Teatro Lambe Lambe

Autora:

María del Mar Martínez Robles

Presentación

Inspiración

- Todos hemos jugado con títeres en algún momento o hemos visto cómo los usan en series de televisión y películas.
- En este proyecto, vamos a crear nuestro propio teatro en miniatura interactivo, con luces, telón y otros añadidos para crear escenas y contar una historia.
- ¡Lo que aprendamos aquí también sirve en otros proyectos! **¿Se te ocurre alguno?**

Presentación

Teatro lambe lambe

- Cada proyecto consistirá en una caja “escenario”, con luces y un telón.
- Tendremos un panel de control detrás con el que controlaremos lo que ocurra en nuestro *show*
- ¡Es tu proyecto! Puedes hacerlo con la temática que quieras, añadir o quitar lo que te guste o incluso sugerir funcionalidades.

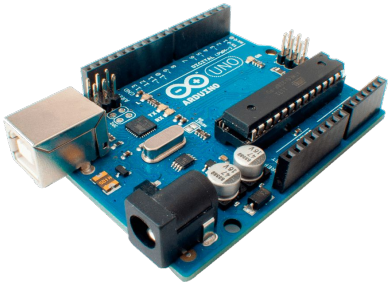
¡No hay preguntas incorrectas!

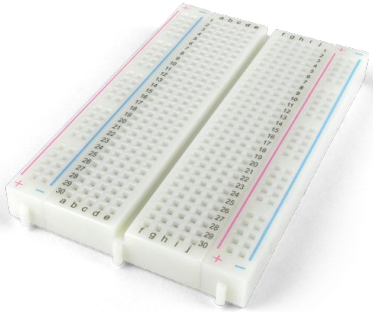
Oficina de Software Libre Universidad de Granada

UNIVERSIDAD DE GRANADA

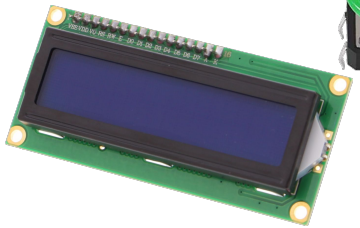
Materiales Necesarios

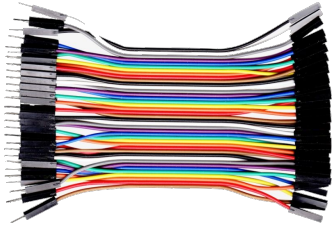
Materiales





- 1 Arduino uno
- 1 placa de pruebas ‘protoboard’ pequeña
- 1 tira de LEDs RGB
- 2 botones
- 1 servomotor 360º
- 1 buzzer
- 1 Resistencias de 220 Ohmios
- 1 pantalla LCD 1602 con I2C HD44780
- 10 cables macho-hembra y 24 cables macho-macho





Oficina de Software Libre Universidad de Granada

UNIVERSIDAD DE GRANADA

Fases del proyecto

Objetivo

El objetivo es llegar a un proyecto que funcione pronto y luego añadir funciones a medida que nos de tiempo.

¡Este método se usa mucho en ingeniería y es muy útil para todos los proyectos que hagáis!

Etapas

Vamos a dividir nuestro proyecto en 3 etapas que se construyen secuencialmente:

1. **Teatro base**: Tendremos nuestro telón, que sube y baja y unas luces intermedias.
2. **Teatro avanzado:** Además de la anterior, varias escenas de luces, controladas por botones, que nosotras ordenaremos para contar la historia.
3. **Teatro extra,** Le podemos añadir un buzzer para crear sonidos según cada escena, servomotores para efectos especiales o una pantalla LCD que nos indique la escena en la que estamos.

Fases

1. Servomotor del telón y botones
   - Aprendemos a usar servomotores de 360º
2. Luces y botones
   - Aprendemos a utilizar luces con botones.
3. Código del teatro
   - Estructura básica de nuestro proyecto que vamos a ir aumentando
4. Escenas extra
   - Cuando ya tengamos lo básico, añadimos escenas con nuestros LEDs
5. Ampliaciones
   - Elige las ampliaciones que quieras (un buzzer, un LCD, servos adicionales…)

Oficina de Software Libre Universidad de Granada

UNIVERSIDAD DE GRANADA

¡EMPEZAMOS!

Fase 1: Servomotor

- Aprenderemos a:
  - Usar el servo
  - Hacer que se mueva con botones

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, pero los nuestros van a girar 360º

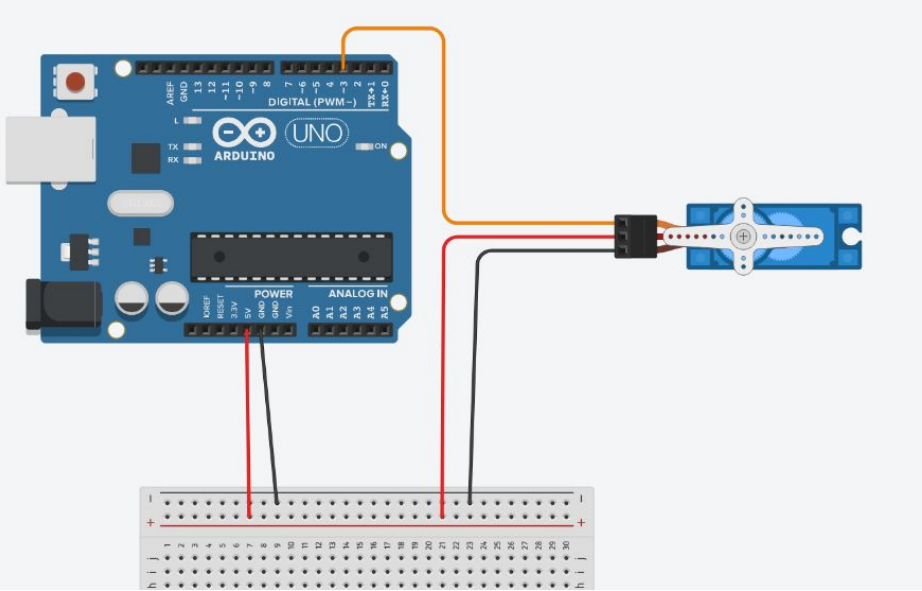
Con este motor abriremos nuestra hucha

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

SERVOMOTORES

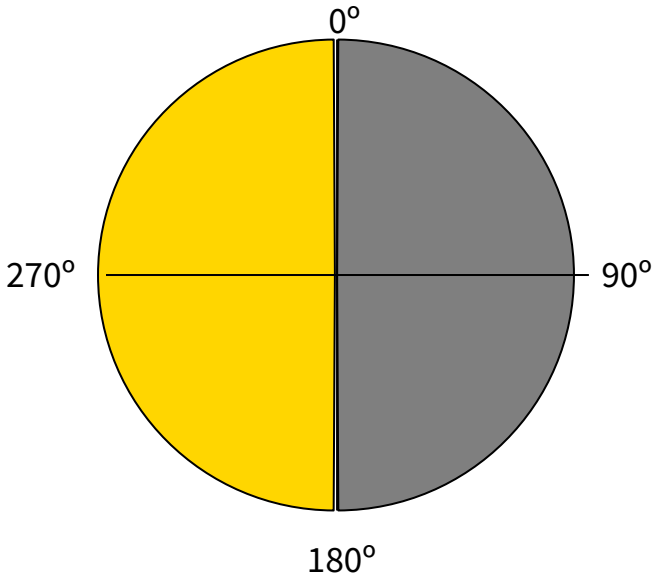
- **Servo motor 360º.**
  - Tienen 3 pines:
    - **DIN** pin de entrada **PWM,** tienen un **~**
    - **+5V** proporciona energía al servo. **VCC**
    - **GND** Toma de tierra.**GND**

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



ServoMotores

Los servomotores normalmente tienen un rango de movimiento limitado, que es de 0 a 180 grados, pero nuestros servomotores están modificados para que giren 360º



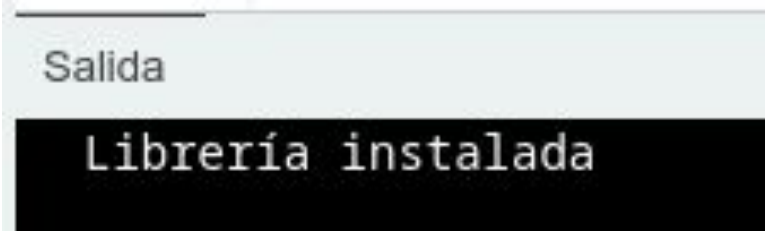
SERVOMOTORES

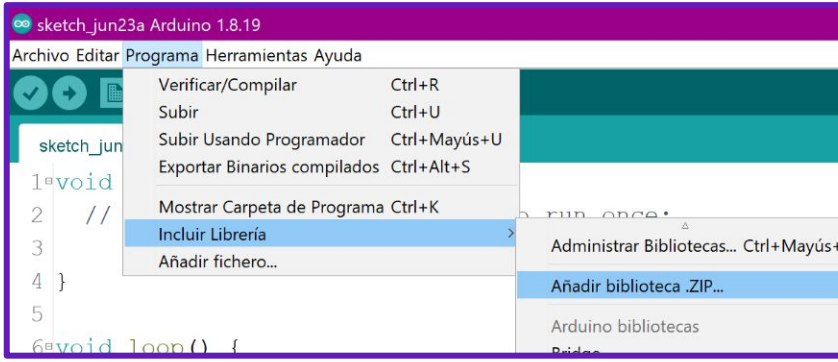
1. Descarga la biblioteca y después la importamos en el arduino:

Está en el enlace:

https://downloads.arduino.cc/libraries/github.com/arduino-libraries/Servo-1.2.2.zip

Sketch > Incluir biblioteca > Añadir biblioteca ZIP





Luego vamos a Sketch > Incluir biblioteca > Servo y nos aparece esto arriba del código

```
#include <Servo.h>
```

ServoMotores

- Las bibliotecas/librerías son código que otras personas crean para facilitar el uso de los componentes. Cuando llamamos a funciones de la biblioteca tenemos que tener cuidado en ver cómo se escriben y qué nos piden

```
Servo motor;      // crear un servo llamado motor
motor.attach(3); // decirle que trabaje en el pin 3
motor.write(90); // mover servo
```

Por ejemplo, aquí creamos un servo llamado motor.
Le decimos que trabajará con el pin 3
Y luego le decimos que se mueva a 90º

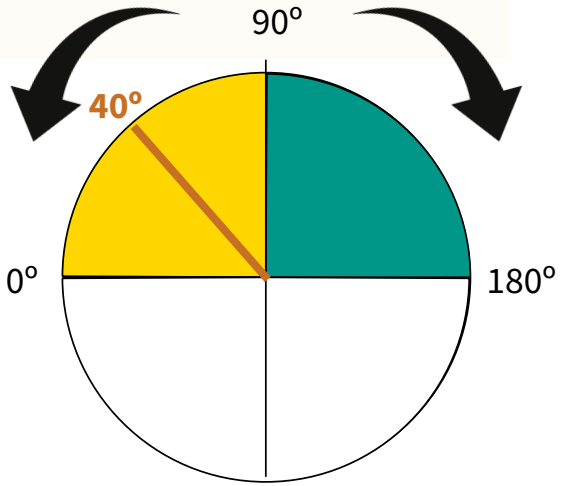
No os preocupéis si no lo entendéis del todo aún, ¡es un ejemplo de función de biblioteca!

ServoMotores

- Hay una función importante que debemos aprender antes de usar nuestro servomotor:

```
motor.write(dirección)
```

  - Cuando escribamos la dirección, el servo se moverá en ese sentido dando vueltas sin parar.
    - 180 es la velocidad máxima hacia la derecha
    - 0 es la velocidad máxima a la izquierda
    - 90 es un servomotor parado.
  - Si queremos que se mueva más lento, usamos grados más cercanos a 90 (por ejemplo, **40** iría girando más lentamente hacia la izquierda).



Servomotores

Este es un pequeño ejemplo de cómo funciona:

Los bucles for son para que gire cada vez más rápido/lento.

