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.

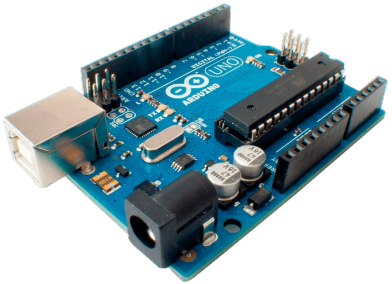
¡No hay preguntas incorrectas!

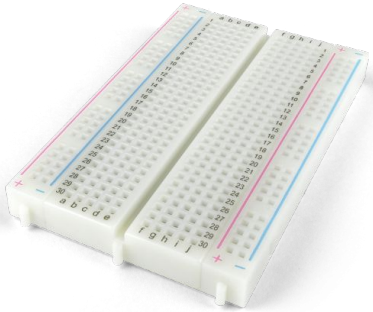
Oficina de Software Libre Universidad de Granada

UNIVERSIDAD DE GRANADA

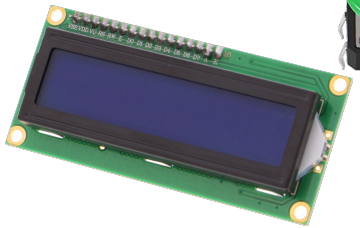
Materiales Necesarios

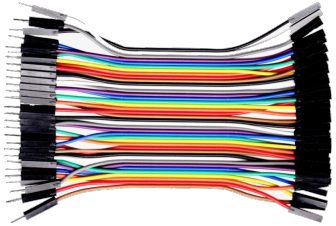
Materiales





- 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

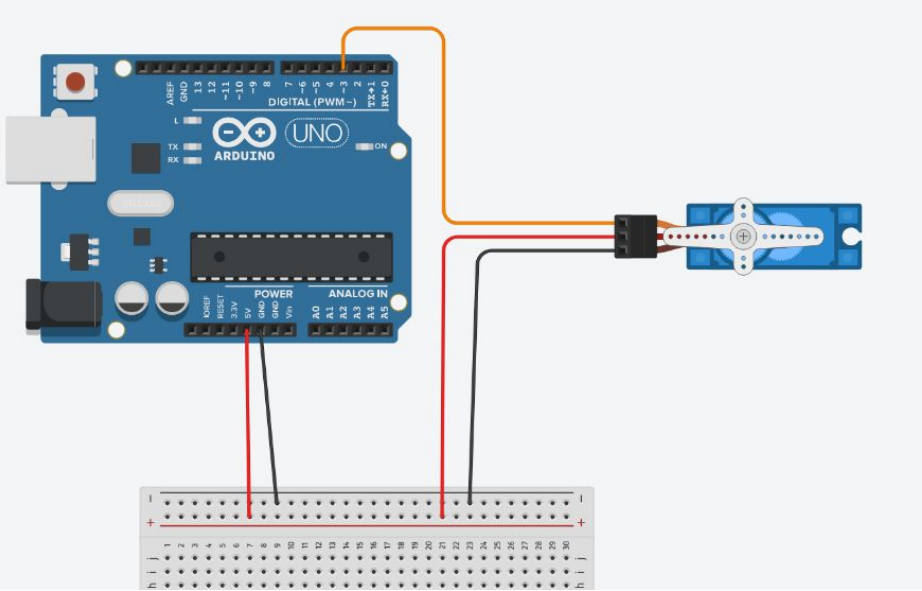
Con este motor abriremos nuestra hucha

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

SERVOMOTORES

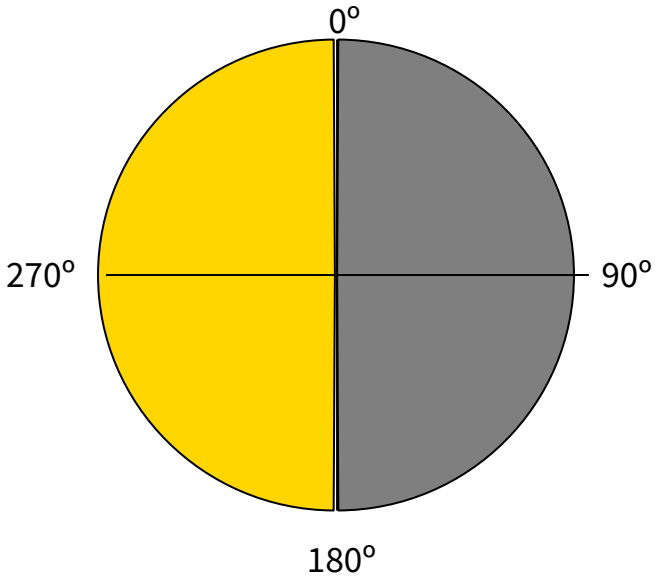
- **Servo motor 360º.**
  - Tienen 3 pines:
    - **DIN** pin de entrada **PWM,** tienen un **~**
    - **+5V** proporciona energía al servo. **VCC**
    - **GND** Toma de tierra.**GND**

