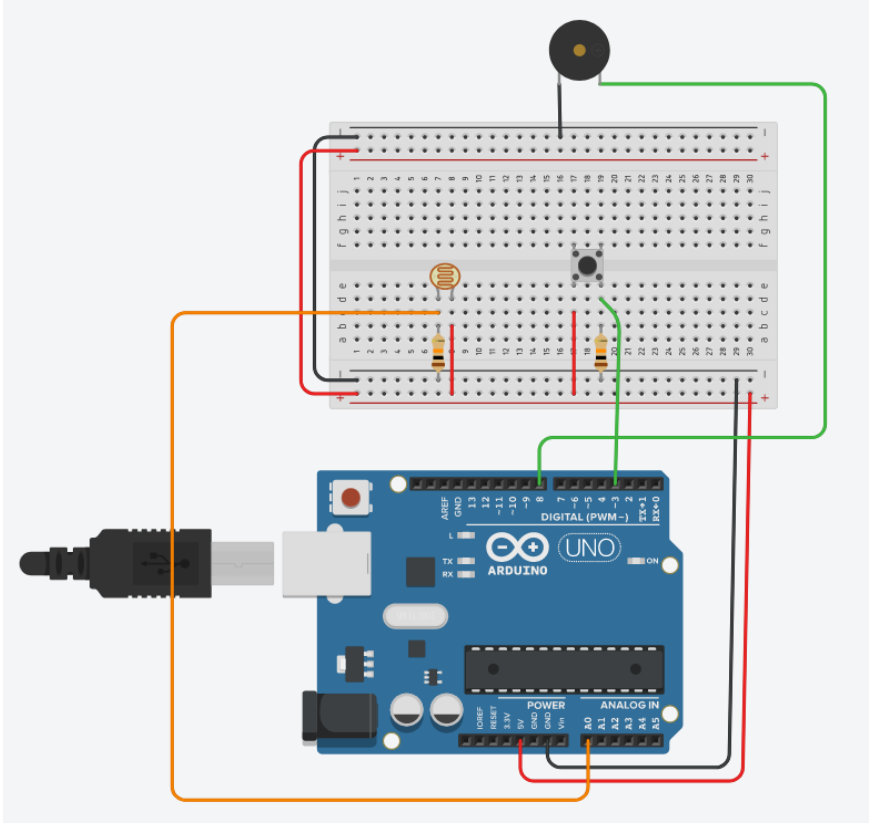
- La señal medida en el sensor LDR (con `analogRead()`) se pasa como frecuencia al zumbador. Para modificar el valor leído con `analogRead(A0)` y pasarlo al pin con `tone()` utilizaremos la función `map()`:

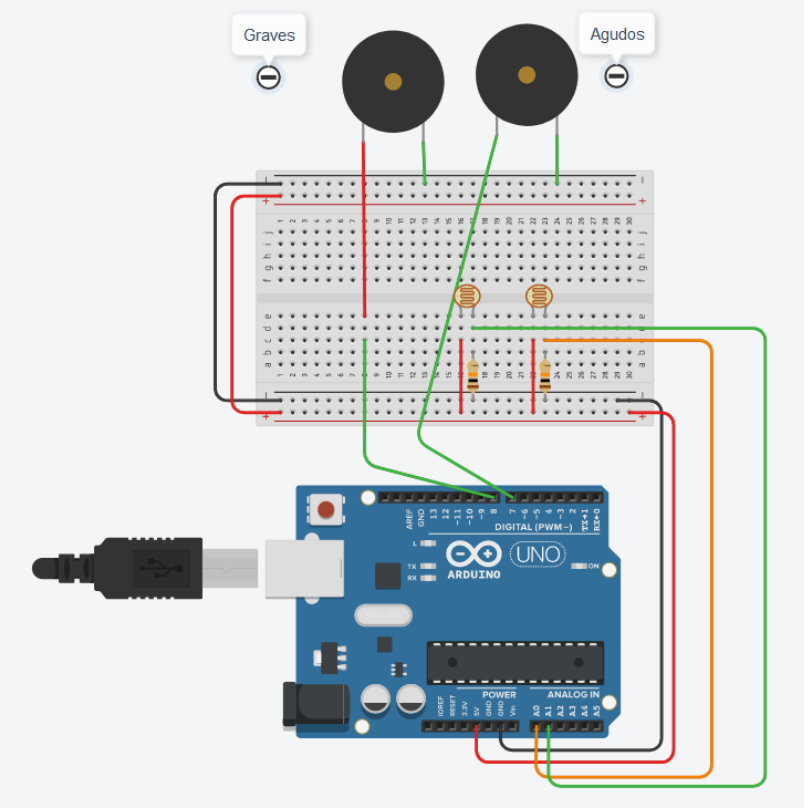
En loop():

```
int sensorValue = analogRead(A0);
// Aquí 50 y 4000 son las frecuencias mínima y máxima donde quiero
// que se mapee el valor detectado en la LDR.
int pitch = map(sensorValue, sensorLow,sensorHigh, 50,4000);
tone(8, pitch, 20);
delay(20);
```

Práctica 5: (4) Pulsador

- Podemos añadir un pulsador para que el sistema solo suene cuando el puslador esté pulsado. Otra opción es que el pulsador haga que el sistema suene o este silenciado cuando lo pulsemos y liberemos.
- Otra opción puede ser tener dos zumbadores y dos LDRs. Un zumbador actua en un rango de frecuencias y el otro en otro rango diferente (agudos y graves respectivamente, por ejemplo)





Práctica 5: (XTRA) Reproducir melodías

Se puede reproducir melodías con el zumbador

Utilizar el ejemplo subido a blackboard (ejemplo 7 del CD de Elegoo)

El fichero pitch.h contiene las equivalencias de notas musicales con frecuencias. Ej:

```
#define NOTE_B0  31
#define NOTE_C1  33
#define NOTE_CS1 35
#define NOTE_D1  37
```

Se reproduce una secuencia de tonos definida en el programa principal.

6

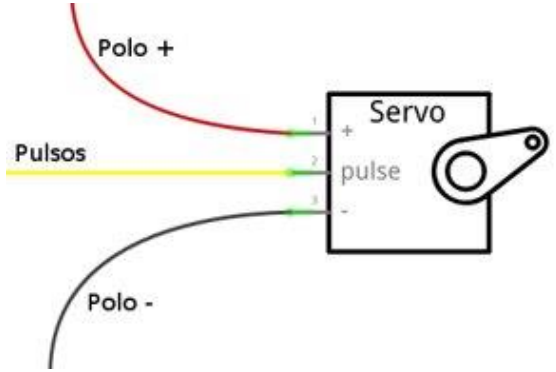
Práctica 6: Servomotor y potenciómetro

Capítulo 5 del Libro
de Proyectos de Arduino:

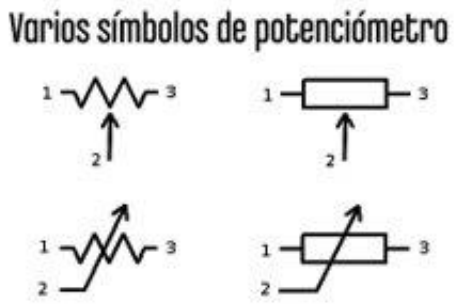
Indicador de estado de ánimo

Práctica 6: Servomotor y potenciómetro

- **Servomotor**:
  mecanismo que permite posicionar el giro de un eje a un a ángulo determinado
  Gira hasta 180º
  Funciona por pulsos (PWM)
  Librería servo.



- **Potenciómetro**:
  Resistencia que varía su valor con un mando físico.
  No tiene polaridad.

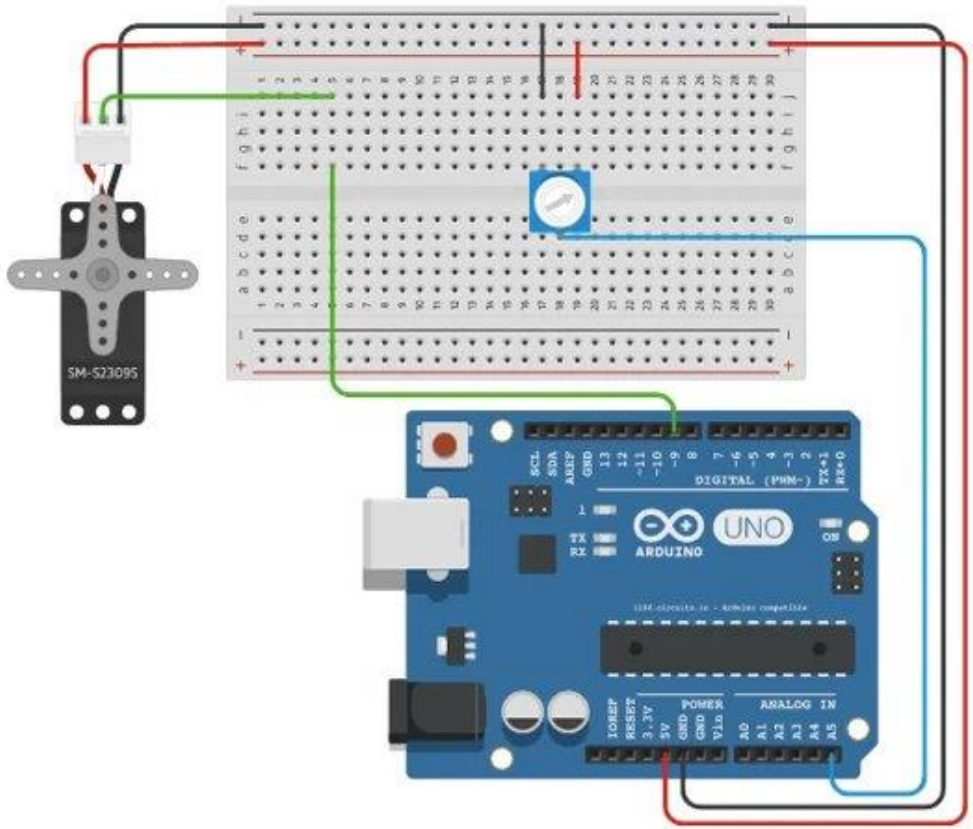


Práctica 6: Servomotor y potenciómetro

- **Objetivo**:
- Controlar la posición del servomotor girando un potenciómetro.

