Oficina de Software Libre Universidad de Granada

UNIVERSIDAD DE GRANADA

Cubo Luminoso

¡Creemos una lámpara mágica!

Autora:

Inés Jiménez Díaz

@https_rim
@_https.rim_

Oficina de Software Libre Universidad de Granada

UNIVERSIDAD DE GRANADA

Hardware necesario para CUBO LUMINOSO

Cubo luminoso

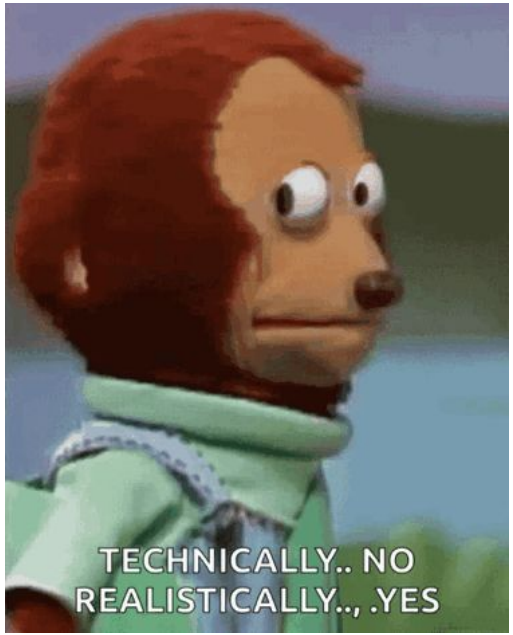
- Arduino Uno
- Sensor de proximidad HSR-04
- Tira de Leds neopixel
- Pantalla LCD
- Cables arduino
- 2 Servomotores
- 3 botones

Antes de empezar...

¿Qué tengo que hacer para cargar una librería?

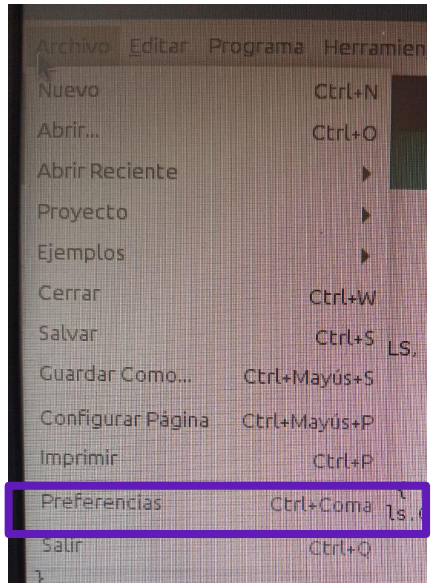
Cargar librería

- **Las librerías las vamos a tener que cargar TODOS los días en los ordenadores del campus.**
- Pero primero, antes de cargarlas, tenemos que configurar el IDE….



Cargar librería

- Para ello, entramos en el IDE de arduino, y nos vamos a preferencias.

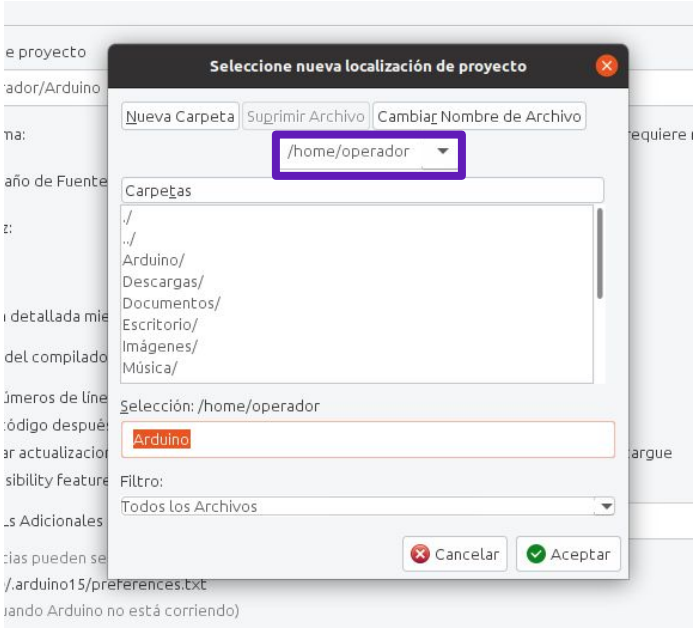


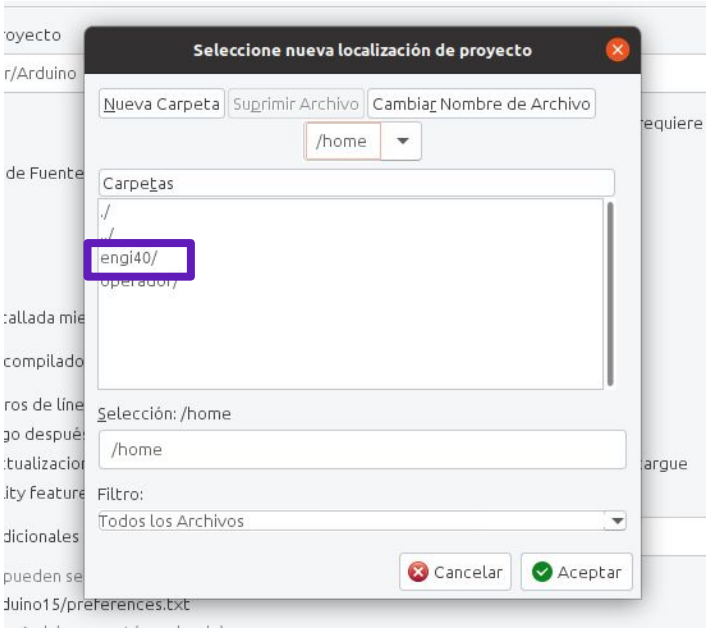
Cargar librería

- Aquí nos saldrá una pestaña como la que sale aquí, vemos que sale “Operador”.
- Esto significa que se está usando la carpeta de operador, pero nosotras queremos nuestra carpeta “engiXX”.
- Para ello…

