Oficina de Software Libre Universidad de Granada

UNIVERSIDAD DE GRANADA

Simon says

¡Creemos un juego!

Autora:

Inés Jiménez Díaz

@https_rim
@_https.rim_

Oficina de Software Libre Universidad de Granada

UNIVERSIDAD DE GRANADA

Hardware necesario para Simon

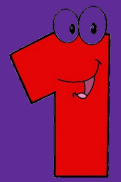
Piano musical

- Arduino Nano
- 4 sensores de presión
- 4 resistencias 2.7k
- Tira de 4 Led Adafruit
- Buzzer
- Cables arduino

Oficina de Software Libre Universidad de Granada

UNIVERSIDAD DE GRANADA

Montaje del Simon



¡Hágase la luz!

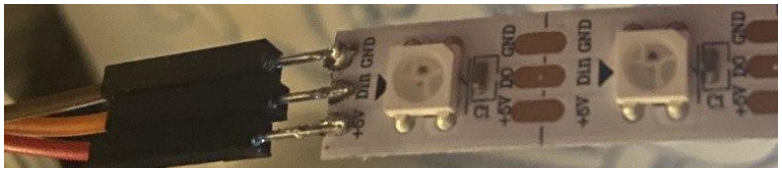
¡HÁGASE LA LUZ!

- **Leds RGB direccionables.**
  - Tienen 4 pines:
    - **DIN** pin de entrada
    - **DO** pin de salida
    - **+5V** proporciona energía a los leds
    - **GND** Toma de tierra

Antes de cortar los leds fijaros que los **DIN** están conectados a los **DO** recuerdalo cuando vayas a incluirlos en el proyecto final al conectarlos.

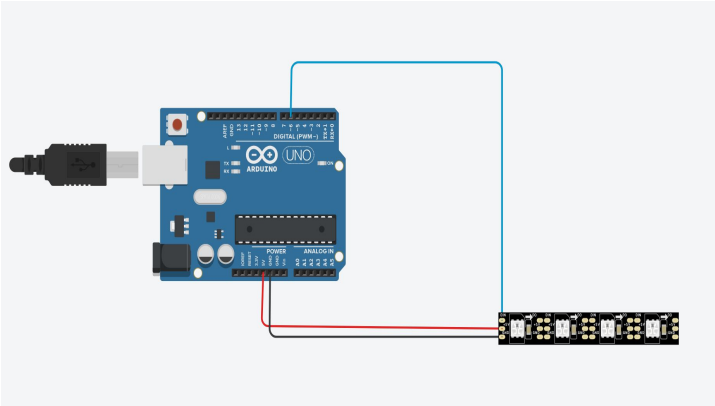
¡HÁGASE LA LUZ!

- **Leds RGB direccionables.**
- Se llaman direccionables porque las señales se transmiten en una dirección a través de los leds.
- **¡Fíjate cómo están conectados!**



¡HÁGASE LA LUZ!

- **Leds RGB direccionables.**
- Se llaman direccionables porque las señales se transmiten en una dirección a través de los leds. **Del Arduino al primer led, al segundo, al tercero… hasta el último**



1. Incluir en el IDE la librería Adafruit_NeoPixel

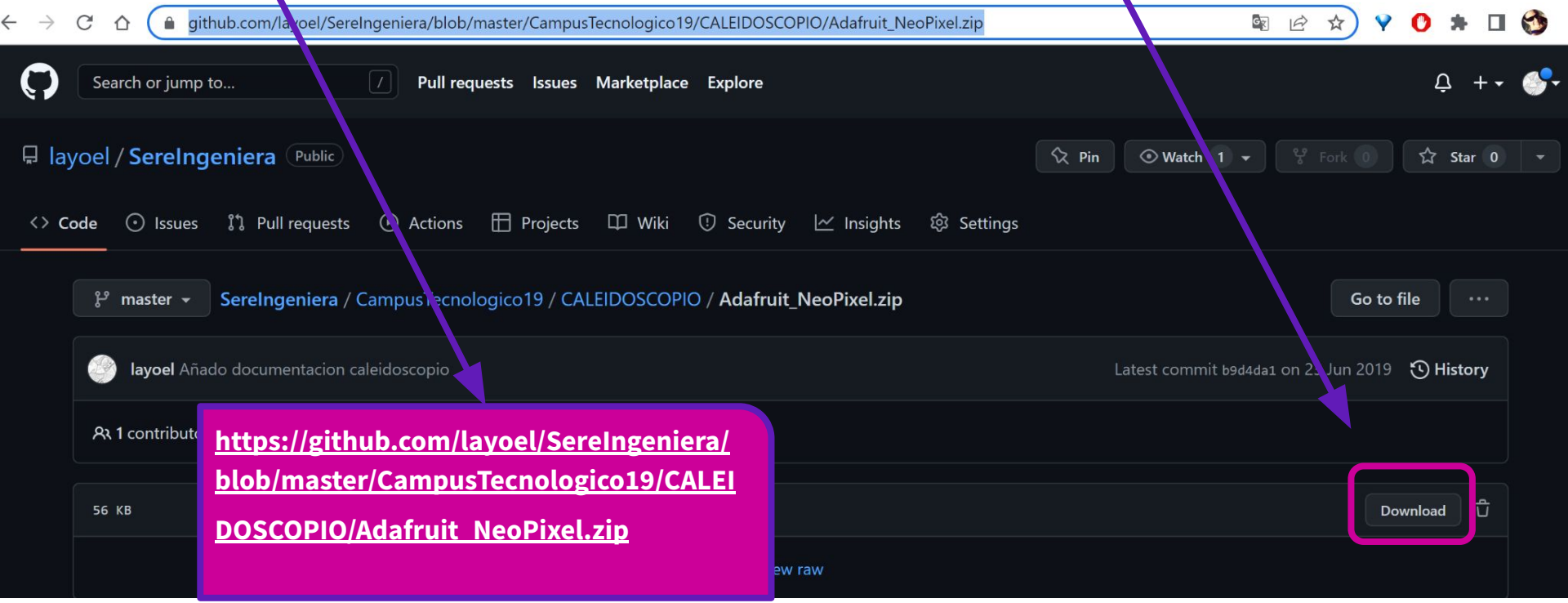
- Hay **dos formas de incluir librerías** a la biblioteca de nuestro arduino.
  - Si tenemos el archivo comprimido con extensión .zip
  - Buscarlas en el repositorio de Arduino

- Para la tira de led tenemos el archivo comprimido con el nombre **Adafruit_NeoPixel.zip**
- La **descargamos** y la incluimos de la 1º forma.