Dentro del bucle, motor.write(pos) es para que cambie el servo a esa velocidad.

```
for (int pos = 90; pos >=0; pos = pos - 1){
  motor.write(pos)
  delay(15);
}
```

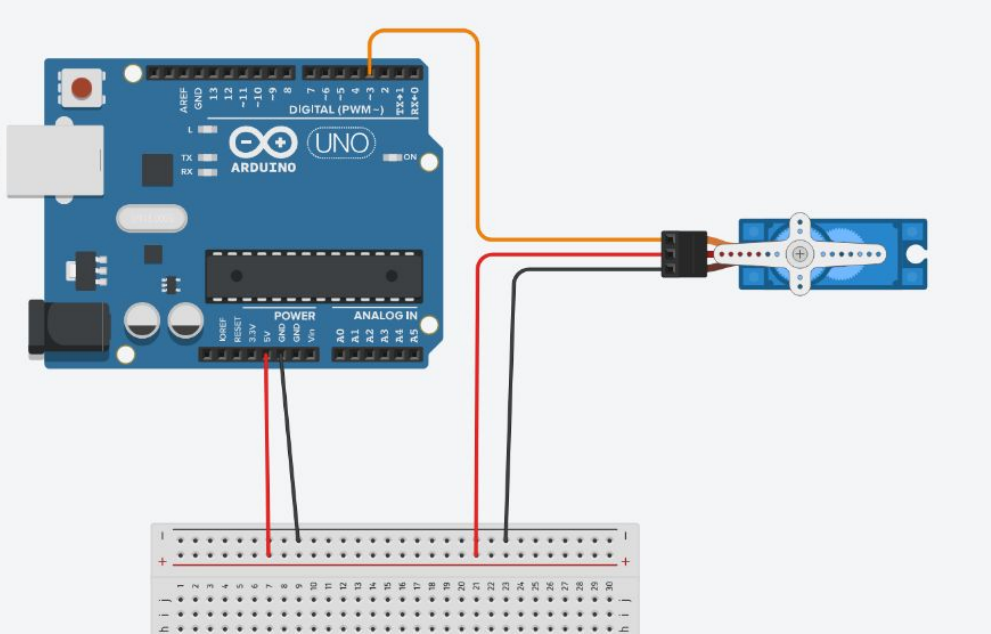
NOTA 1: Aquí empieza parado y va girando cada vez más rápido a la izquierda

```
for (int pos = 0; pos <=90; pos = pos + 1){
  motor.write(pos)
  delay(15);
}
```

NOTA 2: Aquí va reduciendo la velocidad hasta volver a 90 (parado).

Fase 1 - Servomotores

Reto 1 - Hacer que el servo vaya a un lado, pare, luego vaya al otro y pare




```
#include <Servo.h>

//Este es el pin en el que conectamos nuestro motor
#define PIN_SERVO 3

// Este es nuestro servo
Servo motor;
//Aquí vamos a definir las variables de las velocidades de giro
int derecha, izquierda, parado;

void setup() {
  // El servo va a usar el pin 3
  motor.
  //Definimos las velocidades


  //Empezamos con el servo parado
  motor.
}

void loop() {
  // Giramos hacia la derecha
  delay(1000);
  //paramos
  delay(1000);
  //giramos hacia la izquierda
  delay(1000);
  //paramos
  delay(1000);
}
```

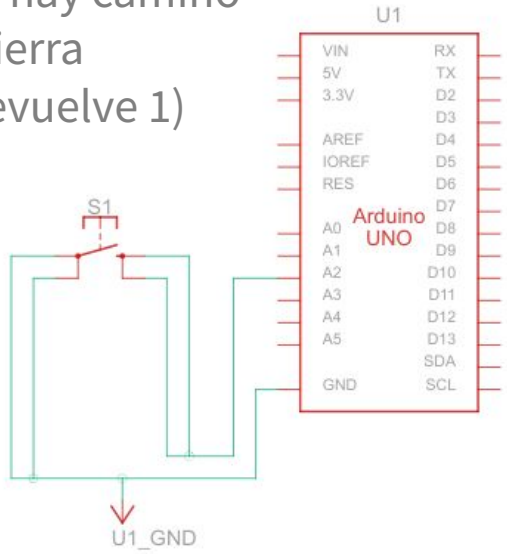
hay que rellenar en los cuadrados verdes

Si todo funciona, ¡¡Guardad el código en drive, haced fotos y vídeos, copia de seguridad!!

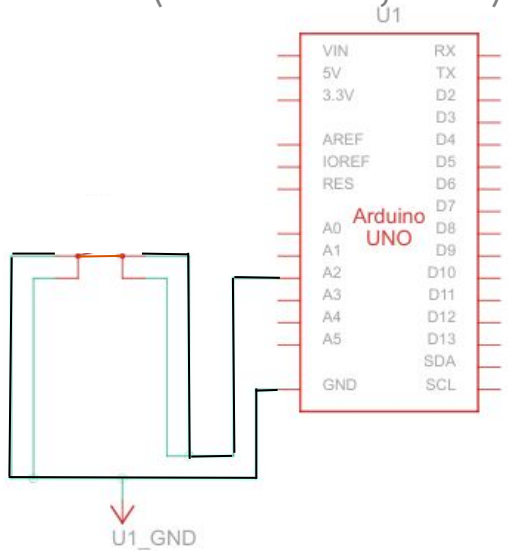
Fase 2 - Botones

Vamos a conectar botones a GND y a un pin, de forma que cuando los pulsemos, el 0 de GND llegará al pin, y cuando no, llegará un 1.

No hay camino a tierra (devuelve 1)



Hay un camino a tierra (devuelve 0, GND)



Botones

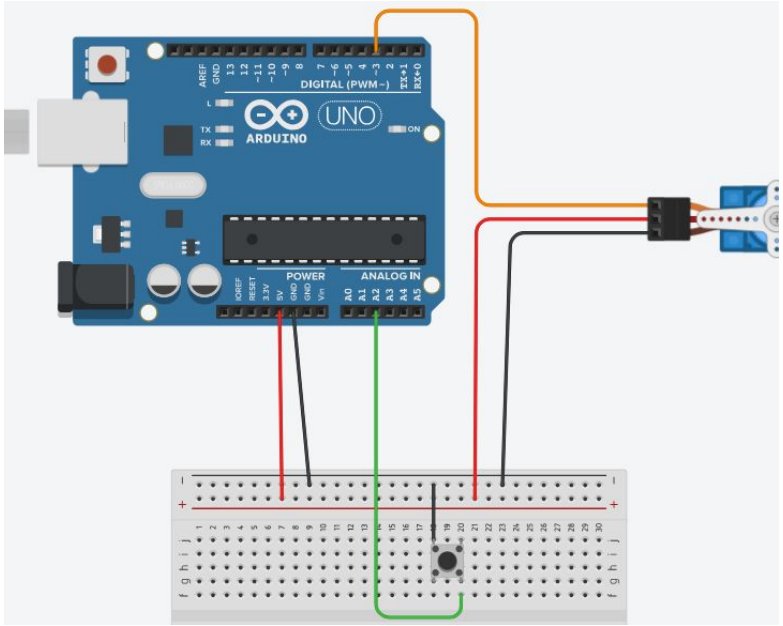
- Para leer de un pin (como, por ejemplo, el de nuestros botones) usamos esta función:

```
a = digitalRead(PIN);
```

  - Esta función nos devuelve un valor (que estamos metiendo en la variable a). Lee el pin (llamado PIN) y nos dice qué hay en él.

Fase 2 - Botones

Reto 2 - Hacer que el servo se mueva a la derecha y a la izquierda SOLO cuando se le de a un botón




```ino
#include <Servo.h>

//Este es el pin en el que conectamos nuestro motor
#define PIN_SERVO 3
//Este es el pin en el que conectamos el botón
[redacted]

//definimos estados de los botones
bool nuevoEstado, estadoA;

// Este es nuestro servo
Servo motor;
//Aquí vamos a definir las variables de las velocidades de giro
int derecha, izquierda, parado;

void setup() {
  //declaramos el pin del botón como INPUT (de entrada) y
  //PULLUP (usando una resistencia del arduino)
  pinMode(PIN_BOTON, INPUT_PULLUP);

  // El servo va a usar el pin 3
  [redacted]
  //Definimos las velocidades
  [redacted]
  //Empezamos con el servo parado
  [redacted]

  //nuevo estado empieza a 1
  [redacted]
  //estadoA empieza a 1
  [redacted]
}
```

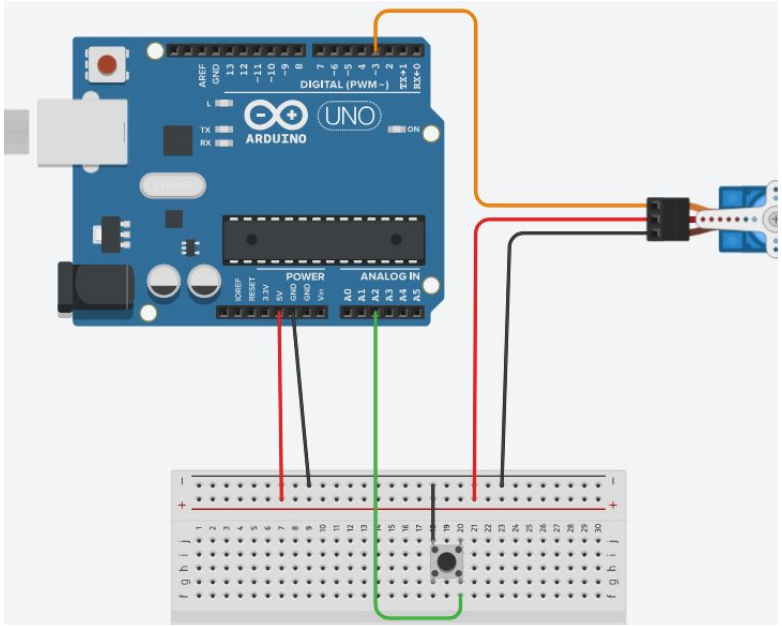
hay que rellenar en los cuadrados verdes

esto es lo del reto 1

El código sigue en la siguiente diapositiva ->

Fase 2 - Botones

Reto 2 - Hacer que el servo se abra y luego se cierre SOLO cuando se le de a un botón



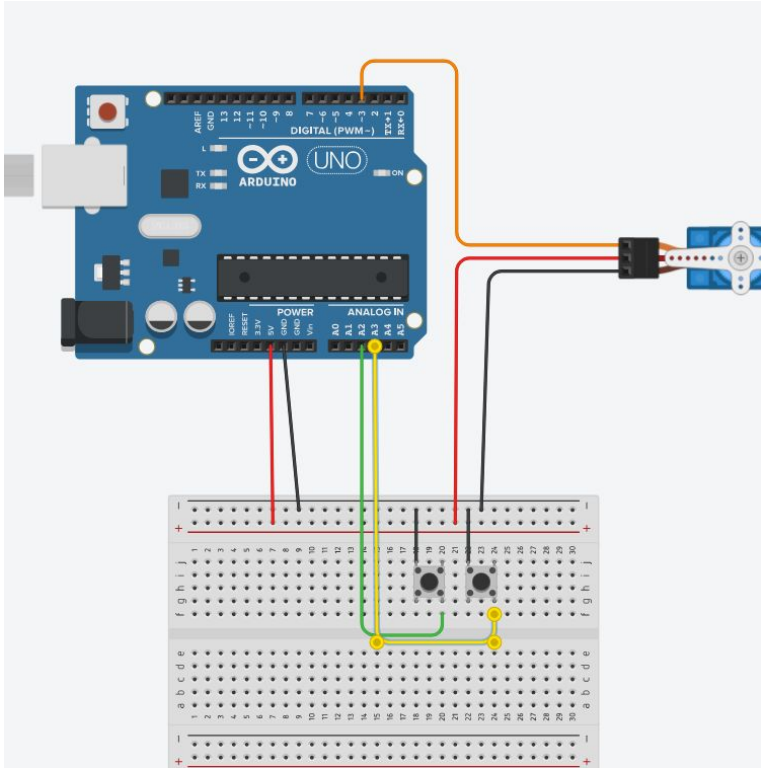

```
void loop(){
  nuevoEstado =  Leer pin botón
  if(  estadoA es diferente a nuevoEstado  ){
    //Ha cambiado de estado
    if(  ¿es nuevo estado igual a 0?  ){
      //el nuevo estado es una pulsación
      esto es el movimiento de motor del reto 1
    }
    estadoA=  asignar nuevoEstado a estadoA
}
```

Si todo funciona, ¡¡Guardad el código en drive, haced fotos y vídeos, copia de seguridad!!

Fase 2 - Botones

Reto 3 - Hacer que el servo se mueva a la derecha y pare con un botón y se mueva a la izquierda y pare con otro

Vamos a añadir un botón más, así que todo lo que hicimos para el botón anterior, lo necesitamos hacer aquí.



Fase 2 - Botones

En rojo está lo que ya hemos hecho en retos anteriores, en verde lo que es nuevo. Tenemos que:

- **Añadir el PIN del botón B**
- añadir su **estado**
- Poner el "**pinmode**" del botón B.
- **Iniciar el estado** del botón B a 1.