Cargar librería

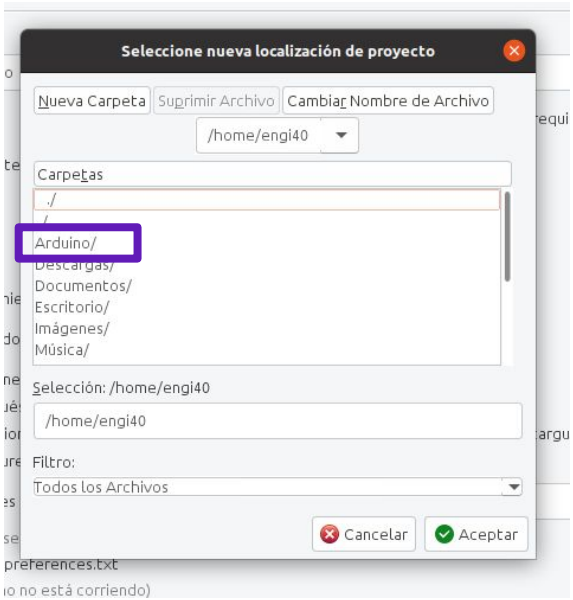
Para ello…

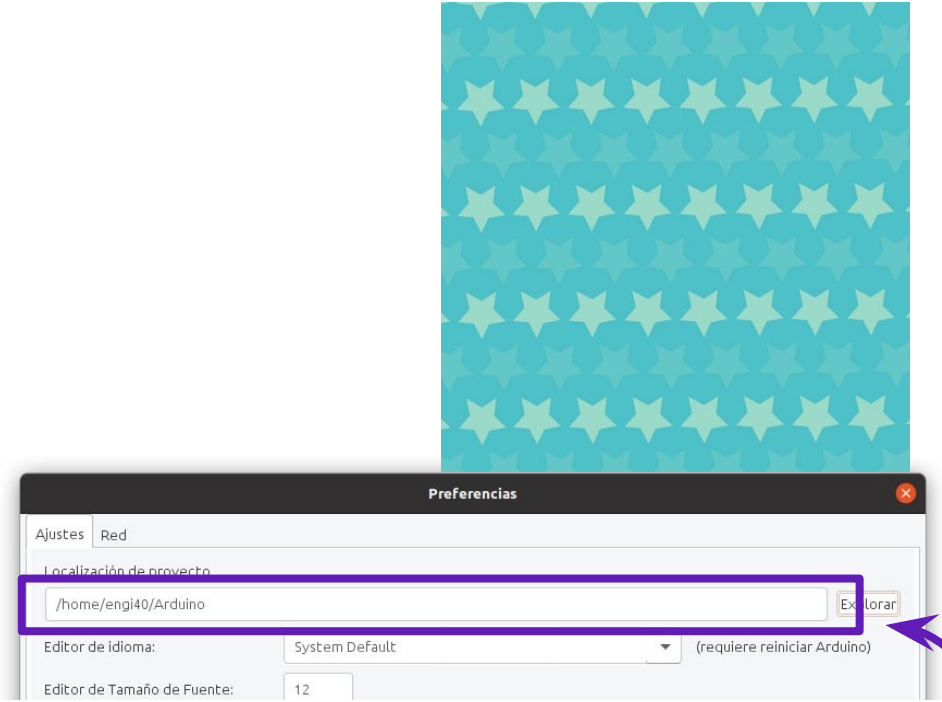




Cargar librería

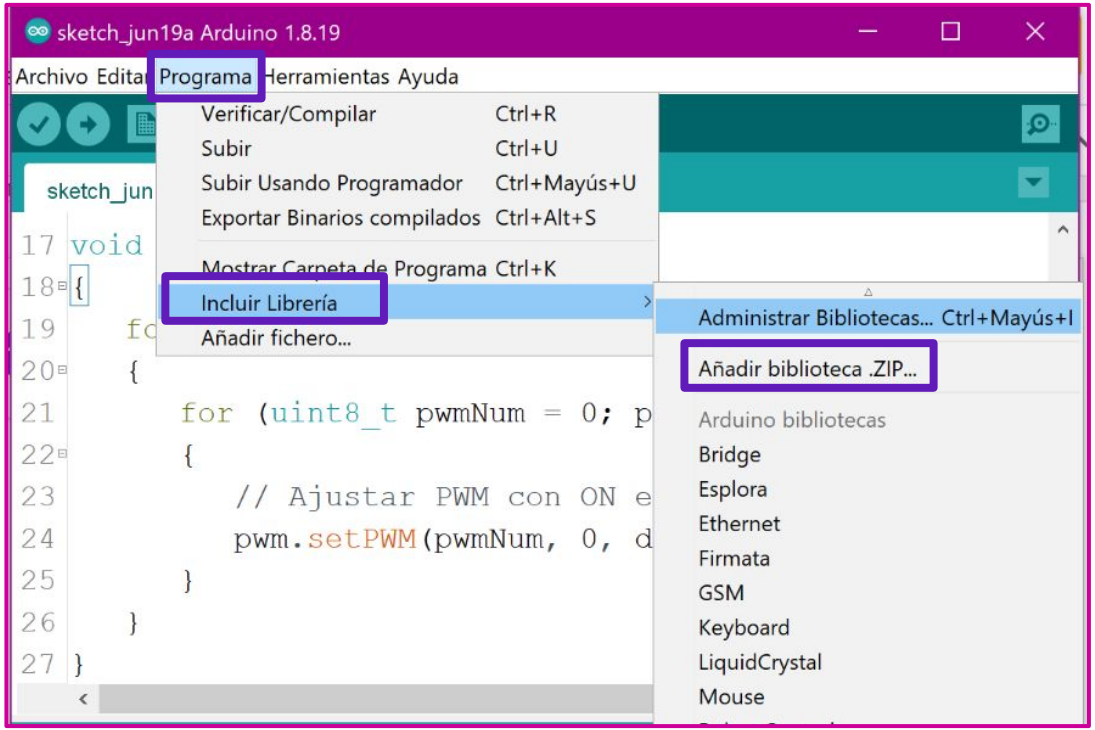
Para ello…





Así debería de quedar

Cargar librería



Ya después de esta configuración,

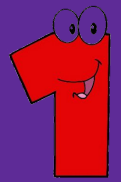
podemos añadimos las **librerías** necesarias.

Repetir este paso para cada una de las librerías.

Oficina de Software Libre Universidad de Granada

UNIVERSIDAD DE GRANADA

Montaje del Cubo Luminoso



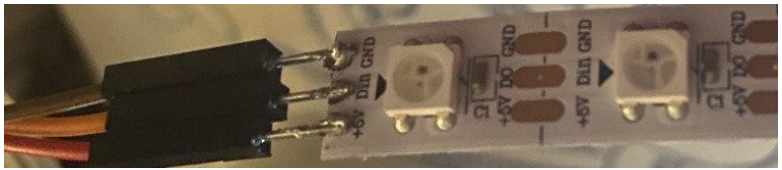
¡Hágase la luz!

¡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

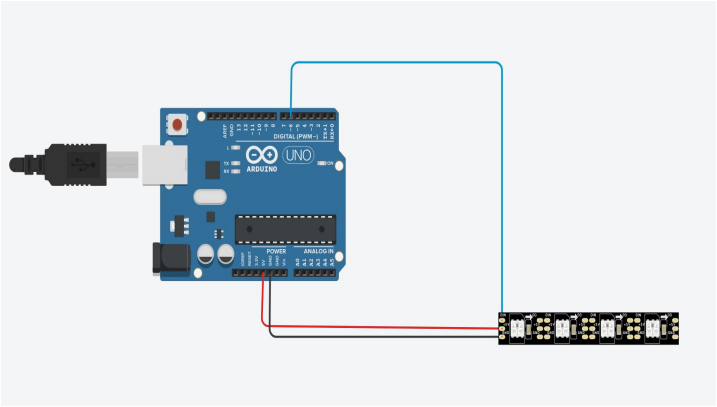
¡HÁGASE LA LUZ!

- **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

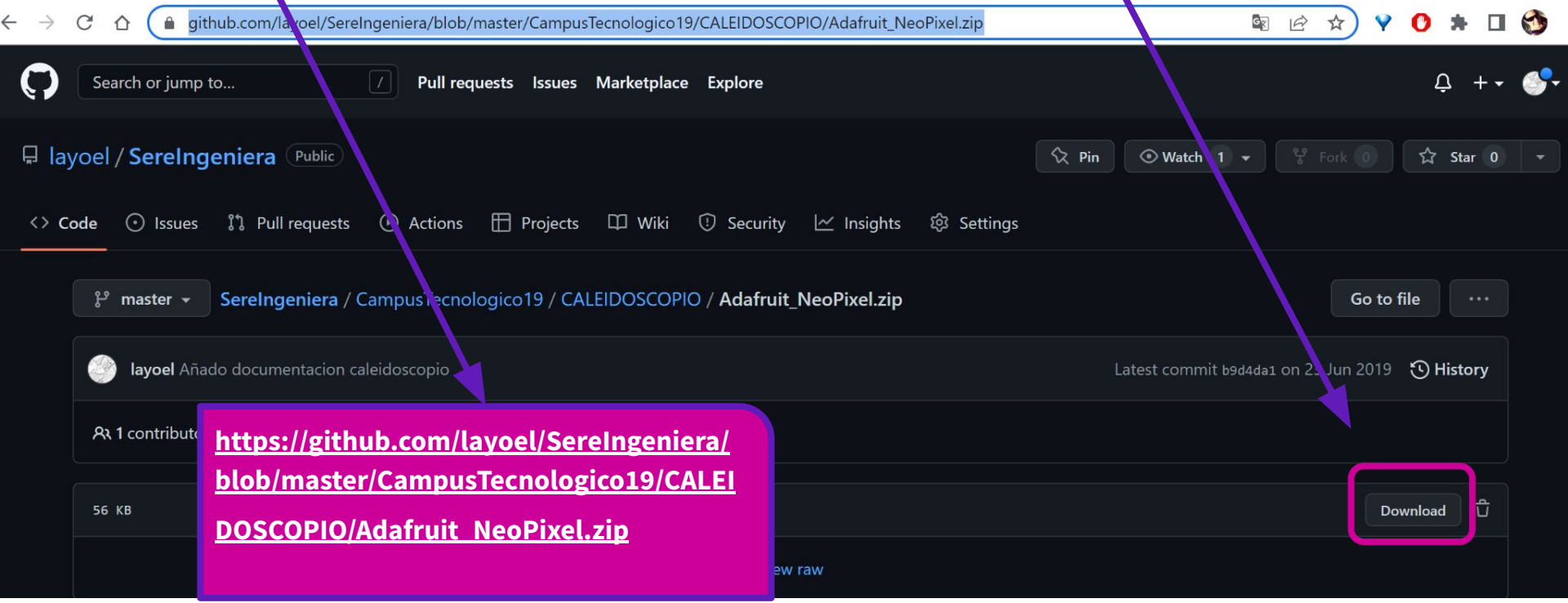
- Hay **dos formas de incluir librerías** a la biblioteca de nuestro arduino.
  - Si tenemos el archivo comprimido con extensión .zip
  - Buscarlas en el repositorio de Arduino