¡HÁGASE LA LUZ!

1. Incluir en el IDE la librería Adafruit_NeoPixel

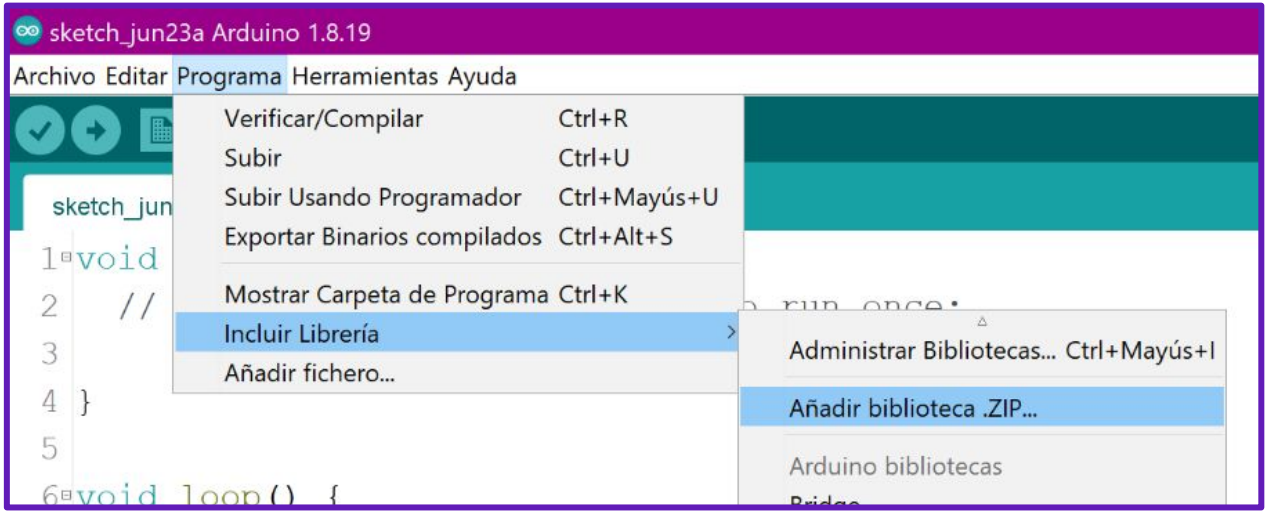
Vamos al enlace y descargamos la librería haciendo clic en download



https://github.com/layoel/SereIngeniera/blob/master/CampusTecnologico19/CALEIDOSCOPIO/Adafruit_NeoPixel.zip

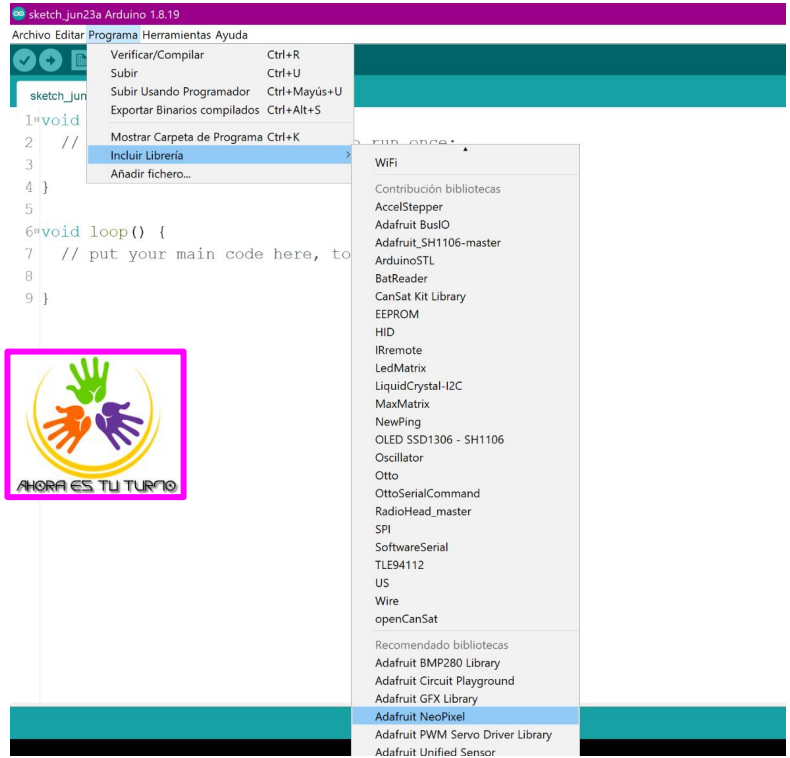
¡HÁGASE LA LUZ!

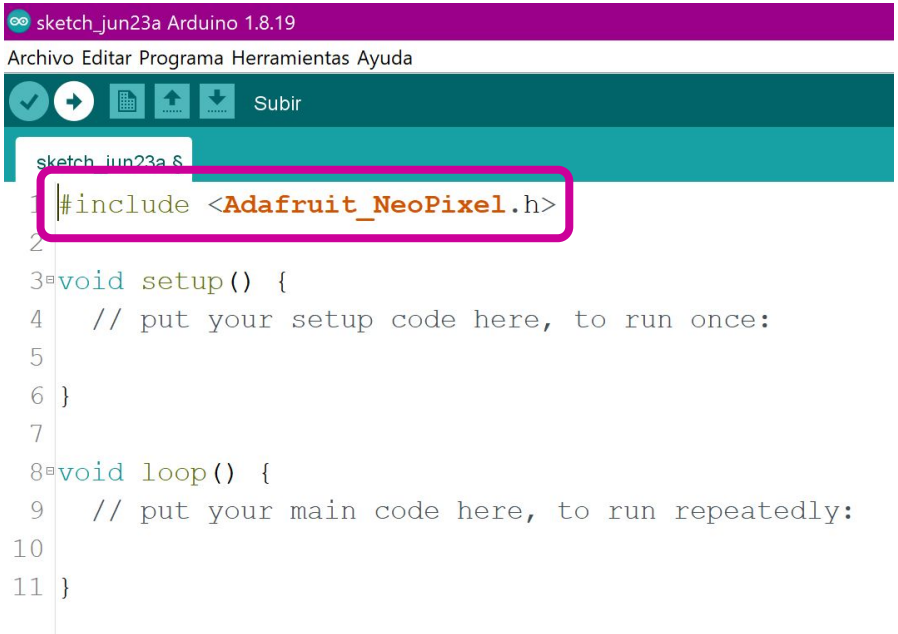
1. Incluir en el IDE la librería Adafruit_NeoPixel



¡HÁGASE LA LUZ!