El código sigue en la siguiente diapositiva ->

```
#include <Servo.h>

//Este es el pin en el que conectamos nuestro motor
#define PIN_SERVO 3
//Este es el pin en el que conectamos el botón
[redacted]

//definimos estados de los botones
bool nuevoEstado, estadoA, [redacted]

// Este es nuestro servo
Servo motor;
//Aquí vamos a definir las variables de las velocidades de giro
int derecha, izquierda, parado;

void setup() {
  //declaramos el pin del botón como INPUT (de entrada) y
  //PULLUP (usando una resistencia del arduino)
  pinMode(PIN_BOTON, INPUT_PULLUP);
  [redacted]

  // El servo va a usar el pin 3
  [redacted]
  //Definimos las velocidades
  [redacted]
  //Empezamos con el servo parado
  [redacted]

  //nuevo estado empieza a 1
  [redacted]
  //estadoA empieza a 1
  [redacted]
  //estadoB empieza a 1
  [redacted]

}
```

hay que rellenar en los cuadrados verdes

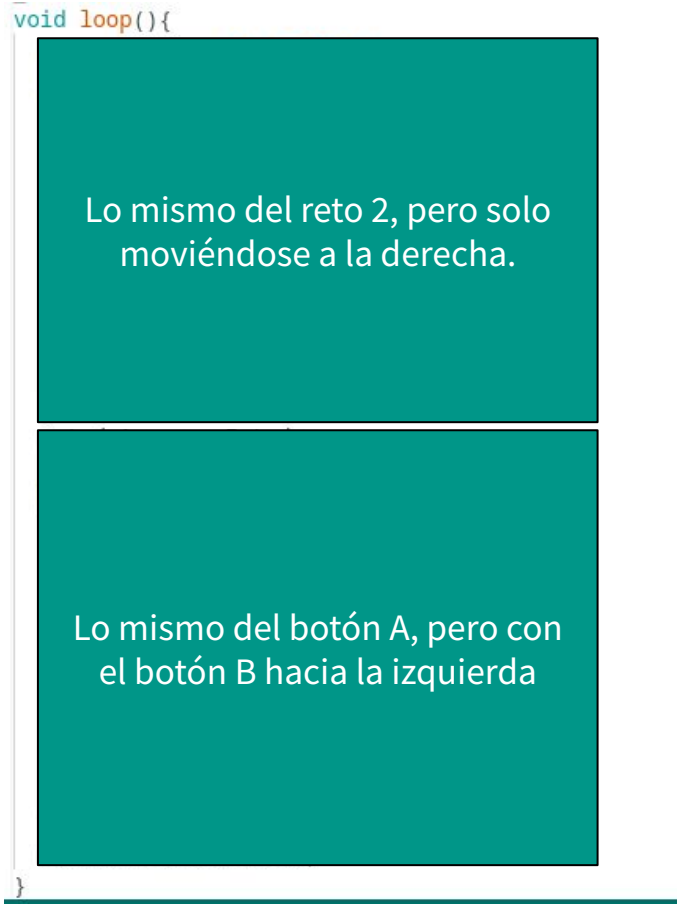
esto es lo del reto 1 y 2

Fase 2 - Botones

En rojo está lo que ya hemos hecho en retos anteriores, en verde lo que es nuevo. Tenemos que:

- Modificar el botón A para que **solo tenga el código de abrir**
- Crear el **mismo código para el botón B**, pero con el código de cerrar.

Si todo funciona, ¡¡Guardad el código en drive, haced fotos y vídeos, copia de seguridad!!



Servomotores y botones

¡Fase superada!

¡Ya sabemos utilizar nuestro servo y nuestros botones para el telón!

Si todo funciona, **guardad todo vuestro progreso y haced foto de los pines**, porque vamos a empezar a aprender otros componentes y desconectaremos los de ahora para usarlos luego.

Fase 2: luces y botones

- Aprenderemos a:
  - Los leds neopixel
  - Hacer que cambien de color con nuestros botones

Leds y botones

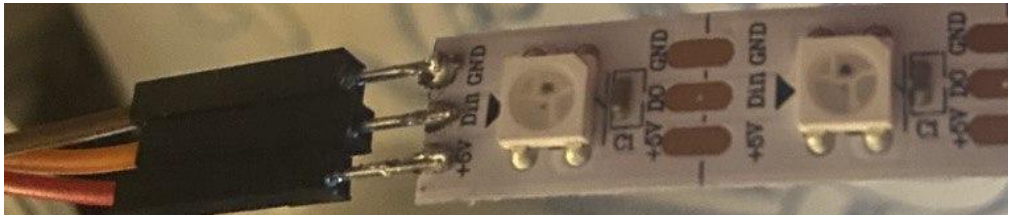
- **Leds RGB direccionables.**
  - Tienen 4 pines:
    - **DIN** pin de entrada
    - **DO** pin de salida
    - **+5V** proporciona energía a los leds
    - **GND** Toma de tierra

Antes de cortar los leds fijaos que los **DIN** están conectados a los **DO.**

Recordadlo cuando vayáis a incluirlos en el proyecto final al conectarlos.

Leds y botones

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



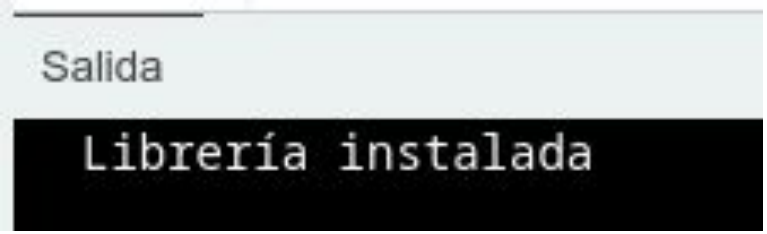
LEDS Y BOTONES

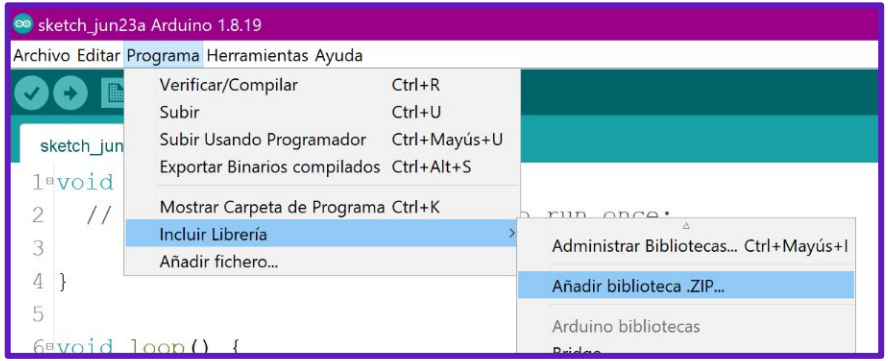
1. Descarga la biblioteca y después la importamos en el arduino:

Está en el enlace:

https://downloads.arduino.cc/libraries/github.com/adafruit/Adafruit_NeoPixel-1.15.2.zip

Sketch > Incluir biblioteca > Añadir biblioteca ZIP





Luego vamos a Sketch > Incluir biblioteca > Adafruit Neopixel y nos aparece esto arriba del código

```
1 #include <Adafruit_NeoPixel.h>
```

LEDS Y BOTONES

- Igual que en los servos, los leds necesitan que usemos funciones de librería:

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

Esta función tiene los siguientes parámetros:

- **NUMPIXELS**: el número de leds que queremos controlar
- **PIN**: el pin de arduino donde hemos conectado el DIN de la tira de led
- **NEO_GRB + NEO_KHZ800**: este parámetro indica que usaremos leds de colores y su controlador

Con esta función declararemos nuestros leds para utilizarlos más adelante.

Leds y botones

- Igual que en los servos, los leds necesitan que usemos funciones de librería:

```
pixels.begin();     // iniciar los leds en setup
pixels.show(); // mostrar los colores que le hemos dicho con setPixelColor
pixels.setPixelColor(i, pixels.Color(rojo, verde, azul)); // decir el color de
                                                          // los leds
```

Esta función tiene los siguientes parámetros:

- **i**: el número del led que cambiamos (de 0 a NUM-1)
- **pixels.Color(int, int, int)**:es la función que le dice a nuestros leds el color rgb que queremos, compuesto por rojo, verde y azul.
- **rojo, verde, azul**: son tres int de 0 a 255 siguiendo los colores RGB

Con esta funciones controlaremos nuestros leds.

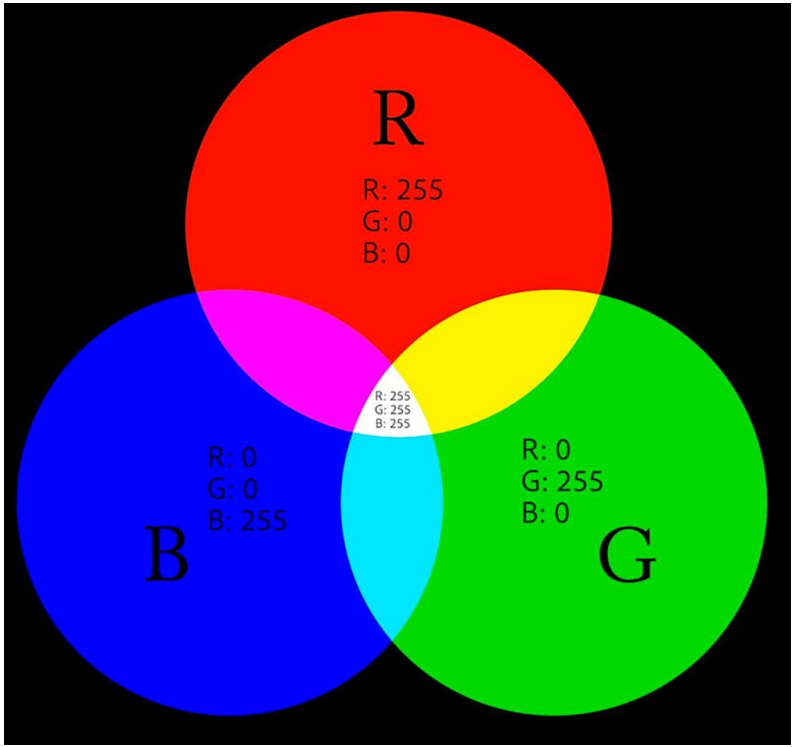
Colores RGB

Los códigos RGB son los que usan nuestros leds. Tienen tres componentes (**rojo**, **verde** y **azul**) que van de 0 a 255.

Cuando todos están a 0, es **negro** porque no hay color, cuando están todos a 255 está **blanco** porque es el máximo color.

En internet existen “RGB color pickers” donde podéis elegir un color y os dicen las componentes R(ed) G(reen) B(lue) para usarlas en nuestos leds.

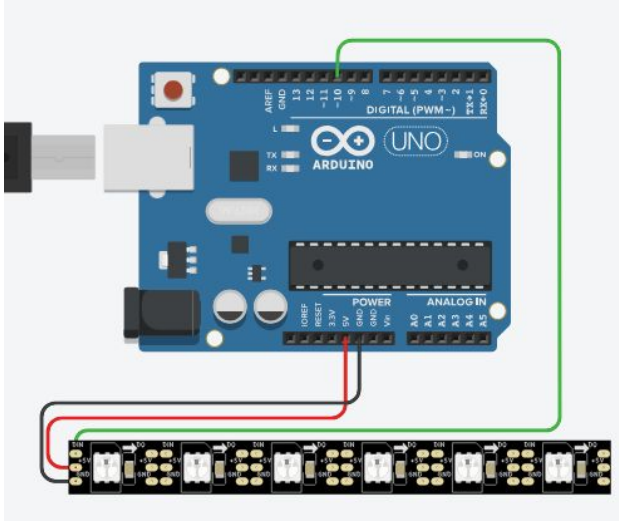
https://www.w3schools.com/colors/colors_rgb.asp



Leds y botones

Fase 3 - Leds

Reto 4 - Hacer que todos los leds de nuestra tira se iluminen del color que queremos. En este código solo uno se ilumina, hay que arreglarlo.




```
// C++ code
//
#include <Adafruit_NeoPixel.h>

#define PINLEDS 10    // pin al que conectamos la tira de neopixels
#define NUMPIXELS 1 //  numero de neopixels

int rojo, verde, azul;
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, PINLEDS, NEO_GRB + NEO_KHZ800);

void setup()
{
  iniciar los leds
}

void loop()
{

  rojo =
  verde =
  azul =

  for (int i=0; i < NUMPIXELS; i++) {
    // mandamos los colores a los pixeles
    pixels.setPixelColor(i, pixels.Color(rojo, verde, azul));
    // Esperamos un poquito en cada led para que no parpadee demasiado rapido
    delay(100);
  }
  pixels.show();
}
```

hay que rellenar en los cuadrados verdes

Si todo funciona, ¡¡Guardad el código en drive, haced fotos y vídeos, copia de seguridad!!

Leds y botones

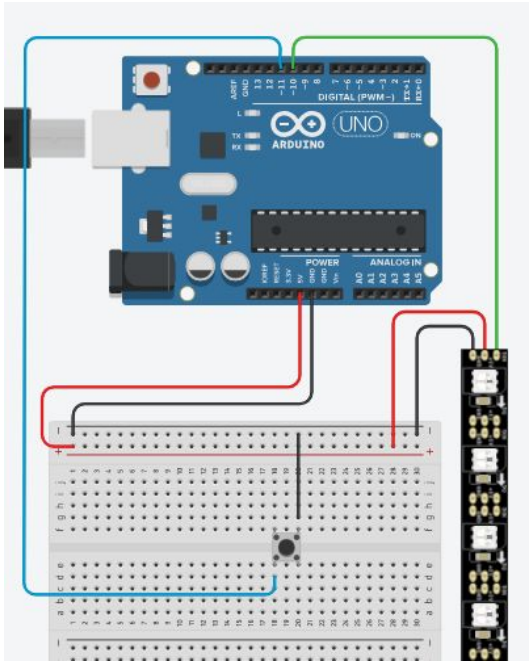
- Para utilizar colores aleatorios, podemos asignar a cada componente un número “random”

```
rojo = random(0, 255)
```

Esta función metería en “rojo” un número aleatorio de 0 a 255

Fase 3 - Leds

Reto 5 - Hacer que los leds cambien de color cuando le demos a un botón.