- 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

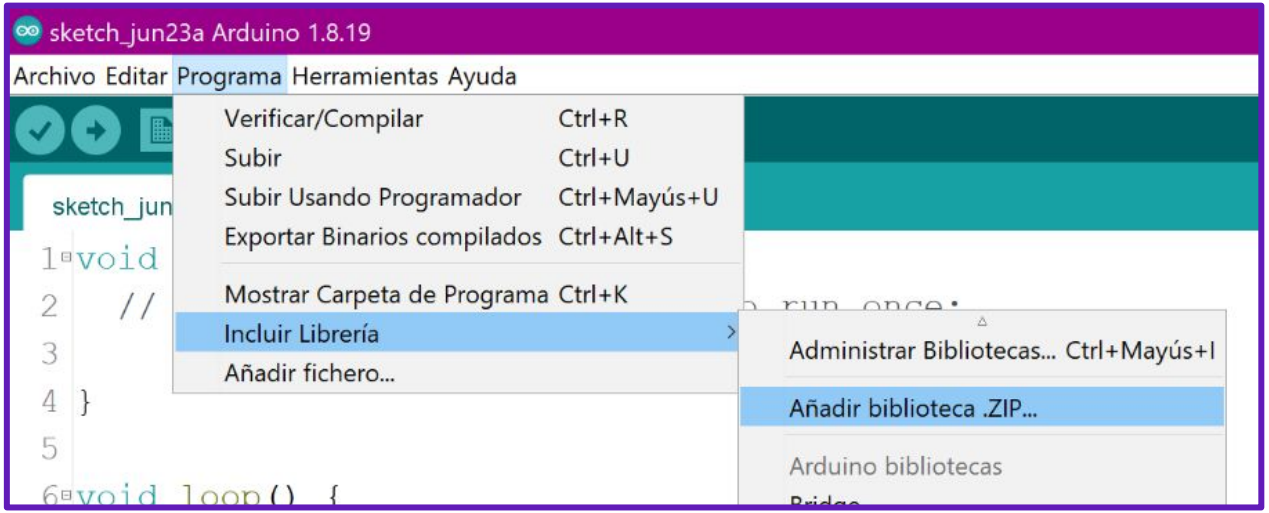
Vamos al enlace y descargamos la librería haciendo clic en download



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

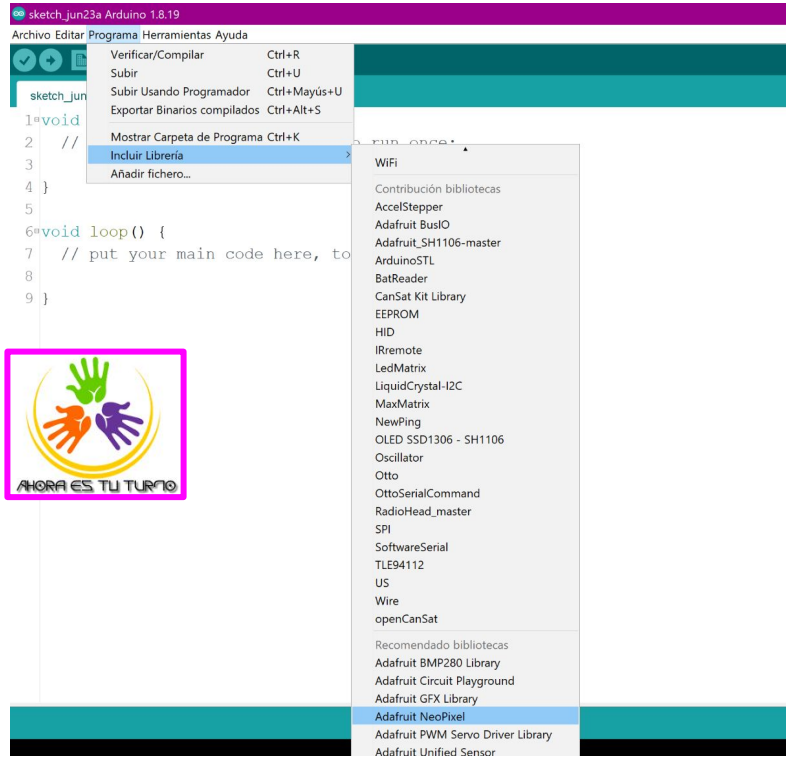
¡HÁGASE LA LUZ!

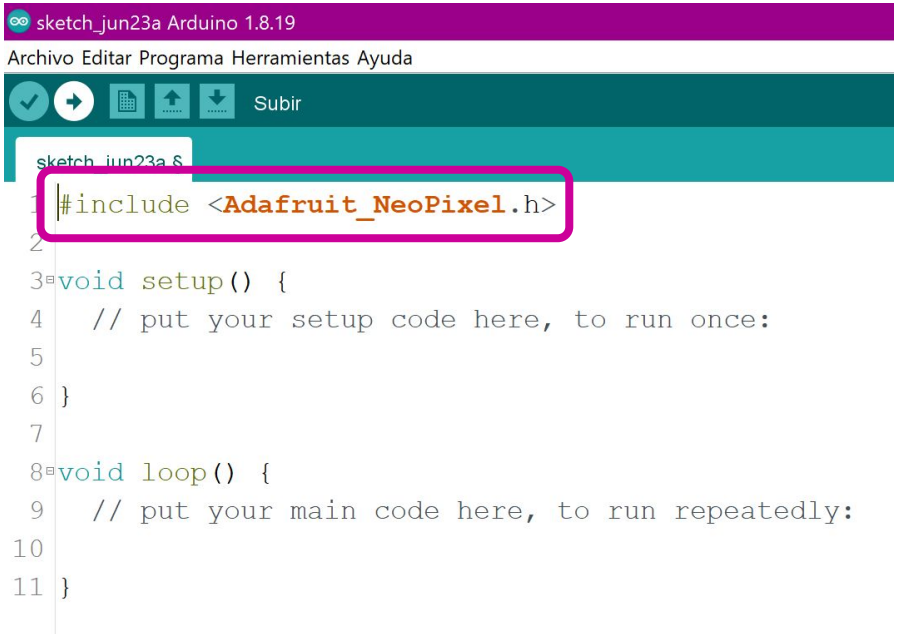
1. Incluir en el IDE la librería Adafruit_NeoPixel



¡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

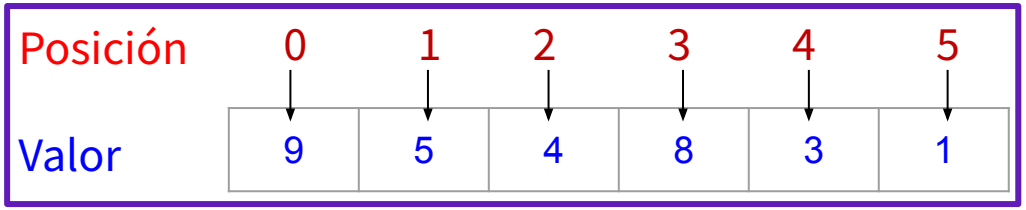
Archivo Editar Programa Herramientas Ayuda

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:

}
```

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

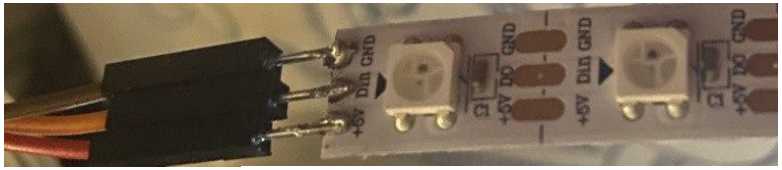
- Antes de empezar a usar las funciones de la librería necesitamos saber cómo acceder a las posiciones de cada led.
- ¿Sabes que es un **vector**?



- 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!

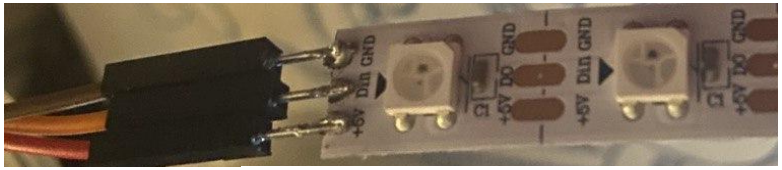
4. **Aprendemos a usar la librería para encender uno a uno los leds de la tira de led de colores.**



- ¿Cómo encedemos un led?

¡HÁGASE LA LUZ!

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!**

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);**
  - **pixels.show();**
    - Muestra el color

¡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.**
  - **Al final del código tenemos que mostrar los leds.**


```
#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.
  - Al final, hay que mostrar las luces.

```
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
  }

¡HÁGASE LA LUZ!

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.