2. Para poder usar la librería tenemos que añadirla en el sckech.





3. Declaramos un objeto con la cadena de leds que vamos a emplear.

- Para ello utilizaremos la siguiente **función de la librería NeoPixel** que acabamos de incluir.
  - **Adafruit_NeoPixel pixels(**NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800**);**
- Los parámetros que se pasan a la función son:
  - **NUMPIXELS** que será una variable donde pondremos el número de leds que queremos controlar
  - **PIN** que será el pin de arduino donde hemos conectado el **IN** de la tira de led
  - **NEO_GRB + NEO_KHZ800** este parámetro indica que usaremos leds de colores y la controladora que utilizan.
- Vale pero, ¿qué tenemos que hacer y cómo?

¡HÁGASE LA LUZ!

3. Declaramos un objeto con la cadena de leds que vamos a emplear.

- Aquí tenéis **un ejemplo para controlar 3 leds**, vuestra tarea es adaptarlo para poder usar los 8 leds.

Esto hay que hacerlo en el scketch donde vamos a crear el piano

sketch_jun23a Arduino 1.8.19

Archivo Editar Programa Herramientas Ayuda

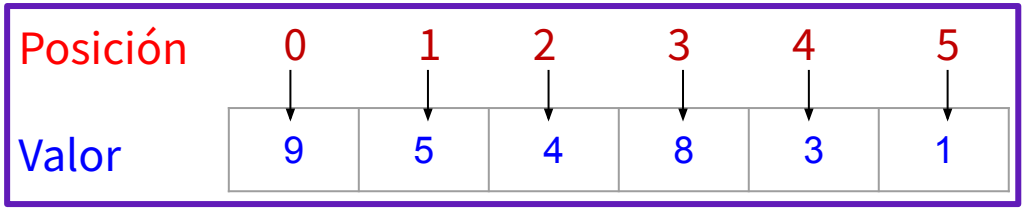
sketch_jun23a §

```
#include <Adafruit_NeoPixel.h>

#define PIN        2
#define NUMPIXELS 3

using namespace std;

Adafruit_NeoPixel pixels(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);

void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

¡HÁGASE LA LUZ!

4. Aprendemos a usar la librería para encender uno a uno los leds de la tira de led de colores.

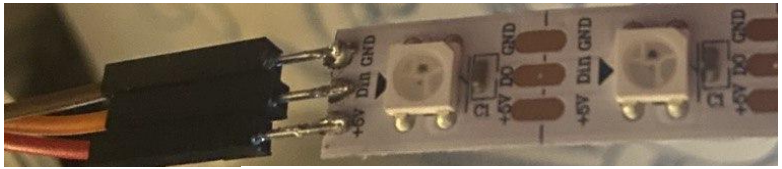
- Antes de empezar a usar las funciones de la librería necesitamos saber cómo acceder a las posiciones de cada led.
- ¿Sabes que es un **vector**?



- El vector tiene **posiciones**
- La primera posición **siempre empieza en 0**
- **Cada posición** tiene **un valor** (en la imagen, la posición 0 tiene valor 9, la posición 4 tiene valor 3…)
- El tamaño máximo del vector es el número de posiciones que tiene. En este ejemplo el vector tiene **6 posiciones que van del 0 al 5**.
- Los led se encienden en el mismo orden que el vector del 0 al 8 en nuestro caso.

¡HÁGASE LA LUZ!

4. **Aprendemos a usar la librería para encender uno a uno los leds de la tira de led de colores.**

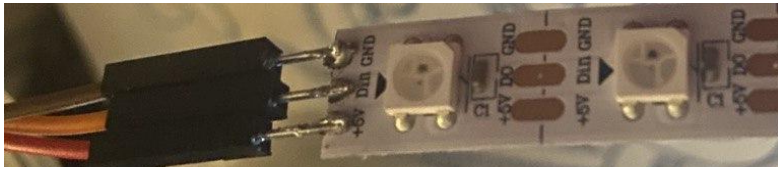


- ¿Cómo encedemos un led?

¡HÁGASE LA LUZ!

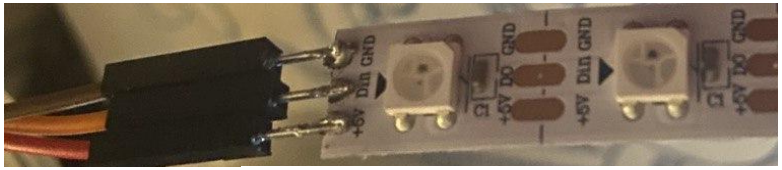
- **Peero antes de seguir...Debemos de soldar la tira de leds**

- Así debería de quedar la soldadura!!!



¡HÁGASE LA LUZ!

4. **Aprendemos a usar la librería para encender uno a uno los leds de la tira de led de colores.**



- ¿Cómo encedemos un led?

- **¡Usaremos la librería Neopixel!**

¡HÁGASE LA LUZ!

- Pero…

CUIDADO

CON

QUEMARSE!!

4. Aprendemos a usar la librería para encender uno a uno los leds de la tira de led de colores.

- Utilizaremos estas funciones de la librería Neopixel.
  - **pixels.begin();** esta función inicializa los leds.
  - **pixels.clear();** esta función borra la configuración guardada en los leds.
  - **pixels.Color(R, G, B);** esta función es para seleccionar el color.
    - **R** es el parámetro donde se pondrá el valor del color **rojo** (toma valores de 0 a 255)
    - **G** es el parámetro donde se pondrá el valor del color **verde** (toma valores de 0 a 255)
    - **B** es el parámetro donde se pondrá el valor del color **azul** (toma valores de 0 a 255)
  - **pixels.setPixelColor(led, color);**
    - **led** es el número de led que queremos que se encienda del **color** que pongamos.
    - **color** es el color que se va a poner y se sustituye por: **pixels.Color(R, G, B);**

¡HÁGASE LA LUZ!

4. Aprendemos a usar la librería para encender uno a uno los leds de la tira de led de colores.

- En este ejemplo se usan 6 leds.
  - **Prueba el código en tu scketch**
    - ¿Qué leds se encienden?
    - ¿En qué color?
  - **Adapta el código para los 4 leds**
  - **Ve haciendo pruebas para encender uno u otro led o varios a la vez y probar los diferentes colores.**