- El sistema debe:
  - leer el valor del potenciómetro desde un puerto analógico
  - convertir el valor leído en el puerto analógico y utilizarlo para variar la posición (giro) del servomotor. Usar map().
  - Se puede adaptar el sistema para que nos sirva de indicador de una medida añadiéndole un marcador físico.

Práctica 6: Servomotor y potenciómetro



Práctica 6: Servomotor y potenciómetro

Código:

Al principio: Importar librería Servo

```
#include <Servo.h>
Servo myServo; // declaramos una variable de tipo Servo
int valPot; // valor del potenciometro
int angServo; // valor del angulo de giro del servo
```

En setup():

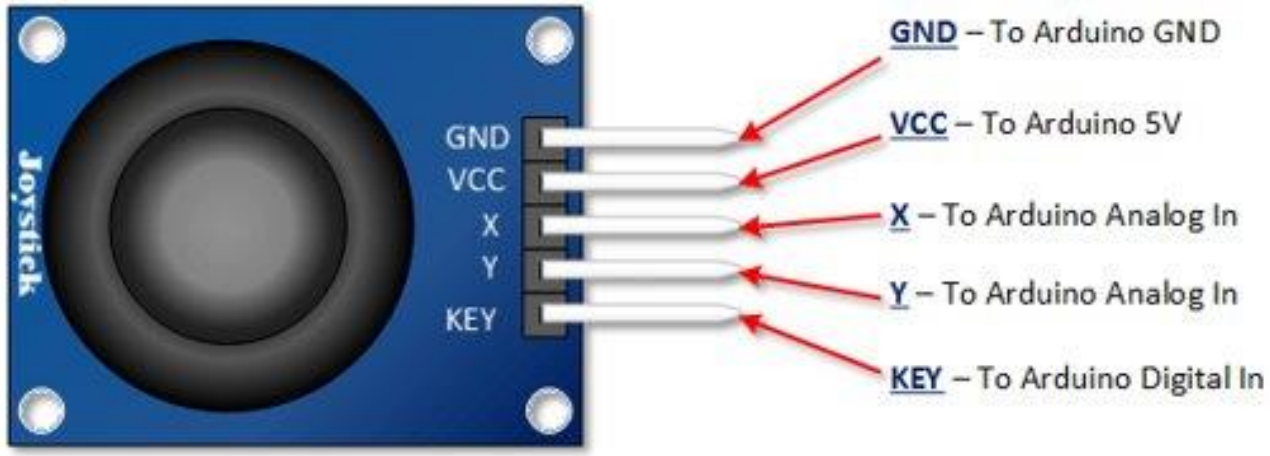
```
myServo.attach(9); // pin digital PWM donde esta el servo
Serial.begin(9600); // iniciar puerto serial para ver que pasa
```

En loop():

```
// Leer el valor del input analógico con analogRead
valPot = analogRead(A0);  // Valor entre 0-1023
// Transformarlo al rango de valores del servomotor con map
angServo = map(valPot, 0,1023, 0,179);
myServo.write(angServo);
delay(15);
```

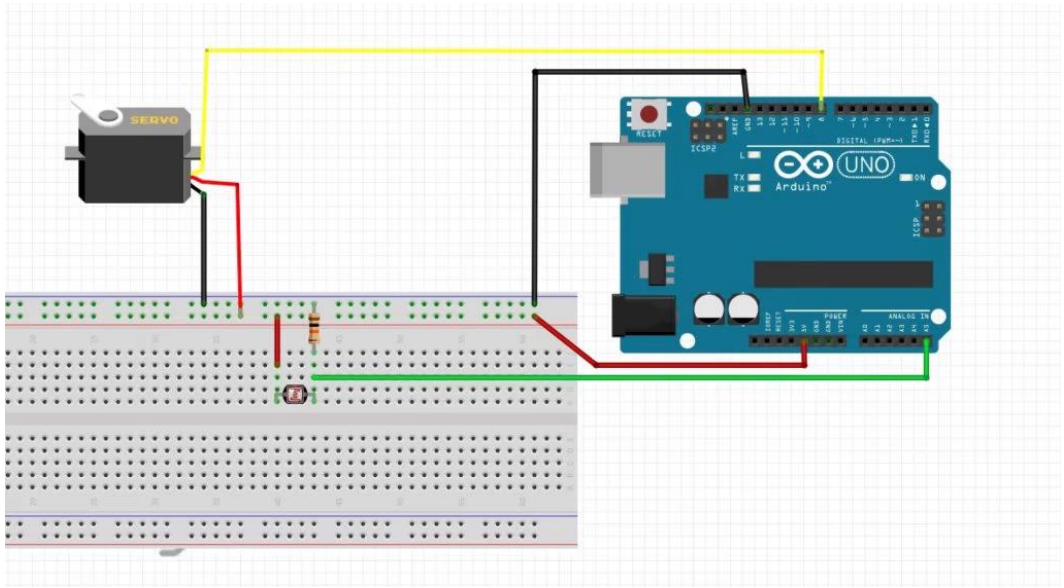
Práctica 6: Servomotor y potenciómetro

- **Joystik Analógico**:
  Podemos sustituir directamente el potenciómetro por un joystick analógico.
- El joystik tiene dos pontenciómetros: uno para el eje X y otro para el eje Y.
- Además puede disponer de un pulsador (interruptor).



Ejemplo de uso de Servomotor y LDR

- https://hackergirl.com.mx/tutoriales/tutorial-superar-el-videojuego-del-dinosaurio-de-chrome-con-arduino/



7

Práctica 7: LCD Screen

Capítulo 11 del Libro de Proyectos de Arduino:

Bola de Cristal

Práctica 7: LCD Screen

- **LCD Screen:**

Pantalla que permite representar hasta 32 caracteres (2 filas de 16 caracteres).

El montaje tiene bastantes cables, así que hay que hacerlo cuidadosemante.

Se incorpora un potenciómetro para controlar el contraste de la pantalla.