```
#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));
        }
```

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 Cubo Luminoso

Servomotores

Servomotores

- **¿Qué es y cómo funciona un servomotor?**

  Un servo es un tipo de motor en el que indicamos directamente el ángulo que queremos y el servo se encarga de posicionarse en este ángulo.

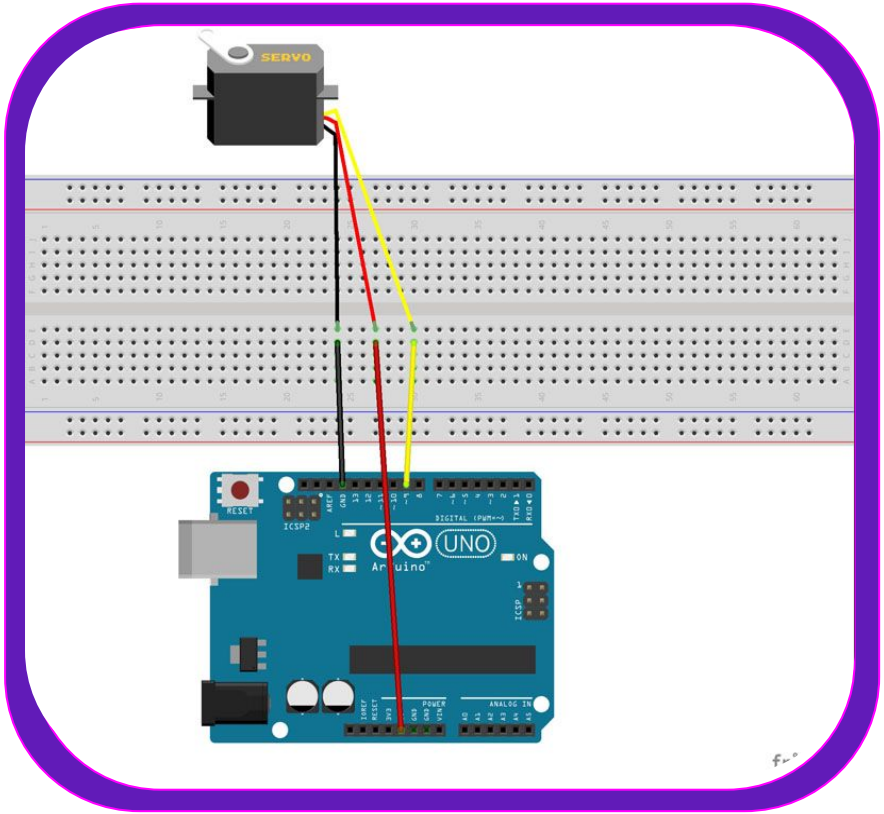
- Típicamente los servos disponen de un rango de movimiento de entre 0 a 180º.

https://www.luisllamas.es/controlar-un-servo-con-arduino/

Servomotores

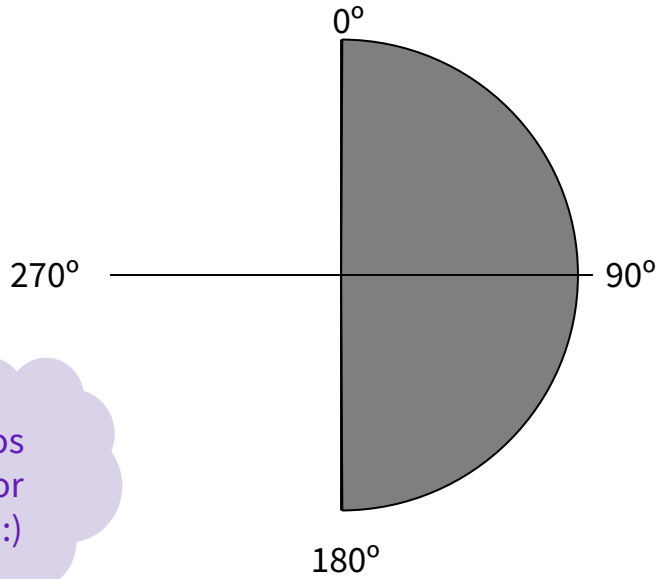
- **Servo motor 180º.**
  - Tienen 3 pines:
    - **DIN** pin de entrada **PWM,** tienen un **~**
    - **+5V** proporciona energía al servo. **VCC**
    - **GND** Toma de tierra.**GND**

¡Ojo! no te equivoques al conectarlo que puedes quemar el motor.



ServoMotor

Los servomotores tienen un rango de movimiento limitado, que es de 0 a 180 grados.



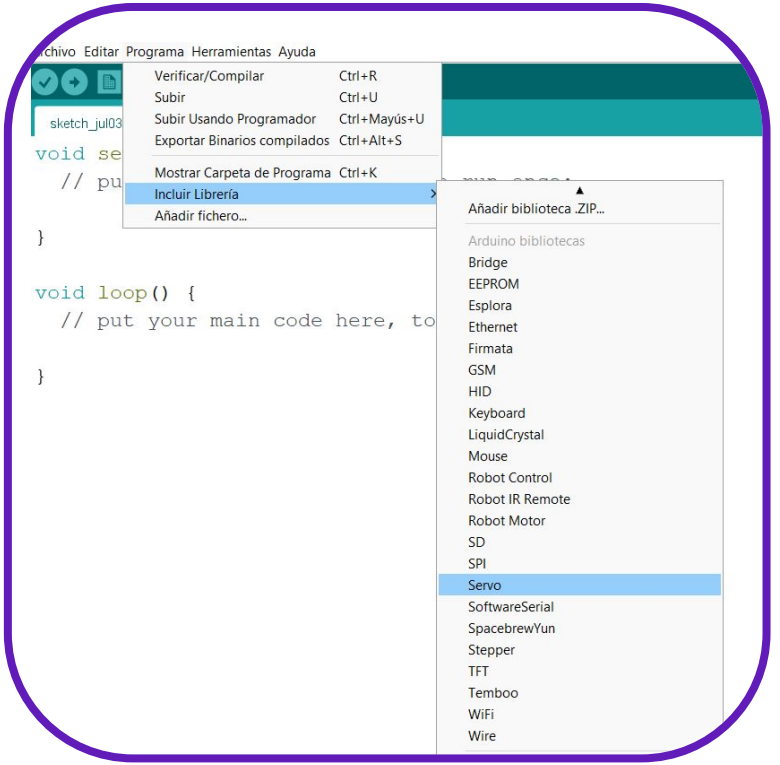
Os dejo los grados por si dudáis :)

SERVOMOTOR

2. ¡Comencemos a programar!

Para este servomotor hace falta incluir una **librería**, para ello pulsamos en:

1. Programa
2. Incluir librería
3. pinchamos en: Servo



ServoMotor

Y para usar el servomotor hay que asociar el objeto **servo** al pin del servo:

servo §

```
#include <Servo.h>

Servo servoMotor; // Crea un objeto Servo

int servoPin = 8; // Pin al que está conectado el servo

void setup() {
  servoMotor.attach(servoPin); // Asocia el objeto Servo al pin del servo
}
```

Servomotores

Este es un pequeño ejemplo de cómo funciona:

Los bucles for son para que gire los grados que queramos.

Dentro del bucle, servo1.write(pos) es para que cambie el servo a ese grado.

NOTA 1: Aquí gira para un sentido

NOTA 2: Aquí gira para el otro sentido.