```
#include <Adafruit_NeoPixel.h>

#define PIN        2
#define NUMPIXELS 3

using namespace std;

Adafruit_NeoPixel pixels(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);

int tablero[]={0,0,0
               0,0,0};

void setup() {
  // put your setup code here, to run once:
  pixels.begin();

}

void loop() {
  // put your main code here, to run repeatedly:
  pixels.clear();
  pixels.setPixelColor(0, pixels.Color(30, 0, 0));
  pixels.setPixelColor(1, pixels.Color(0, 30, 0));
  pixels.setPixelColor(2, pixels.Color(0, 0, 30));
  pixels.setPixelColor(3, pixels.Color(30, 30, 30));
}
```

5. Encender de uno en uno los led pares en rojo y led impares en verde

- Para poder hacer esto, vamos a aprender a escribir condicionales.
  - **Condicional simple**
    - **Si** es cierto… **entonces…**
  - **Condicional doble**
    - **Si** es cierto… **entonces**…. **sino**… **entonces**…
  - **Condicional compuesto**
    - **Si** esto **o** esto es cierto… **entonces…**
    - **Si** esto **y** esto es cierto… **entonces…**

5. Encender de uno en uno los led pares en rojo y led impares en verde

- Para hacer condicionales **necesitamos operadores**

que pueden ser:

- Matemáticos:
  - **+, -, *, /, %.**
- Relacionales:
  - **==, !=, <, >, >=, <=**
- Lógicos
  - **&&, and, ||, or, !**

P	!P
True	False
False	True

P	Q	P&&Q	P||Q
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	False

5. Encender de uno en uno los led pares en rojo y led impares en verde

Podemos usar un condicional simple usando los operadores == (igual) **y !=** (distinto)

```
If (condición){
    sentencias;
}
```

== ➡ ¿Son iguales?
!= ➡ ¿Son distintos?

```
8  Adafruit_NeoPixel pixels(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);
9
10
11
12
13 void setup() {
14   // put your setup code here, to run once:
15   pixels.begin();
16
17 }
18
19 void loop() {
20
21   int numLed = 2; //El led 2
22   /*Calcula par o impar. si el resto es 0 es par sino es impar*/
23   int calculo = numLed % 2;
24
25   if(calculo == 0){
26     pixels.setPixelColor(numLed, pixels.Color(30, 0, 0));
27     }
28
29   if(calculo != 0){
30     pixels.setPixelColor(numLed, pixels.Color(0, 30, 0));
31     }
```

5. Encender de uno en uno los led pares en rojo y led impares en verde

- Veamos que hace el código:
  - Creo una **variable numLed** y le asigno el valor **2. (línea 21)**
  - Creo una **variable calculo** a la que le asigno el **resto de dividir numLed entre 2** (para saber si es par o impar) **línea 23.**
  - Si el valor de calculo es igual a 0
    - entonces enciende el led numLed de color rojo.
  - Si el valor de calculo es distinto de 0
    - entonces enciende el led numLed de color verde.

```
8  Adafruit_NeoPixel pixels(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);
9
10
11
12
13 void setup() {
14   // put your setup code here, to run once:
15   pixels.begin();
16
17 }
18
19 void loop() {
20
21   int numLed = 2; //El led 2
22  /*Calcula par o impar. si el resto es 0 es par sino es impar*/
23   int calculo = numLed % 2;
24
25   if(calculo == 0){
26     pixels.setPixelColor(numLed, pixels.Color(30, 0, 0));
27     }
28
29   if(calculo != 0){
30     pixels.setPixelColor(numLed, pixels.Color(0, 30, 0));
31     }
```

5. Encender de uno en uno los led pares en rojo y led impares en verde

- Veamos que hace el código:
  - Creo una **variable numLed** y le asigno el valor **2. (línea 21)**
  - Creo una **variable calculo** a la que le asigno el **resto de dividir numLed entre 2** (para saber si es par o impar) **línea 23.**
  - Si el valor de calculo es igual a 0
    - entonces enciende el led numLed de color rojo.
  - Si el valor de calculo es distinto de 0
    - entonces enciende el led numLed de color verde.

```
8  Adafruit_NeoPixel pixels(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);
9
10
11
12
13 void setup() {
14   // put your setup code here, to run once:
15   pixels.begin();
16
17 }
18
19 void loop() {
20
21   int numLed = 2; //El led 2
22  /*Calcula par o impar. si el resto es 0 es par sino es impar*/
23   int calculo = numLed % 2;
24
25   if(calculo == 0){
26     pixels.setPixelColor(numLed, pixels.Color(30, 0, 0));
27     }
28
29   if(calculo != 0){
30     pixels.setPixelColor(numLed, pixels.Color(0, 30, 0));
31     }
```

5. Encender de uno en uno los led pares en rojo y led impares en verde

- ¿Y si queremos que haga las comprobaciones de manera automática para todos los leds?
  - **¡Usaremos bucles!**

Bucles o ciclos

Bucles

```
Loop(){
  Lo que se va a repetir siempre
}

for (inicio; fin; incremento){
  Lo que se va a repetir desde inicio a fin
}

while(condición){
lo que quiero que haga mientras se cumpla la condición
}
```

5. Encender de uno en uno los led pares en rojo y led impares en verde

- Bucles:
-
```
Loop(){
  Lo que se va a repetir siempre
}
```
-
```
for (inicio; fin; incremento){
  Lo que se va a repetir desde inicio a fin
}
```
-
```
while(condición){
lo que quiero que haga mientras se cumpla
la condición
}
```

5. Encender de uno en uno los led pares en rojo y led impares en verde

- Bucles:
- **Loop(){**
  Lo que se va a repetir siempre
  }

```
18
19 void loop() {
20
21   int numLed = 2; //El led 2
22  /*Calcula par o impar. si el resto es 0 es par sino es impar*/
23   int calculo = numLed % 2;
24
25   if(calculo == 0){
26     pixels.setPixelColor(numLed, pixels.Color(30, 0, 0));
27   }else{
28       pixels.setPixelColor(numLed, pixels.Color(0, 30, 0));
29         }
```

¡HÁGASE LA LUZ!

5. Encender de uno en uno los led pares en rojo y led impares en verde

- Bucles:
- **for (inicio; fin; incremento){**
  Lo que se va a repetir desde inicio a fin
  }

5. Encender de uno en uno los led pares en rojo y led impares en verde

- Bucles:
- **while(*condición*){**
  lo que quiero que haga mientras
  se cumpla la condición
  }

6. Aprendemos el condicional simple y el condicional compuesto

Condicional doble

```
If (condición ){
    sentencia1;
}else{
    sentencia2;
}
```

- Este condicional traducido a lenguaje significa :
- si se cumple condicion
  - entonces ejecuta sentencia1
- sino se cumple condición
  - entonces ejecuta sentencia2

6. Aprendemos el condicional simple y el condicional compuesto

- Veamos el ejemplo de par o impar, usando un condicional doble.
- Sería:
  - Si el cálculo es igual a cero,
    - entonces enciende el led2 de color rojo.
  - Sino,
    - entonces enciende el led 2 de color verde.