¡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



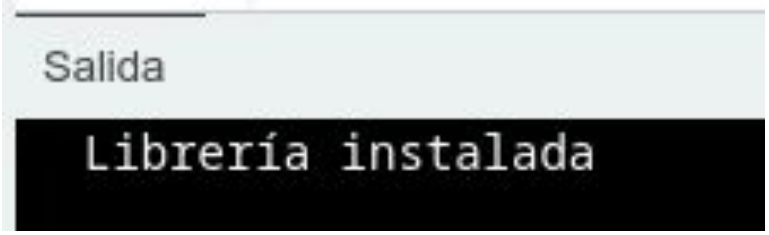
SERVOMOTORES

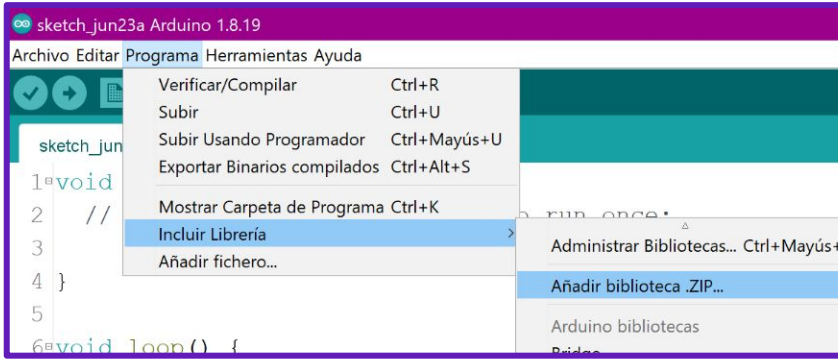
1. Descarga la biblioteca y después la importamos en el arduino:

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

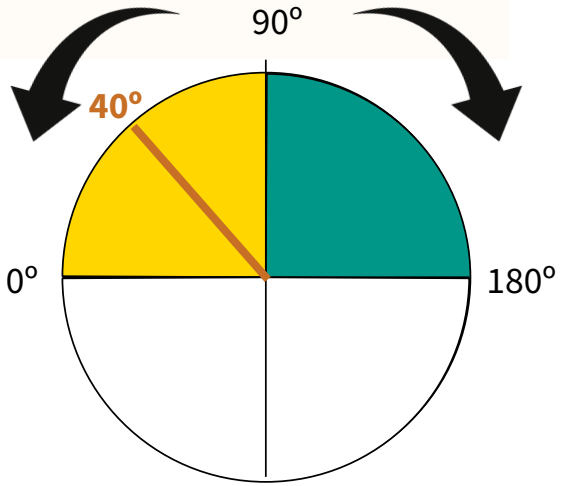
No os preocupéis si no lo entendéis del todo aún, ¡es un ejemplo de función de biblioteca!

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);
}
```

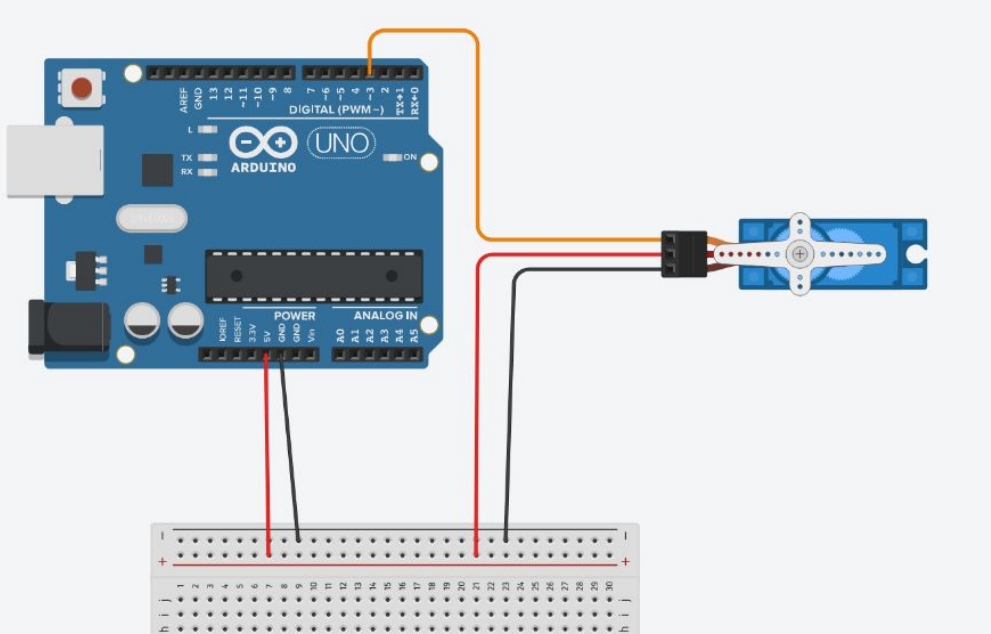
NOTA 1: Aquí empieza parado y va girando cada vez más rápido a la izquierda