```
for (pos = 0; pos <= 50; pos += 1) {
  servo1.write(pos);
  delay(15);
}
delay(1000);
for (pos = 50; pos >= 0; pos -= 1) {
  servo1.write(pos);
  delay(15);
}
```

Servomotores

Utilizamos el servo - Vamos a hacer que gire 90 º cada segundo:

- **servoMotor.write(0);**
  - El servomotor se pone a 0 grados
- **delay(1000);**
  - Esperamos 1 segundo

servo §

```
#include <Servo.h>

Servo servoMotor; // Crea un objeto Servo

int servoPin = 8; // Pin al que está conectado el servo

void setup() {
  servoMotor.attach(servoPin); // Asocia el objeto Servo al pin del servo
}
```

escribimos el punto 0 en el objeto servo
esperamos 1 seg

. . . vosotras . . .

(Hasta 180)

```
}
```

Oficina de Software Libre Universidad de Granada

UNIVERSIDAD DE GRANADA

Montaje del Cubo Luminoso

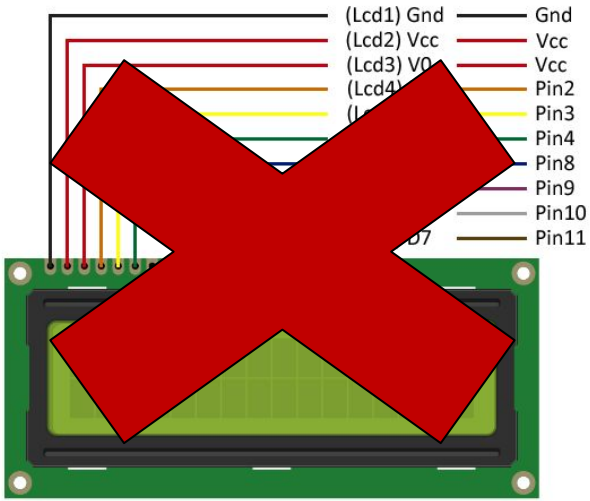
¡Pantalla LCD!

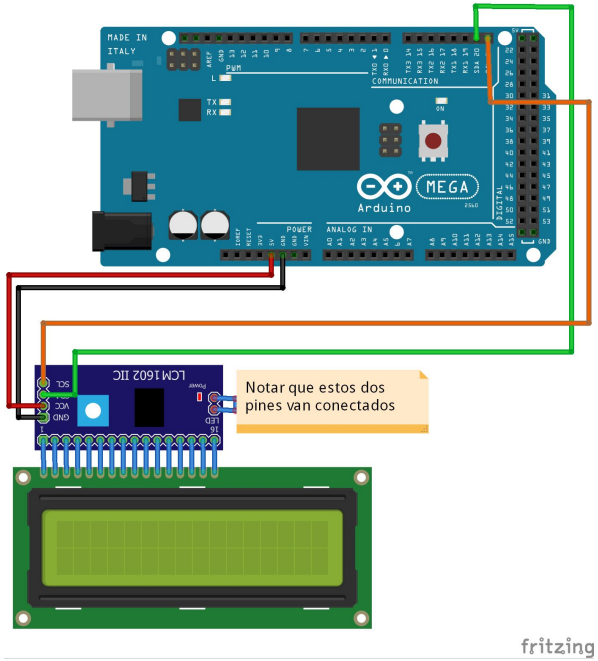
Pantalla LCD

- ¿Qué es una pantalla LCD?
  - Pantalla de cristal líquido
  - Funciona con cristales líquidos que reaccionan a la electricidad. Estos se alinean o no dependiendo de la corriente eléctrica.
  - Estas pantallas se utilizan en ordenadores, tablets, relojes, …
- ¿Para qué puede servir nuestra pantalla LDC?
  - Para mostrar datos por pantalla

Pantalla LDC

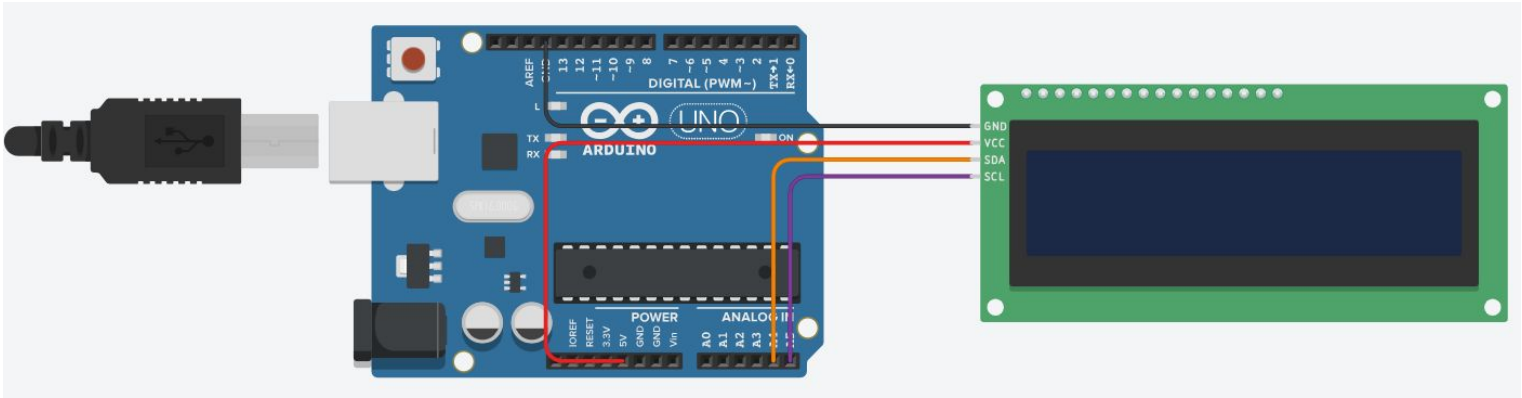
- La pantalla que vamos a utilizar: 16x02
  - con un controlador para no malgastar pines.





Juguemos con la pantalla

- Tiene 4 pines:
  - **SCL** y **SDA** conexión analógica
  - **+5V** proporciona energía a los leds
  - **GND** Toma de tierra



Pantalla LCD

- 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 pantalla tenemos el archivo comprimido con el nombre **Arduino-LiquidCrystal-I2C.zip**
- La **descargamos** y la incluimos de la 1º forma.

Pantalla LCD

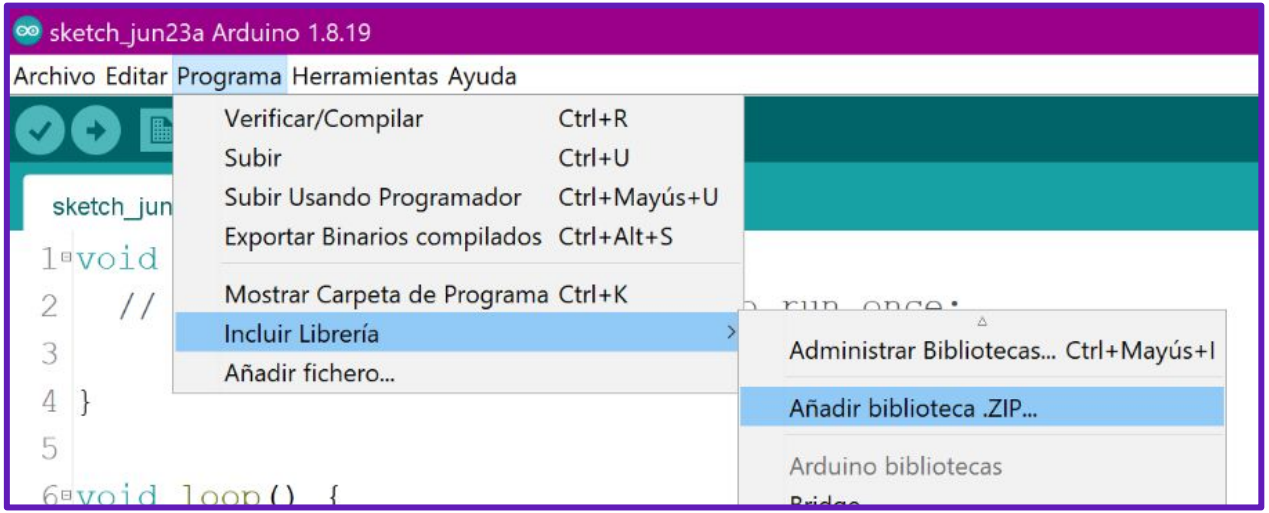
1. Incluir en el IDE la librería LiquidCrystal.h

Vamos al enlace y descargamos la librería haciendo clic en download

https://www.arduino.cc/reference/en/libraries/liquidcrystal-i2c/

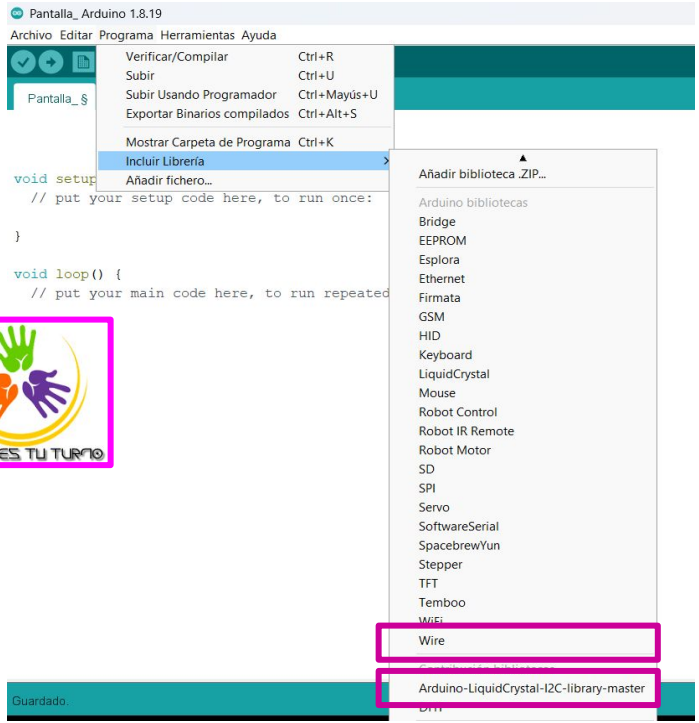
Pantalla LCD

1. Incluir en el IDE la librería **LiquidCrystal.h** ✓



Pantalla LCD

2. Para poder usar la librería tenemos que añadirla en el archivo. Vamos a añadir también wire.



Pantalla LCD

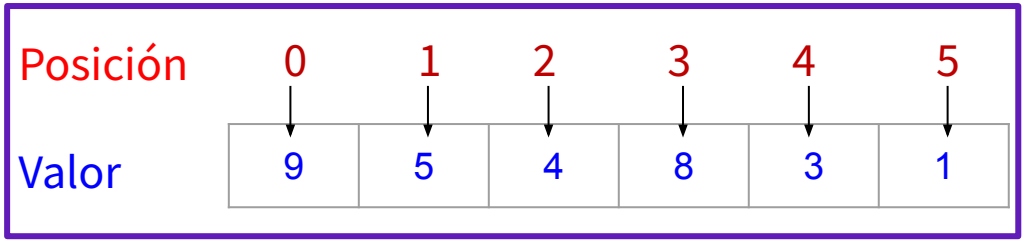
3. Declaramos un objeto basándonos en nuestra pantalla

- Para ello utilizaremos la siguiente **función de la librería LiquidCrystal_I2C** que acabamos de incluir.
  - **LiquidCrystal_I2C lcd(0x27, 16, 2);**
- Se inicializa con:
  - 0x27 porque se inicializa en esa dirección
  - 16 = número de caracteres por línea
  - 2 = número de líneas

- Vale pero, ¿qué tenemos que hacer y cómo? Poco a poco

4. Aprendemos a usar la librería para utilizar cada punto de nuestra pantalla.

- ❑ Antes de empezar a usar las funciones de la librería necesitamos saber cómo usar las posiciones de cada línea.
- ❑ ¿Sabes qué 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**.
- En nuestra pantalla, habrá dos vectores, que irán del… **0 al 15**

Pantalla LDC

4. Aprendemos a usar la librería para mostrar mensajes por pantalla:

- Para poder utilizar nuestra pantalla vamos a tener que aprender las funciones de la librería:
  - lcd.begin() se utiliza para inicializar la comunicación con la pantalla LCD.
  - lcd.backlight() enciende la retroiluminación de la pantalla LCD.
  - lcd.setCursor(0, 0) establece la posición del cursor en la columna 0 y la fila 0.
  - lcd.print("Hola") muestra el texto "Hola" en la pantalla LCD.
  - lcd.clear() borra el contenido de la pantalla LCD.

¿Creéis que podríais hacer que la pantalla muestre en la primera línea “Hola”?

Pantalla LDC

4. Aprendemos a usar la librería para mostrar mensajes por pantalla:

- lcd.begin()
- lcd.backlight()
- lcd.setCursor(0, 0)
- lcd.print("Hola")
- lcd.clear()

pantalla