```
4  #define NUMPIXELS 3
5
6  using namespace std;
7
8  Adafruit_NeoPixel pixels(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);
9
10
11
12
13 void setup() {
14   // put your setup code here, to run once:
15   pixels.begin();
16
17 }
18
19 void loop() {
20
21   int numLed = 2; //El led 2
22  /*Calcula par o impar. si el resto es 0 es par sino es impar*/
23   int calculo = numLed % 2;
24
25   if(calculo == 0){
26     pixels.setPixelColor(numLed, pixels.Color(30, 0, 0));
27   }else{
28       pixels.setPixelColor(numLed, pixels.Color(0, 30, 0));
29         }
```

6. Aprendemos el condicional simple y el condicional compuesto

- Veamos el ejemplo de par o impar, usando un condicional doble.
- Sería:
  - Si el cálculo es igual a cero,
    - entonces enciende el led2 de color rojo.
  - Sino,
    - entonces enciende el led 2 de color verde.

```
#define NUMPIXELS 3

using namespace std;

Adafruit_NeoPixel pixels(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);



void setup() {
  // put your setup code here, to run once:
  pixels.begin();

}

void loop() {

  int numLed = 2; //El led 2
 /*Calcula par o impar. si el resto es 0 es par sino es impar*/
  int calculo = numLed % 2;

 if(calculo == 0){
   pixels.setPixelColor(numLed, pixels.Color(30, 0, 0));
 }else{
    pixels.setPixelColor(numLed, pixels.Color(0, 30, 0));
       }
```

5. Encender de uno en uno los led pares en rojo y led impares en verde

Ahora que ya sabes encender los leds, edita el código para que, se encienda de uno en uno los led pero que se apague el anterior.

Te dejo el algoritmo. Piensa como se programa aplicando lo aprendido antes.


```
Para i=0 mientras i sea menor que el número de pixel incrementa i en 1
        función que pone el color al pixel i
        si i es distinto de 0
                función que pone el color al pixel i anterior y poner el color 0 que es apagado
        aplicar la configuración a los pixel
        esperar medio segundo
```

Oficina de Software Libre Universidad de Granada

UNIVERSIDAD DE GRANADA

Montaje del Simon

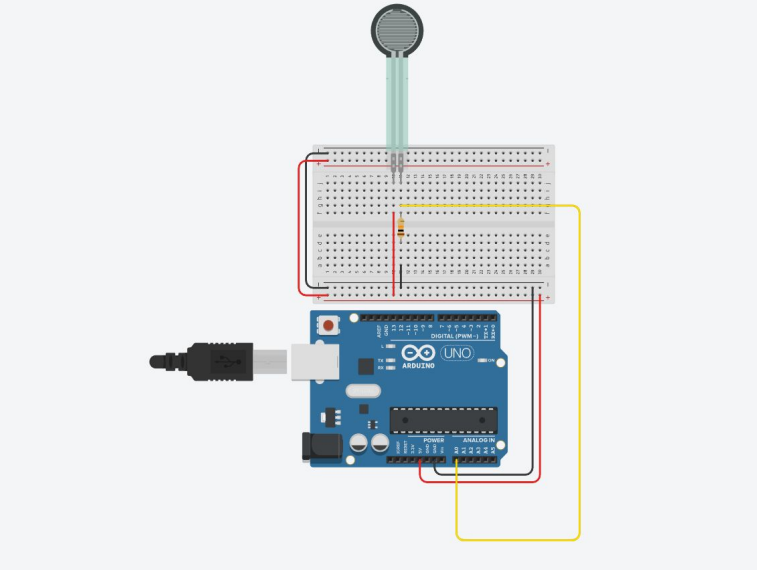
Sensores de presión

ESQUEMA

- **Los sensores de presion…**

  Tienen 2 pines:

  - Uno de ellos va a **5V**,
  - Y el otro se lleva a **GND**, poniendo una ***resistencia de 2.7K*** y el otro va a un **pin analógico**.



¿CÓMO SE QUE VALORES DA?

Pues lo vamos a hacer con una instrucción que se llama analogRead().

analogRead(*pin*) se utiliza para leer el valor de un **pin analógico**.

Nosotras queremos que salga en el **Serial monitor** ese valor.

¿CÓMO SE QUE VALORES DA?

¿Y…como hacemos para que salgan en el Serial monitor los valores?

Usaremos dos instrucciones:

Serial.begin(9600) → Se debe de poner en el **void setup()**

Serial.print() ó **Serial.println()** → Estas se deben de poner en **void loop()**

¿CÓMO SE QUE VALORES DA?

¿Y…como hacemos para que salgan en el Serial monitor los valores?

Usaremos dos instrucciones:

Serial.begin(9600) → Sirve para establecer la velocidad de datos en bits por segundo (baudios) para la transmisión de datos en serie.

Serial.print() ó **Serial.println()** → Las dos sirven para enseñar la información por el **Serial monitor**. La única diferencia es que…

- ★ **print()** te pone todos los valores juntos,
- ★ **println()** te pone cada valor en una línea diferente.

¿CÓMO SE QUE VALORES DA?

Ahora te dejo el algoritmo. Piensa como se programa aplicando lo aprendido antes.

```
void setup()
        establezca la velocidad
        pongo el pin del sensor en OUTPUT.

void loop()
        valor = leo el valor del pin del sensor
        Escribo en el Serial monitor el valor que me da
        espero 500 milisegundos
```

¿CÓMO SE QUE VALORES DA?

Teniendo este código hecho, vemos que valores nos da cuando pulsamos el sensor y cuando no pulsamos el sensor.