```
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);
}
```

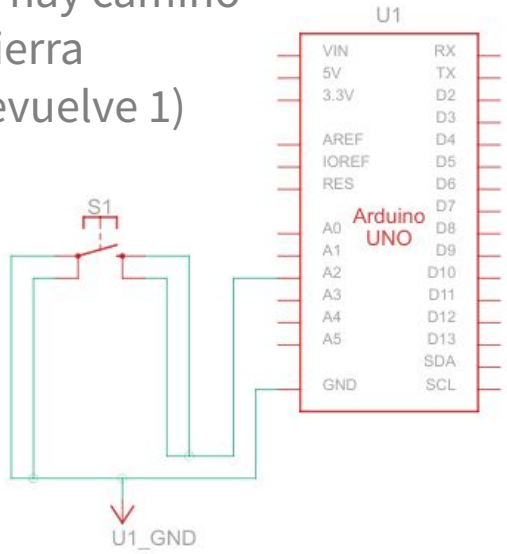
hay que rellenar en los cuadrados verdes

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

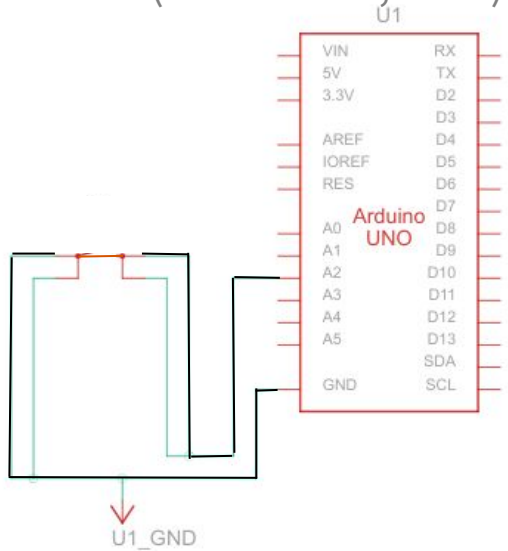
Fase 2 - Botones

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

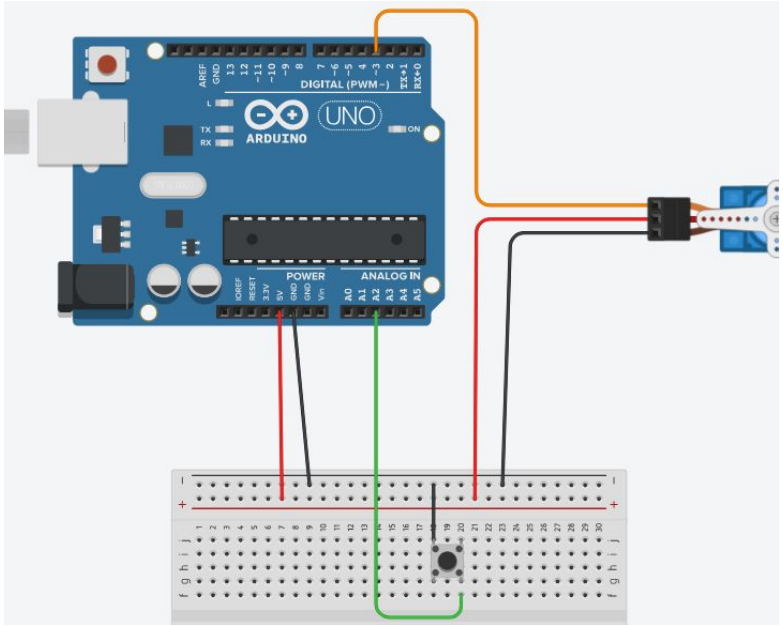
- Para leer de un pin (como, por ejemplo, el de nuestros botones) usamos esta función:

```
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]
}
```

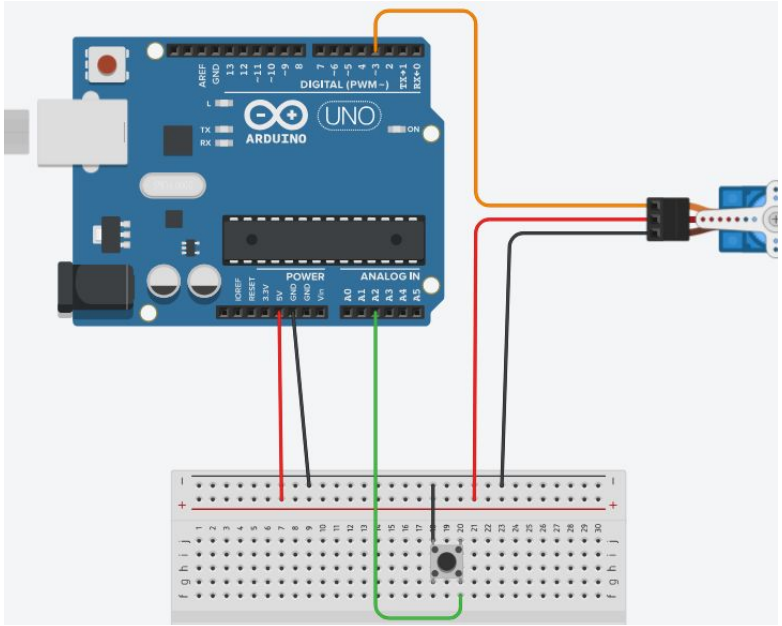
hay que rellenar en los cuadrados verdes

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



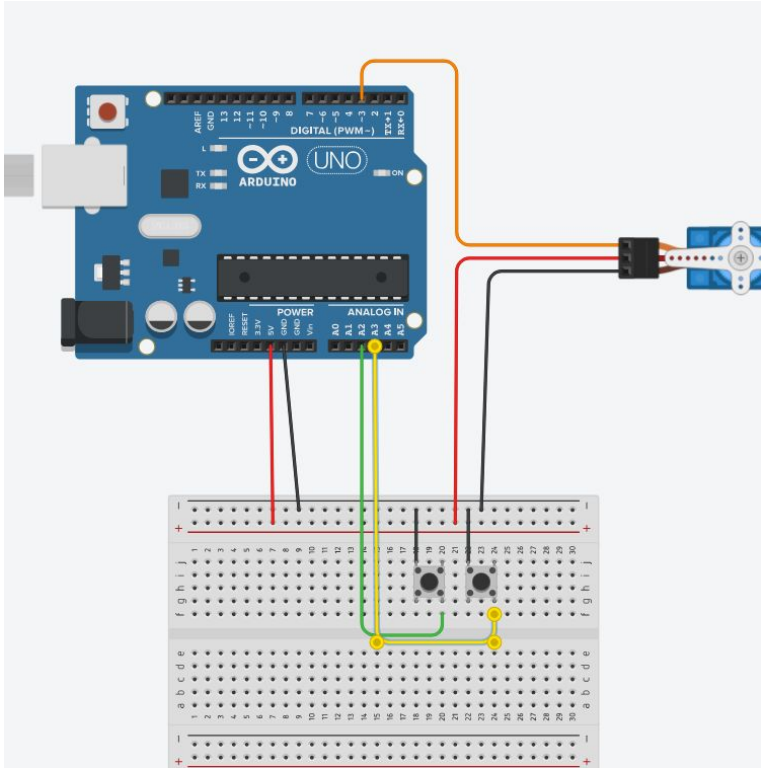

```
void loop(){
  nuevoEstado =  Leer pin botón
  if(  estadoA es diferente a nuevoEstado  ){
    //Ha cambiado de estado
    if(  ¿es nuevo estado igual a 0?  ){
      //el nuevo estado es una pulsación
      esto es el movimiento de motor del reto 1
    }
    estadoA=  asignar nuevoEstado a estadoA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

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

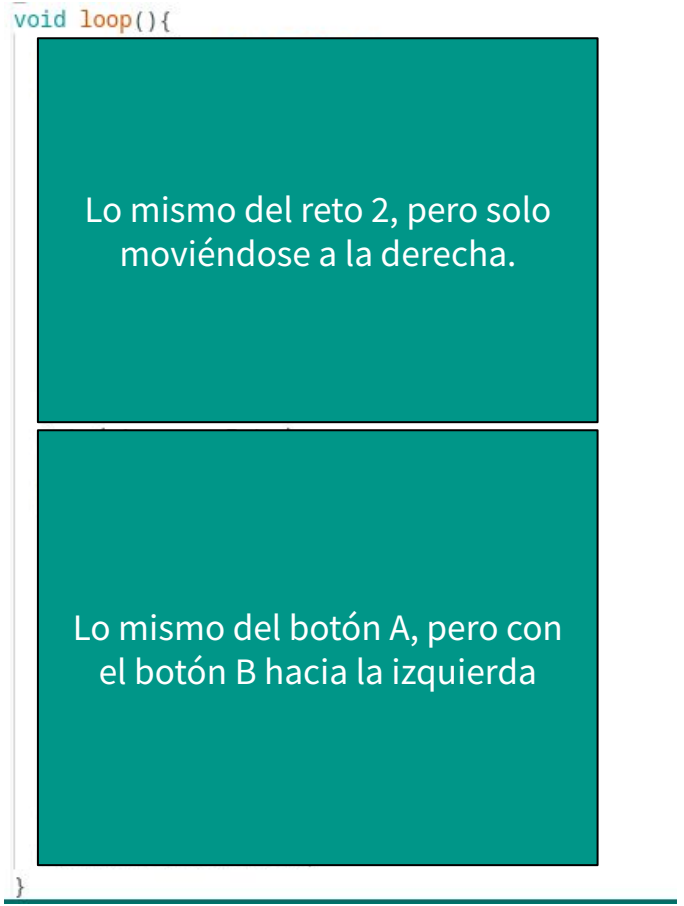
esto es lo del reto 1 y 2

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

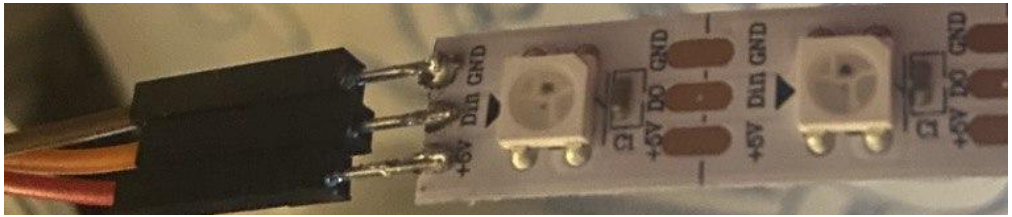
- **Leds RGB direccionables.**
  - Tienen 4 pines:
    - **DIN** pin de entrada
    - **DO** pin de salida
    - **+5V** proporciona energía a los leds
    - **GND** Toma de tierra