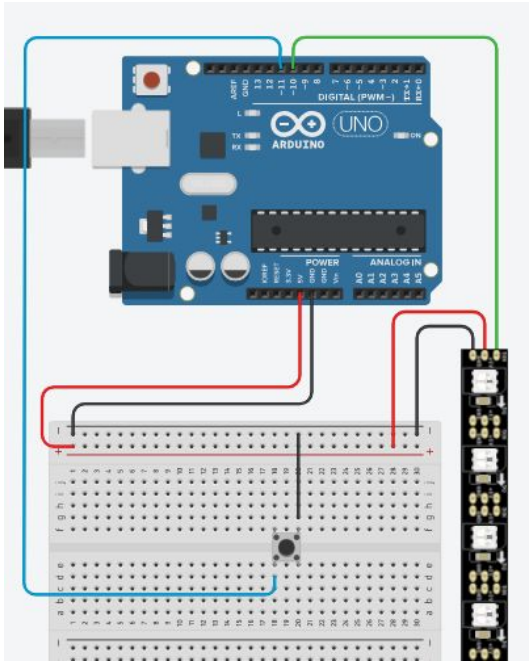

```
//
#include <Adafruit_NeoPixel.h>

// Pin al que conectamos el boton

// definir estados de los botones

void setup()
{
  iniciar los leds

  //iniciamos el boton a input pullup

  //nuevoEstado es 1

  //estadoA es 1

}
```

añadimos botones como en el reto 2

El código sigue en la siguiente diapositiva ->

Fase 3 - Leds

Reto 5 - Hacer que los leds cambien de color cuando le demos a un botón.



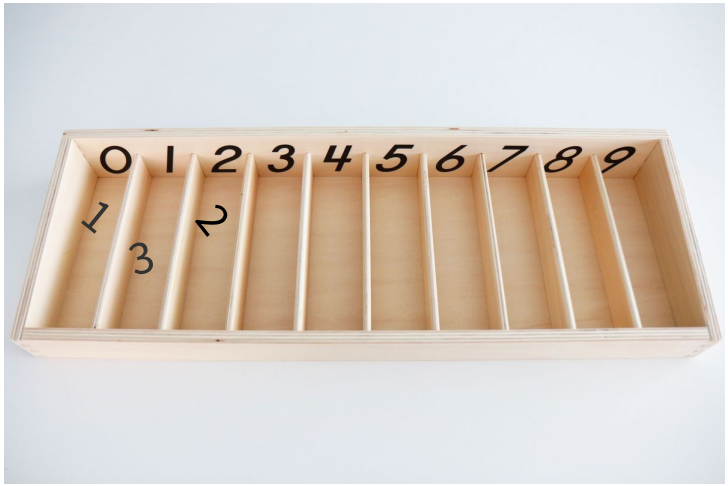

```
void loop()
{
  //
}
```

Igual que el reto 2, pero cuando pulsamos el botón se ponen colores (como el reto 4) aleatorios en los pixel

Si todo funciona, ¡¡Guardad el código en drive, haced fotos y vídeos, copia de seguridad!!

LEDS y BOTONES

- Antes de poder seguir haciendo nuestros ejercicios, vamos a ver algunos conceptos básicos de programación que van a ser muy útiles en el proyecto entero.
- Un **vector** en programación es el equivalente a una serie de cajitas numeradas donde guardamos datos:

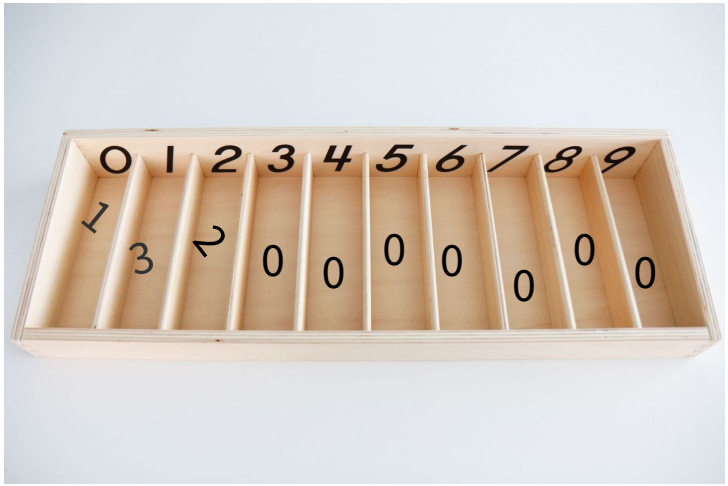


Si nosotros tenemos un tamaño **n** (por ejemplo, n=10), empezamos a enumerarlos por el **0 hasta n-1** (en esta imagen, n-1 es 9).
Si queremos acceder al primer hueco, llamaríamos a la posición 0 (en nuestra imagen, hay un valor 1 ahí).

LEDS Y BOTONES

Para declarar un **vector:**

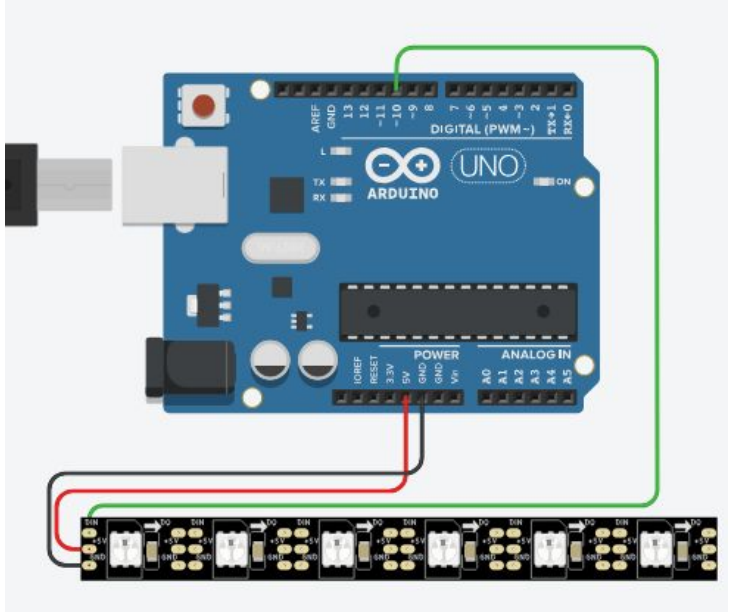
```
tipoDeDato nombre[n] = {valor,...};
```



Este por ejemplo sería

```
int vector[10] = {1,3,2,0,0,0,0,0,0,0};
```

LEDS Y BOTONES

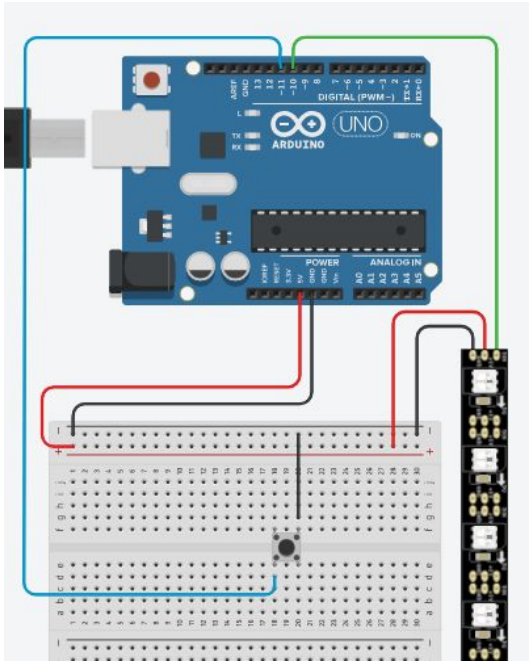


Nuestros leds son como vectores, y cuando usamos `pixels.setPixelColor(i, pixels.Color(rojo, verde, azul));` estamos dándole color al led i.

Sabiendo esto, podemos darle los colores que queramos a cada uno de los leds para conseguir efectos diferentes.

Fase 3 - Leds

Reto 6 - Poner cada led de un color cuando se presiona el botón



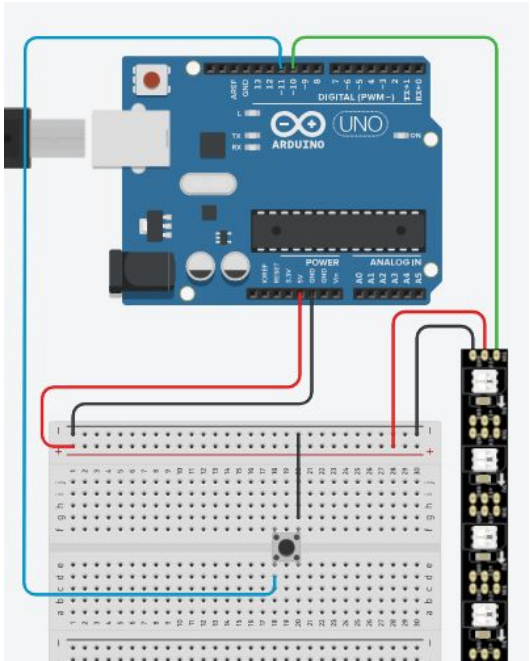

```
//
#include <Adafruit_NeoPixel.h>
[redacted]

// Pin al que conectamos el boton
[redacted]

// definir estados de los botones
[redacted]

[redacted]

void setup()
{
  iniciar los leds
  //iniciamos el boton a input pullup
  [redacted]
  //nuevoEstado es 1
  [redacted]
  //estadoA es 1
  [redacted]
}
```

añadimos botones como en el reto 2

El código sigue en la siguiente diapositiva ->

Fase 3 - Leds

Reto 6 - Poner cada led de un color cuando se presiona el botón




```
void loop()
{
  //
}
```

Igual que el reto 4, pero cambiamos el color por cada pixel.

Si todo funciona, ¡¡Guardad el código en drive, haced fotos y vídeos, copia de seguridad!!

Leds y botones

¡Fase superada!

¡Ya sabemos utilizar nuestros pixels y nuestros botones para las escenas!

Si todo funciona, **guardad todo vuestro progreso y haced foto de los pines**, porque vamos a empezar a desarrollar nuestro teatro básico y lo necesitaremos todo.

Fase 3: Teatro base

- Aprenderemos a:
  - Crear la estructura sobre la que nuestro teatro funcionará.
  - Unir nuestros conocimientos de los componentes al código.
  - ¡Tener un teatro funcionando!

Teatro base

¿En qué va a consistir nuestro teatro base?

Parte 1: Esperamos a que se presione el botón del telón. Cuando se pulse, subirá.

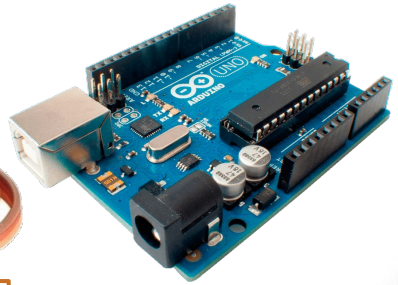
Parte 2: Una vez subido el telón, se encenderán las luces.

Parte 3: Cuando volvamos a presionar el botón del telón, bajará y se apagarán las luces.

Vamos a ir poco a poco añadiendo partes hasta tenerlas todas.

Fase 3 - Código

Preliminares: Vamos a juntar todo lo que hemos aprendido hasta ahora en un solo proyecto. Es decir, vamos a conectar todos los componentes como los teníamos en los retos y prepararlos para usarlos



Revisad las fotos que habéis ido guardando para conectarlos, si tenéis problemas, ¡podéis revisar las diapositivas o preguntar!

Fase 3 - Código

Preliminares: Necesitamos preparar nuestros componentes igual que lo hicimos anteriormente, así que vamos a ir añadiendo lo que necesitan para funcionar en nuestro código.

Revisad el código que habéis ido guardando para prepararlos, si tenéis problemas, ¡podéis revisar las diapositivas o preguntar!

```
// C++ code
//
// Bibliotecas de neopixel y servo

//pin de los leds
// número de leds en nuestra tira
// pin botón
// pin servo
```

estados botones

int colores

```
// crea el objeto servo
// velocidad del servo
```

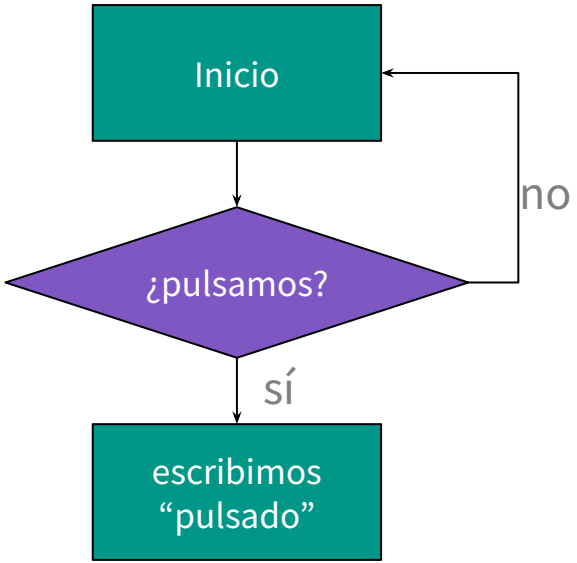
usamos la biblioteca de neopixel para declarar los pixels

```
void setup()
{
  // para poder escribir en pantalla
  Serial.begin(9600);
  //configurar pin botón
  //iniciar los leds
  // vincula el servo al pin del motor
  // iniciamos los bools de los botones
}
```

Fase 3 - Código

- **Parte 1**: Esperamos a que se presione el botón del telón. Cuando se pulse, subirá.