```
#include <LiquidCrystal_I2C.h>
#include <Wire.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);
  // Inicia el LCD en la dirección 0x27, con 16 caracteres y 2 líneas

void setup()
{
}
```

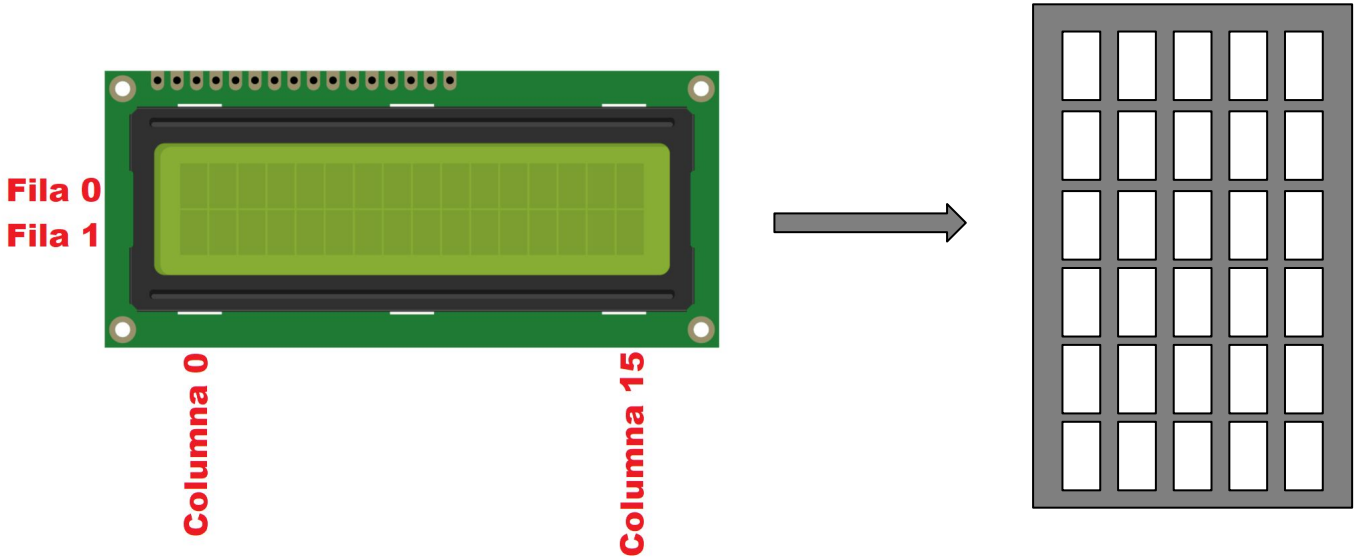
```
loop{
      inicio la pantalla
      luz de fondo
      empiezo a escribir en línea
      añado el texto que quiera
      espero 2 segundos
      limpio pantalla
}
```

Pantalla LCD

4. Aprendemos a usar la librería para mostrar mensajes por pantalla:

   EXTRA:

- ¿Molaría poder poner iconos un poco más complicados?
- Vamos a entrar más en profundidad en cómo se muestran las letras en pantalla:

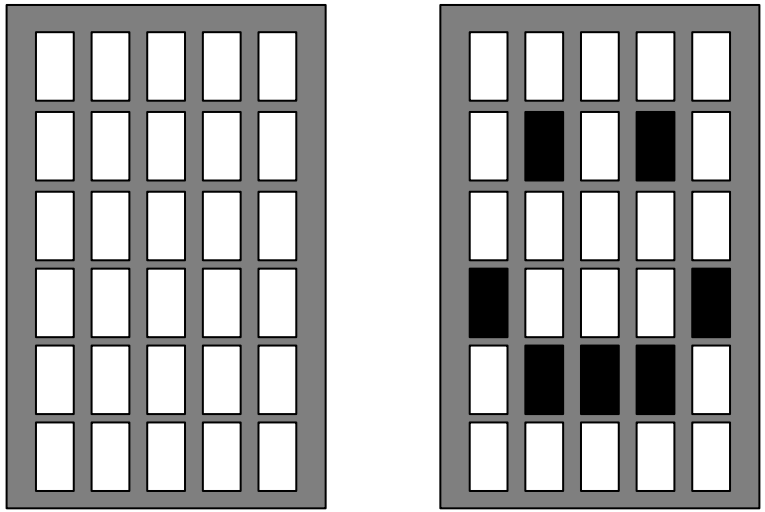


Pantalla LCD

4. Aprendemos a usar la librería para mostrar mensajes por pantalla:

   EXTRA:

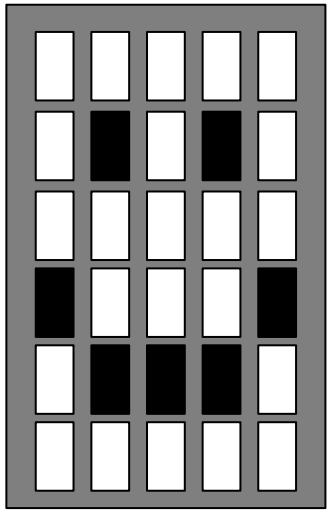
- ❑ Si quisiéramos poner una carita sonriente, ¿Qué puntos blancos colorearíais?



Pantalla LCD

4. Aprendemos a usar la librería para mostrar mensajes por pantalla:

EXTRA:

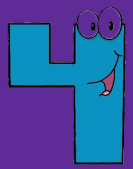