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!**



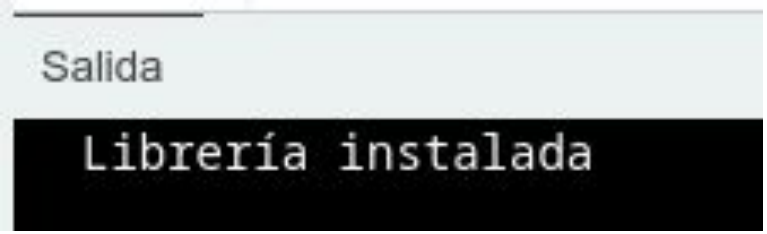
LEDS Y BOTONES

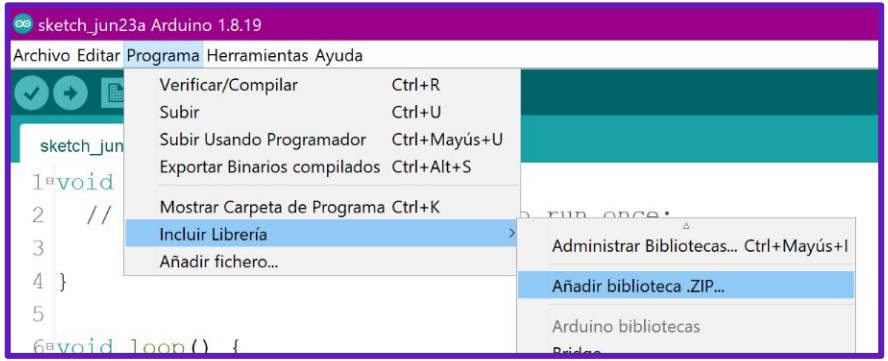
1. Descarga la biblioteca y después la importamos en el arduino:

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.

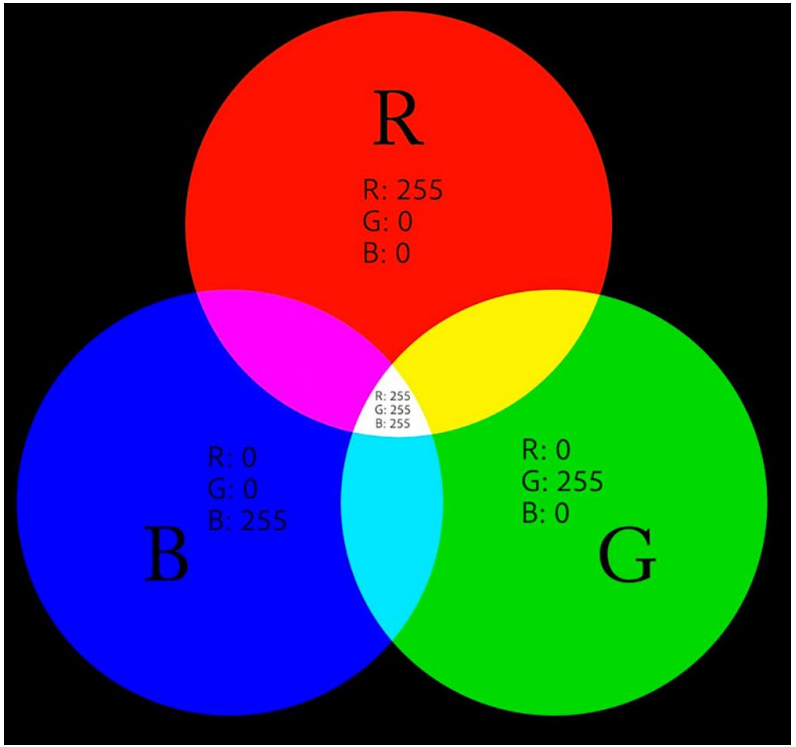
Colores RGB

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.

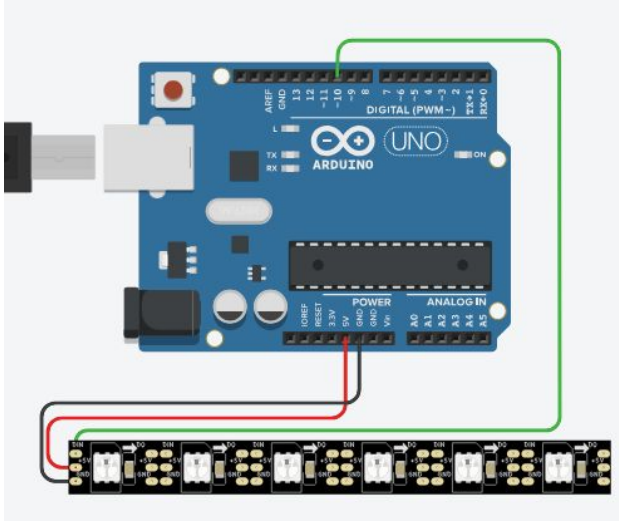
https://www.w3schools.com/colors/colors_rgb.asp