```
void setup()
    establezca la velocidad
    pongo el pin del sensor en OUTPUT.

void loop()
    valor = leo el valor del pin del sensor
    Escribo en el Serial monitor el valor que me da
    espero 500 milisegundos
```

¿CÓMO SE QUE VALORES DA?

Teniendo este código hecho, vemos que valores nos da cuando pulsamos el sensor y cuando no pulsamos el sensor.

```
void setup()
    establezca la velocida
    pongo el pin del sens

void loop()
    valor = leo el valor del pin del sensor
    Escribo en el Serial monitor el valor que me da
    espero 500 milisegundos
```

Aquí solo se indica el algoritmo para 1 sensor… pero se debe de hacer el algoritmo para los 4 sensores!!

Comprobación

Teniendo en cuenta el valor que nos da cuando lo tenemos pulsado y cuando no, hacemos una comprobación para que nos vaya todos los sensores.

Ahora te dejo el algoritmo. Piensa como se programa aplicando lo aprendido antes.

si valor del sensor > valor más alto que nos da cuando no lo pulsamos, **entonces**

imprimimos por **Serial monitor** “OK, funciona sensor X”

esperamos 500 **milisegundos**

Comprobación

Teniendo en cuenta el valor que nos da cuando lo tenemos pulsado y cuando no, hacemos una comprobación para que nos vaya todos los sensores.

Ahora te dejo el algoritmo. Piensa como se programa aplicando lo aprendido antes.

si valor del sensor > val… no lo pulsamos, **entonces**

Aquí solo se indica el algoritmo para 1 sensor… pero se debe de hacer el algoritmo para los 4 sensores!!

imprimimos por **Serial monitor** “OK, funciona sensor X”

esperamos 500 **milisegundos**

¿Recuerdas este algoritmo?