Vamos a empezar por la primera parte. Esperaremos esperando una pulsación para subir el telón y cuando lo hagamos escribiremos por pantalla que lo hemos hecho.



Fase 3 - Código

Vamos a ver cómo subir el telón

Paso 1: Declarar variable booleana “empezamos” junto al resto de variables al inicio. Dentro de setup, vamos a darle el valor “false”

Paso 2: En loop(), vamos a implementar el siguiente pseudocódigo:

```
si (no empezamos){
        leemos el botón
        si (ha cambiado de estado){
                si(es una pulsación){
                        movemos el servo
                        empezamos = true
                }
        }
        estado es nuevoestado
}
sino{
        escribimos “subido”
}
```

Fase 3 - Código

¿Qué nos debería salir?

Si le damos al botón: Subirá el telón y saldrá “subido” en pantalla

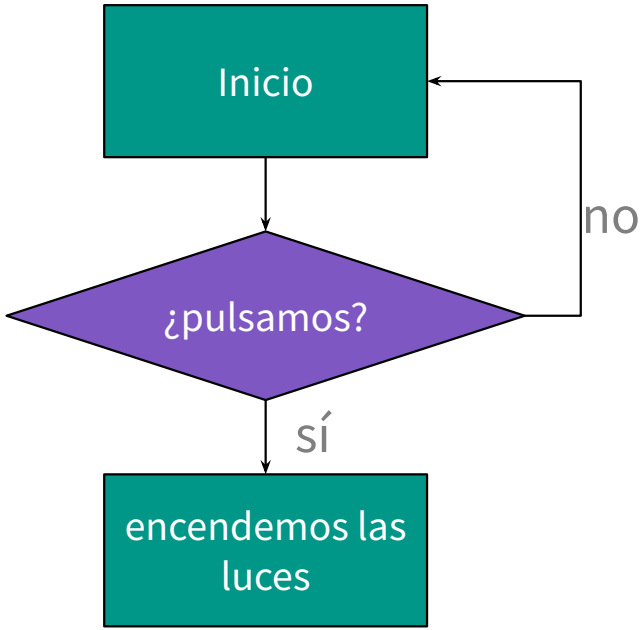
Si no le damos al botón: Nada se mueve y nada sale en pantalla

```
subido
subido
subido
subido
subido
subi
```

Si todo funciona, ¡¡Guardad el código en drive, haced fotos y vídeos, copia de seguridad!!

Fase 3 - Código

- **Parte 2:** Una vez abierto el telón, encendemos las luces



Fase 3 - Código

Vamos a ver cómo encender los leds tras subir el telón

En loop(), vamos a implementar el siguiente pseudocódigo:

```
si (no empezamos){
        leemos el botón
        si (ha cambiado de estado){
                si(es una pulsación){
                        movemos el servo
                        empezamos = true
                }
        }
        estado es nuevoestado
}
sino{
        encendemos las luces como en el reto 4
}
```

igual que antes

nuevo

Fase 3 - Código

¿Qué nos debería salir?

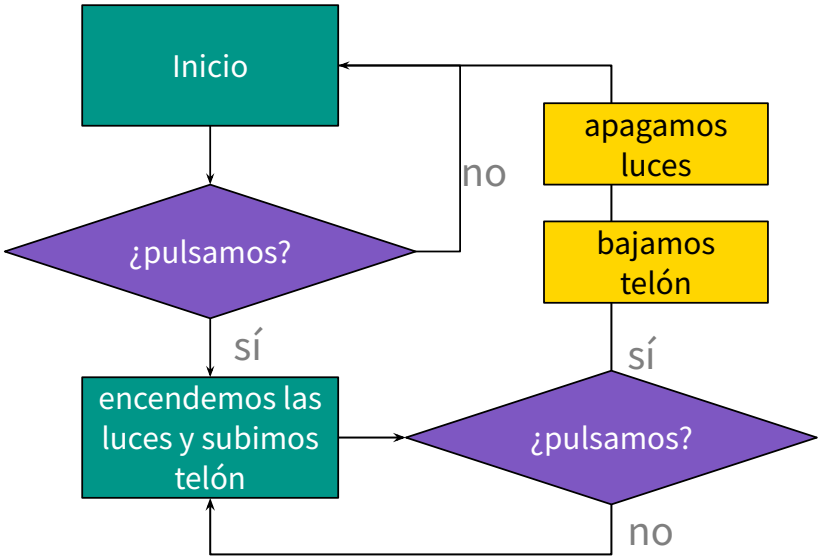
Si le damos al botón: Subirá el telón y se encenderán las luces

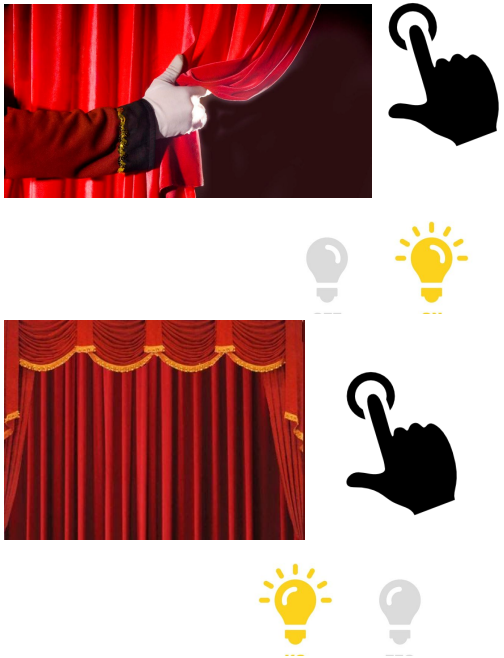
Si no le damos al botón: Nada se mueve y nada sale en pantalla

Si todo funciona, ¡¡Guardad el código en drive, haced fotos y vídeos, copia de seguridad!!

Fase 3 - Código

- **Parte 3:** Cuando volvamos a presionar el botón del telón, bajará y se apagarán las luces





Fase 3 - Código

Vamos a ver cómo bajar el telón, apagar los leds y volver al inicio.

En loop(), vamos a implementar el siguiente pseudocódigo:

```
si (no empezamos){
        leemos el botón
        si (ha cambiado de estado){
                si(es una pulsación){
                        movemos el servo
                        empezamos = true
                }
        }
        estado es nuevoestado
}
sino{
        encendemos las luces como en el reto 4
        leemos el botón
        si (ha cambiado de estado){
                si(es una pulsación){
                        movemos el servo
                        empezamos = false
                        apagamos las luces
                }
        }
        estado es nuevoestado

}
```

igual que antes

nuevo

Fase 3 - Código

¿Qué nos debería salir?

Si le damos al botón: SUbe el telón y se encienden las luces

Si damos al botón otra vez: Baja el telón y se apagan las luces

¡¡Debería ser infinito!! Probad a presionar más de dos veces (subir, bajar, subir, bajar...)

Si todo funciona, ¡¡Guardad el código en drive, haced fotos y vídeos, copia de seguridad!!

Teatro base

¡FASE SUPERADA!

Nuestro teatro está más que funcionando,
¡ya es un proyecto completo!

A partir de aquí, podéis empezar a **Decorar**

y hay más diapositivas para hacer al proyecto aún más chulo y aprender a usar los componentes que quedan en la bolsa.

¡Haced vídeos! ¡¡Guardadlo todo!!

¿División de tareas?

¡¡Este teatro es totalmente funcional!!

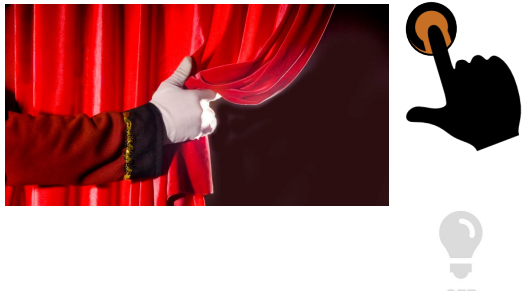
Una parte del equipo puede seguir programando para mejorar el código y otra parte diseñar el montaje de la estructura

Fase 4: Teatro avanzado

- Aprenderemos a:
  - Añadir escenas de luces secuenciales
  - Pasar esas escenas con un botón adicional.

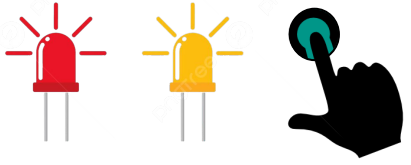
Teatro avanzado

¿En qué va a consistir nuestro teatro avanzado?



Parte 1: Esperamos a que se presione el botón del telón. Cuando se pulse, subirá.

Parte 2: Una vez subido el telón, cuando pulsemos el segundo botón, se cambian las luces con las escenas programadas

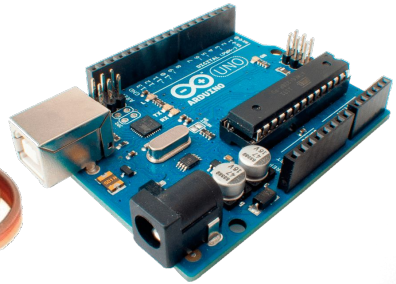


Parte 3: Cuando volvamos a presionar el botón del telón, bajará y se apagarán las luces.

En esta fase modificaremos la **parte 2** para añadir un botón adicional

Fase 4

Preliminares: Necesitaremos conectar un botón adicional a nuestro proyecto para controlar las escenas.



Revisad las fotos que habéis ido guardando para conectarlos, si tenéis problemas, ¡podéis revisar las diapositivas o preguntar!

TEATRO AVANZADO

Fase 4

Preliminares: Vamos a añadir nuestro botón igual que el otro. Tenemos que:

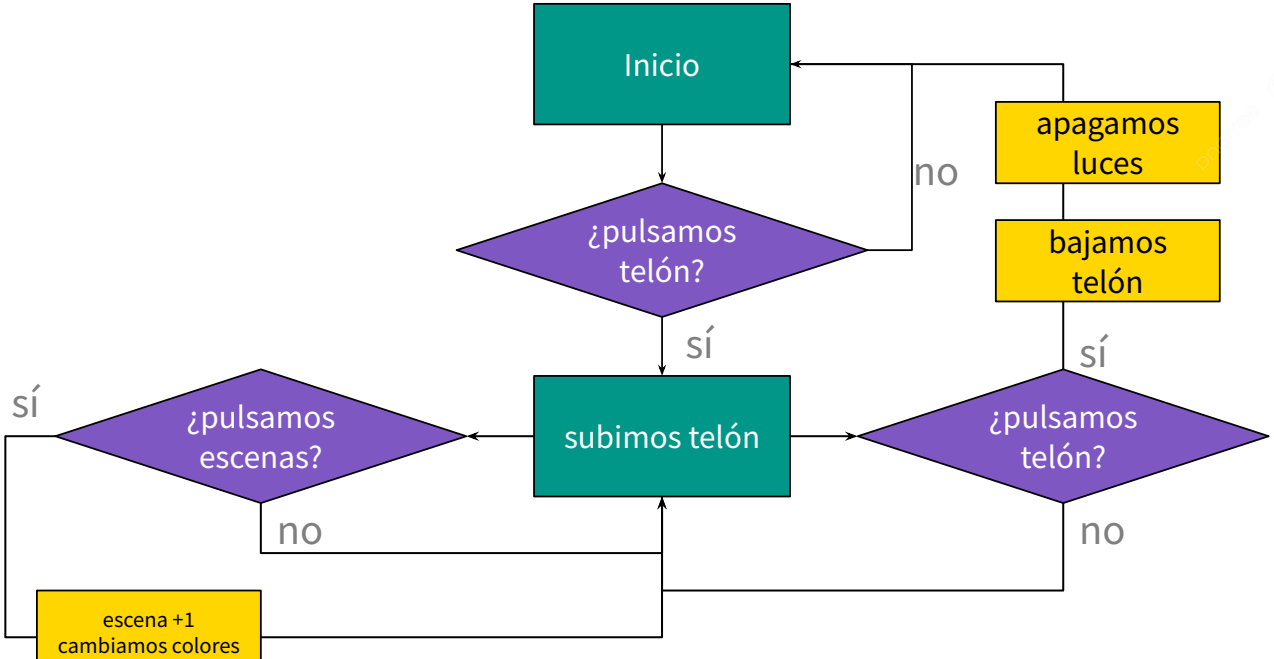
- Poner el pin del nuevo botón
- Añadir un estado para el nuevo botón
- En setup configurar el pin del botón
- Iniciar el bool del estado nuevo
- Añadir un nuevo entero que simboliza nuestra **escena**

Revisad el código que habéis ido guardando para prepararlos, si tenéis problemas, ¡podéis revisar las diapositivas o preguntar!

```
// C++ code
//
bibliotecas pixel y servo

pines de teatro base

pines de teatro base
pin del botón nuevo

estados botones + nuevo

int escena

int colores

; // crea el objeto servo
  // velocidad del servo

usamos la biblioteca de neopixel para declarar los pixels ;

void setup()
{
  Serial.begin(9600);
  iniciamos pin boton nuevo ;

  iniciamos el nuevo estado de
  botón y el int de escena a 1
}
```

Fase 4

- **Parte 2:** Una vez subido el telón, cuando pulsemos el segundo botón, se cambian las luces con las escenas programadas



Fase 4

Vamos a ver cómo bajar el telón, apagar los leds y volver al inicio.