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

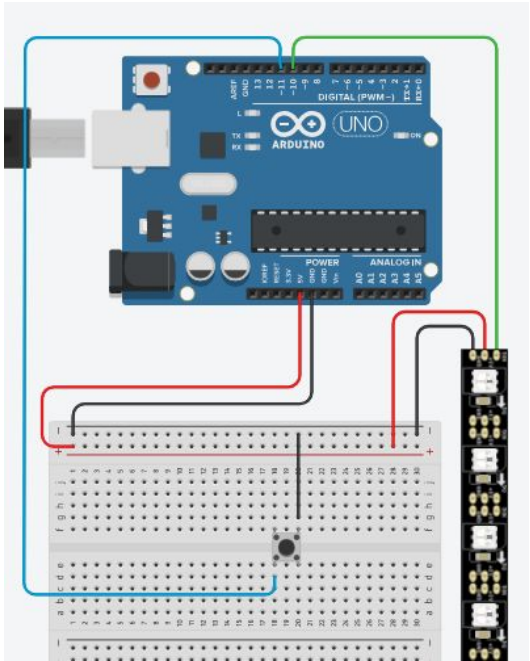
- Para utilizar colores aleatorios, podemos asignar a cada componente un número “random”

```
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.



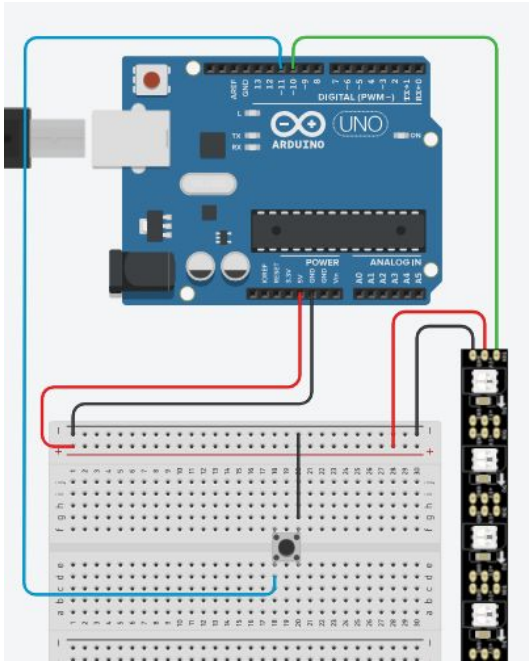

```
//
#include <Adafruit_NeoPixel.h>

// 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.



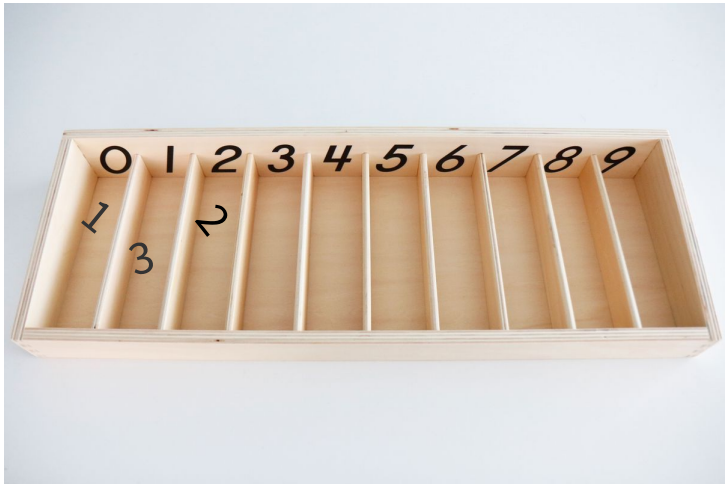

```
void loop()
{
  //
}
```

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:

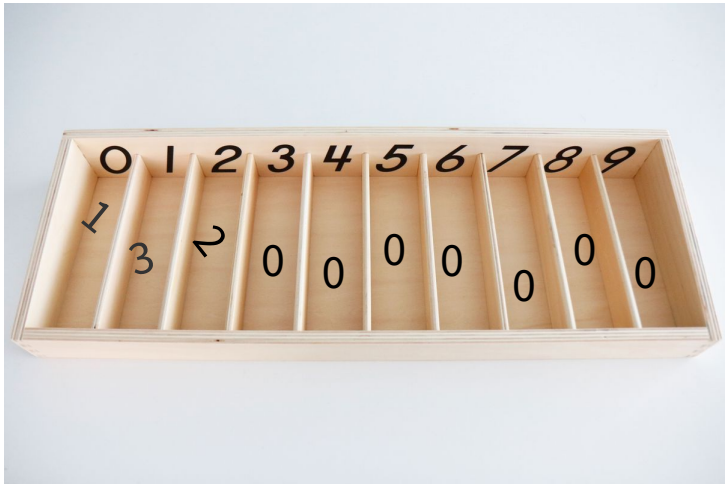


Si nosotros tenemos un tamaño **n** (por ejemplo, n=10), empezamos a enumerarlos por el **0 hasta n-1** (en esta imagen, n-1 es 9).
Si queremos acceder al primer hueco, llamaríamos a la posición 0 (en nuestra imagen, hay un valor 1 ahí).