Librería LiquidCrystal

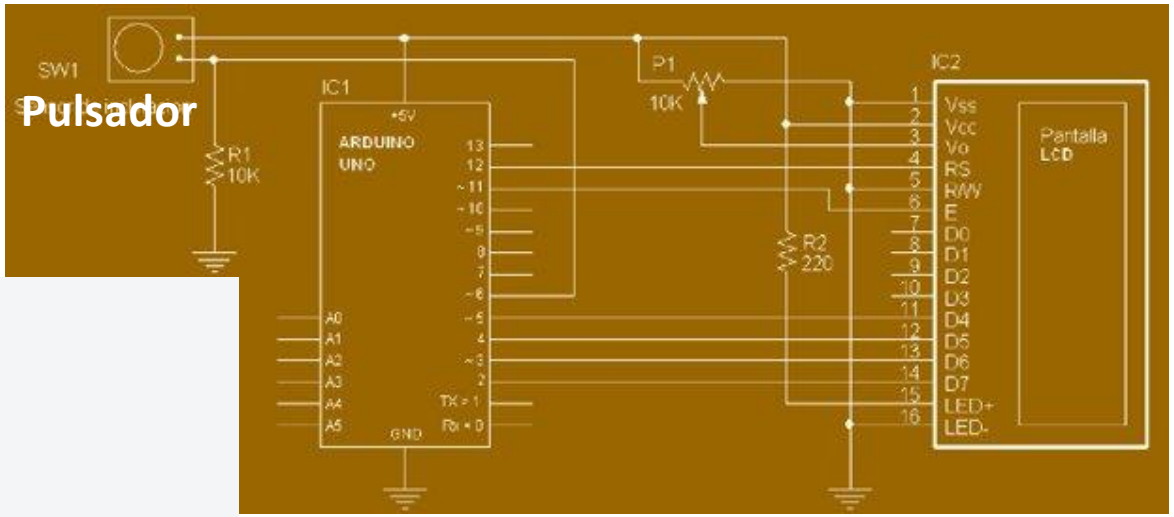
- **Funcionamiento**

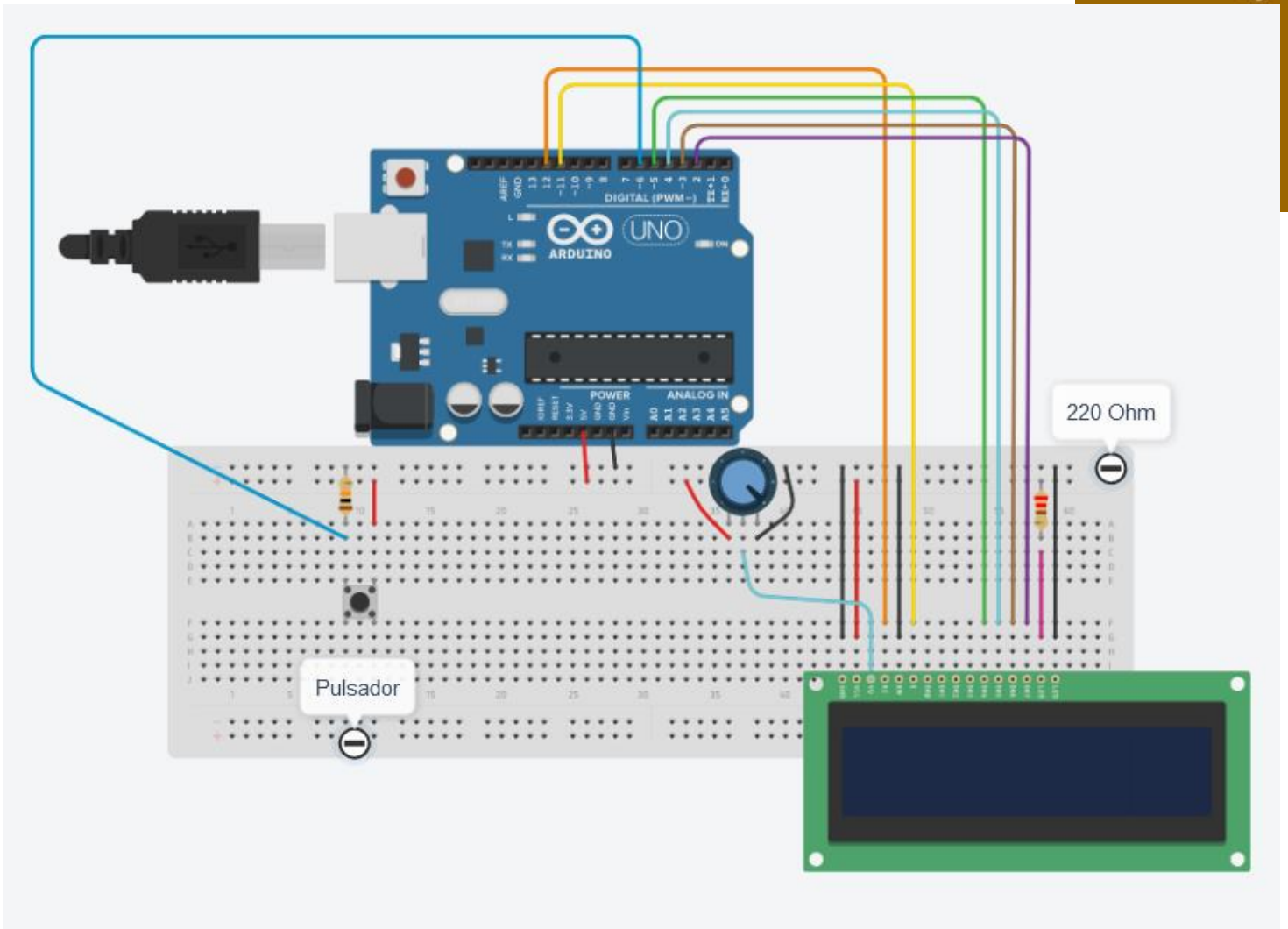
La pantalla (bola) sugiere que hagas una pregunta y luego pulses un interruptor (o un sensor tilt). Entonces la pantalla responde una de entre 8 opciones posibles elegida al azar. Al cabo de un tiempo el sistema debería ponerse en el modo de inicio.

- **Orden switch() / case / break**

Ve para realizar diferentes acciones dependiendo del valor de una variable.

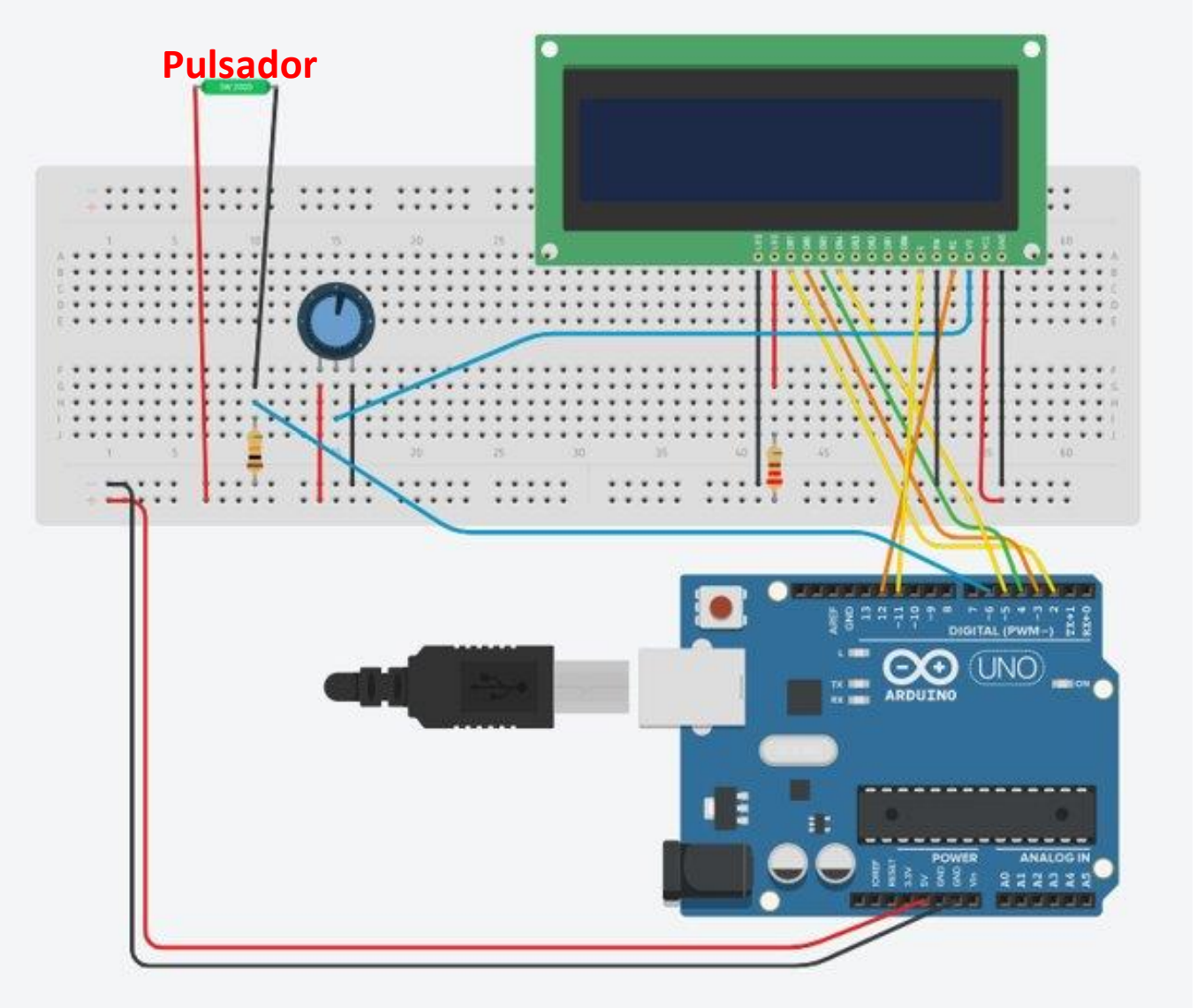
Práctica 7: LCD Screen





En el libro de proyectos se explican brevemente qué es cada pin de la pantalla y para qué sirve.

Práctica 7: LCD Screen



Práctica 7: LCD Screen

- **Código:**

Chequear que la pantalla esta bien conectada cargando el ejemplo de arduino:

- **Código. Seguir el código del libro de proyectos:**

Hay que importar la librería LiquidCrystal y declarar un objeto LiquidCrystal que podemos llamar `lcd`, y hay que indicar los números de PIN a los que esta conectada.

- **Mensajes de la bola**

Estado inicial: "Que dice / La bola"

Respuestas: "La bola dice:"

0 -"Si"; 1-"Probablemente"; 2-"Desde luego"; 3-"Tiene buena pinta";

4-"No estoy segura"; 5-"Vuelve a preguntar"; 6-"Tengo dudas"; 7-"No"

Práctica 7: LCD Screen

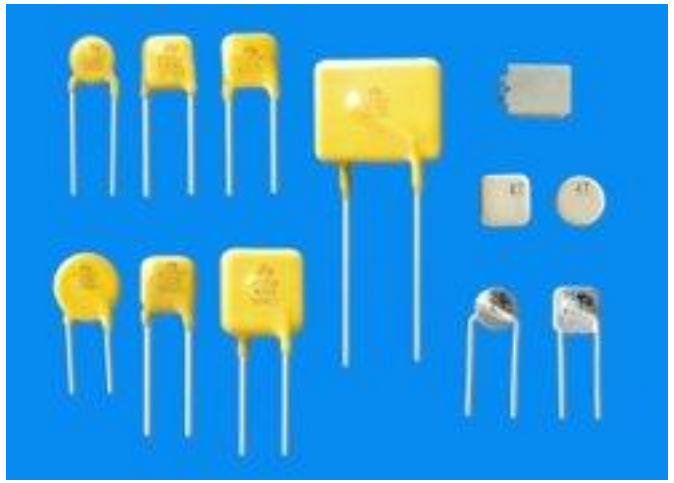
Librería LiquidCrystal

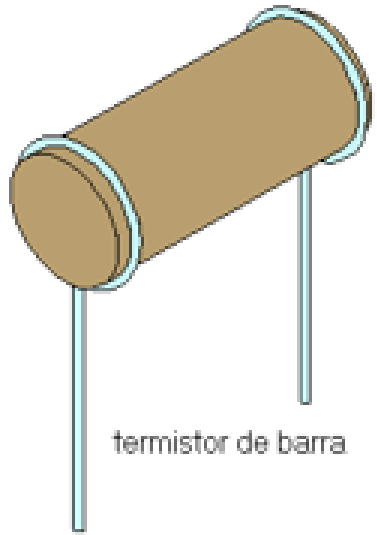
La librería tiene mas funcionalidades: link del cursor, scroll de textos...