En loop(), vamos a implementar el siguiente pseudocódigo:

```
si (no empezamos){
        igual que antes la subida de cortina
}
sino{
        revisamos el botón de bajar la cortina como antes
                cuando subamos la cortina  ponemos
                también escena = 1
        leemos el botón de luces
        si (ha cambiado de estado){
                si(es una pulsación){
                        si (escena es 1){
                        colores de escena 1
                        escena +1
                        }
                        sino{
                                si (escena es 2){
                                colores escena 2
                                escena +1
                                }
                                sino{...}
                        }
                        encendemos luces
                }
        }
        estado es nuevoestado
}
```

igual que antes

nuevo

Fase 3 - Código

¿Qué nos debería salir?

Si le damos al botón de luces después de subir la cortina: Va cambiando las luces por cada presión según lo programado

Si seguimos dándole luego de todas nuestras luces programadas: Se quedará con la última luz

¡¡Debería ser infinito!! Probad a presionar más de dos veces (subir, cambiar luces, bajar, subir, cambiar luces, bajar…)

Si todo funciona, ¡¡Guardad el código en drive, haced fotos y vídeos, copia de seguridad!!

Teatro Avanzado

¡Fase superada!

Nuestro teatro tiene escenas de luces, ya podemos cambiar la ambientación en cada momento.

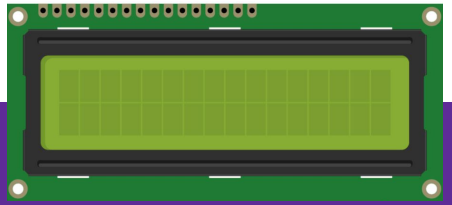
Los demás componentes de la caja se explican en las siguientes fases.

¡Haced vídeos! ¡¡Guardadlo todo!!

Fase 5: Teatro extra

- Aprenderemos a:
  - Usar y añadir un buzzer
  - Usar y añadir un LCD
  - Algunas ideas para añadir a nuestro teatro

Pantalla LCD



La pantalla LCD con I2C nos dejará mandar mensajes a nuestro público

Pantalla LCD

- **¿Qué es y cómo funciona una pantalla LCD?**

  Una **pantalla LCD** es una pantalla de retroiluminación LED que permite mostrar dos filas de 16 caracteres con los que podemos escribir texto.

- Realmente no vamos a aprender a conectar la pantalla LCD directamente, ya que tiene muchos pines y si la conectamos nos quedamos sin pines en el Arduino para todo lo demás. Por eso vamos a utilizar un **controlador I2C**.

En esta pantalla mostraremos mensajes de nuestra hucha.

Pantalla LCD

- ¿**Que es I2C y para qué sirve**?

  El controlador I2C nos permite controlar la pantalla utilizando solamente dos cables de control (además del cable de 5v y la toma a tierra GND) a través de un tipo de comunicación entre placas llamada I2C.

- Los dos cables utilizados son SCL, que se conecta al pin analógico A5, y SDA, que se conecta al pin analógico A4:

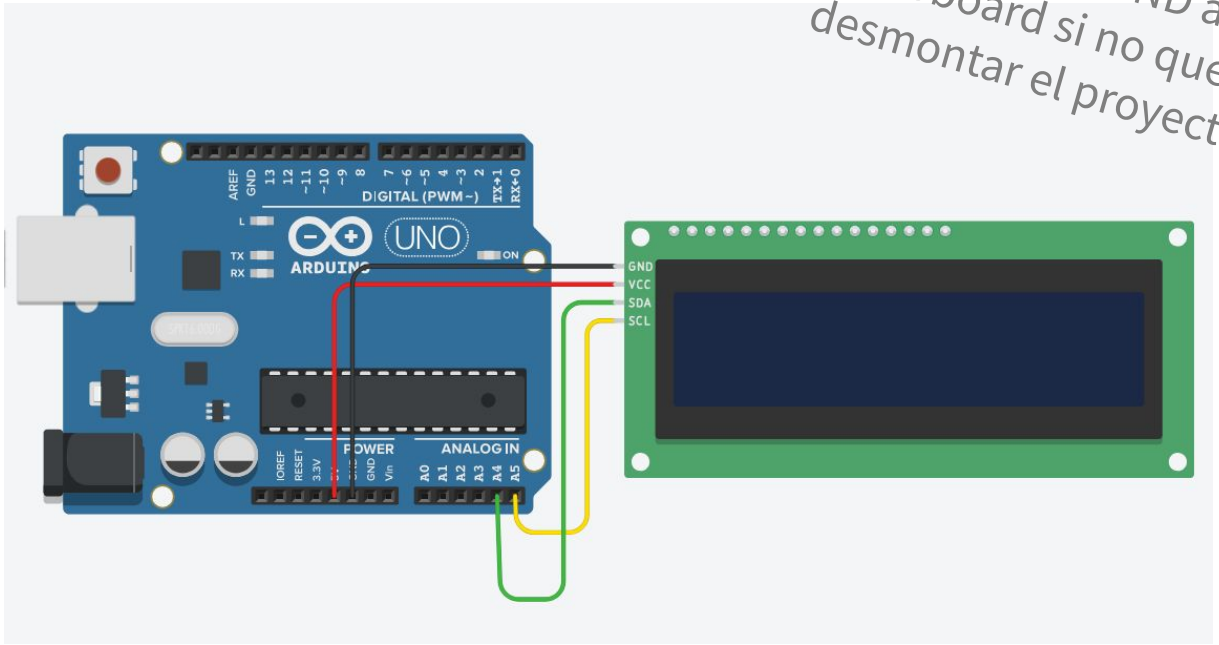
- SCL envía señales de reloj (clock), que se encargan de decidir quién habla en cada momento, para evitar conflictos y SDA envía los datos.

Pantalla LCD

SCL-> A5 GND-> GND

SDA-> A4 VCC-> 5V

Recordad que podéis conectar 5V y GND a vuestra protoboard si no queréis desmontar el proyecto!!

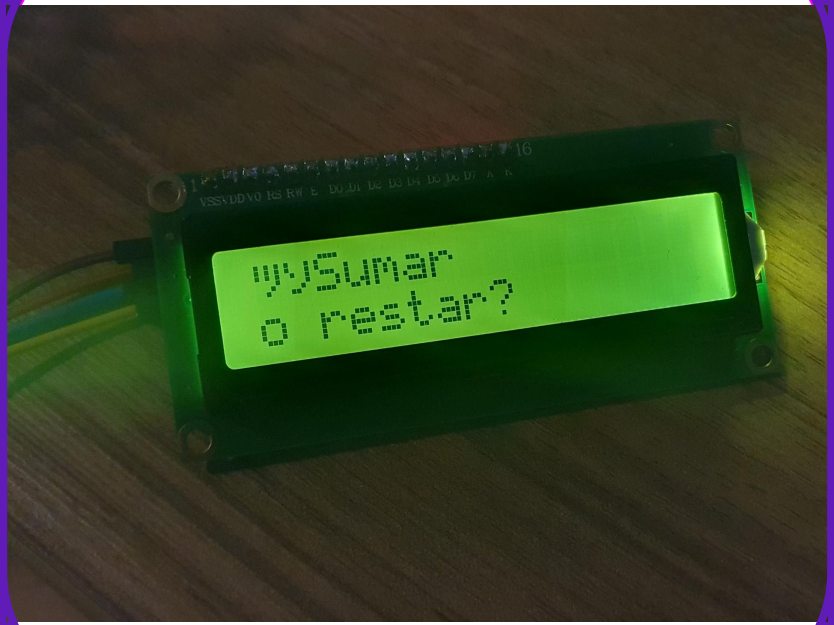


Pantalla LCD

En esta pantalla **mostraremos todo lo que antes mostrábamos en Println**

Recuerda los pines son:

SCL-> A5
GND-> GND
SDA-> A4
VCC-> 5V



Pantalla LCD

1. Incluir en el IDE la biblioteca LiquidCrystal_I2C.h

- Hay **dos formas de incluir bibliotecas** 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 **LiquidCrystal_I2C-1.1.2.zip**
- La **descargamos** y la incluimos de la 1º forma que vemos a continuación.

Pantalla LCD

1. Descarga la biblioteca y después la importamos en el arduino:

Está en el enlace:
http://downloads.arduino.cc/libraries/github.com/marcoschwartz/LiquidCrystal_I2C-1.1.2.zip

Pantalla LCD

- Para poder utilizar la pantalla con I2C además de descargarnos la librería de Arduino llamada **<LiquidCrystal_I2C.h>** , también necesitamos cargar otra llamada **<Wire.h>**.

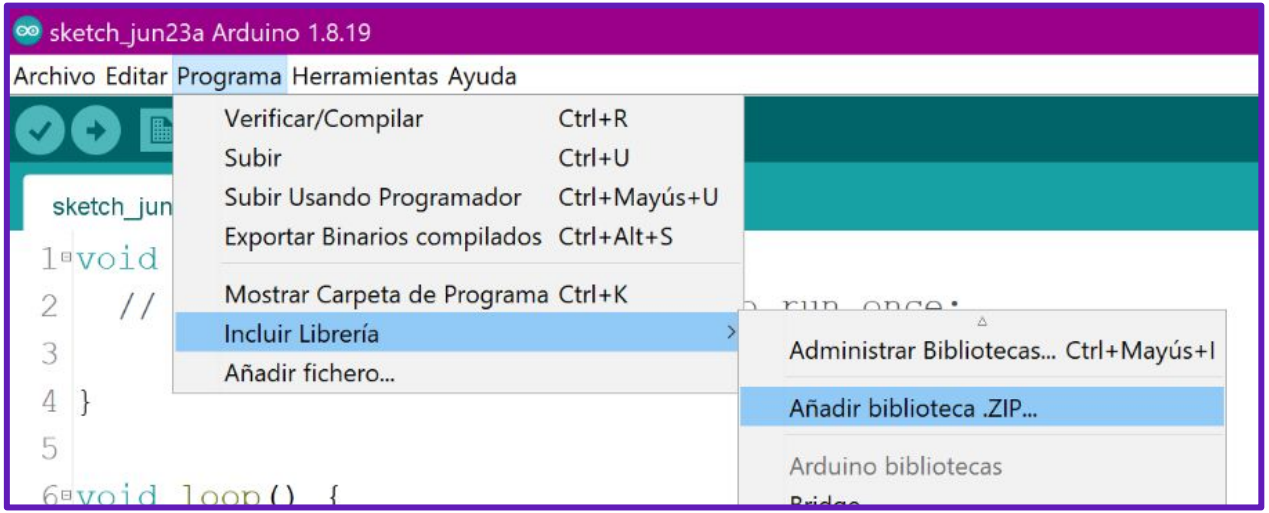
- Al trabajar con funciones de la librería descargada, a las que no les hemos puesto el nombre nosotras, tenemos que acostumbrarnos y entender lo que hacen.

NOTA: Las librerías son como funciones creadas por otras personas y que nosotras podemos aprovechar y utilizar siempre que lo necesitemos.