```
byte sonrisa[8] = {
  B00000,
  B01010,
  B00000,
  B00000,
  B10001,
  B01110,
  B00000,
};

void setup()
{
  lcd.begin();
  lcd.createChar(0, sonrisa);
}

void loop()
{
  lcd.setCursor(0, 0);
  lcd.write((byte)0);
  lcd.setCursor(1, 0);
  lcd.print("Hola");
  lcd.setCursor(0, 1);
  lcd.print("Ingenieras!");
  delay(2000);
  lcd.clear();
  delay(1000);
}
```

Oficina de Software Libre Universidad de Granada

UNIVERSIDAD DE GRANADA

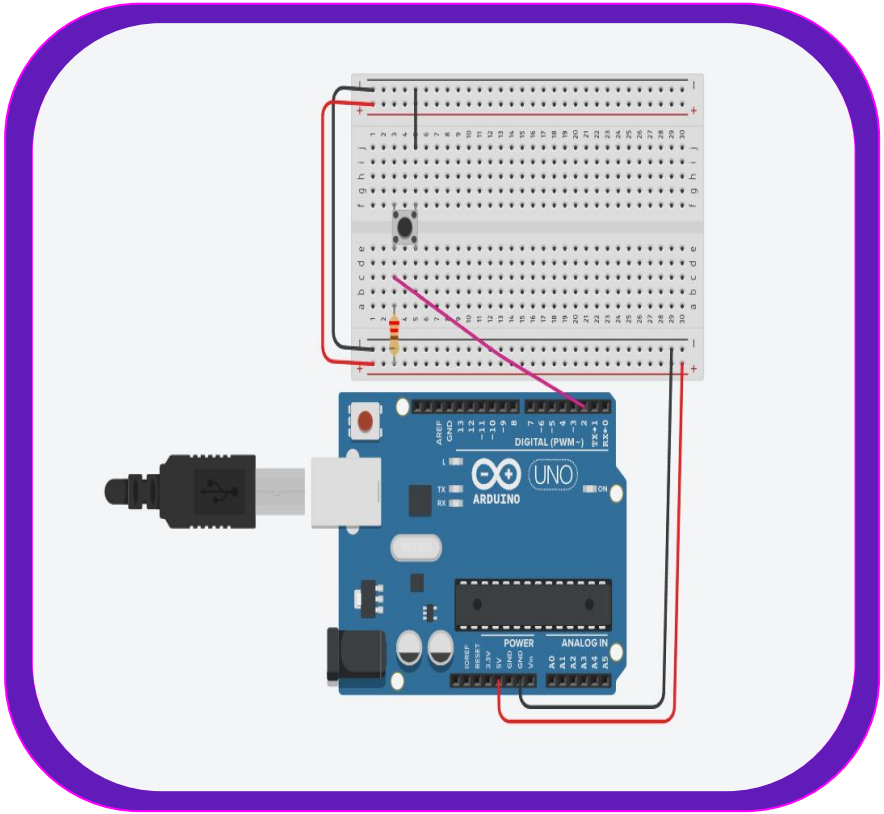
Montaje del Cubo Luminoso



¡Botones!

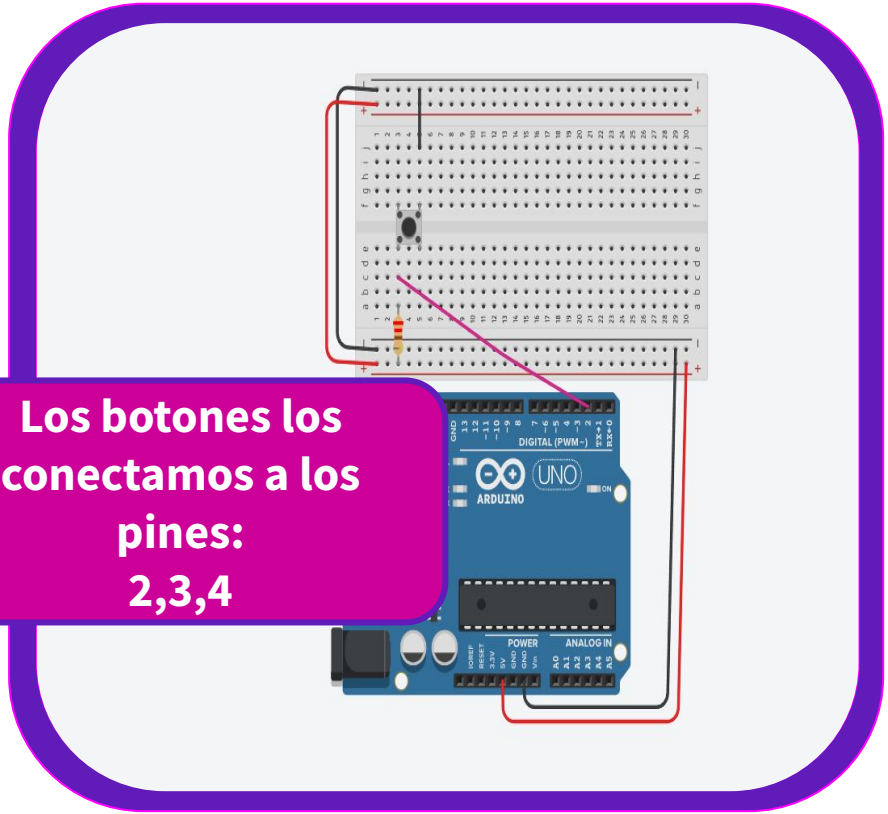
Botones

- Conectamos **un botón al Arduino UNO**
  - Pull up y Pull Down
    - **Pata 1 del botón**
    - **Pata 4 del botón**
    - **Resistencia**
- **Pata de la parte de abajo** del botón va al pin de arduino que enviará la señal **pin2** y a **VCC** respectivamente para cada botón.
- **La otra pata(la de la parte de arriba)** del botón va a **GND.**
- **Resistencia** va de la **pata de abajo** del botón a **vcc**.



Botones

- Conectamos **un botón al Arduino UNO**
  - Pull up y Pull Down
    - **Pata 1 del botón**
    - **Pata 4 del botón**
    - **Resistencia**
- **Pata de la parte de abajo** del botón va al pin de arduino que enviará la señal **pin2** y a **VCC** respectivamente para cada botón. (Pull up)
- **La otra pata(la de la parte de arriba)** d botón va a **GND.**
- **Resistencia** va de la **pata de abajo** del botón a **vcc** si es pull up.



Oficina de Software Libre Universidad de Granada

UNIVERSIDAD DE GRANADA

Montaje del Cubo Luminoso

¡Sensor de proximidad!

Sensor de proximidad

Un sensor ultrasónico mide la distancia a un objeto utilizando **ondas sonoras.**

Componentes Principales:

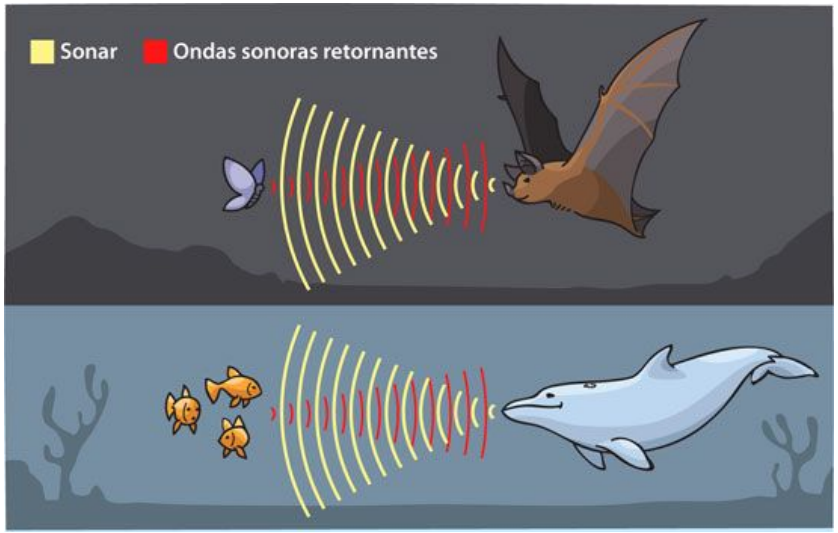
- Emisor: Envía un pulso ultrasónico.
- Receptor: Recibe el pulso reflejado desde el objeto.

El sensor envía un pulso ultrasónico y mide el tiempo que tarda en regresar después de rebotar en un objeto.

La distancia se calcula con la fórmula: ***Distancia = Tiempo * Velocidad del Sonido / 2.***

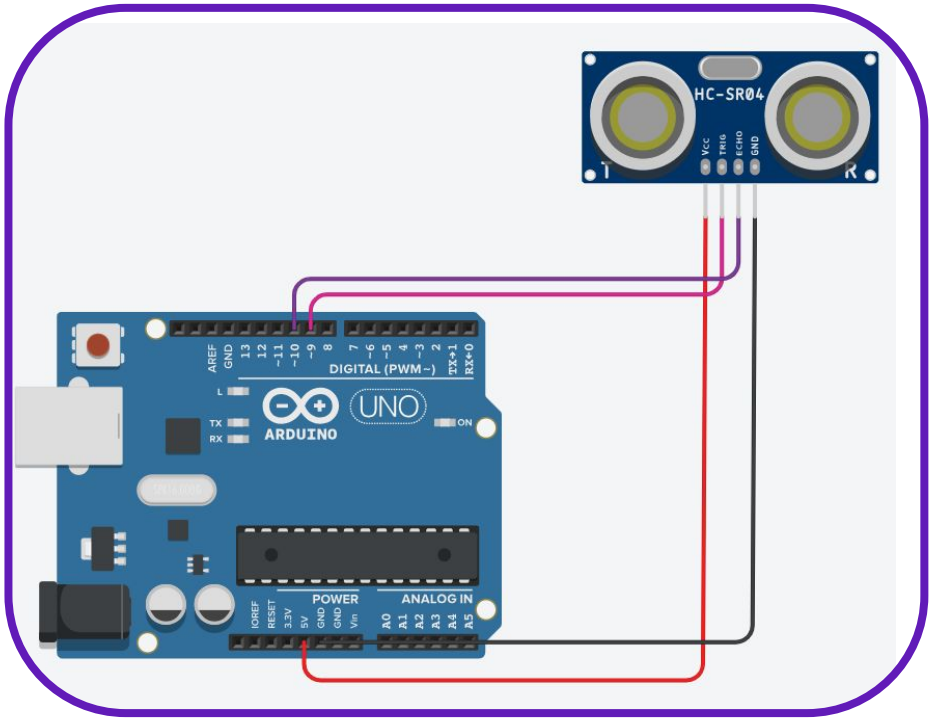
Sensor proximidad

Algunos animales, como los murciélagos y los delfines, utilizan una técnica muy similar al funcionamiento de este sensor, llamada **ecolocación**.



Sensor proximidad

1. Conectamos el sensor al arduino:
   - VCC al **pin de 5V**, positivo
   - GND al **pin GND**, negativo
   - TRIG al **pin 6.**
   - ECHO al **pin 5.**



TRIG se utiliza para enviar un pulso ultrasónico.

ECHO se utiliza para recibir el pulso reflejado y medir el tiempo transcurrido.

Sensor Ultrasónico

2. Utilizamos el sensor:

Vamos a mostrar un mensaje diciendo la distancia a la que está el objeto. Esto se hace dentro del loop().

RECUERDA: trigPin envia señal, echoPin recibe señal.

1. Definimos los pines del Sensor
2. Hacemos la configuración inicial
3. Vamos a una función llamada ping que devuelve los cm. Y cuando los devuelve, los enseña por pantalla
4. Enviamos señal trigger
5. Se calcula la distancia y se devuelve