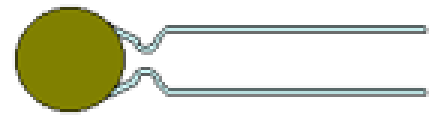
Verlas aquí: https://www.arduino.cc/en/Reference/LiquidCrystal

8

Práctica 8: Termistor







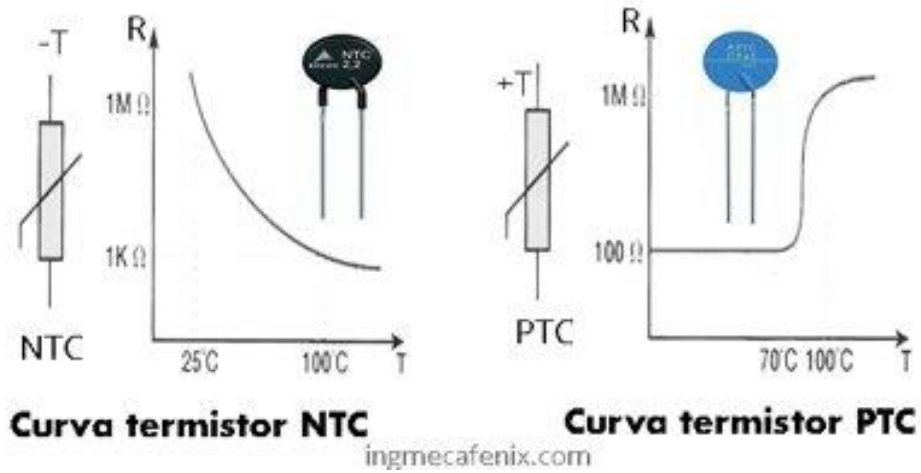
Cap 15 – pdf de Proyectos de Elegoo

Termistor

- Es un sensor compuesto de material **semiconductor**. Su **conductividad** varía con la **temperatura** en la que se encuentre.

- Dos tipos:
  - **NTC: Negative Temperature Coeficient**: resistencia disminute con la Tª
  - **PTC: Positive Temperature Coeficient**: resistencia aumenta con la Tª

- La dependencia de la resistencia con la temperatura no es lineal



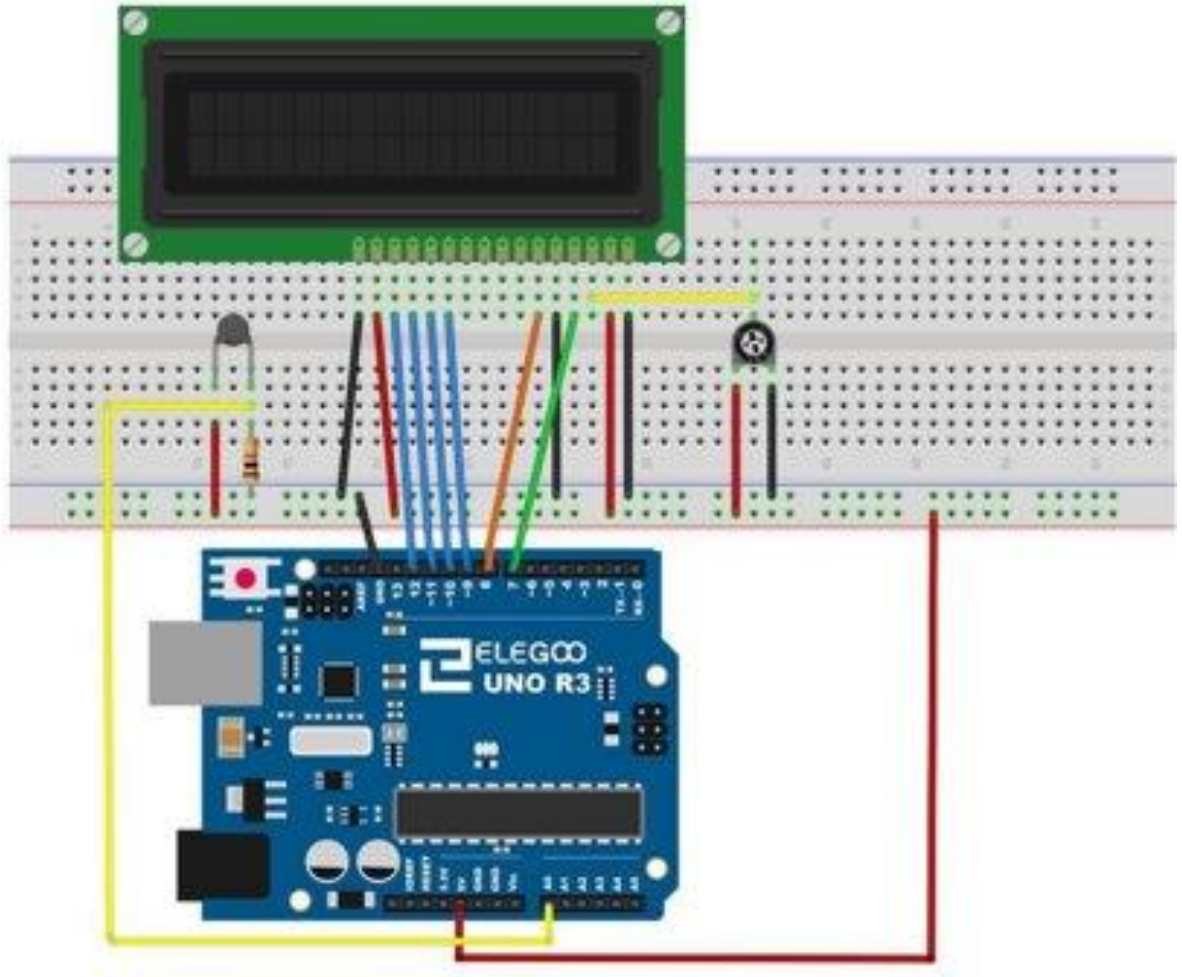
Curva termistor NTC **Curva termistor PTC**

Termistor

- Es un sensor analógico, como un potenciómetro, así que se conectará a un **puerto analógico**.
- Vamos a conectar un **display LCD** para mostrar el valor medido de la temperatura.
- Realizaremos una **media** de los últimos valores.
- También conectaremos un **sensor digital DHT11** y comparamos las medidas.

Termistor - circuito

- Resistencia: 10K
- Recomendable extender el termistor con cables para hacer una sonda.
- **El montaje de la pantalla está simplificado. Mejor utilizar el de la práctica de pantalla LCD.**
- En Tinkercad no hay termistor. Se puede simular con un potenciómetro.



Termistor - Código

- Hay que configurar el pin analógico para medir datos de entrada.
- Hay que configurar la parte de la pantalla LCD.

- **Cálculo de la T:**

```
float tempReading = analogRead(A0);
double tempK = log(10000.0 * ((1024.0 / tempReading - 1)));
tempK = 1 / (0.001129148 + (0.000234125 + (0.0000000876741 * tempK * tempK )) * tempK );
float tempC = tempK - 273.15;
float tempF = (tempC * 9.0)/ 5.0 +32.0;
```

Termistor - Código

- **Cálculo de la media**.

Utilizaremos un array en el que guardar los últimos `N` valores medidos: `valores[]`

Cada vez que se lea un nuevo valor hay que copir primero los valores previos a una posición anteior del array:

```
valores[i+1] = valores[i]
```

```
valores[0] = nuevoValor;
```

La media es la suma de los valores dividido para N.

No es necesario medir el valor cada loop(), sino que se puede hacer cada cierto intervalo de tiempo `dtMedida`.

9

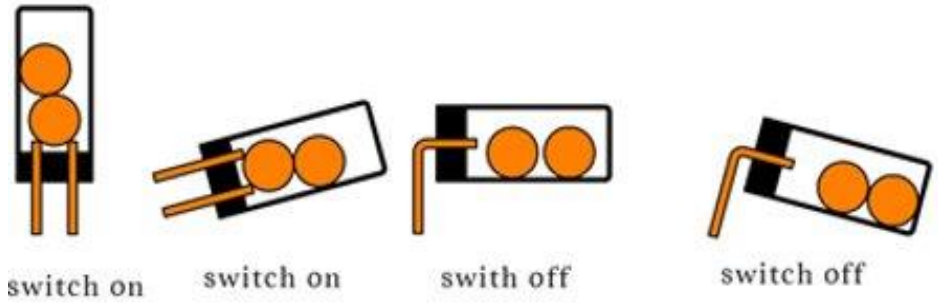
Práctica 9: Sensor tilt de inclinación

Capítulo 6 del Libro
de Proyectos de Arduino:

Reloj de arena digital

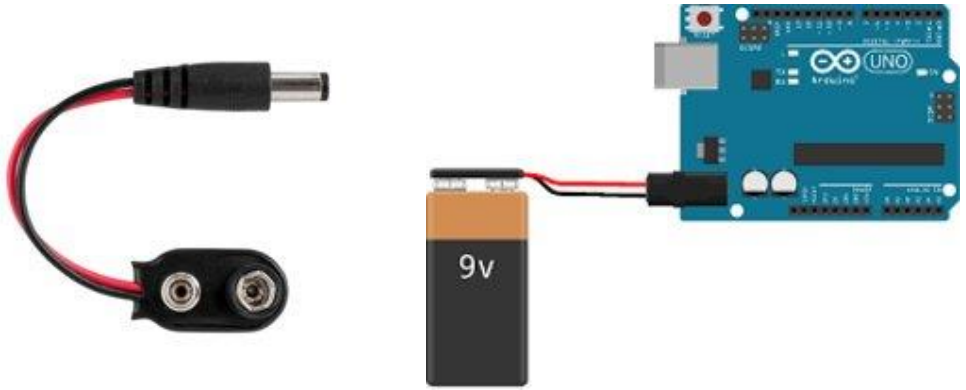
Práctica 9: Sensor tilt de inclinación

- **Sensor tilt de inclinación**:
  Es un sensor que pone
  el circuito abierto cuando no está vertical.
  También se puede utilizar para detectar vibraciones.



- **Circuito de LEDs**:
  Control de varios LEDs.
  Utilizaremos el tiempo para ir encendiendo los
  LEDs uno a uno

- **Uso de baterías**



Práctica 9: Sensor tilt de inclinación

Tiempo en Arduino

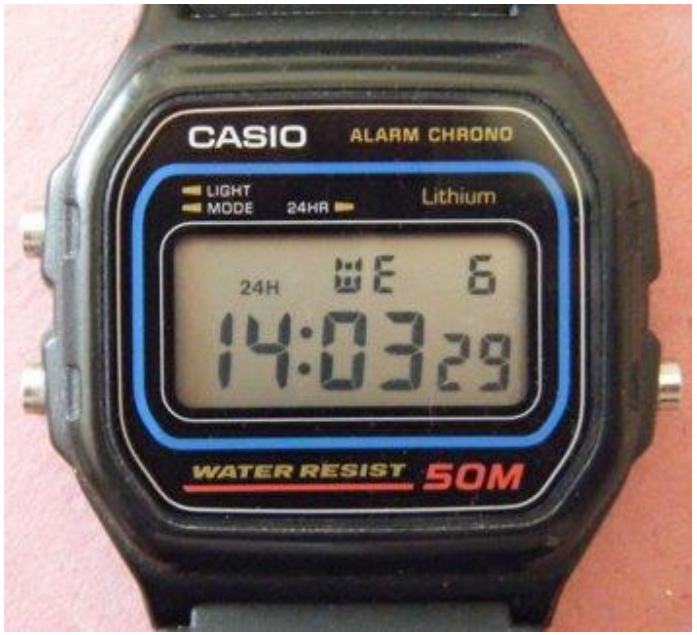
int es de 16 bits: de -32.768 a 32.768

long es de 32 bits: de -2.147.483.648 a 2.147.483.648

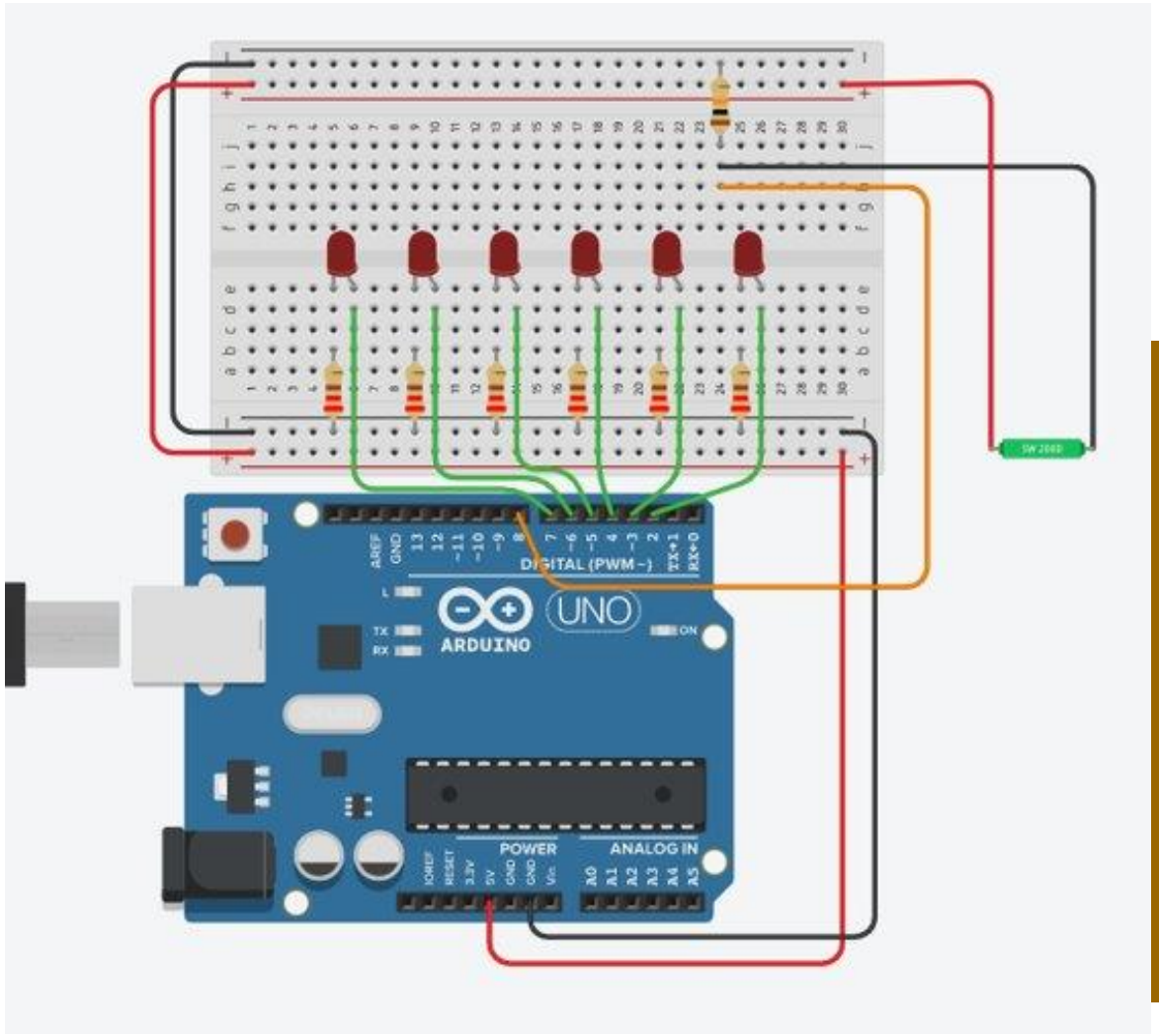
unsigned long es de 32 bits pero sin signo: de 0 a 2 * 2.147.483.648

Guardaremos la variable de tiempo en tipo unsigned long

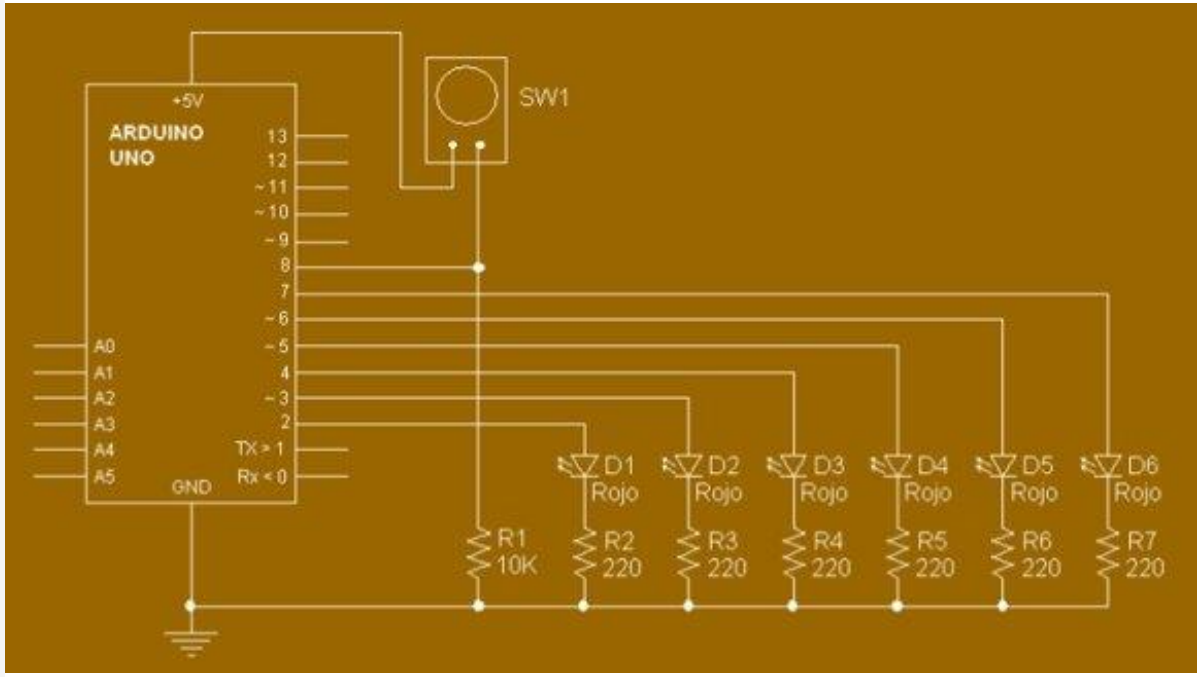
```
unsigned long tiempoActual = millis();
```