LEDS Y BOTONES

Para declarar un **vector:**

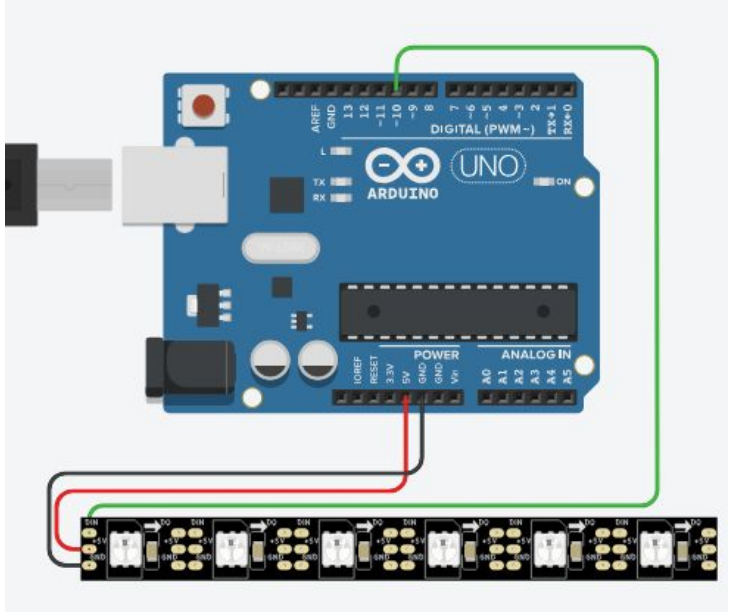
```
tipoDeDato nombre[n] = {valor,...};
```



Este por ejemplo sería

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

LEDS Y BOTONES

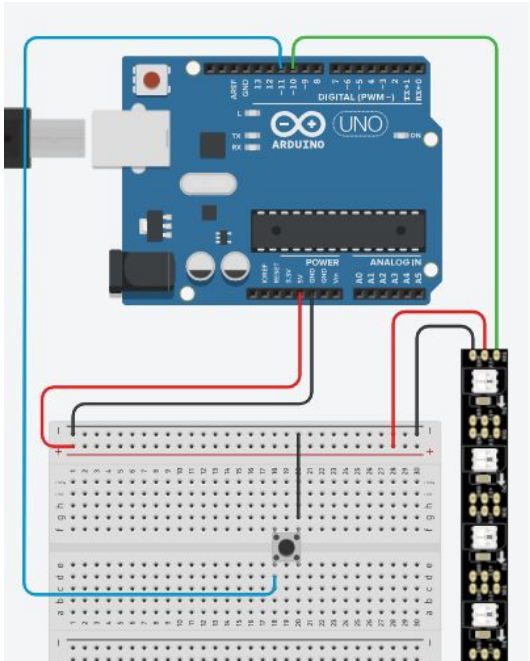


Nuestros leds son como vectores, y cuando usamos `pixels.setPixelColor(i, pixels.Color(rojo, verde, azul));` estamos dándole color al led i.

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



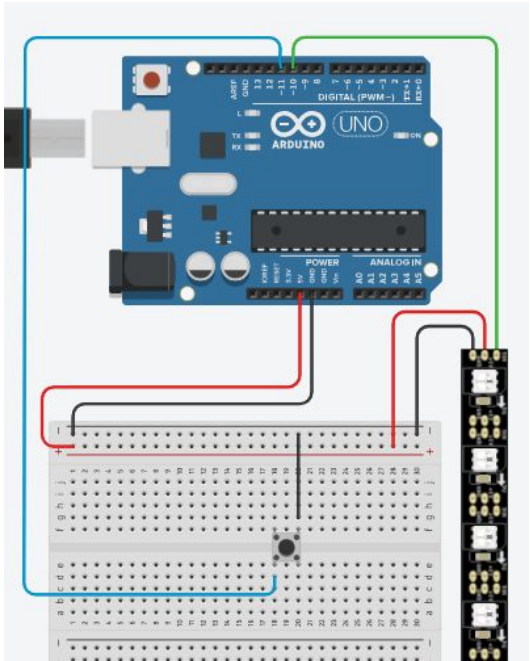

```
//
#include <Adafruit_NeoPixel.h>
[redacted]

// 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.

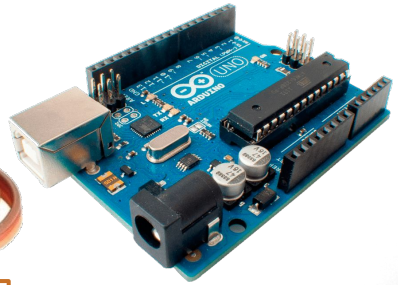
Parte 2: Una vez subido el telón, se encenderán las luces.

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
```