Pantalla LCD

1. Cargar librerías en el arduino:

En arduino hacemos clic en **Programa**-> **incluir libreria**-> **añadir biblioteca Zip.**

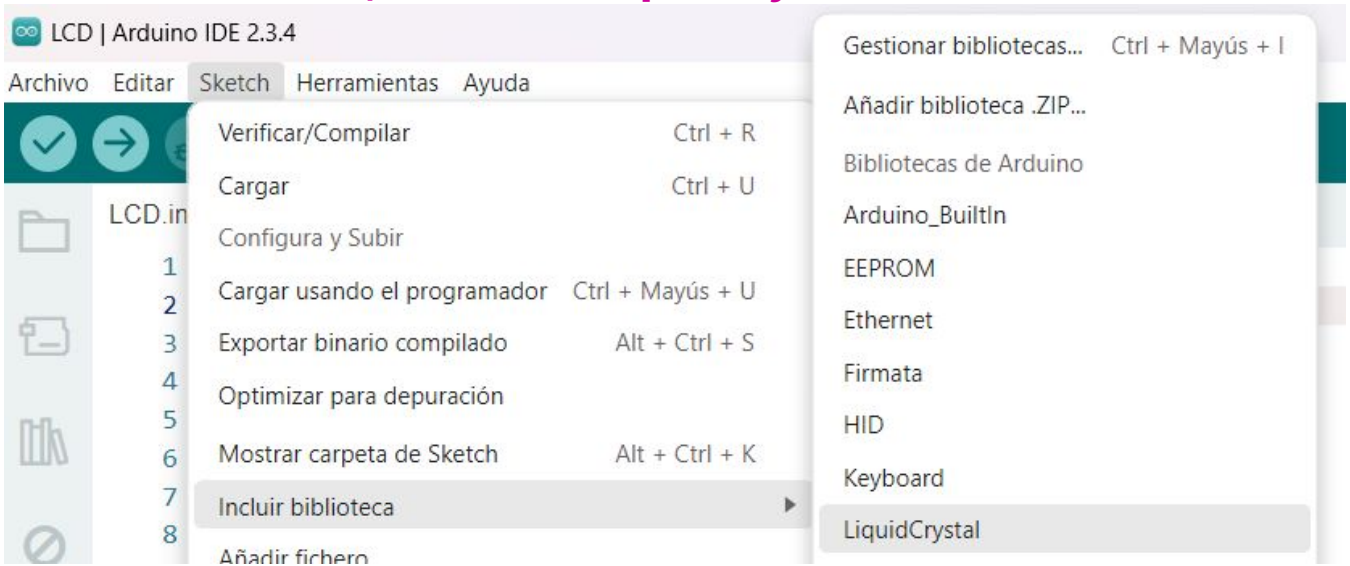


Pantalla LCD

1. Cargar librerías en el arduino:

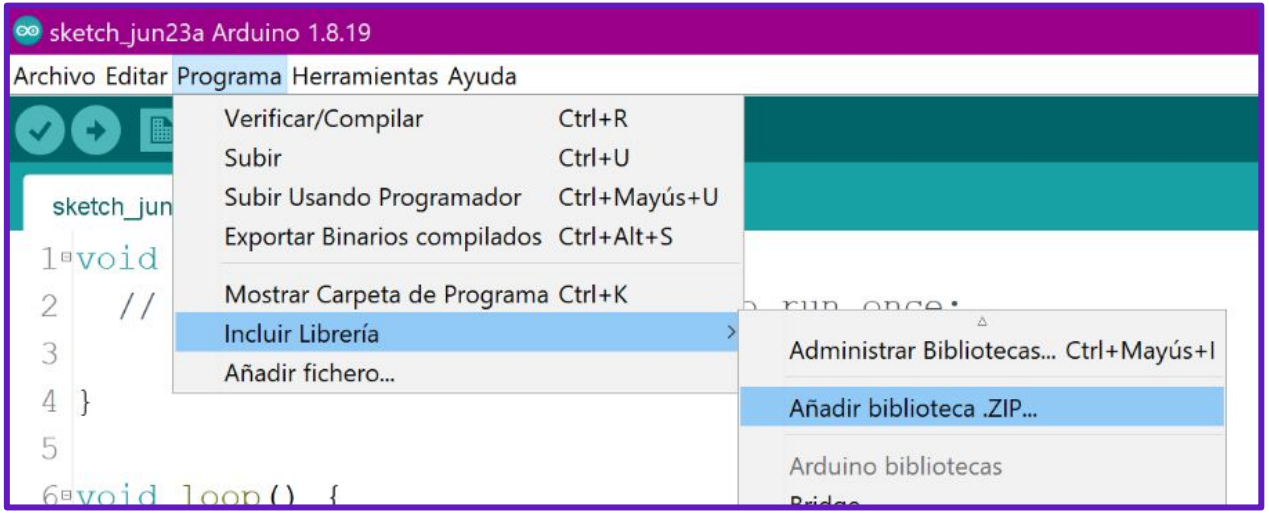
una vez añadida, para poder usarla hay que ir a:

Programa-> **incluir biblioteca/libreria**-> **LiquidCrystal_I2C**



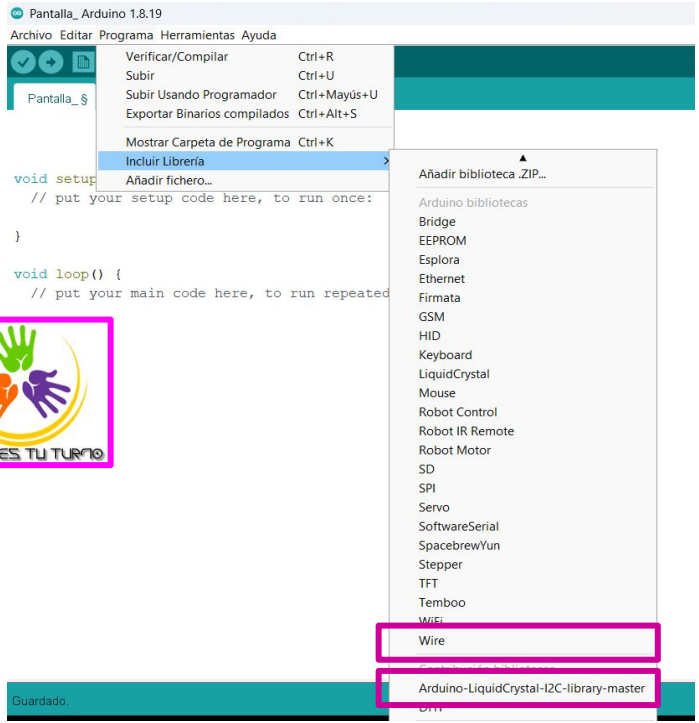
Pantalla LCD

1. Incluir en el IDE la librería **LiquidCrystal_I2C.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

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.init()** 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.
  - Esas son las básicas, pero hay alguna más a ver si eres capaz de localizarlas ;)

¿Os acordáis de los vectores?

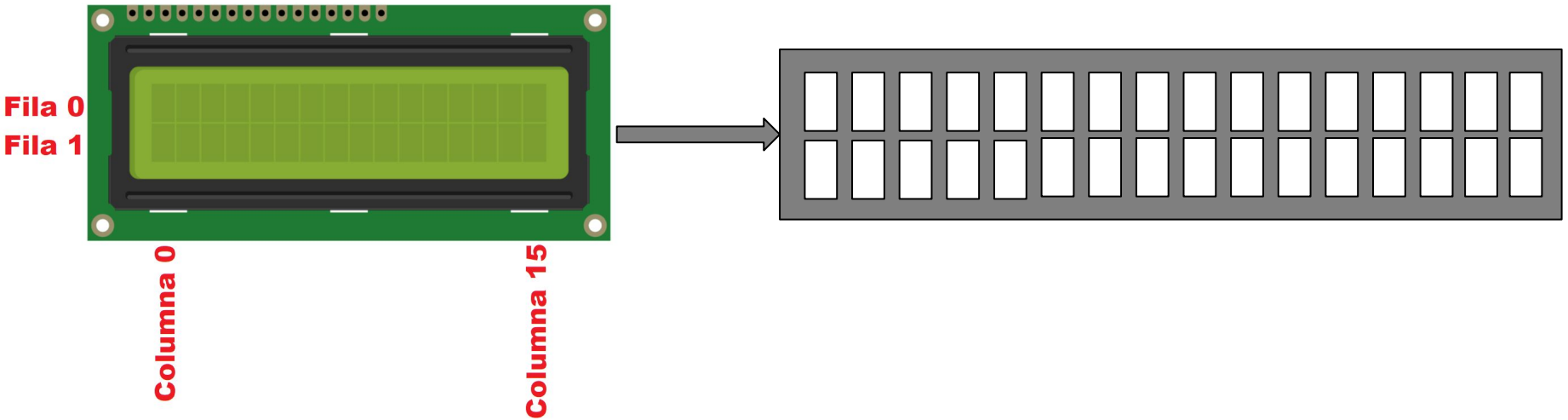
¿Creéis que podríais hacer que la pantalla muestre en la primera línea “Hola”?

Pantalla LCD

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

EXTRA:

- Vamos a entrar más en profundidad en cómo se muestran las letras en pantalla:
  - Siempre se escribe de la fila 0 posición 0 a la fila 0 posición 15
  - segunda fila es la fila 1 posición 0 a la fila 1 posición 15

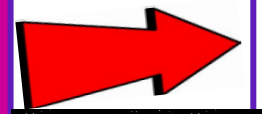


Pantalla LDC

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

- **lcd.init()** inicia la pantalla
- **lcd.backlight()** enciende la luz
- **lcd.setCursor(0, 0)** posiciona el cursor
- **lcd.print("Hola")** escribe en la pantalla
- **lcd.clear()** borra de la pantalla

NOTA: Traduce este pseudocódigo a código real con las funciones que te hemos enseñado



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()
{

      inicio la pantalla
      luz de fondo
 }
 void loop(){
      empiezo a escribir en línea0 pos0
      añado el texto que quiera
      espero 2 segundos
      limpio pantalla
 }
```

Pantalla LCD

Aquí tenéis un ejemplo de cómo usar la pantalla Lcd, como referencia. ¡Probadlo! Os ayudará a entender mejor el lcd.

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd ( 0x27, 16, 2 );
void setup() {
  // put your setup code here, to run once:
    lcd.init ( );
}

void loop() {
  // put your main code here, to run repeatedly:
lcd.backlight ( );                  /*~ Encender la luz de fondo. ~*/
    delay ( 1000 );                   /*~ Esperar 1 segundo. ~*/

  lcd.noBacklight ( );                /*~ Apagar la luz de fondo. ~*/
    delay ( 1000 );                   /*~ Esperar 1 segundo. ~*/

  lcd.backlight ( );                  /*~ Encender la luz de fondo. ~*/
  lcd.setCursor ( 0, 0 );             /*~ ( columnas, filas) Ubicamos el cursor en la primera posición(columna:0) de la primera línea(fila:0) ~*/
  lcd.write ( 0 );                    /*~ Mostramos nuestro primer icono o caracter ~*/
    delay ( 1000 );                   /*~ Esperar 1 segundo ~*/
  lcd.clear ( );                      /*~ Limpiar pantalla ~*/

  lcd.setCursor ( 0, 1 );             /*~ ( columnas, filas) Ubicamos el cursor en la primera posición(columna:0) de la segunda línea(fila:1) ~*/
  lcd.write ( 1 );                    /*~ Mostramos nuestro segundo icono o caracter ~*/
    delay ( 1000 );                   /*~ Esperar 1 segundo ~*/
  lcd.clear ( );                      /*~ Limpiar pantalla ~*/

  lcd.setCursor ( 0, 0 );             /*~ ( columnas, filas) Ubicamos el cursor en la primera posición(columna:0) de la primera línea(fila:0) ~*/
  lcd.print ( "Bienvenido" );         /*~ Mostrar una cadena de texto (no exceder 16 caracteres por línea)~*/
   delay ( 1000 );                    /*~ Esperar 1 segundo ~*/

}
```

Escribiendo mensajes Pantalla LCD

- Ahora vamos a implementar el uso del lcd en nuestro teatro.
- Lo primero será añadir lo necesario para que funcione (bibliotecas, declaración del lcd, setup…) Igual que los ejemplos anteriores.
- Luego vamos a escribir “bienvenido al teatro” al inicio de nuestro programa

Fase 4 - LCD

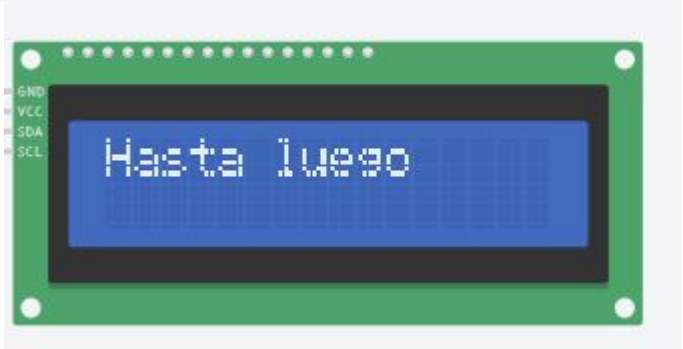
¿Qué nos debería salir?

Nuestro teatro funciona igual, pero nos da la bienvenida!

Si todo funciona, ¡¡Guardad el código en drive, haced fotos y vídeos, copia de seguridad!!

¿Dónde podemos usar el lcd?

Podemos avisar de los nombres de las escenas, despedirnos de nuestro público, saludarlo y ¡muchas otras opciones más!



Si todo funciona, ¡¡Guardad el código en drive, haced fotos y vídeos, copia de seguridad!!

Buzzer

El buzzer hará soniditos cuando se lo pidamos

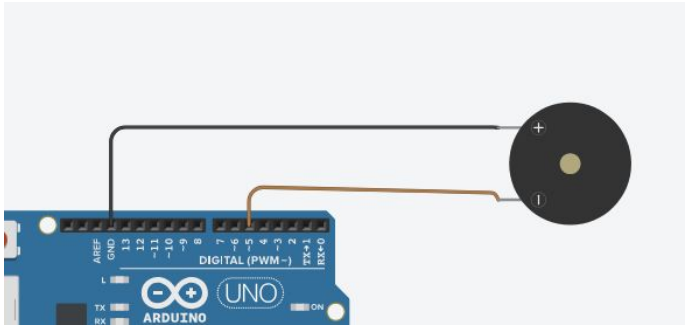
Buzzer

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

★ Uno de **inicio**, cuando subamos la cortina del telón

★ Otro de **final** cuando la bajemos

El buzzer se conecta al pin 5



Buzzer

- Los buzzer tienen dos funciones principales y una de ellas tiene 2 versiones:

```
tone(pin, frecuencia); //activa un tono de frecuencia determinada en un pin dado
tone(pin, frequencia, duracion);  //activa un tono de frecuencia y duracion
                                  //determinados en un pin dado
noTone(pin);  //detiene el tono en el pin
```

Es importante entender que como la función tone usa el Timer 2 de arduino, **los pines 3 y 11 no funcionarán mientras suene el buzzer** (para que lo tengamos en cuenta con nuestro motor)

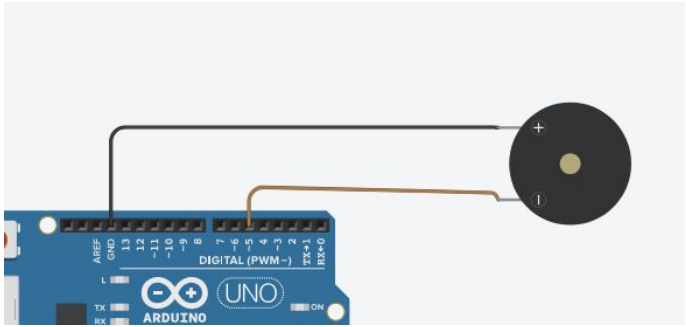
Ahora después os enseñaré un código de ejemplo de cómo funcionan estas funciones

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

Vamos a ver un ejemplo.