```
Para i=0 mientras i sea menor que el número de pixel incrementa i en 1
        función que pone el color al pixel i
        si i es distinto de 0
                función que pone el color al pixel i anterior y poner el color 0 que es apagado
        aplicar la configuración a los pixel
        esperar medio segundo
```

¿Recuerdas este algoritmo?

Pues lo vamos a cambiar para que se encienda un led cuando pulsemos un pulsador.

Por ejemplo, si queremos encender el led 0, tendríamos que pulsar el primer sensor.

Si queremos encender el led 1, se pulsa el segundo sensor… y así con los 4 sensores y leds.

¿Recuerdas este algoritmo?

Ahora te dejo el algoritmo. Piensa como se programa aplicando lo aprendido antes.

si valor del sensor > valor más alto que nos da cuando no lo pulsamos, **entonces**

encendemos el led correspondiente con un color

lo enseñamos

esperamos 1000ms

limpiamos la configuración

lo enseñamos

¿Recuerdas este algoritmo?

Ahora te dejo el algoritmo. Piensa como se programa aplicando lo aprendido antes.

si valor del sensor > valor más alto que nos da cuando no lo pulsamos, **entonces**

encendemos el led cor

lo enseñamos

esperamos 1000ms

limpiamos la configura

lo enseñamos

Aquí solo se indica el algoritmo para 1 sensor… pero se debe de hacer el algoritmo para los 4 sensores!!
(y en este con cuatro colores diferentes)

Oficina de Software Libre Universidad de Granada

UNIVERSIDAD DE GRANADA

Montaje del Simon

¡Vamos a programar!

¡VAMOS A PROGRAMAR!

★ **Tomando como referencia el código anterior…**

- Vamos a comenzar a crear niveles, y para ello, primero pensamos en la **secuencia que queremos conseguir al final**.
- Además, tenemos que pensar cuántos niveles queremos, ya que vamos a usar variables **booleanas**(de verdadero y falso) para pasar de un nivel a otro.

¡Vamos a programar!

★ **Vamos por partes…**

- ❑ Tenemos que crear **variables booleanas** para **pasar de nivel**.
- ❑ También tenemos que crear **variables booleanas** para **indicar que se ha pulsado ya un color** (cuando hay 2 o más colores en la secuencia.
- ❑ Al principio, todas las variables deben de estar en *false*, menos la del primer nivel, que debe de estar en *true*.

¡VAMOS A PROGRAMAR!

- ★ Creemos la secuencia para los niveles.
- ❑ Ahora te dejo el algoritmo para cuando tienes que pulsar más de un sensor.

Piensa como se programa aplicando lo aprendido antes.

si nivel X, **entonces**

    encendemos el led correspondiente con el color del primer elemento de la secuencia

    lo enseñamos

    esperamos 500ms

    limpiamos la configuración

    lo enseñamos

ponemos a false la variable para ese nivel

¡VAMOS A PROGRAMAR!

- ★ Creemos la secuencia para los niveles.
- ❑ Ahora te dejo el algoritmo para cuando tienes que pulsar más de un sensor.

Piensa como se programa aplicando lo aprendido antes.

si nivel X, **entonces**

encendemos el led correspondiente con el color del primer elemento de la secuencia

lo enseñamos

esperamos 500ms

limpiamos la configuración

lo enseñamos

ponemos a false la variable para ese nivel

y así con todos los colores que tenga la secuencia en ese nivel.

¡VAMOS A PROGRAMAR!

Este algoritmo nos va a ayudar cuando tengamos que ir pulsando los sensores para repetir la secuencia que nos había enseñado las luces.

si valor del sensor > valor más alto que nos da cuando no lo pulsamos, **entonces**

encendemos el led correspondiente con un color

lo enseñamos

esperamos 500ms

limpiamos la configuración

lo enseñamos

ponemos a true para que siga al siguiente nivel

¡VAMOS A PROGRAMAR!

Ahora te dejo el algoritmo para cuando tienes que pulsar más de un sensor. Piensa como se programa aplicando lo aprendido antes.

si valor del sensor > valor más alto que nos da cuando no lo pulsamos, **entonces**

encendemos el led correspondiente con un color

lo enseñamos

esperamos 500 ms

limpiamos la configuración

lo enseñamos

ponemos a true la variable que tengamos para el siguiente color

¡VAMOS A PROGRAMAR!

Ahora te dejo el algoritmo para cuando tienes que pulsar más de un sensor. Piensa como se programa aplicando lo aprendido antes.

si valor del sensor > valor más alto que nos da cuando no lo pulsamos **y** siguiente color, **entonces**

encendemos el led correspondiente con un color

lo enseñamos

esperamos 500 ms

limpiamos la configuración

lo enseñamos

(ponemos a true la variable que tengamos para el siguiente color) → si hubiera más colores

&& → y
|| → o

¡VAMOS A PROGRAMAR!

Ahora te dejo el algoritmo para cuando tienes que pulsar más de un sensor. Piensa como se programa aplicando lo aprendido antes. Este caso es cuando es el último color de la secuencia de ese nivel.

si valor del sensor > valor más alto que nos da cuando no lo pulsamos **y** siguiente color, **entonces**

encendemos el led correspondiente con un color

lo enseñamos

esperamos 500 ms

limpiamos la configuración

lo enseñamos

poner a true para que pase de nivel

&& → y
|| → o

¡VAMOS A PROGRAMAR!

Al final te quedaría algo así. Te dejo el algoritmo para el nivel 1, y de ahí se va haciendo los demás. Deberás de pasarlo a código con lo aprendido

si nivel X, **entonces**

encendemos el led correspondiente con el color del primer elemento de la secuencia

lo enseñamos

esperamos 500ms

limpiamos la configuración

lo enseñamos

ponemos a false la variable para ese nivel

1

si valor del sensor > valor más alto que nos da cuando no lo pulsamos, **entonces**

encendemos el led correspondiente con un color

lo enseñamos

esperamos 500ms

limpiamos la configuración

lo enseñamos

ponemos a true para que siga al siguiente nivel

2

¡VAMOS A PROGRAMAR!

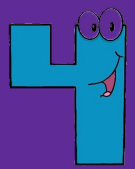
Para vuestro diseño, se puede hacer en un guante, o en cualquier cosa que se os ocurra…

Dadle rienda suelta a vuestra imaginación!!!!

Oficina de Software Libre Universidad de Granada

UNIVERSIDAD DE GRANADA

Plus para el Simon

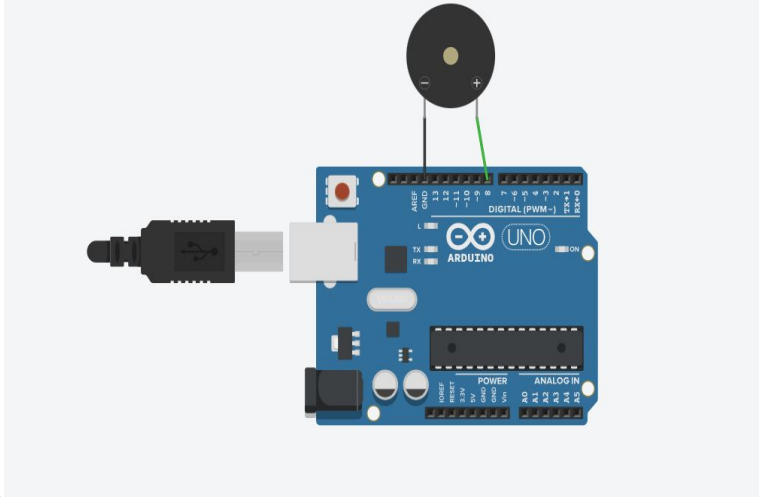


Buzzer para hacer sonidos de victoria

Buzzer

- Usaremos ahora el zumbador(***buzzer***) para crear una melodía:

- ★ Uno de **victoria**, cuando ganemos todas las rondas del Simon

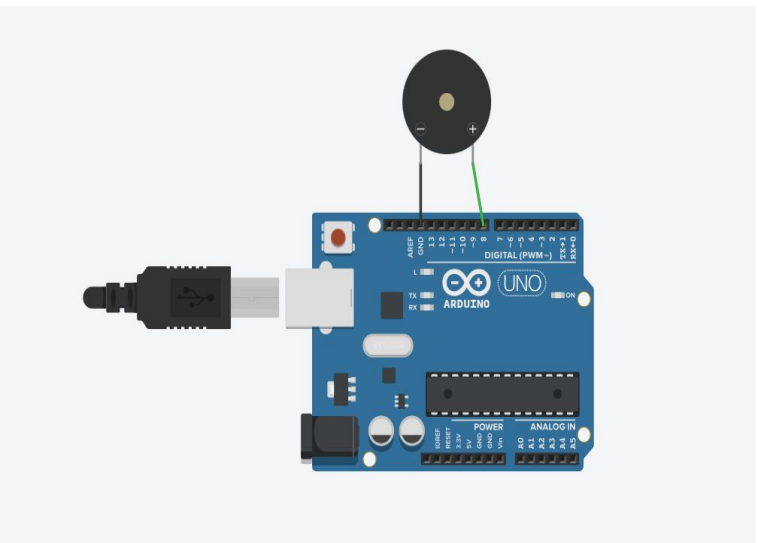


Buzzer

- Usaremos ahora el zumbador(***buzzer***) para crear una melodía:

- ★ Uno de **victoria**, cuando ganemos todas las rondas del Simon

El buzzer se conecta al pin 9

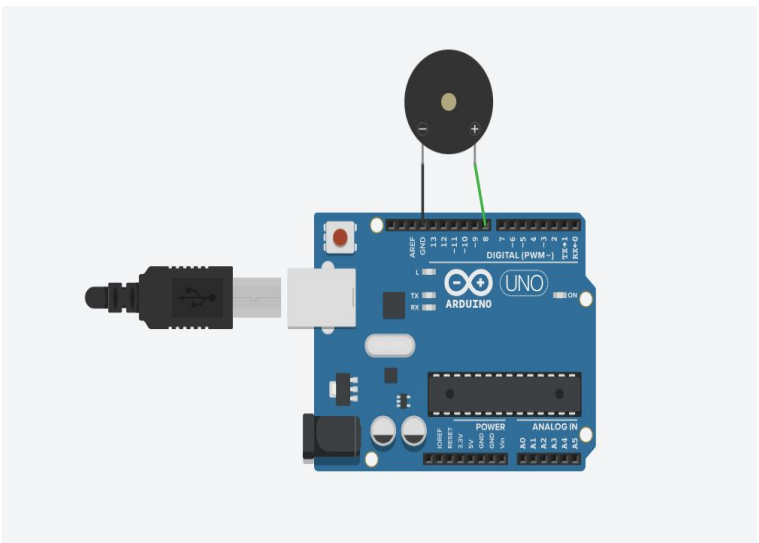


Buzzer

❑ Recordad...

★ **No** podemos usar dos funciones **tones()** al mismo tiempo.
Se debe de dejar un pequeño intervalo de tiempo para que suene la nota → **delay(*time*)**

★ La frecuencia de sonido va desde **31 Hz** a **65535 Hz**.



Buzzer

- Recordad

FRECUENCIA DE LAS NOTAS MUSICALES EN HERCIOS (Hz)									
	OCTAVA 0	OCTAVA 1	OCTAVA 2	OCTAVA 3	OCTAVA 4	OCTAVA 5	OCTAVA 6	OCTAVA 7	OCTAVA 8
Do	16,3516	32,7032	65,4064	130,813	261,626	523,251	1046,50	2093,00	4186,01
Do# / Reb	17,3239	34,6479	69,2957	138,591	277,183	554,365	1108,73	2217,46	4434.92
Re	18,3540	36,7081	73,4162	146,832	293,665	587,330	1174,66	2349,32	4698,64
Re# / Mib	19,4454	38,8909	77,7817	155,563	311,127	622,254	1244,51	2489,02	4978,04
Mi	20,6017	41,2035	82,4069	164,814	329,628	659,255	1318,51	2637,02	5274,04
Fa	21,8268	43,6536	87,3071	174,614	349,228	698,456	1396,91	2793,83	5587,66
Fa# / Solb	23,1246	46,2493	92,4986	184,997	369,994	739,989	1479,98	2959,96	5919,92
Sol	24,4997	48,9995	97,9989	195,998	391,995	783,991	1567,98	3135,96	6271,92
Sol# / Lab	25,9565	51,9130	103,826	207,652	415,305	830,609	1661,22	3322,44	6644,88
La	27,5000	55,0000	110,000	220,000	440,000	880,000	1760,00	3520,00	7040,00
La# / Sib	29,1353	58,2705	116,541	233,082	466,164	932,328	1864,66	3729,31	7458,62
Si	30,8677	61,7354	123,471	246,942	493,883	987,767	1975,53	3951,07	7902,14

- ★ **N**
  fu
  tie
  Se
  pe
  tie
  no
- ★ L
  desde **31 Hz** a **65535 Hz**.

¡Vemos la estructura del programa!

★ **Programar un buzzer**

- Pasa este pseudocódigo a lenguaje del arduino.