```
const int EchoPin = 5;
const int TriggerPin = 6;

void setup() {
                ;)
}

void loop() {
                ;)
}

int ping(int TriggerPin, int EchoPin) {
  long duration, distanceCm;

    Mando señal baja de trigger
    espero 4 microsegundos
    Mando señal alta de trigger
    espero 10 microsegundos

  duration = pulseIn(EchoPin, HIGH);  //

  distanceCm = duration * 10 / 292/ 2;
  return distanceCm;
}
```

Tenéis que investigar sobre los distintos tipos de delay que existen en arduino!!!

Oficina de Software Libre Universidad de Granada

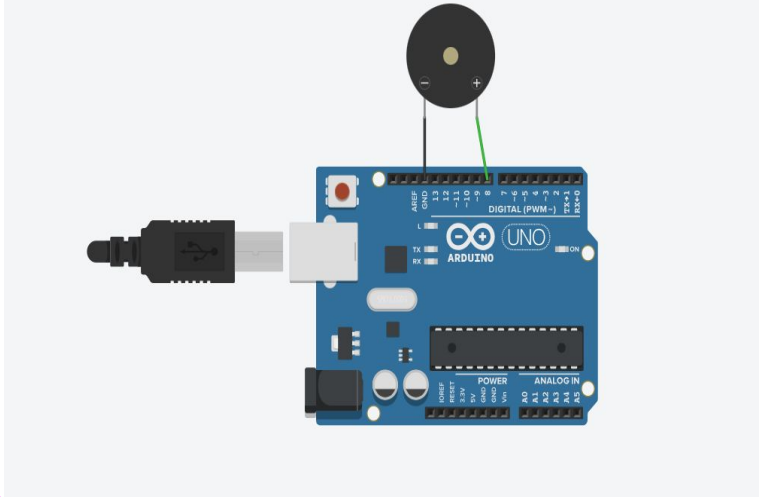
UNIVERSIDAD DE GRANADA

Montaje del Cubo Luminoso

¡Buzzer!

Buzzer

- Recordad que no podemos usar dos funciones tones() al mismo tiempo. Deberemos apagar el tono con la funcón noTone() antes de poder usarlo en otro pin.
- Los rangos de la función tone son de 31 Hz a 65535 Hz.

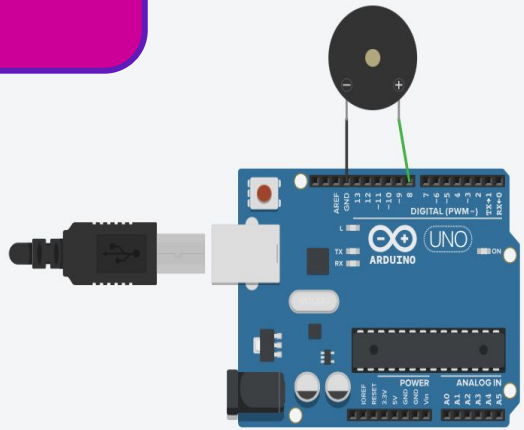


/

Buzzer

- Recordad que no podemos usar dos funciones tones() al mismo tiempo. Deberemos apagar el tono con la fu noTone() antes de poder usarlo en
- Los rangos de la función tone son d a 65535 Hz.

Lo vamos a conectar al pin 9



https://www.luisllamas.es/reproducir-sonidos-arduino-buzzer-pasivo-altavoz/

Buzzer

- Recordad que no podemos usar dos funciones tones() al mismo tiempo. Deberemos apagar el tono con la funcón noTone() antes de poder usarlo en otro pin.
- Los rangos de la función tone a 65535 Hz.

Tabla de frecuencias:

NOTA	FECUENCIA
DO	532,25
DOS	554,37
RE	584,33
RES	622,25
MI	659,26
FA	698,46
FAS	739,99
SOL	789,99
SOLS	830,61
LA	880
LAS	932,33
SI	987,77
RE2	1174,66
FAS2	1479,98

Oficina de Software Libre Universidad de Granada

UNIVERSIDAD DE GRANADA

Montaje del Cubo Luminoso

¡Programación!

Recordemos...

Condicionales if e if else

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

    sentencias;
}
```

Atención!!! Estructura nueva!

switch

```
switch(variable ){
    case variable1:
        sentencias;
        break;
    ...
    case variableN:
        sentencias;
        break;

}
```

¿Qué vamos a hacer con esta estructura?

Así vamos a crear distintas acciones con los botones y el sensor de proximidad, por ejemplo:

Mientras esté a menos de 10 cm:

Si pulso el boton X, que suene una canción.

Si pulso el boton Y o Z, que se sume a 1 una variable llamada modo, y cambio el tipo de luz dependiendo del valor de la variable.

¿Qué vamos a hacer con esta estructura?

De ahí, con los componentes que hemos aprendido a programar antes, tenéis rienda suelta a vuestra imaginación para hacer las **funciones que queráis**!!

Oficina de Software Libre Universidad de Granada

UNIVERSIDAD DE GRANADA

Montaje del Cubo Luminoso

¡Extra!

EXTRA 1

¡Vamos a decorarlo! Por el momento solo tenemos una simple estructura…¡Vamos a darle nuestro toque añadiendole colores, piezas creadas por vosotras, o impresas con vuestros diseños!

Extra 2

Todo el código lo vamos a ir dividiendo en funciones, de modo que en el void loop, solo haya llamadas a funciones dentro del switch…¡Así queda más limpio y profesional!