Práctica 9: Sensor tilt de inclinación



Poner al menos 4 LEDs

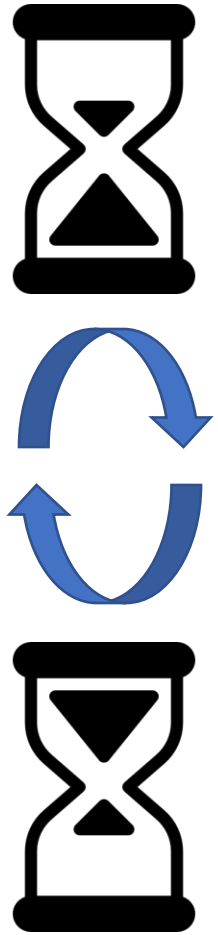


Práctica 9: Sensor tilt de inclinación

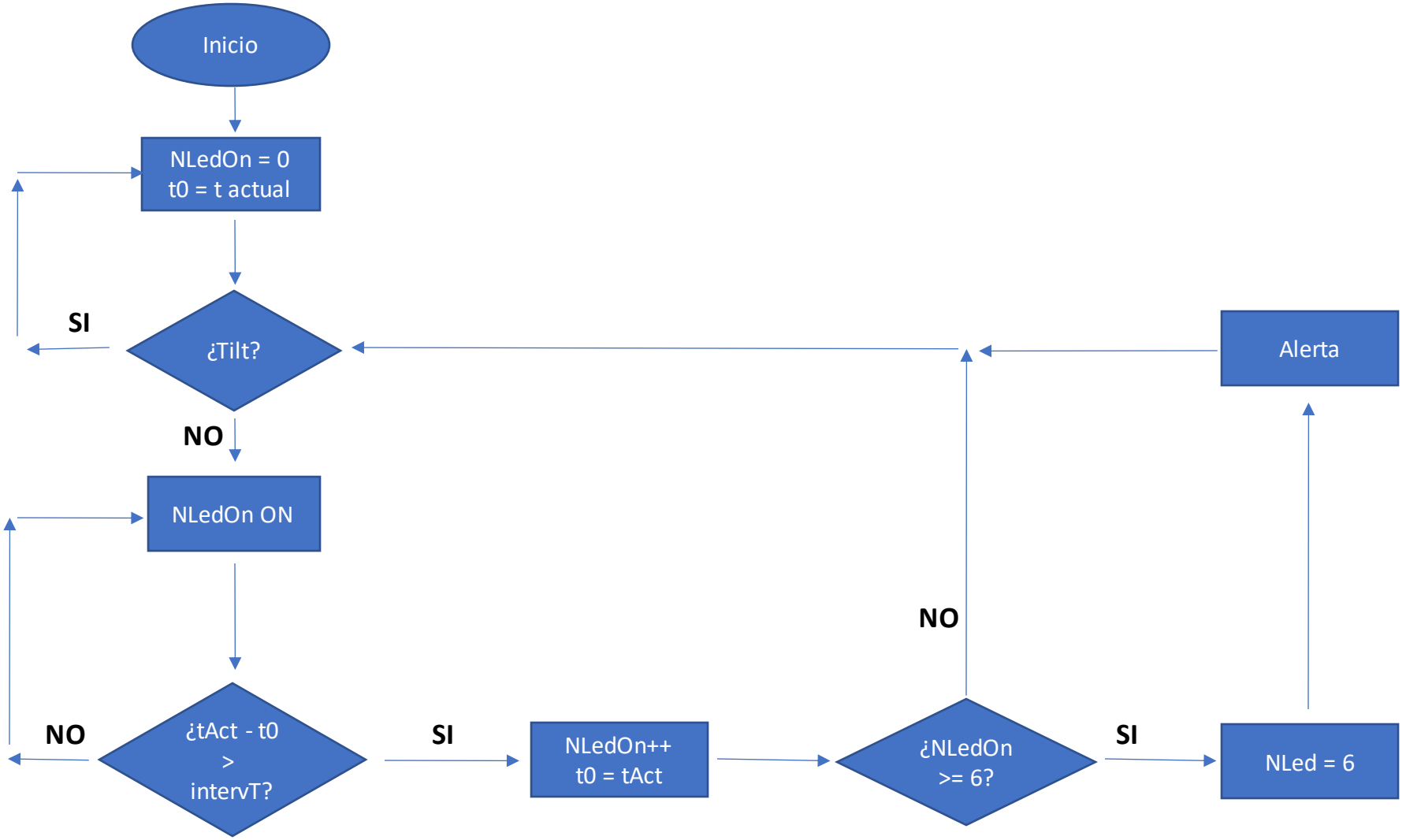
- El **objetivo** es que cada cierto **intervalo de tiempo** se encienda un LED, así hasta que se enciendan los 6 LEDs del circuito.

- En ese momento, para que comience a contar habrá que darle la vuelta al reloj. El sensor tilt indicará que se ha dado la vuelta y hay que comenzar el proceso. Si no se etectan cambios en el sensor tilt el sistema se queda en ese estado. Se pueden quedar los **LEDs parpadeando** para llamar la atención, o poner un **zumbador** para lanzar una alarma sonora.

- Si cambia el tilt mientras ocurre el proceso de encendido de LEDs con el paso del tiempo, debe reiniciarse el proceso.: todos los LEDs apagados y comienza el contador.



Práctica 9: Sensor tilt de inclinación



Práctica 9: Sensor tilt de inclinación

Variables globales:

```
// variable para saber la ultima vez que se encenció un LED
unsigned long tUltLedON;
// variable que indique el intervalo de tiempo
unsigned long pasoT = 1000;
// ultimo led encendido o numero de LEDs ON
int pinUltLedON = 0;
```

En setup()

```
// poner los pin de los LEDs como OUTPUT

// poner el pin del sensor tilt como INPUT
```

Práctica 9: Sensor tilt de inclinación

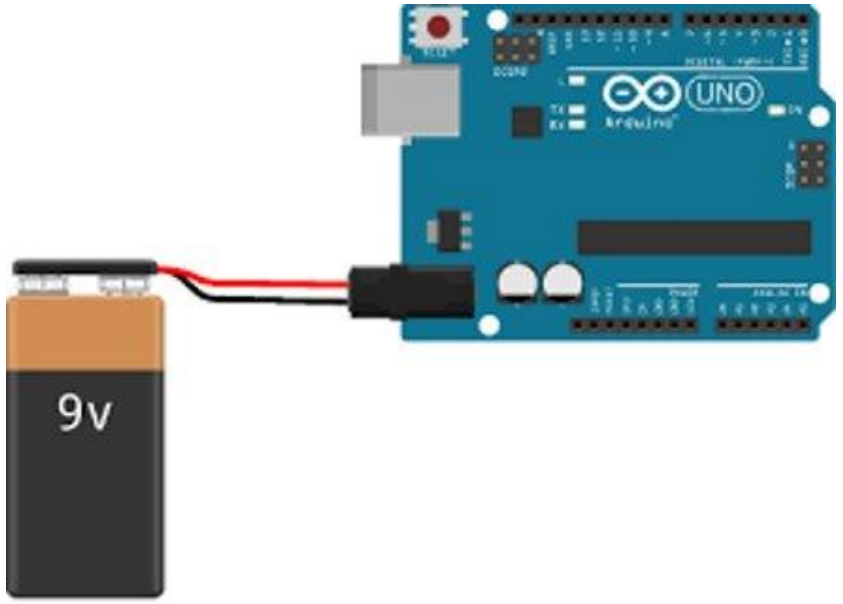
En loop()

```
// Medir cuanto tiempo llevamos
unsigned long tActual = millis();

// ver si hay que encender un LED: si ha pasado un cierto
// tiempo desde el ultimo led encendido

// si ha pasado ese tiempo encender otro led mas

// Si estan encencedidos todos los LEDs ya se puede medir el valor
// del tilt y reiniciar el proceso si cambia
```

Práctica 9: Sensor tilt de inclinación

Batería

Así liberamos a nuestro dispositivo del cable USB que le aporta energía

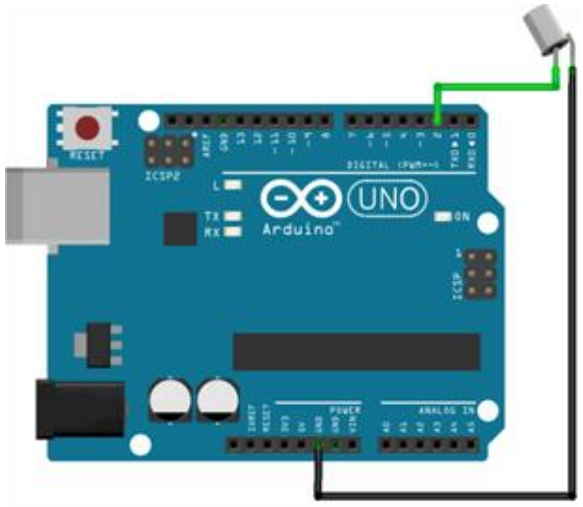


Práctica 9: Sensor tilt de inclinación

- **Potenciómetro** (Xtra):
  Con esto podremos controlar el intervalo de tiempo de encendido y apagado de los LEDs, o el brillo de los LEDs.
- ¿Cómo lo conectaríamos?