```
/*
Inicializamos buzzer
*/

void setup() {
  // put your setup code here, to run once:
  buzzer es un pin de salida (OUTPUT)
}

void loop() {
  // put your main code here, to run repeatedly:
  suena el buzzer con una frecuencia
  dejar x milisegundos
  apagar el sonido del buzzer
}

```

¡Vemos la estructura del programa!

★ **Programar un buzzer**

- Pasa este pseudocódigo a lenguaje del arduino.

Pero,...¿Esto cómo se vería en código?

```
1   /*
2   Inicializamos buzzer
                                  tup code here, to run once:
                                  in de salida (OUTPUT)
9
10  void loop() {
11    // put your main code here, to run repeatedly:
12    suena el buzzer con una frecuencia
13    dejar x milisegundos
14    apagar el sonido del buzzer
15  }
16
```

¿CÓMO LO HARÍA?

Se haría algo parecido a como se explica en este código.

- Veamos el ejemplo de música.
- Primero, inicializamos el buzzer.
  - Se conecta al pin 9.
  - Además, en ***void setup()***, se indica en pinMode(buzzer), OUTPUT) que es OUTPUT(*salida*).
  - Después, en el ***void loop()***, indicamos con **tone()**, la frecuencia que queremos.
  - Después dejamos que suene durante x milisegundos.
  - Y después hacemos que deje de sonar con **noTone()**.

```
1  int buzzer = 9;
2
3  void setup() {
4    Serial.begin(9600);
5    pinMode(buzzer,OUTPUT);
6  }
7
8  void loop() {
9    tone(buzzer,311.13);
10   delay(480);
11   noTone(buzzer);
12   delay(50);
13
14   tone(buzzer,311.13);
15   delay(260);
16   noTone(buzzer);
17   delay(50);
18
19    tone(buzzer,261.63);
20   delay(250);
21    noTone(buzzer);
22   delay(50);
23
24 }
```

¿CÓMO LO HARÍA?

Se haría algo parecido a como se explica en este código.

- Veamos el ejemplo de música.
- Primero, inicializamos el buzzer.
  - Se conecta al pin 9.
  - Además, en ***void setup()***, se indica en pinMode(buzzer), OUTPUT) que es OUTPUT(*salida*).
  - Después, en el ***void loop()***, indicamos con **tone()**, la frecuencia que queremos.
  - Después dejamos que suene durante x milisegundos.
  - Y después hacemos que deje de sonar con **noTone()**.

```
1  int buzzer = 9;
2
3  void setup() {
4    Serial.begin(9600);
5    pinMode(buzzer,OUTPUT);
6  }
7
8  void loop() {
9    tone(buzzer,311.13);
10   delay(480);
11   noTone(buzzer);
12   delay(50);
13
14   tone(buzzer,311.13);
15   delay(260);
16   noTone(buzzer);
17   delay(50);
18
19    tone(buzzer,261.63);
20   delay(250);
21    noTone(buzzer);
22   delay(50);
23
24 }
```

Buzzer

- En el código indicaremos estas instrucciones donde ponemos la **condición** de que hemos **ganado** (*la melodía de victoria*).
- Tendremos que inventarnos esa melodía, e indicar en el código cuando debería de sonar ;)

BUZZER

- Las funciones son **agrupaciones de código**
- Ya conocemos las funciones
  - void setup(){ … }
  - void loop() { … }
- Cuando delante del nombre de la función aparece **void** significa que la función **no devuelve nada.**
- Las funciones pueden devolver variables.
- Delante del nombre de la función hay que añadir el **tipo de variable** que devuelve.
- **Esta función es void porque no devuelve nada, ejecuta la música solo.**

```
void victoria(){
    METE AQUÍ DENTRO EL CÓDIGO
    QUE HAS ESCRITO
}
```

Esto hay que hacerlo en el scketch donde vamos a crear el simon. **Las funciones se añaden debajo del setup**

BUZZER

Imaginemos que el código de abajo es el nivel final…para decir que hemos ganado, la función de victoria que hemos creado, la tenemos que poner **aquí**.

si nivel X, **entonces**

encendemos el led correspondiente con el color del primer elemento de la secuencia

lo enseñamos

esperamos 500ms

limpiamos la configuración

lo enseñamos

ponemos a false la variable para ese nivel

1

si valor del sensor > valor más alto que nos da cuando no lo pulsamos, **entonces**

encendemos el led correspondiente con un color

lo enseñamos

esperamos 500ms

limpiamos la configuración

lo enseñamos

función de victoria

2