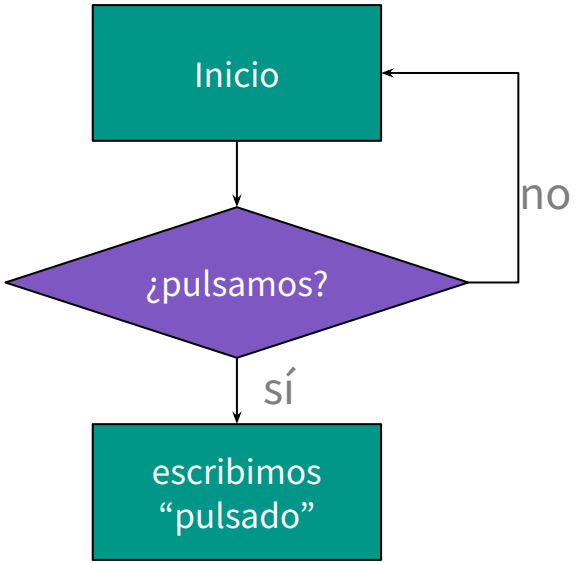
usamos la biblioteca de neopixel para declarar los pixels

```
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

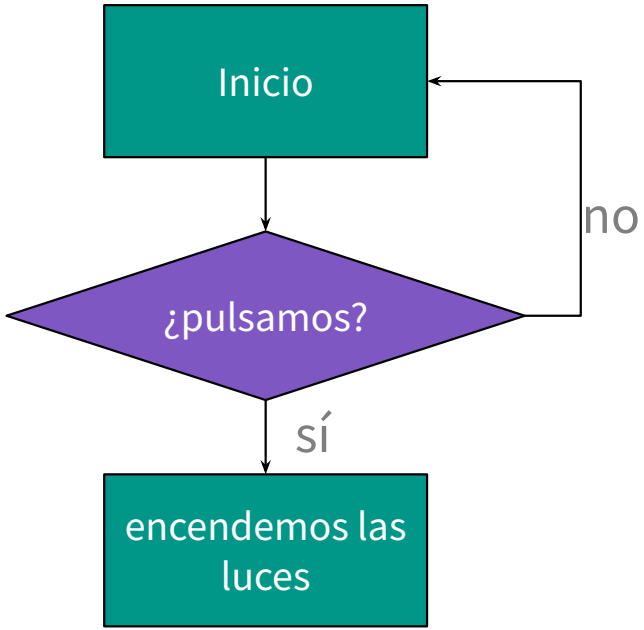
Si no le damos al botón: Nada se mueve y nada sale en pantalla

```
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?

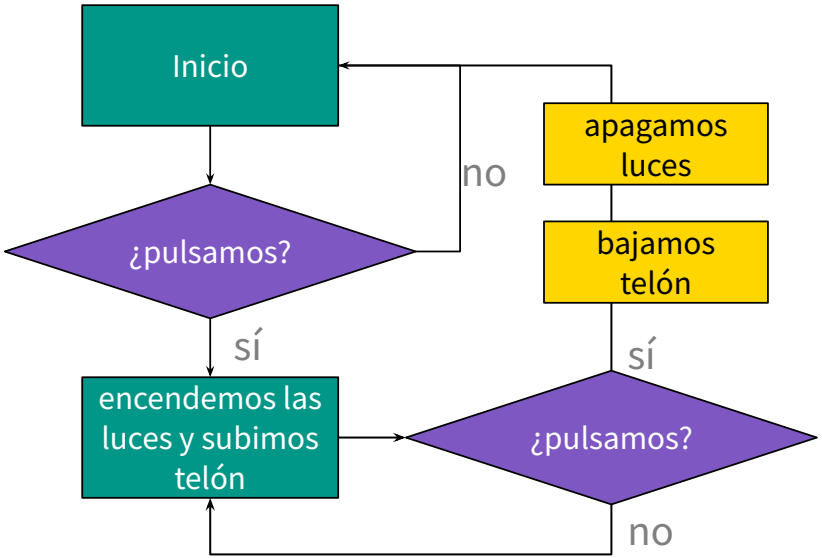
Si le damos al botón: Subirá el telón y se encenderán las luces

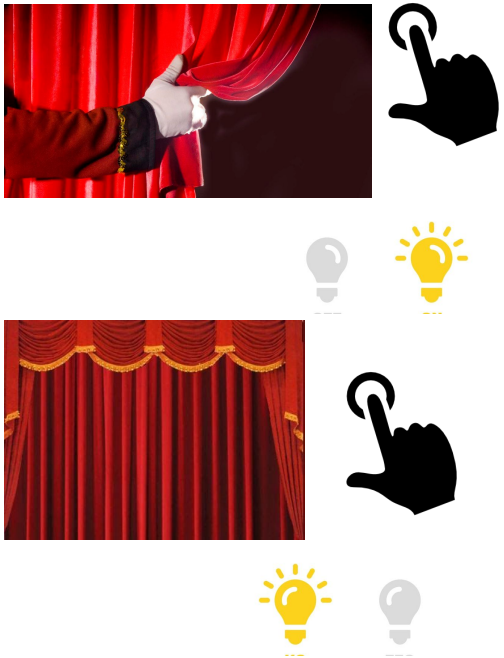
Si no le damos al botón: Nada se mueve y nada sale en pantalla

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