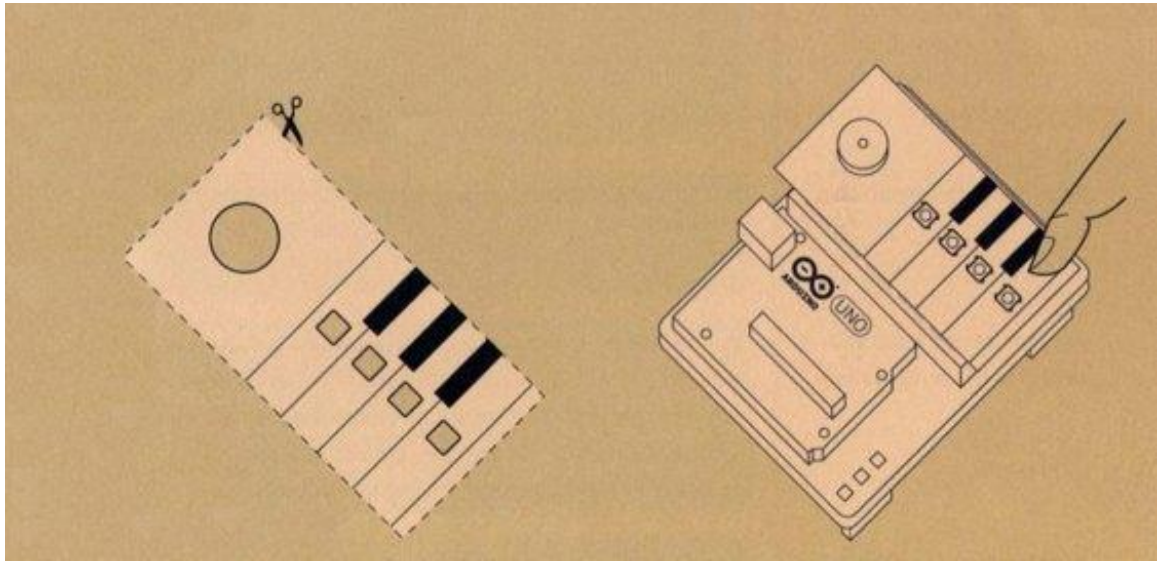
Práctica 9: Sensor tilt de inclinación

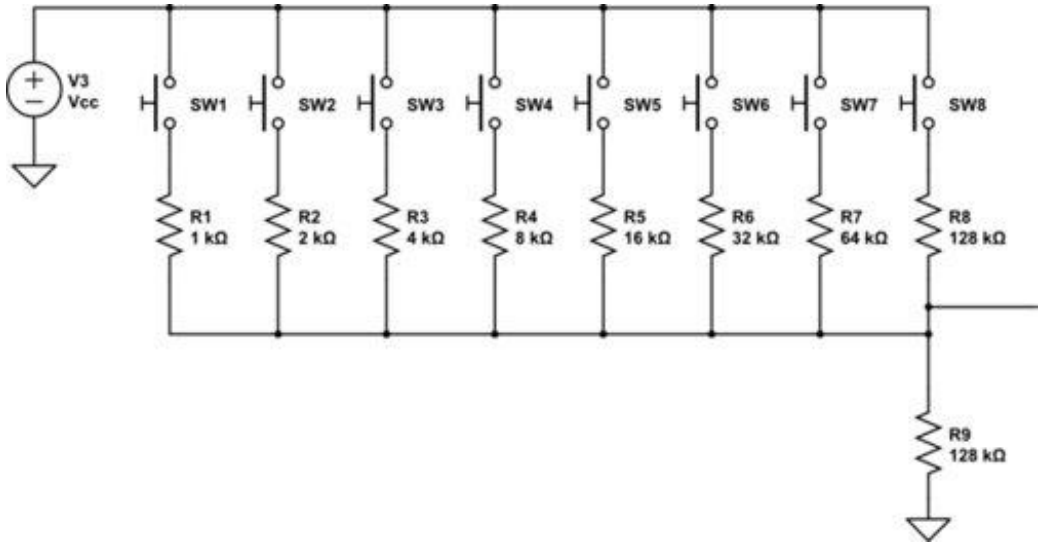
- **Referencias web**:
- https://www.luisllamas.es/medir-inclinacion-con-arduino-y-sensor-tilt-sw-520d/
- Varios montajes: http://rufianenlared.com/tilt-switch/
- https://www.prometec.net/tilt-switch/



10

Práctica 10: Piano – Escalera de resistencias





Capítulo 7 del Libro de Proyectos de Arduino:

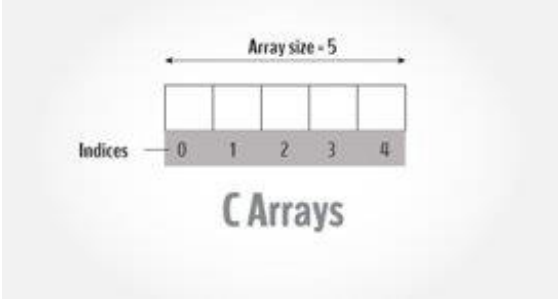
Teclado musical

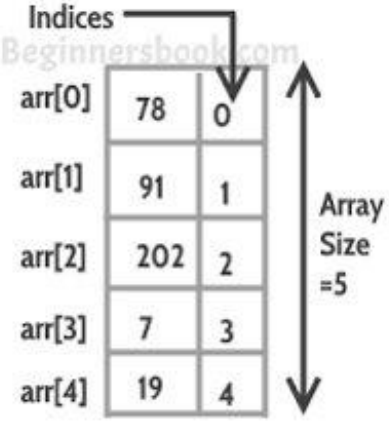
Práctica 10: Piano – Escalera de resistencias

- **Escalera de resistencias:**
- Utilizamos varios interruptores sin ocupar puertos digitales, sólo un puerto analógico.

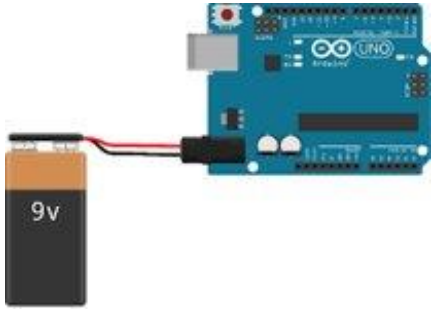
  Al pulsar cualquier conector se modifica el voltaje medido en un punto del circuito. Este valor se mide con un puerto analógico y se utiliza para realizar unas acciones u otras.

- **Array de valores**
- Nuevo tipo de variable.





- **Uso de baterías**
- Utilizaremos baterías para independizar el instrumento



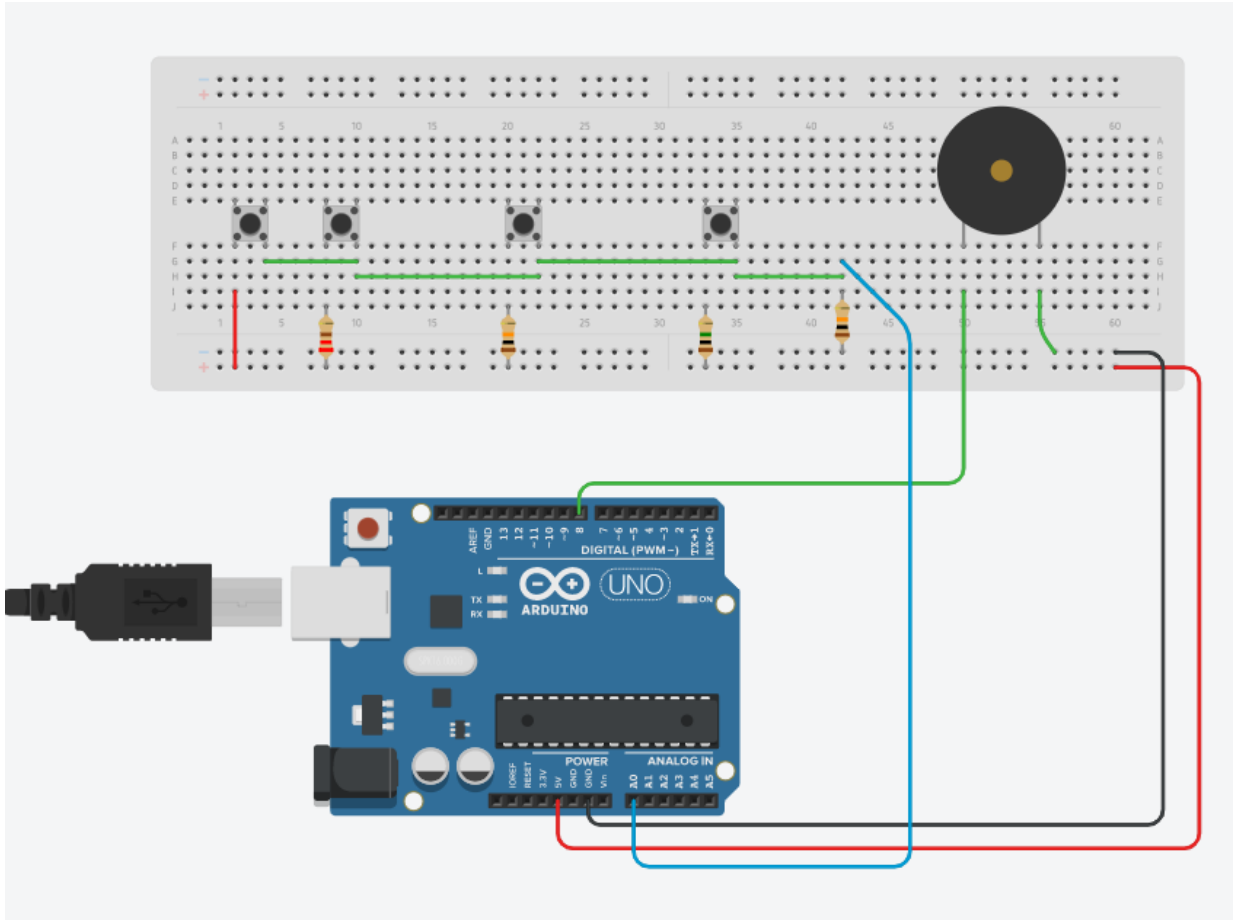
Práctica 10: Piano – Escalera de resistencias

- **Escalera de resistencias:**

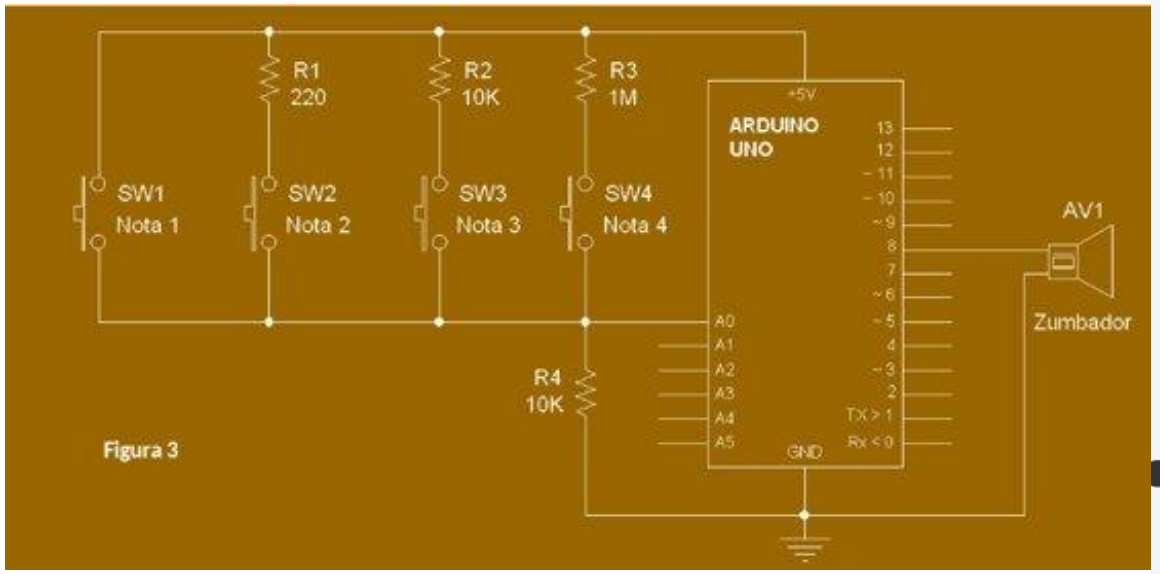
Utilizamos varios interruptores sin ocupar puertos digitales, sólo un puerto analógico.

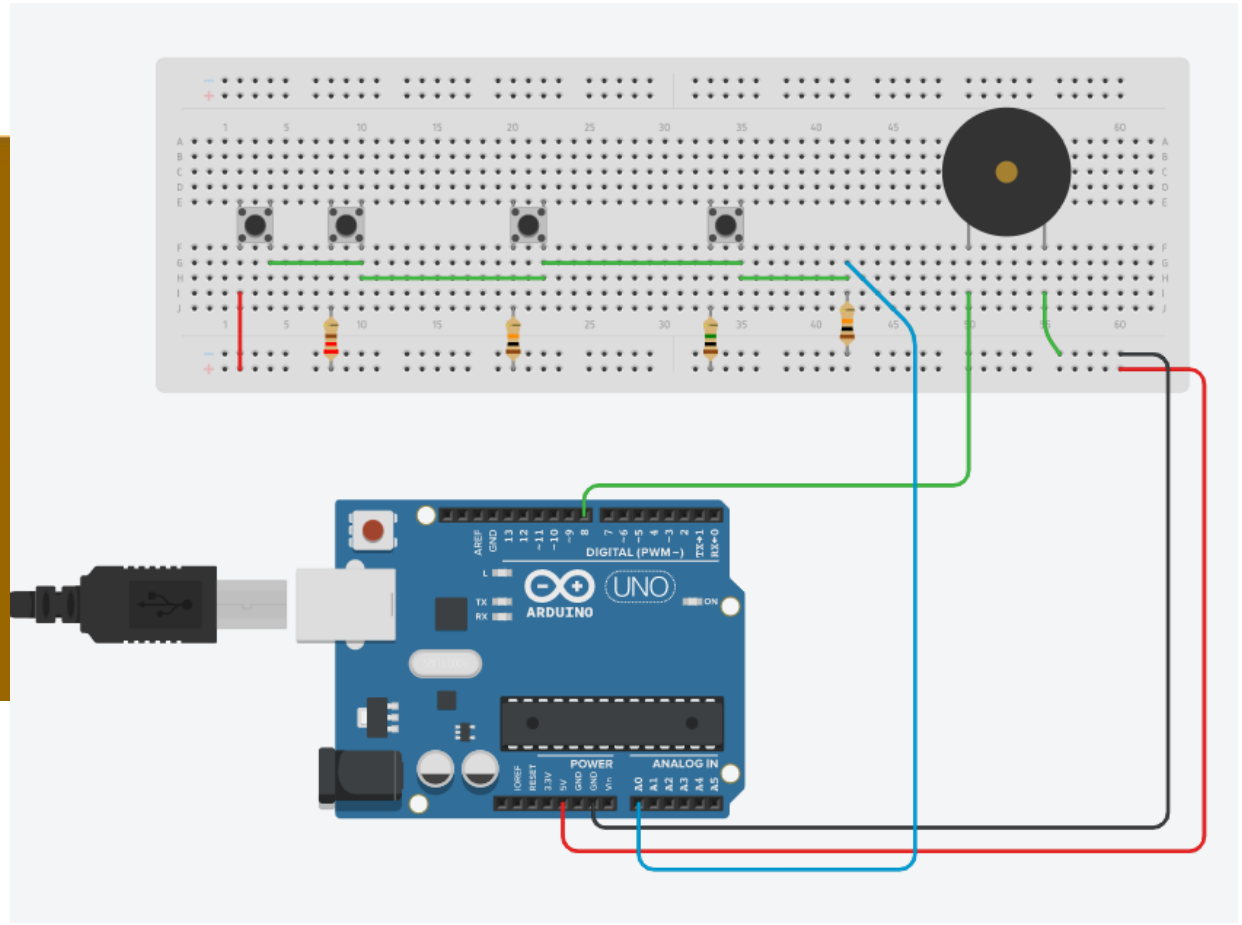
Al pulsar cualquier conector se modifica el voltaje medido en un punto del circuito. Este valor se mide con un puerto analógico y se utiliza para realizar unas acciones u otras.

El valor leído se utilizará para lanzar un sonido de zumbador con una nota (tono) u otra.



Práctica 10: Piano – Escalera de resistencias





Práctica 10: Piano – Escalera de resistencias

- **Comprobar valores:**

Con el puerto serie leemos los valores que da el puerto analógico donde hemos conectado el sensor de la escalera de resistencias.

Hay que anotarlos para poder utilizarlos en la programación.

Btn1	Btn2	Btn3	Btn4	Valor A0
X				...
	X			
		X		
			X	
X	X			
X		X		
X			X	
etc	etc	etc	etc	

Práctica 10: Piano – Escalera de resistencias

- **Código: array de valores**

En un array podemos guardar en una variable un cierto número de valores diferentes. Se accede a ellos con un índice que indica el elemento (0,1,2,3...)

```
int notas[] = {262,294,330,349};
// se accede a los datos con el indice: notas[0] es 262, por ejemplo
int valores[] = {1023,...,...,...}; // Estos valores se miden
                                    // experimentalmente
```

- **Código: asociar valores a notas**

Con una estructura condicional (if...else if...else) asociar cada posible valor a una acción dada.

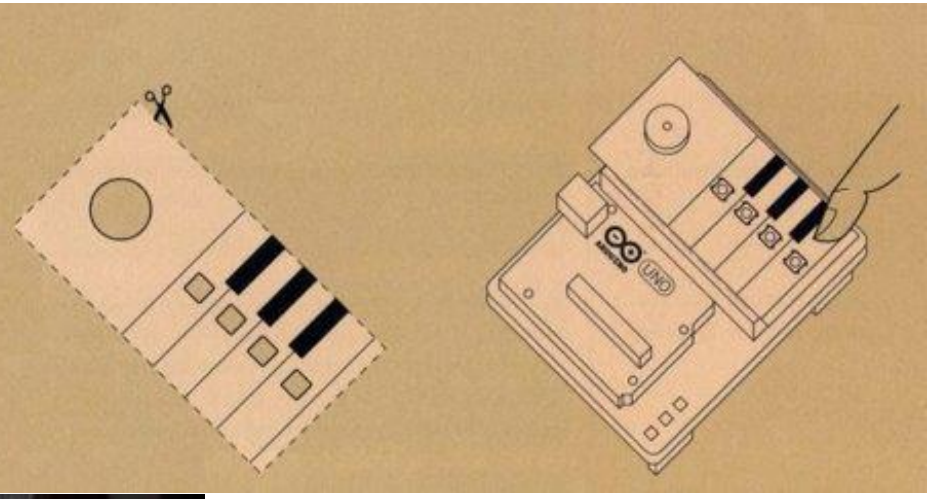
Esa acción será tocar un tono con el zumbador.

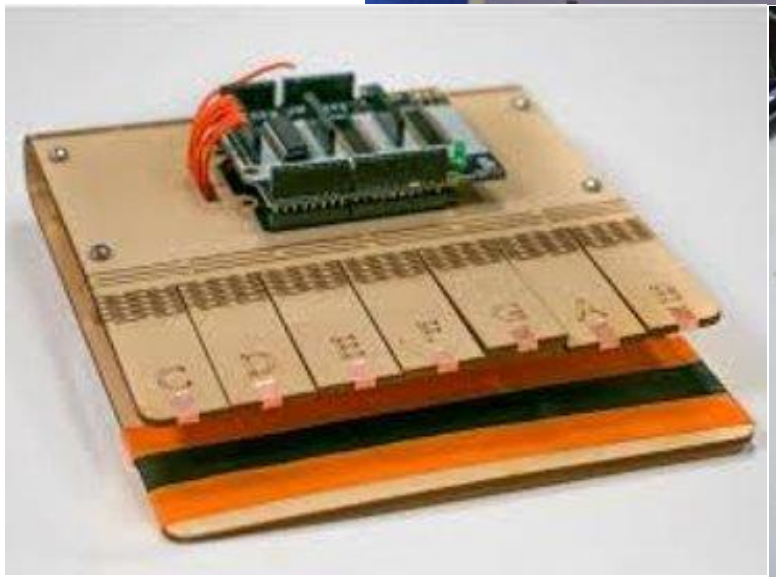
Se puede explorar y hacer que se reproduzcan melodías, que varíe el tono, que se encienda un LED...

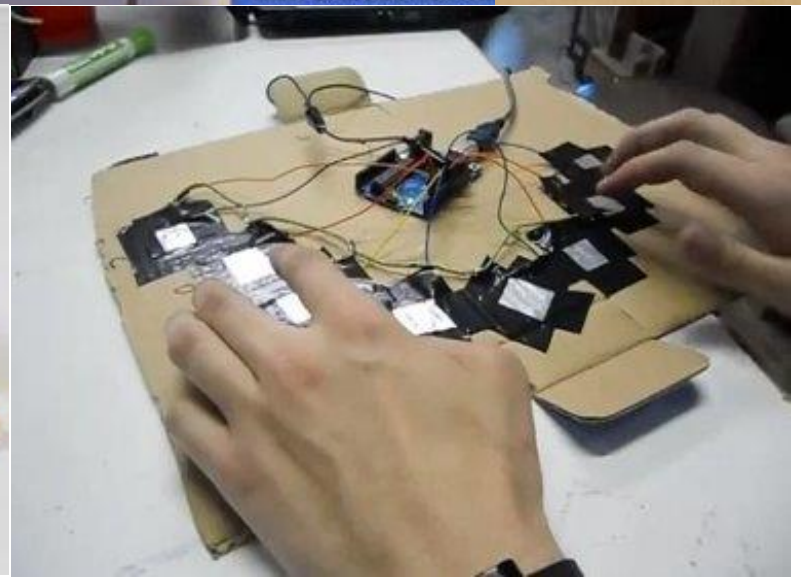
Arduino.cc/frequencies aqui encontraréis las equivalencias de notas

Práctica 10: Piano – Escalera de resistencias

Personalización







http://www.futureworkss.com/arduino/Proyecto07TecladoMusical01Descripcion.html