En este código, ponemos el pin del buzzer en el pin 5

Le pedimos que genere tonos de diferentes frecuencias usando delay y noTone

Y usamos la versión alternativa de la función tone para que tenga en cuenta la duración:

```
#define BUZZER 5

void setup()
{
}

void loop()
{
  //generar tono de 440Hz durante 1000 ms
  tone(BUZZER, 440);
  delay(1000);

  //detener tono durante 500ms
  noTone(BUZZER);
  delay(500);

  //generar tono de 523Hz durante 500ms, y detenerlo durante 500ms.
  tone(BUZZER, 523, 300);
  delay(500);
}
```

¿Qué pasa si quiero hacer una melodía?

Cuando queremos hacer una melodía necesitamos decir las notas específicas y su duración. **¿Cómo hacemos eso?**

Uno de ellos tendrá nuestras **notas** y otro las **duraciones**, y luego los recorreremos con un for.

Lo primero que veremos son las notas y cómo definirlas.

¿Qué pasa si quiero hacer una melodía?

Primero vamos a ver algunos conceptos básicos.

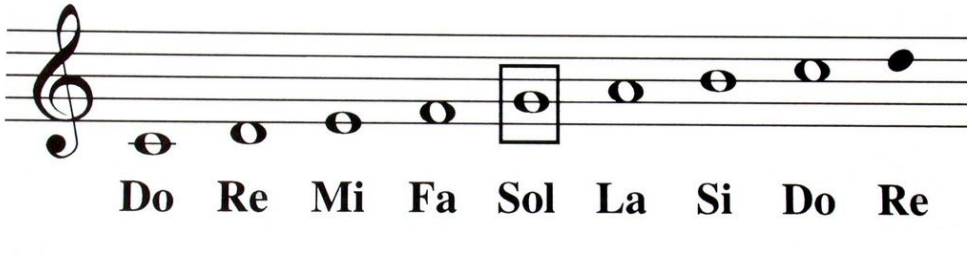
Las notas musicales pueden denominarse de dos formas:

- Sistema latín: Do, re, mi…
- Sistema anglosajón: A, B, C…

Os dejo en la foto una referencia de cómo es cada nota en los dos

Esto es útil para saber qué notas estamos tocando en las tablas de referencia.

```
// C++ code
//
#define BUZZER 5

/*
A - LA
B - SI
C - Do
D - Re
E - Mi
F - Fa
G - Sol
*/
```



¿Qué pasa si quiero hacer una melodía?

Con tablas como estas podemos traducir las notas de cualquier melodía a frecuencias, que será lo que necesitaremos para nuestro buzzer:

(podéis ignorar los decimales o redondearlos)

Music Note To Frequency Chart

MixButton

NOTE	OCTAVE 0	OCTAVE 1	OCTAVE 2	OCTAVE 3	OCTAVE 4	OCTAVE 5	OCTAVE 6	OCTAVE 7	OCTAVE 8
C	16.35 Hz	32.70 Hz	65.41 Hz	130.81 Hz	261.63 Hz (A piano middle C)	523.25 Hz	1046.50 Hz	2093.00 Hz	4186.01 Hz (A piano's highest note)
C#/D♭	17.32 Hz	34.65 Hz	69.30 Hz	138.59 Hz	277.18 Hz	554.37 Hz	1108.73 Hz	2217.46 Hz	4434.92 Hz
D	18.35 Hz	36.71 Hz	73.42 Hz	146.83 Hz	293.66 Hz	587.33 Hz	1174.66 Hz	2349.32 Hz	4698.63 Hz
D#/E♭	19.45 Hz	38.89 Hz	77.78 Hz	155.56 Hz	311.13 Hz	622.25 Hz	1244.51 Hz	2489.02 Hz	4978.03 Hz
E	20.60 Hz	41.20 Hz (A bass's lowest note)	82.41 Hz (A guitar's lowest note)	164.81 Hz	329.63 Hz	659.25 Hz	1318.51 Hz	2637.02 Hz	5274.04 Hz
F	21.83 Hz	43.65 Hz	87.31 Hz	174.61 Hz	349.23 Hz	698.46 Hz	1396.91 Hz	2793.83 Hz	5587.65 Hz
F#/G♭	23.12 Hz	46.25 Hz	92.50 Hz	185.00 Hz	369.99 Hz	739.99 Hz	1479.98 Hz	2959.96 Hz	5919.91 Hz
G	24.50 Hz	49.00 Hz	98.00 Hz	196.00 Hz (A violin's lowest note)	392.00 Hz	783.99 Hz	1567.98 Hz	3135.96 Hz	6271.93 Hz
G#/A♭	25.96 Hz	51.91 Hz	103.83 Hz	207.65 Hz	415.30 Hz	830.61 Hz	1661.22 Hz	3322.44 Hz	6644.88 Hz
A	27.50 Hz (A piano's lowest note)	55.00 Hz	110.00 Hz	220.00 Hz	440.00 Hz	880.00 Hz	1760.00 Hz	3520.00 Hz	7040.00 Hz
A#/B♭	29.14 Hz	58.27 Hz	116.54 Hz	233.08 Hz	466.16 Hz	932.33 Hz	1864.66 Hz	3729.31 Hz	7458.62 Hz
B	30.87 Hz (A 5 string bass's lowest note)	61.74 Hz	123.47 Hz	246.94 Hz	493.88 Hz	987.77 Hz	1975.53 Hz	3951.07 Hz	7902.13 Hz

¿Qué pasa si quiero hacer una melodía?

Ahoravoy a hacer mi canción.

Primero defino mis notas

- NOTE_C5 es la octava 5 de Do (C)
- NOTE_E5 es la octava 5 de Mi (E)

Y luego defino un vector con mi melodía (DO DO MI)

Solo necesito definir cada nota una vez, pero puedo ponerlas varias veces en mi melodía.

Recordad las comas en el vector!!

```
#define NOTE_C5  523
#define NOTE_E5  659

int melodia[] = {
  NOTE_C5, NOTE_C5, NOTE_E5
};
```

Music Note To Frequency Chart

NOTE	OCTAVE 0	OCTAVE 1	OCTAVE 2	OCTAVE 3	OCTAVE 4	OCTAVE 5	O
C	16.35 Hz	32.70 Hz	65.41 Hz	130.81 Hz	261.63 Hz (A piano middle C)	523.25 Hz	10
C#/D♭	17.32 Hz	34.65 Hz	69.30 Hz	138.59 Hz	277.18 Hz	554.37 Hz	11
D	18.35 Hz	36.71 Hz	73.42 Hz	146.83 Hz	293.66 Hz	587.33 Hz	11
D#/E♭	19.45 Hz	38.89 Hz	77.78 Hz	155.56 Hz	311.13 Hz	622.25 Hz	12
E	20.60 Hz	41.20 Hz (A bass's lowest note)	82.41 Hz (A guitar's lowest note)	164.81 Hz	329.63 Hz	659.25 Hz	13
F	21.83 Hz	43.65 Hz	87.31 Hz	174.61 Hz	349.23 Hz	698.46 Hz	13

¿Qué pasa si quiero hacer una melodía?

Y nos faltan las duraciones, quiero que unas notas duren más que otras.

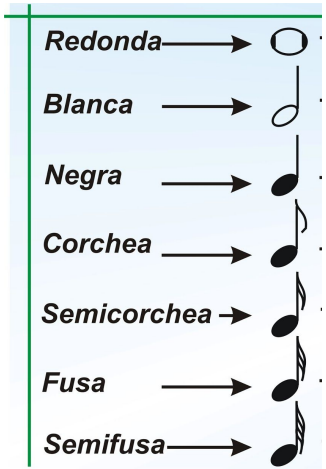
Vamos a definir 3 duraciones.

- La primera va a ser una corchea (8),
- la segunda una negra (4)
- y la tercera una blanca (2).

Estos números se sacan de cuántas veces dividimos un segundo (cuanto más grande, más dividimos y menos dura)

Yo voy a hacer corchea, negra, negra

```
/*
 8 corta
 4 normal
 2 larga
*/
int duraciones[] = {
  8, 4, 4
};
```



¿Qué pasa si quiero hacer una melodía?

En el setup vamos a definir nuestro buzzer como output (puesto que le mandamos información para que suene)

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

¿Qué pasa si quiero hacer una melodía?

Ahora vamos a poner nuestro código.

Primero vamos a ver cuántas notas tenemos viendo el tamaño del vector de duraciones

Luego vamos a ir nota por nota viendo su duración y su melodía

Por último paramos después de cada nota para pasar a la siguiente

```
void loop()
{
  int tam = sizeof(duraciones) / sizeof(int);

  for (int nota = 0; nota < tam; nota++) {
    // Para calcular la duración de la nota, tomamos un segundo
    //dividido por el tipo de nota (corta larga etc)
    //ej. un cuarto = 1000 / 4, un octavo = 1000/8, etc.
    int duracion = 1000 / duraciones[nota];
    tone(BUZZER, melodia[nota], duracion);

    //Para distinguir las notas entre sí, tenemos que poner un
    //tiempo entre ellas
    //la duración de la nota + 30% funciona bien:
    int pausa = duracion * 1.30;
    delay(pausa);

    //stop the tone playing:
    noTone(BUZZER);
  }
}
```

¿Dónde podemos usar el buzzer?

Podemos hacer melodías **cuando subimos la cortina, cuando la bajamos o para cada escena.** Recordad que **mientras suena el buzzer** no podemos usar nuestro programa, así que ¡**no podemos pulsar botones**!

Si todo funciona, ¡¡Guardad el código en drive, haced fotos y vídeos, copia de seguridad!!

Oficina de Software Libre Universidad de Granada

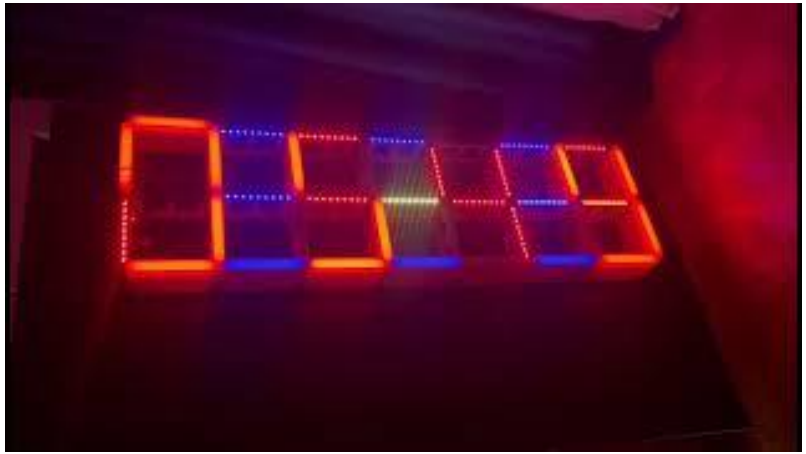
UNIVERSIDAD DE GRANADA

Otras ampliaciones

Algunas ideas que podéis implementar si os sobra tiempo

Leds y botones

Podéis hacer secuencias de colores más avanzadas que se encienden y apagan con efectos para algunas escenas:



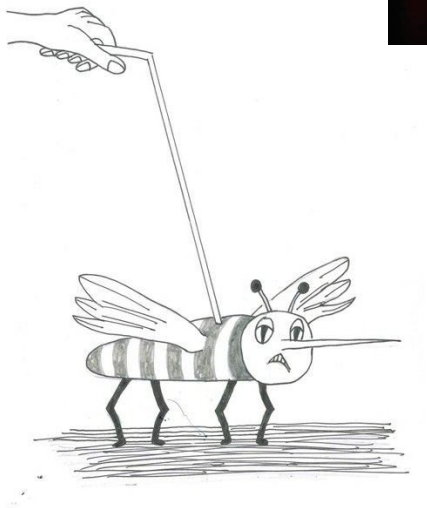
https://youtu.be/k8q8fUaaqEs

TÍTERES Y DECORADO

Hay muchos tipos de títeres que se pueden diseñar para nuestro teatro:

- Varilla superior
- guante
- dedo
- Varilla inferior

TITELLA DE VARETA SUPERIOR:



Titella de vareta superior. Foto Didascàlia

Trampilla secreta

Con un servomotor de 180º adicional podéis diseñar una trampilla secreta en el escenario que levante un cartón o deje que pase un títere: