Oficina de Software Libre Universidad de Granada

UNIVERSIDAD DE GRANADA

Coco Secreto

Para guardar tus objetos más valiosos

Autora:

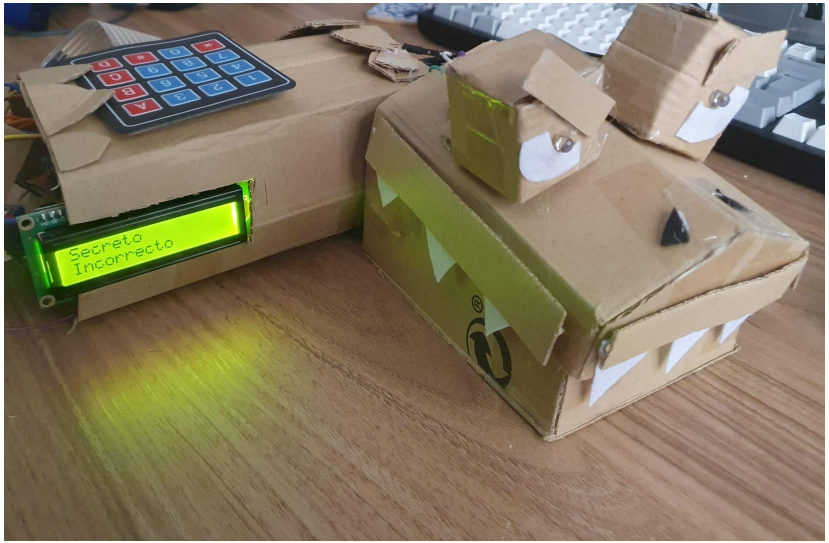
María del Mar Martínez Robles

Inspiración

- De toda la vida se han vendido cajas fuertes de juguete, con contraseña y hechas de plástico, pero ¿nunca te has preguntado cómo de difícil sería hacer una por tu cuenta?
- Este proyecto es una base para aprender cómo desarrollar sistemas con contraseñas y teclados numéricos, que puede ser extendido luego o personalizado como queramos para tener nuestra caja secreta.
- ¡Lo que aprendamos aquí también sirve en otros proyectos! **¿Se te ocurre alguno?**

Presentación

Coco Secreto



- Cada proyecto consistirá en una caja secreta, con contraseña, que nos dará información de si estamos escribiendo bien la contraseña o no.
- Si escribimos bien la contraseña, se abrirá y podremos meter lo que queramos, y al final se cerrará.
- ¡Es tu proyecto! Puedes hacerlo con la temática que quieras, añadir o quitar lo que te guste o incluso sugerir funcionalidades.

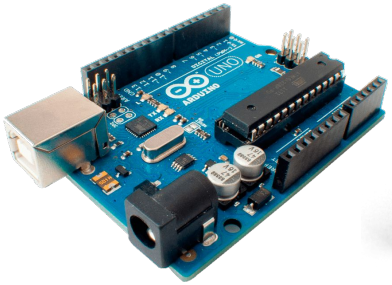
¡No hay preguntas incorrectas!

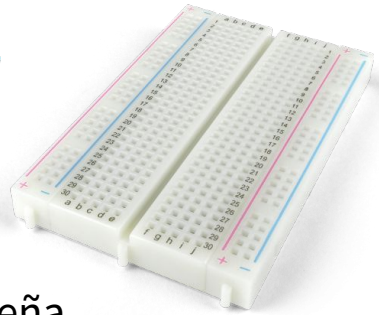
Oficina de Software Libre Universidad de Granada

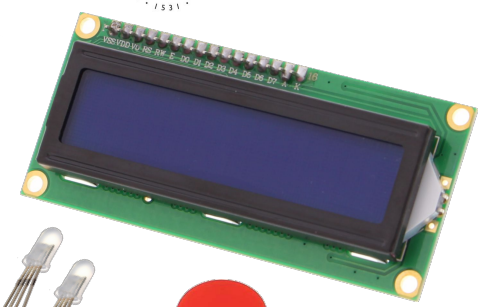
UNIVERSIDAD DE GRANADA

Materiales Necesarios

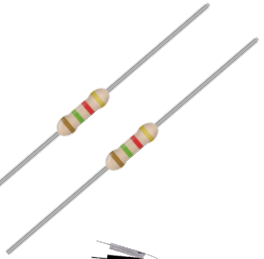
Materiales

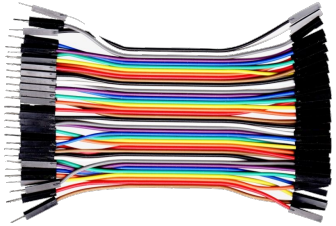






- 1 Arduino uno
- 1 placa de pruebas ‘protoboard’ pequeña
- 2 LEDs RGB
- 2 botones, uno verde y otro rojo
- 2 servomotores
- 1 pantalla LCD 1602 con I2C HD44780
- 1 numpad 4x4
- 2 Resistencias de 220 Ohmios
- 10 cables macho-hembra y 24 cables macho-macho





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. **Hucha base**: Se abrirá y cerrará con contraseña y contará el dinero que tenemos.
2. **Hucha avanzada:** Además de la anterior, tendrá lucecitas en los ojos y una pantalla que nos informa.
3. **Hucha extra,** Le podemos añadir un segundo motor para añadir una “cola”. También meterle botones secretos, sonidos o sensores de presión.

Fases

1. Numpad
   - Meter información a través del numpad.
2. Servomotores y botones
   - Mover los motores con los botones según diferentes criterios
3. Código de la hucha
   - Estructura básica de nuestro proyecto que vamos a ir aumentando
4. LCD y LED
   - Mostrar información por la pantalla y colores en los led
5. Ampliaciones
   - Elige las ampliaciones que quieras (una cola, un buzzer, un botón secreto, sensor de presión…)

Oficina de Software Libre Universidad de Granada

UNIVERSIDAD DE GRANADA

¡EMPEZAMOS!

Fase 1: Numpad

- Aprenderemos a:
  - Usar el numpad
  - Programar contraseñas
  - Usar variables numéricas

NUMPAD

- **¿Qué es y cómo funciona un teclado matricial?**

  Un teclado matricial agrupa varios botones en filas columnas y permite controlarlos con menos pines que si los usáramos sueltos.

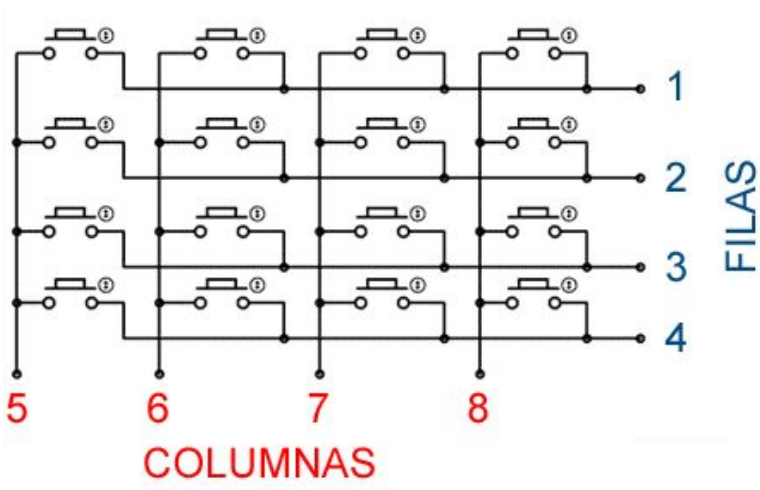
- Los teclados matriciales son frecuentes en electrónica e informática. De hecho, los teclados de ordenador normales son teclados matriciales.

Este teclado nos permite mandar información a nuestro arduino

https://www.luisllamas.es/arduino-teclado-matricial/

- Al detectar la pulsación en la columna X y la fila Y, sabremos que se ha pulsado la tecla (X,Y).

- Funciona internamente similar a los botones individuales. Solo que tendremos que leer cada fila y por cada una sus columnas para saber qué está pulsado. Por eso cuesta ver dos pulsaciones simultáneas.



https://www.luisllamas.es/arduino-teclado-matricial/

1. Incluir en el IDE la biblioteca Keypad.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 **Keypad-3.1.1.zip**
- La **descargamos** y la incluimos de la 1º forma que vemos a continuación.

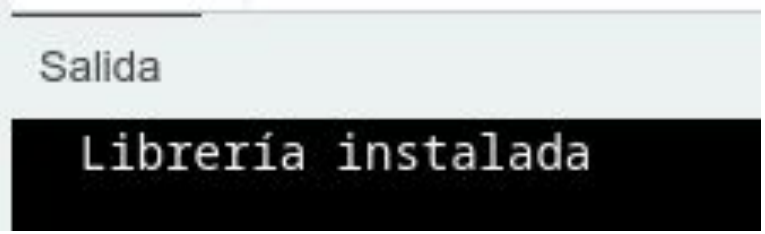
NUMPAD

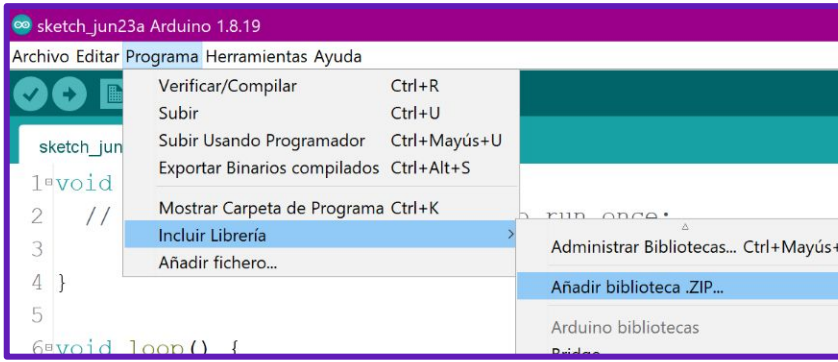
1. Descarga la biblioteca y después la importamos en el arduino:

Está en el enlace:

https://downloads.arduino.cc/libraries/github.com/Chris--A/Keypad-3.1.1.zip

Sketch > Incluir biblioteca > Añadir biblioteca ZIP





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

```
#include <Key.h>
#include <Keypad.h>
```

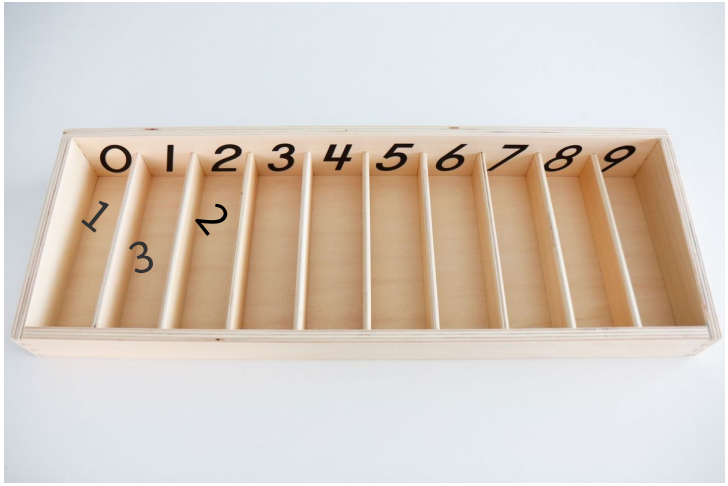
- Las bibliotecas/librerías son código que otras personas crean para facilitar el uso de los componentes. Cuando llamamos a funciones de la biblioteca tenemos que tener cuidado en ver cómo se escriben y qué nos piden

```
Keypad customKeypad = Keypad(makeKeymap(teclas),
                             filPins, colPins, FILAS,
                             COLUMNAS);
```

Por ejemplo, aquí creamos un teclado (Keypad) con un mapa de teclas (makeKeymap(teclas)), con unos pines en las filas (filPins), otros en las columnas (colPins), y un número de filas (FILAS) y columnas (COLUMNAS).

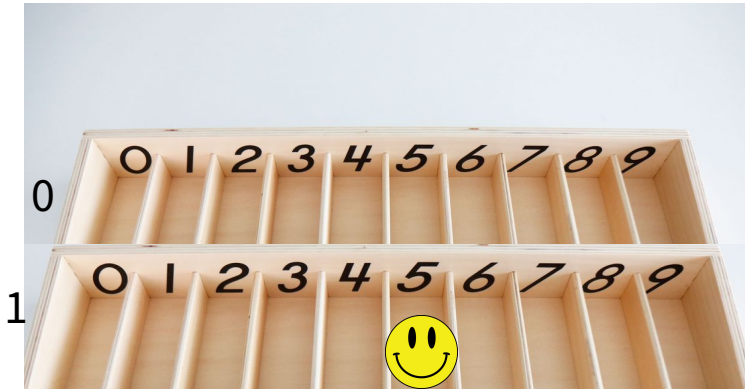
No os preocupéis si no lo entendéis del todo aún, ¡es un ejemplo de función de biblioteca!

- Antes de poder hacer 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).

- Similar al vector, existen las **matrices**, que son varios vectores apilados en filas:



Si tenemos **m filas** (en nuestro ejemplo m=2), las numeramos **de 0 a m-1** (0 a 1).

Para acceder a nuestra carita sonriente, necesitamos ir a la fila 1 y a la columna 5.

Para declarar un **vector:**

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

Para declarar una **matriz:**

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

- Hay dos funciones importantes que debemos aprender antes de usar nuestro teclado:

```
Keypad(makeKeymap(matriz_de_teclas),vector_pines_filas,vector_pines_columnas,numero_filas,numero_columnas);
```

  - En la primera función, creamos nuestro teclado. Vamos a meterle una matriz con todas las teclas de nuestro teclado (por ejemplo, ‘1’, ‘2’, ‘3’, ‘4’ …)
  - Luego tenemos un vector en el que hemos metido los pines en los que hemos conectado nuestras filas (A0, 1, 14, los que sean) y otro vector con los pines de nuestras columnas.
  - Por último, le decimos cuántas filas y columnas tiene nuestro teclado.

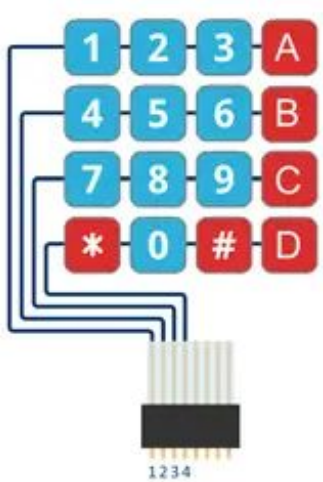
- Hay dos funciones importantes que debemos aprender antes de usar nuestro teclado:

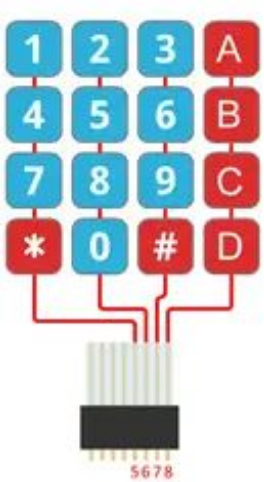
```
char tecla = teclado.getKey();
```

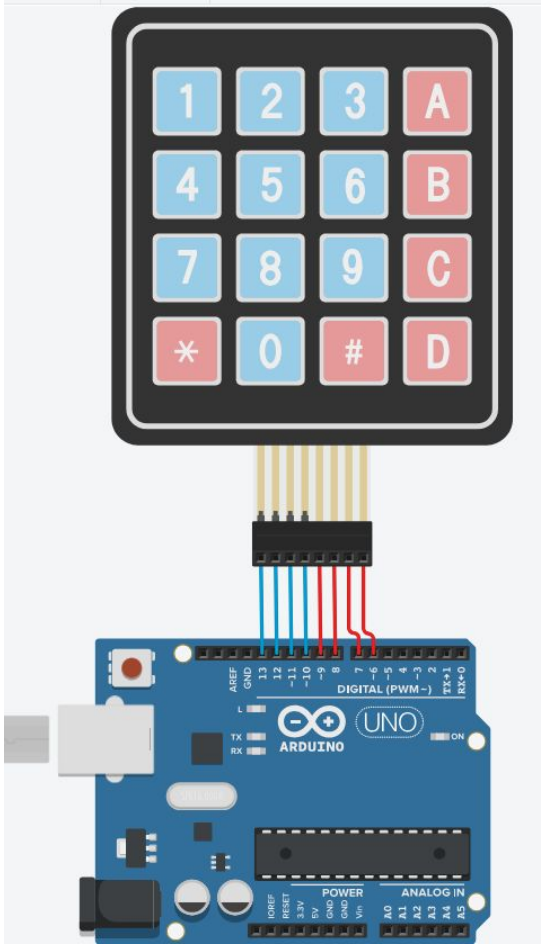
  - En la segunda función, guardamos en una variable llamada “tecla” el resultado de pedirle a nuestro teclado que nos diga qué está siendo pulsado ahora mismo (keypad.getKey()). En este ejemplo, el teclado se llama “teclado”.

Fase 1 - Numpad

Reto 1 - Conectar Numpad y mandar teclas al Arduino.







- Vamos a declarar nuestro keypad en el código para poder utilizarlo

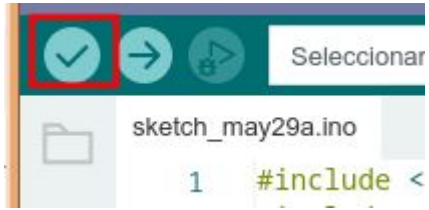
Aquí faltan varias cosas para que funcione:

- **Número de filas y columnas** de nuestro teclado
- **Las teclas** que están en nuestro teclado en una matriz (entre comillas, ‘a’) y su n y m
- **Los pines** en los que hemos conectado nuestras filas y columnas, separados por coma ,
- La **declaración del teclado** rellenando las variables, matrices y vectores que hemos ido creando
- **Conseguir una tecla** y meterla en char tecla

```
#include <Key.h>
#include <Keypad.h>

 //Cuántas filas y columnas tenemos
const int FILAS = ???;
const int COLUMNAS = ???;

 //escribimos las teclas del teclado en una matriz de caracteres
char teclas[??][??] = {
  {...},
  {...},
  {...},
  {...}

}
 // ponemos los pines a los que hemos conectado filas y columnas en vectores
byte filPin[FILAS]={...};
byte colPin[COLUMNAS]={...};

 // iniciamos el teclado para usarlo
 //   recordad: Keypad(makeKeymap(teclas),pines filas,pines columnas,numero filas,numero columnas);
 Keypad teclado = Keypad(makeKeymap(teclas),???,???,???,???);
void setup() {
  //Iniciamos la conexión con el monitor en serie, para mandar
  // mensajes
   Serial.begin(9600);
}

void loop() {
  // recibimos tecla
  //    recordad ¿cómo recibimos una tecla del teclado?
  char tecla = ???
  if (tecla) {
      // mostramos tecla en monitor serie
      Serial.println(tecla);
  }
}
```

Fase 1 - Numpad

Reto 1 - Conectar Numpad y mandar teclas al Arduino.



Cuando creamos que está bien nuestro código, le damos a este botón y lo revisará por nosotros

Si no sale ningún error, podemos conectar nuestro teclado y ver el resultado:

Deberían salir en pantalla las teclas que vayamos tocando

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

Fase 1 - Numpad

Reto 2 - Elegir teclas secretas

Vamos a modificar nuestro loop() para que cuando pulsemos una tecla específica nos diga que está correcta y con el resto no, esto nos servirá luego para la contraseña.

```
void loop() {
  // recibimos tecla
  |
  // si hemos recibido una tecla
  if(tecla){
    // si (tecla igual 'caracter')
    |   // escribir "tecla correcta"
    // sino
    |   // escribir "tecla incorrecta"
  }
}
```

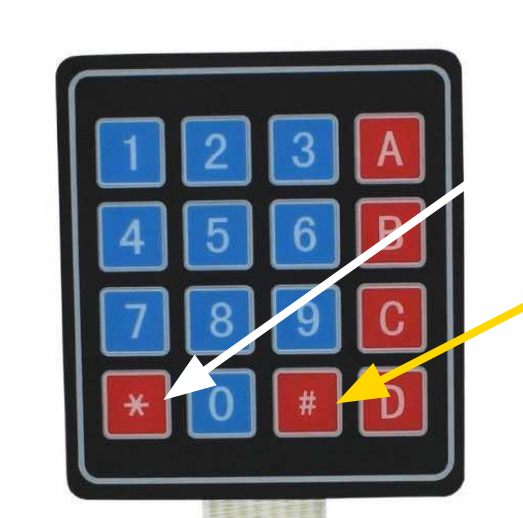
Cuando hayamos escrito este código, lo comprobamos y probamos en nuestro arduino

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

Fase 1 - Numpad

Reto 3 - Escribir palabras enteras

Ya sabemos cómo escribir letras y números sueltos, ¿cómo se escriben las palabras? Para eso vamos a elegir una tecla (o varias) que sirva para “enviar” la palabra que estamos escribiendo.



Yo por ejemplo, he decidido usar la tecla * para borrar la contraseña que estaba escribiendo y la tecla # para decir que he terminado de escribir

Para nuestras contraseñas vamos a usar variables **String.** Contienen texto.

Fase 1 - Numpad

Reto 3 - Escribir palabras enteras

Lo primero que vamos a hacer es declarar variables “String” después de donde habíamos iniciado el teclado. Una de ellas va a ser nuestra contraseña y la otra va a ser donde vamos a ir metiendo las pulsaciones del teclado.

```
18
19  // iniciamos el teclado para usarlo
20  Keypad teclado = Keypad
21
22
23  // contraseña
24  const String secreto = "777A"; // pon tu contraseña aquí
25  String in_secreto;             // aquí vamos a ir metiendo las pulsaciones
26
```

- Vamos a hacer la estructura que necesitamos para que las contraseñas funcionen:

Aquí faltan varias cosas para que funcione:

- **Conseguir una tecla** y meterla en char tecla
- Varios **condicionales** del tipo **if.**
  - Si la tecla es borrar
  - Si la tecla es enviar
  - Si la contraseña es igual al secreto
- La última limpieza de la **contraseña escrita**

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

```
void loop() {
  // recibimos tecla
  char tecla =
  // si hemos recibido la tecla
  if (tecla){
    // si la tecla es "borrar"
    {
      // Limpiamos la contraseña escrita igualando a texto vacio ""
      in_secreto = "";
    }
    // si la tecla no es borrar, pero es enviar
    else if(tecla ==  ) {
      si la contraseña escrita es igual al secreto {
        Serial.println("La contraseña es correcta");


      } else {
        Serial.println("La contraseña es incorrecta, inténtalo de nuevo");

      }
    // una vez hemos comprobado si es correcta o no, tenemos que limpiarla
      // Limpiamos la contraseña escrita
    } else {
      in_secreto += tecla; // añadir caracter a contraseña escrita
    }
  }
}
```

¡FASE SUPERADA!

¡Ya sabemos utilizar nuestro teclado numérico para la hucha!

Si todo funciona, **guardad todo vuestro progreso y haced foto de los pines**, porque vamos a empezar a aprender otros componentes y desconectamos el teclado para usarlo luego.

Fase 2: Servomotores

- Aprenderemos a:
  - Usar servomotores
  - Hacer que se muevan usando 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.

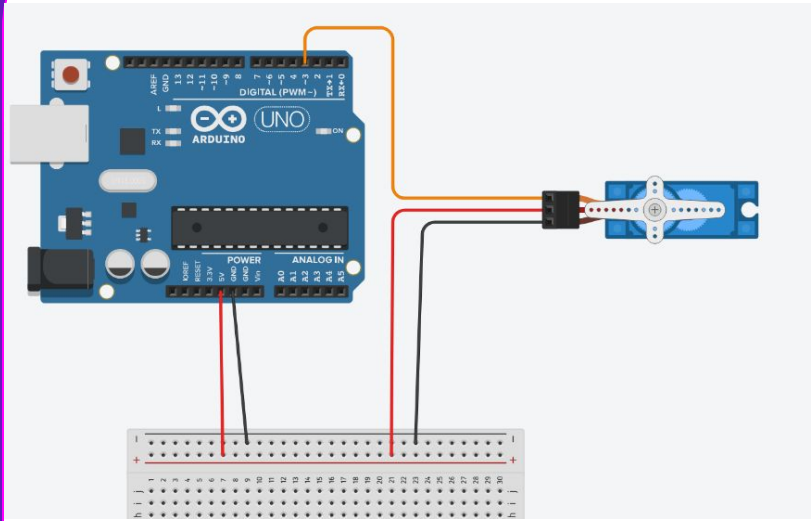
Con este motor abriremos nuestra hucha

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

Servomotores

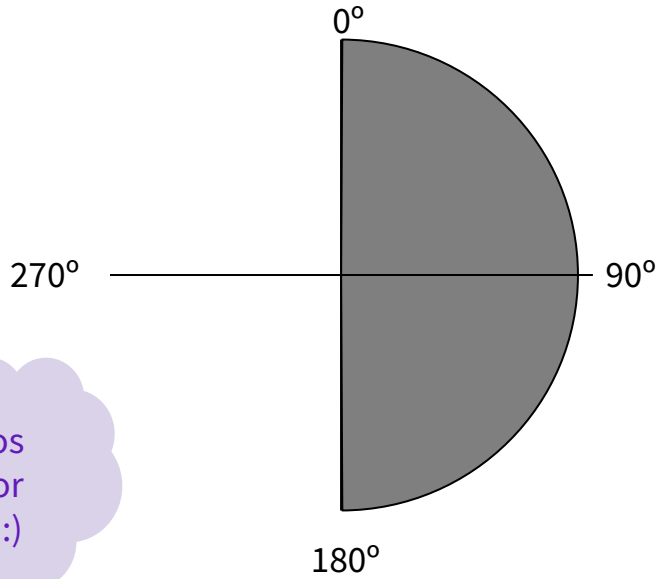
- **Servo motor 180º.**
  - Tienen 3 pines:
    - **DIN** pin de entrada **PWM,** tienen un **~**
    - **+5V** proporciona energía al servo. **VCC**
    - **GND** Toma de tierra.**GND**

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



ServoMotores

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



Os dejo los grados por si dudáis :)

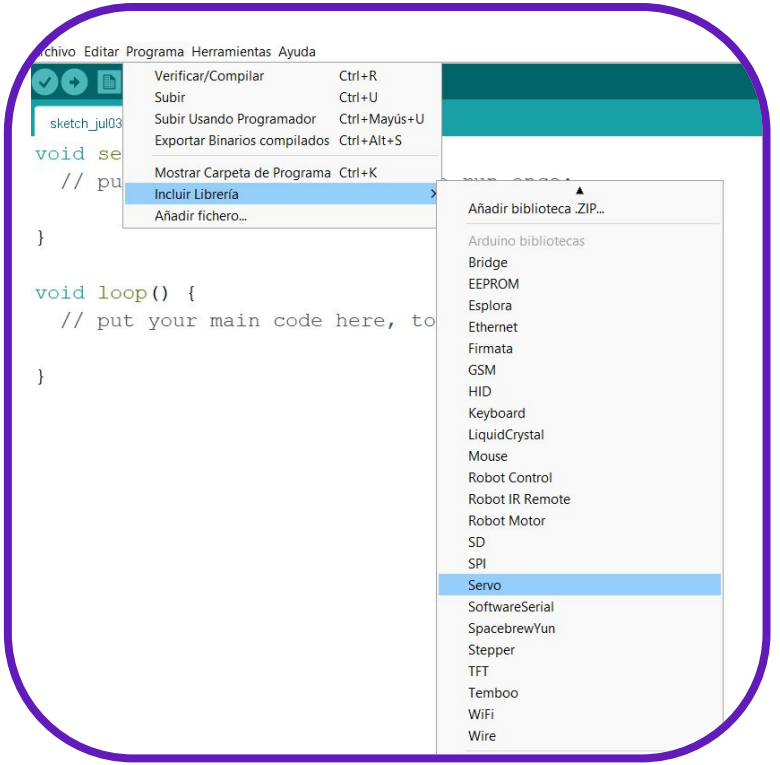
ServoMotores

2. ¡Comencemos a programar!

Para este servomotor hace falta incluir una **librería**, para ello incluimos el siguiente zip:

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

Y la incluimos igual que hicimos con el numpad



ServoMotores

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

servo1.ino

```
1   #include <Servo.h>
2
3   //Este es el pin al que conectamos la señal
4   #define PIN_BOCA 3
5
6   // este es el nombre del servo que vamos a usar
7   Servo Boca;
8
9   void setup()
10  {
11    //Aquí le decimos que el servo de la boca va a usar el pin 3
12    Boca.attach(PIN_BOCA);
13  }
```

ServoMotores

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

```
servo.write(posicion)
```

  - Esta función hace que nuestro servo (llamado servo en esta situación) se ponga en el ángulo especificado en posición (que puede ser una variable o un número entero)

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 esa posición.

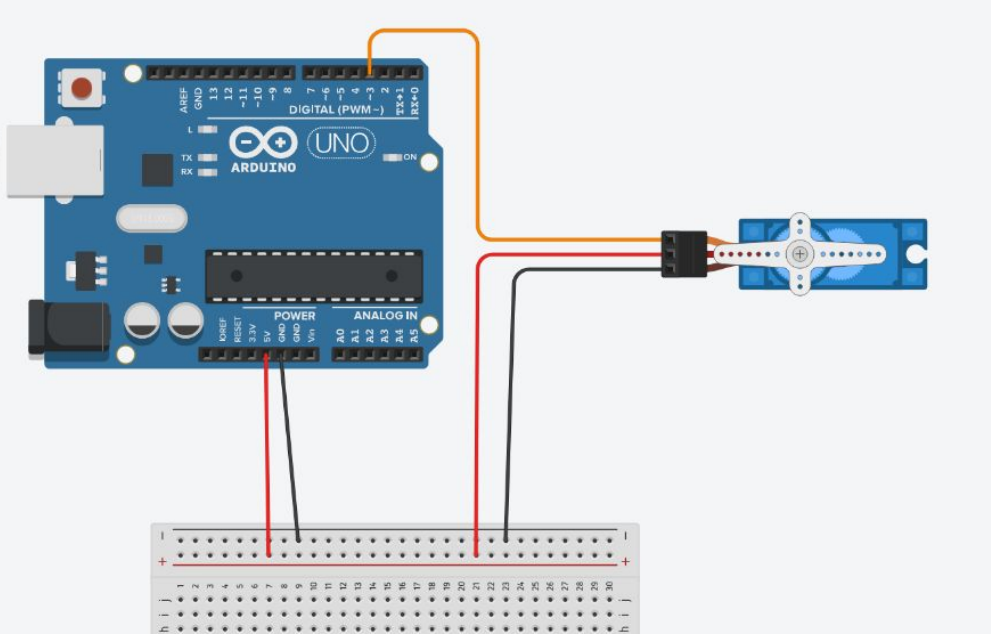
NOTA 1: Aquí gira para un sentido hasta los 50 grados

NOTA 2: Aquí gira para el otro sentido hasta volver a 0.

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

Fase 2 - Servomotores

Reto 1 - Hacer que el servo se abra y luego se cierre.



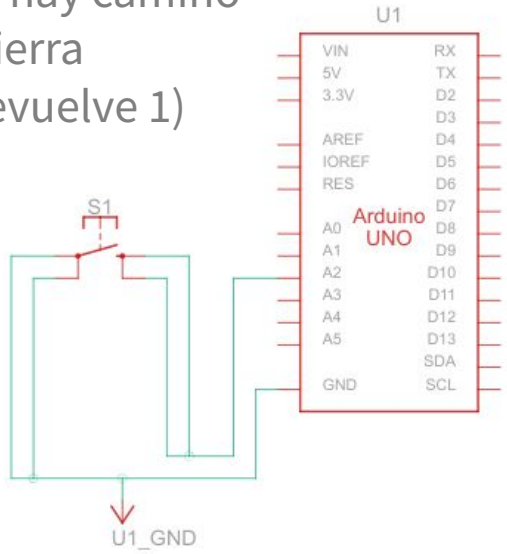

```
#include <Servo.h>

//Este es el pin al que conectamos la señal
#define PIN_BOCA 3

// este es el nombre del servo que vamos a usar
Servo Boca;
 // Aquí vamos a definir cuántos grados son "boca abierta" y cuántos "boca cerrada"
int grados_abierto, grados_cerrado;

void setup()
{
  //Aquí le decimos que el servo de la boca va a usar el pin 3
  Boca.
  // definimos los grados (grados_abierto y grados_cerrado)

  // empezamos con la boca cerrada
}

void loop()
{
        Primero abrimos hasta "grados abiertos" en un bucle for
        delay de 1000
        cerramos hasta "grados cerrado" en otro bucle for
        delay de 100
}
```

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

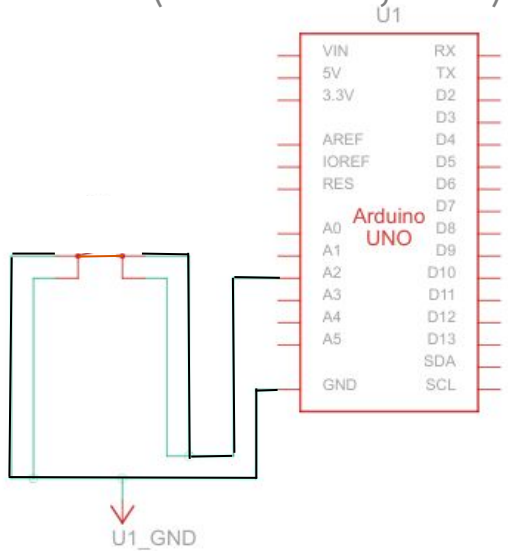
Fase 2 - Botones

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)



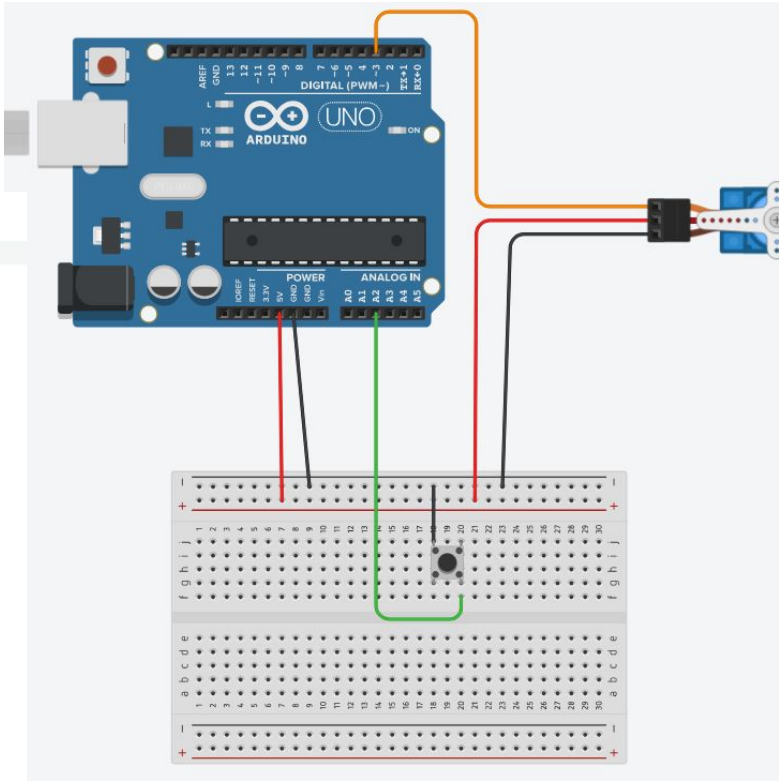
Hay un camino a tierra (devuelve 0, GND)



Fase 2 - Botones

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

```
#define PIN_BOCA 3
                                  Definimos el pin donde conectemos el botón
// este es el nombre del servo que vamos a usar
Servo Boca;
 // Aquí vamos a definir cuántos grados son "boca abierta" y cuántos "boca cerrada"
int grados_abierto, grados_cerrado;

// estados de los botones
bool nuevoEstado, EstadoA;

void setup()
{
  //Aquí le decimos que el servo de la boca va a usar el pin 3
  Boca
  // definimos los grados (grados_abierto y grados_cerrado)
                                  Esto es lo del reto 1 de los servomotores
  // empezamos con la boca cerrada
  // declaramos el pin del boton como INPUT (queremos que nos de información)
  //  y PULLUP (con una resistencia)
  pinMode(BOTON_A, INPUT_PULLUP);
  // estados iniciales de los botones
                                  Estarán los dos a 1
}
```



Botones

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

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

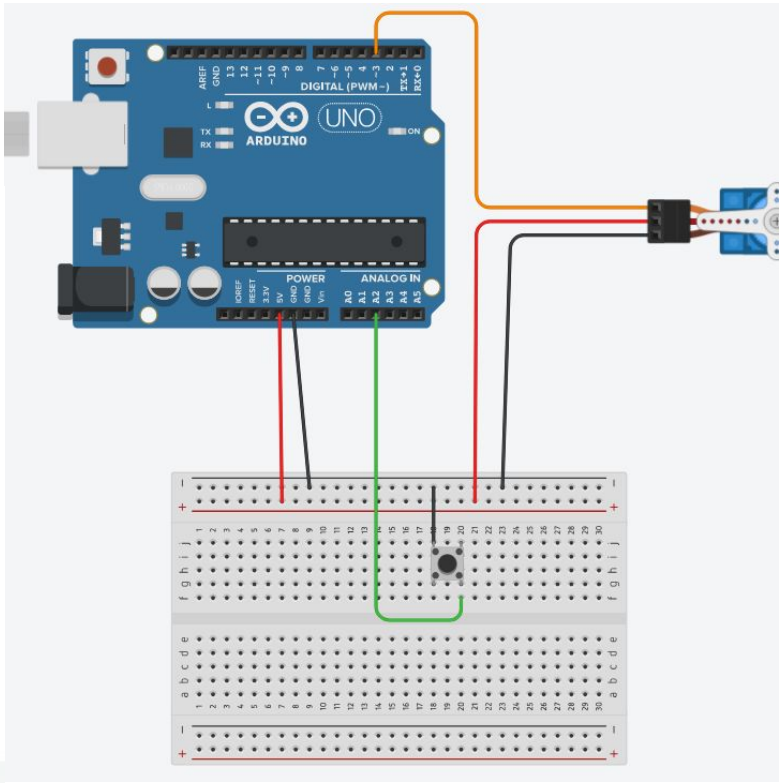
  - Esta función nos devuelve un valor (que estamos metiendo en la variable a). Lee el pin (llamado PIN) y nos dice qué hay en él.

Fase 2 - Botones

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

```
void loop()
{
  Leemos lo que hay en BOTON_A y lo metemos en nuevoEstado
  Si EstadoA no es igual a nuevoEstado {
    // Ha cambiado de estado
    Si nuevoEstado es un 0 (pulsado) {
      // el nuevo estado es una pulsación
      Código de mover servomotor
    }
  }
}
```

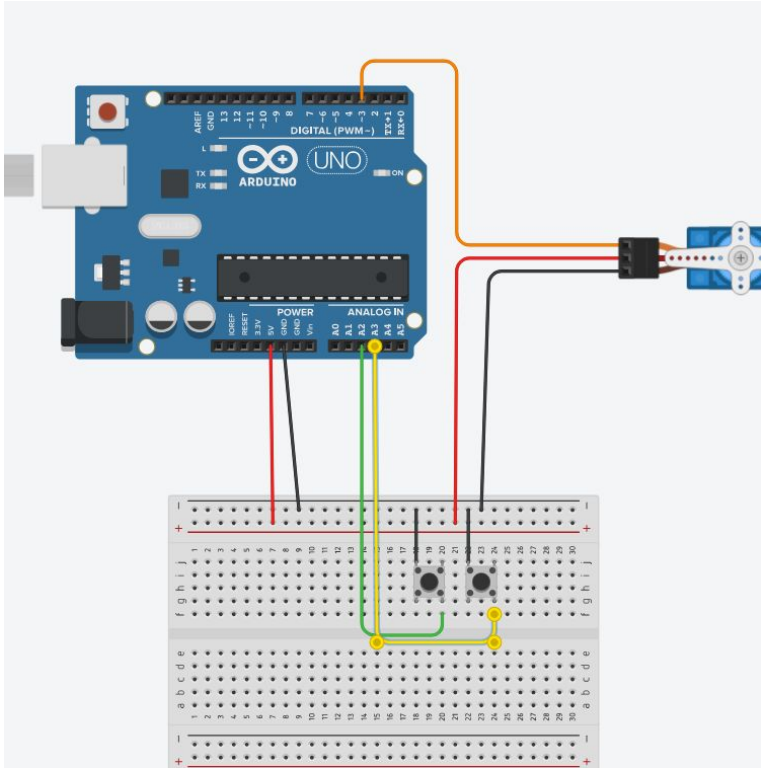
Si todo funciona, ¡¡Guardad el código en drive, haced fotos y vídeos, copia de seguridad!!



Fase 2 - Botones

Reto 3 - Hacer que el servo se abra con un botón y se cierre 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.

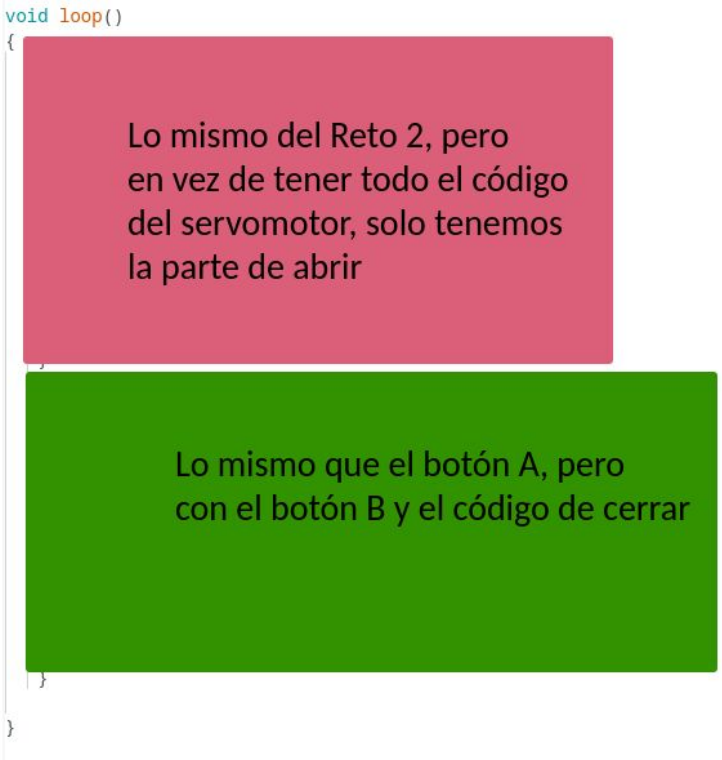
```
//Este es el pin al que conectamos la señal
#define PIN_BOCA 3
[redacted]
[redacted]

// este es el nombre del servo que vamos a usar
Servo Boca;
 // Aquí vamos a definir cuántos grados son "boca abierta" y cuántos "boca cerrada"
int grados_abierto, grados_cerrado;

// estados de los botones
bool nuevoEstado, EstadoA, EstadoB;

void setup()
{
  //Aquí le decimos que el servo de la boca va a usar el pin 3
  Boca[redacted]
  // definimos los grados (grados_abierto y grados_cerrado)
  [redacted]
  // empezamos con la boca cerrada
  [redacted]
  // declaramos el pin del boton como INPUT (queremos que nos de información)
  //  y PULLUP (con una resistencia)
  pinMode(BOTON_A, INPUT_PULLUP);
  [redacted]
  // estados iniciales de los botones
  [redacted]
  [redacted]
  [redacted]
}
```

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 la hucha!

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.

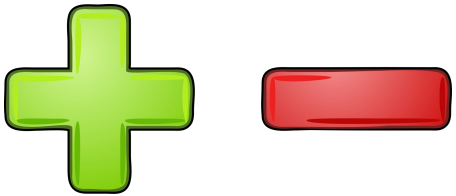
Fase 3: Código

- Aprenderemos a:
  - Crear la estructura sobre la que nuestra hucha funcionará.
  - Unir nuestros conocimientos de los componentes al código.
  - ¡Tener una hucha funcionando!

CÓDIGO

¿En qué va a consistir nuestra hucha?

Parte 1: Preguntamos por la contraseña y comprobamos que está bien. Si está bien, abrimos la caja.



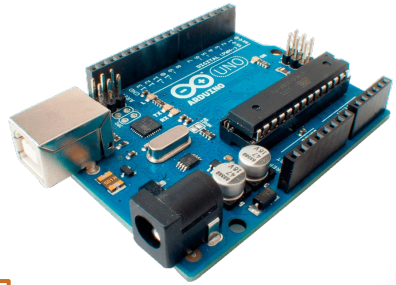
Parte 2: Una vez abierta la caja, preguntamos si queremos sumar o restar dinero, con los botones.

Parte 3: Cuando sabemos si sumar o restar, vamos a preguntar la cantidad y la vamos a guardar. Luego, por último, vamos a cerrar la caja y volver a la fase 1.

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

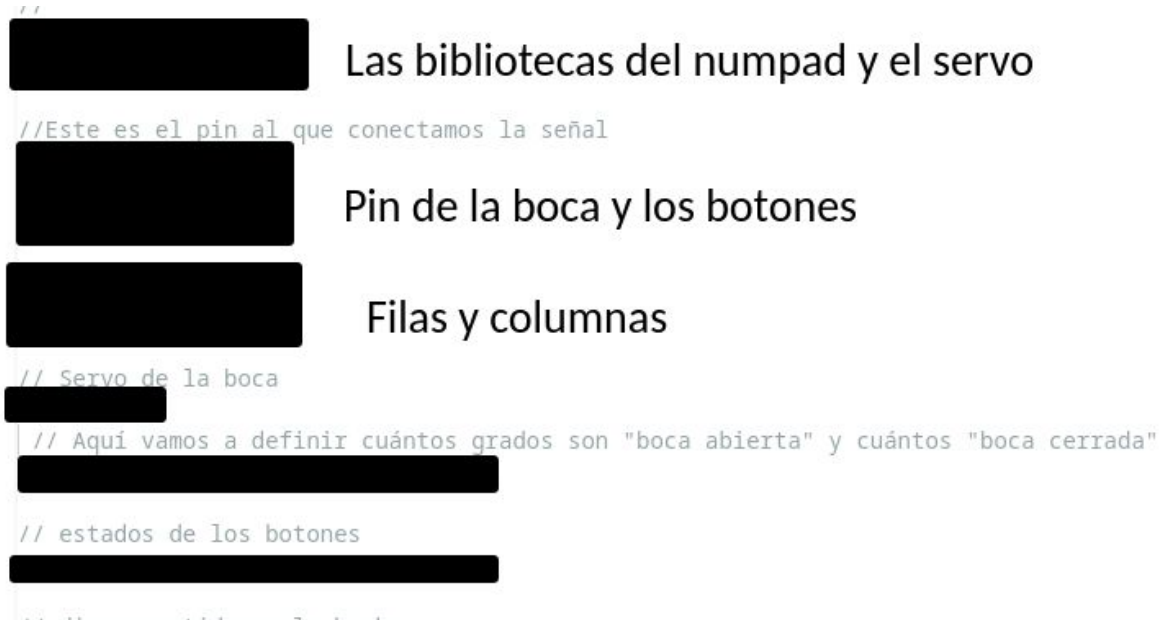


Revisad las fotos que habéis ido guardando para conectarlos, si tenéis problemas, ¡podéis revisar las diapositivas o preguntar!

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!



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!

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.

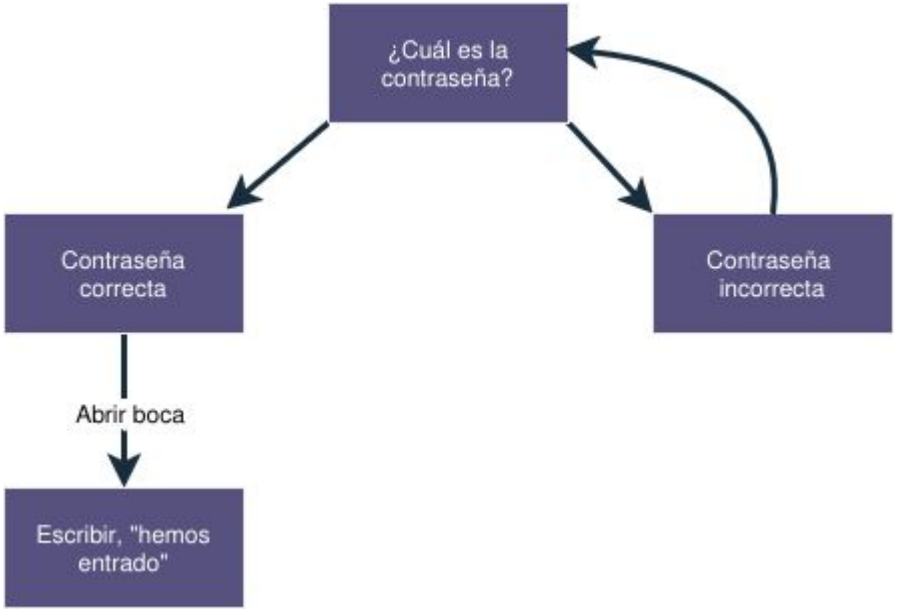
```
void setup()
{
  Serial.begin(9600);
  [redacted]  Ponemos el pin de la boca
  // definimos los grados (grados_abierto y grados_cerrado)
  [redacted]
// empezamos con la boca cerrada
  [redacted]
  // declaramos el pin del boton como INPUT (queremos que nos de información)
  //  y PULLUP (con una resistencia)
  [redacted]
  // estados iniciales de los botones
  [redacted]




  in_secreto.reserve(10); // máximo número de caracteres de la contraseña
}
```

Fase 3 - Código

- **Parte 1**: Preguntamos por la contraseña y comprobamos que está bien. Si está bien, abrimos la caja.

Para hacer esto, vamos a usar una variable booleana llamada “Secreto_correcto” Controlará si hemos puesto una contraseña correcta o incorrecta.

Por ahora, para hacer las cosas simples, vamos a seguir el siguiente diagrama:



Fase 3 - Código

Vamos a ver cómo preguntar cuál es la contraseña:

Paso 1: Declarar variable booleana “secreto_correcto” 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:


```
void loop()
{
  if(!Secreto_correcto){
    // recibimos tecla del numpad
    if (tecla){
      //si queremos ver la tecla que hemos pulsado
                          Tecla de borrar
      if(tecla == '[ ]') {
        // Limpiamos la contraseña escrita
                          Tecla de enviar palabra
      } else if(tecla == '[ ]') {
        if(secreto == in_secreto) {
          //escribir "La contraseña es correcta"


          //abrir la boca del cocodrilo
          // Guardar que hemos puesto bien la contraseña
        } else {
          //Escribir que hemos puesto mal la contraseña
        }
        // Limpiamos la contraseña escrita
      } else {
        // añadir caracter a contraseña escrita
      }
    }
  }
  else
  {
    //contraseña correcta, escribir "hemos llegado a la segunda parte"
  }
```

Fase 3 - Código

¿Qué nos debería salir?

Si ponemos la contraseña correcta: Escribir en pantalla un montón de veces “hemos llegado a la segunda parte” y abrir la boca

Si ponemos la contraseña incorrecta: Escribir “la contraseña es incorrecta, inténtalo de nuevo” una vez.

```
Hemos llegado a la segunda parte
Hemos llegado a la segunda parte
Hemos llegado a la segunda parte
Hemos llegado a la segunda parte
Hemos llegado a la segunda parte
Hemos llegado a la segunda pa
```

```
7
#
La contraseÃ±a es incorrecta, intÃ©ntalo de nuevo
```

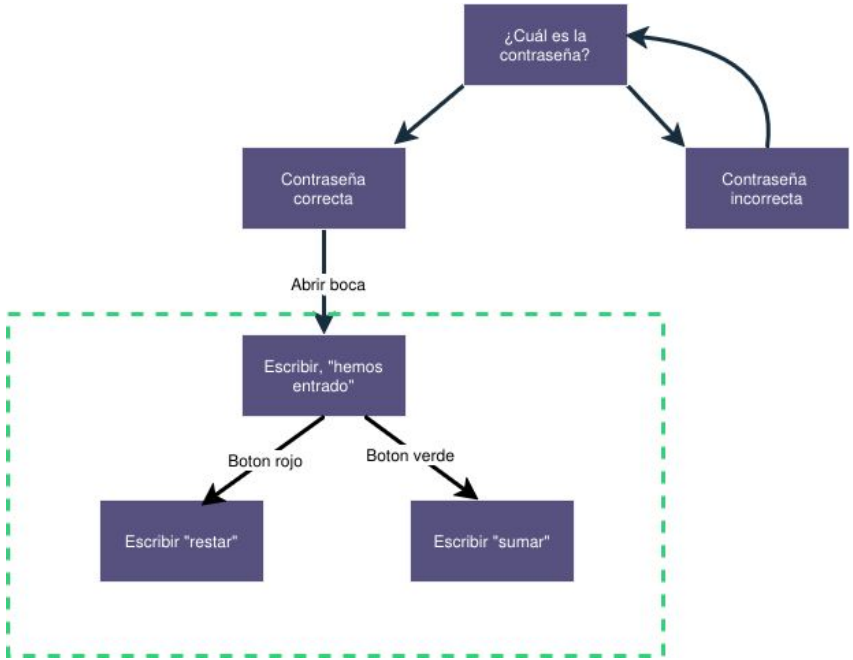
Si todo funciona, ¡¡Guardad el código en drive, haced fotos y vídeos, copia de seguridad!!

Fase 3 - Código

- **Parte 2:** Una vez abierta la caja, preguntamos si queremos sumar o restar dinero, con los botones.

En esta parte, vamos a empezar a usar los botones.

Queremos que, una vez puesta la contraseña correctamente, veamos si vamos a **sumar** o **restar**.



Fase 3 - Código

Lo primero es declarar dos nuevos booleanos junto a Secreto_correcto, llamados suma y resta. Tenemos que iniciarlos en setup a false.

```
// Secreto correcto -> hemos escrito bien la contraseña
//          y vamos a meter o sacar dinero
// suma -> cuando hemos dado al boton de sumar
// resta -> cuando hemos dado al boton de restar

                                        // pon tu contraseña aquí
                                      lo que escribimos con el numpad

//escribimos las teclas del teclado
```

```
void setup()
{
  Serial.begin(9600);
...
                                  suma y resta a false
  Secreto_correcto=false;

  in_secreto.reserve(10); // máximo número de caracteres de la contraseña
}
```

Fase 3 - Código

En el loop, cuando escribimos “la contraseña es correcta” y abrimos la boca, vamos a escribir también “¿quieres sumar o restar?

```
void loop()
{
  if(!Secreto_correcto){
    // recibimos tecla del numpad

    if (tecla){
      //si queremos ver la tecla que hemos pulsado

                          Tecla de borrar
     if(tecla == '[illegible]') {
       // Limpiamos la contraseña escrita
                                Tecla de enviar palabra
     } else if(tecla == '[illegible]') {
       if(secreto == in_secreto) {
         //escribir "La contraseña es correcta"


         //abrir la boca del cocodrilo

         // Guardar que hemos puesto bien la contraseña


       } else {
         //Escribir que hemos puesto mal la contraseña

       }
       // Limpiamos la contraseña escrita

     } else {
       // añadir caracter a contraseña escrita

     }
    }
  }
  else
  {
    //contraseña correcta, escribir "hemos llegado a la segunda parte"

  }
```

Fase 3 - Código

Ahora, en el else, que era donde entrábamos cuando Secreto_correcto era verdad y donde teníamos nuestro "*escribir, hemos llegado a la segunda parte*", vamos a **implementar el código de los botones** igual que en la Fase 2 y **vamos a quitar ese Print**.

A diferencia de la Fase 2, ya no vamos a mover un servo, sino que vamos a poner que suma y resta son verdad.

```
else
{
  //contraseña correcta, escribir "hemos llegado a la segunda parte"

  //leer nuevoEstado del botón A
  // si el estado A no es igual al nuevoEstado
  if(                    ){
    // Si nuevoestado es un 0 (pulsación)
    if (                 ){
      //suma es verdad
    }
    //estadoA es igual a nuevoestado
  }

  //leer nuevoEstado del botón B
  // si el estado A no es igual al nuevoEstado
  if(                    ){
    // Si nuevoestado es un 0 (pulsación)
    if (                 ){
      //resta es verdad
    }
    //estadoB es igual a nuevoestado
  }
  //si se ha pulsado el botón de suma (suma es true)
  if(    ){
    // suma y resta se ponen a false
    //se escribe por pantalla "vas a sumar"
  }
  // si se ha pulsado el botón de resta (resta es true)
  else if(    ){
    // suma y resta se ponen a false
    //se escribe por pantalla "vas a restar"
  }
}
```

Esto es muy parecido a vuestro código, miradlo para tener una referencia.

Fase 3 - Código

¿Qué nos debería salir?

Si ponemos la contraseña correcta: Escribir en pantalla “contraseña correcta” y “¿vas a sumar o restar?”

Si damos a uno de los botones después de eso: Escribir “vas a sumar” o “vas a restar” una vez.

```
/
A
#
La contraseÃ±a es correcta
Â¿Quieres aÃ±adir o quitar dinero?
```

Aquí he pulsado cada botón dos veces, alternando:

```
Vas a sumar
Vas a restar
Vas a sumar
Vas a restar
```

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

Fase 3 - Código

- **Parte 3:** Cuando sabemos si sumar o restar, vamos a preguntar la cantidad y la vamos a guardar. Luego, por último, vamos a cerrar la caja y volver a la fase 1.

Para esto, vamos a usar una **función.**

Una función es un código que escribimos aparte de nuestro loop() y que podemos llamar desde ahí. Nos sirve para no tener que repetir una y otra vez el mismo código y resumir las cosas.

```
int funcion(){
    codigo;
    …
}

void loop(){

    funcion();
    funcion();
    …
}
```

Fase 3 - Código

En realidad, es como si pusiéramos el código de la función en nuestro loop varias veces, es muy útil cuando tenemos código muy largo que se repite:

```
int funcion(){
    codigo;
    …
}

void loop(){

    funcion();
    funcion();
    …
}
```

```
int funcion(){
    codigo;
    …
}

void loop(){

    codigo;
    codigo;
    …
}
```

Fase 3 - Código

Las funciones tienen las siguientes partes:

```
int nombre(int valor)
{
    valor+=2;
    return valor;
}
```

Esta función recibe un valor entero, le suma 2 y lo devuelve.
El resultado de llamar variable =nombre(2); sería que en la variable habría un 4.

`int nombre ->` Esto nos indica que nuestra función se llama “nombre” y que devuelve un int. Es decir, que cuando llamemos a nuestra función, nos dará un número entero.
`int valor ->` Esto es un argumento. Los argumentos son valores que le damos a nuestras funciones para que trabajen. En este caso, esta función pide un valor entero.
`return valor ->`Después de calcular todo, devolvemos un valor del tipo de la función, en este caso int.

Fase 3 - Código

Vamos a implementar la siguiente función al principio de nuestro código:

Es muy parecido a lo que hacíamos para recibir la contraseña, pero esta vez nos vale cualquier tecla como si termináramos el número.

Donde haya huecos con comentarios, hay que rellenar código.

¿Has terminado la función? ¡Llámame y la revisaré para comprobar que está todo bien y pasar al siguiente paso!

```
void setup()
{
  ...
}

int recibir_dinero(){
  // creamos una variable bool llamada num_completo que empieza en false

  // creamos un string que almacenará nuestra cantidad

  // decimos que el tamaño máximo de la cantidad es 4 dígitos
  cantidad.reserve(4);
  // mientras el número no esté completo
  while (!num_completo){
    // recibimos tecla del numpad

    // si recibimos una tecla
    if (tecla){
      //Vemos la tecla que hemos pulsado en pantalla

      // si la tecla no es un número, es decir, es cualquiera de las otras teclas
      if(tecla == '*'|| tecla == '#' || tecla == 'A' || tecla == 'B' || tecla == 'C' || tecla == 'D') {
        // el número completo es true

      }
      //sino
      else {
        // añadir caracter a la cantidad

      }
    }
  }
  // convertir a int y devolver
  Serial.println(cantidad.toInt());
  return (cantidad.toInt());
}

void loop()
{
  ...
}
```

Fase 3 - Código

Necesitamos una nueva variable llamada “dinero”. La declararemos como un número entero al principio de nuestro código y la igualamos a 0 en el setup.

```
void setup()
{
...

  dinero = 0; // dinero inicial

...
}
```

```
// estados de los botones

// dinero metido en la hucha
int dinero;
// booleanos que se usan en la lógica
// fin de cantidad -> para escribir cuánto dinero tenemos
// Secreto correcto -> hemos escrito bien la contraseña
//          y vamos a meter o sacar dinero
// suma -> cuando hemos dado al boton de sumar
// resta -> cuando hemos dado al boton de restar
```

Fase 3 - Código

Ahora, necesitamos modificar este dinero con la función que hemos hecho y dentro de los botones de sumar y restar.

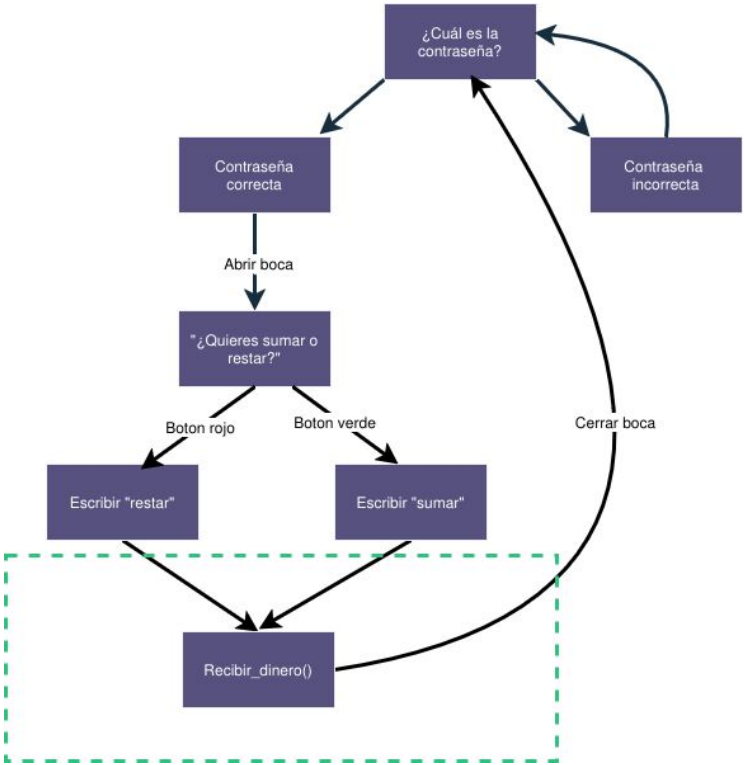
```
//si se ha pulsado el botón de suma (suma es true)
if(    ){
  // suma y resta se ponen a false

  //se escribe por pantalla "vas a sumar"

  // sumamos a dinero el resultado de llamar a la función recibir_dinero

}
// si se ha pulsado el botón de resta (resta es true)
else if(     ){
  // suma y resta se ponen a false

  //se escribe por pantalla "vas a restar"

  // restamos a dinero el resutlado de llamar a la función recibir_dinero

  // si el dinero es menor que 0, lo igualamos a 0,
  // porque no podemos tener dinero negativo en la hucha
  if(       )

}
```

Fase 3 - Código

Ahora, vamos a ver la última parte de nuestro código.

Vamos a **cerrar la boca y volver a empezar.**

Para esto, vamos a utilizar un último booleano llamado "`finDeCantidad`" que indicará si ya hemos recibido el número que modificará el dinero.



Fase 3 - Código

Vamos a añadirlo e iniciarlo a false en el setup

```
// booleanos que se usan en la lógica
// fin de cantidad -> para escribir cuánto dinero tenemos
// Secreto correcto -> hemos escrito bien la contraseña
//          y vamos a meter o sacar dinero
// suma -> cuando hemos dado al boton de sumar
// resta -> cuando hemos dado al boton de restar
bool finDeCantidad, Secreto_correcto, suma, resta;

suma = false;
resta = false;
Secreto_correcto=false;
finDeCantidad = false;
```

Fase 3 - Código

Por último, vamos a prepararlo todo para volver al inicio.

Vamos a decir que sea suma o resta, vamos a **entrar en "fin de cantidad"** una vez tengamos el dinero.

Vamos a **enseñar el dinero que tenemos** por pantalla y vamos a **devolver los booleanos a false**.

También, por supuesto, tenemos que **cerrar nuestra caja fuerte**

```
  //si se ha pulsado el botón de suma (suma es true)
  if(    ){
    // suma y resta se ponen a false

    //se escribe por pantalla "vas a sumar"

    // sumamos a dinero el resultado de llamar a la función recibir_dinero

    //fin de cantidad es verdad, vamos a volver al inicio

  }
  // si se ha pulsado el botón de resta (resta es true)
  else if(     ){
    // suma y resta se ponen a false

    //se escribe por pantalla "vas a restar"

    // restamos a dinero el resutlado de llamar a la función recibir_dinero

    // si el dinero es menor que 0, lo igualamos a 0,
    // porque no podemos tener dinero negativo en la hucha

    //fin de cantidad es verdad, vamos a volver al inicio

  }
// al final del else, después de if suma y resta:
// si fin De cantidad es verdad
if(          )
  {
    // Escribir "el dinero total es:"

    // escribir la variable dinero

    // cerrar la boca

    // secreto_correcto es falso (volvemos al inicio)

    // fin de cantidad es falso (ya hemos hecho todo para volver)

  }
}
```

Fase 3 - Código

¿Qué nos debería salir?

Si ponemos la contraseña correcta: Escribir en pantalla “contraseña correcta” y “¿vas a sumar o restar?”

Si damos a uno de los botones después de eso: Escribir “vas a sumar” o “vas a restar” una vez. Después, nos va a pedir cuánto dinero. Lo metemos con el numpad, nos enseñará el dinero total y se cerrará la caja. Además, si tratamos de restar más de lo que tenemos, nos dirá que tenemos 0

```
7
7
7
A
#
La contraseÃ±a es correcta
Â¿Quieres aÃ±adir o quitar dinero?
Vas a sumar
8
#
8
El dinero total es:8
```

```
El dinero total es:8
7
7
7
A
#
La contraseÃ±a es correcta
Â¿Quieres aÃ±adir o quitar dinero?
Vas a restar
9
#
9
El dinero total es:0
```

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

CÓDIGO

¡FASE SUPERADA!

Nuestra hucha 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?

¡¡Esta hucha 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: LCD y LEDs

- Aprenderemos a:
  - Usar una pantalla LCD con I2C
  - Utilizar leds RGB
  - Integrarlos en nuestro proyecto

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.

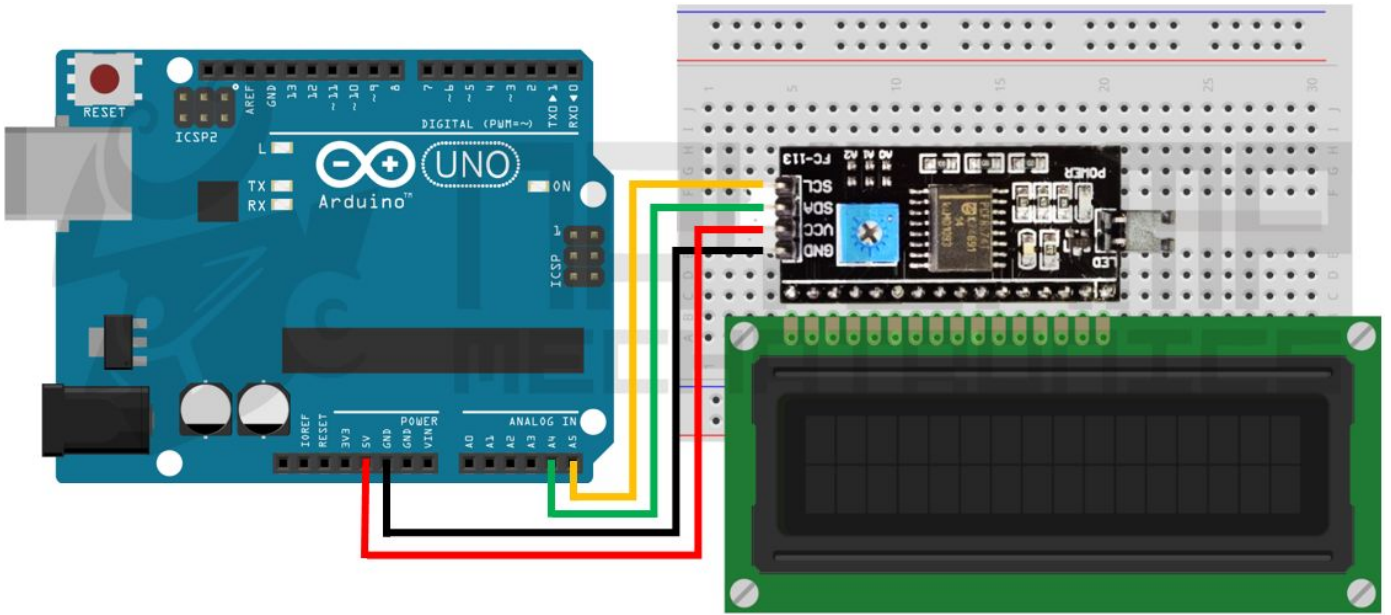
- Los dos cables utilizados son SCL, que se conecta al pin analógico A5, y SDA, que se conecta al pin analógico A4:

- 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

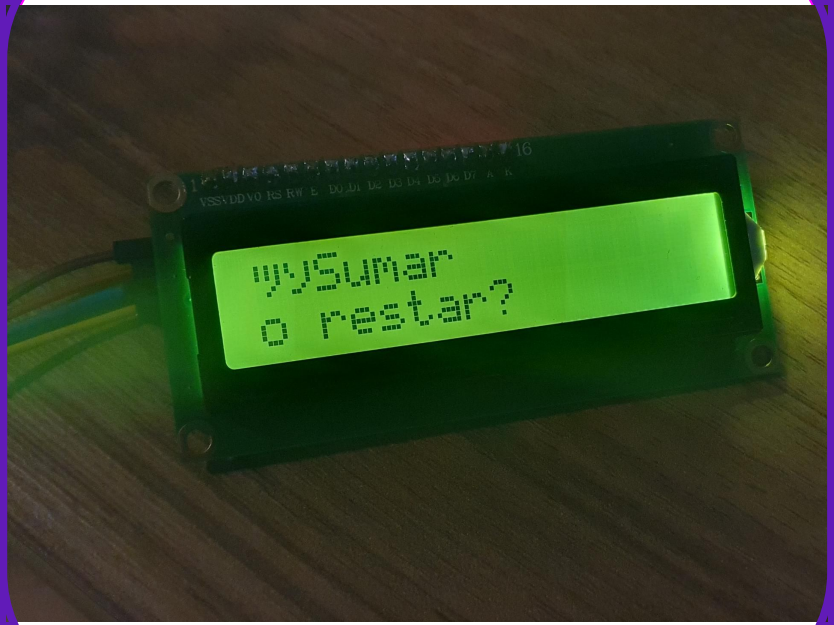


Pantalla LCD

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>**.

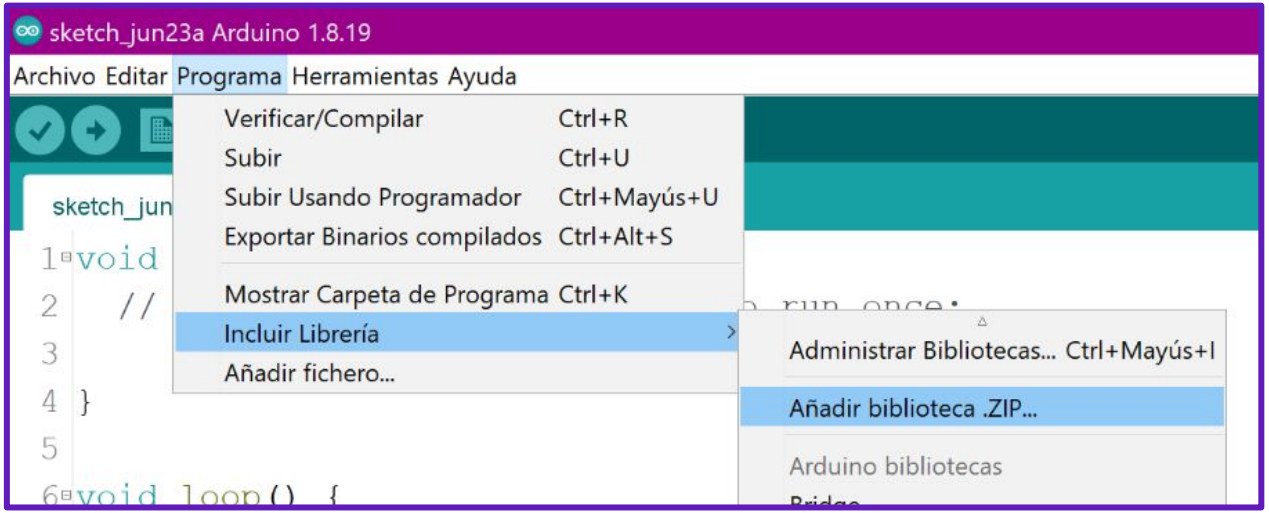
- Al trabajar con funciones de la librería descargada, a las que no les hemos puesto el nombre nosotras, tenemos que acostumbrarnos y entender lo que hacen.

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.**

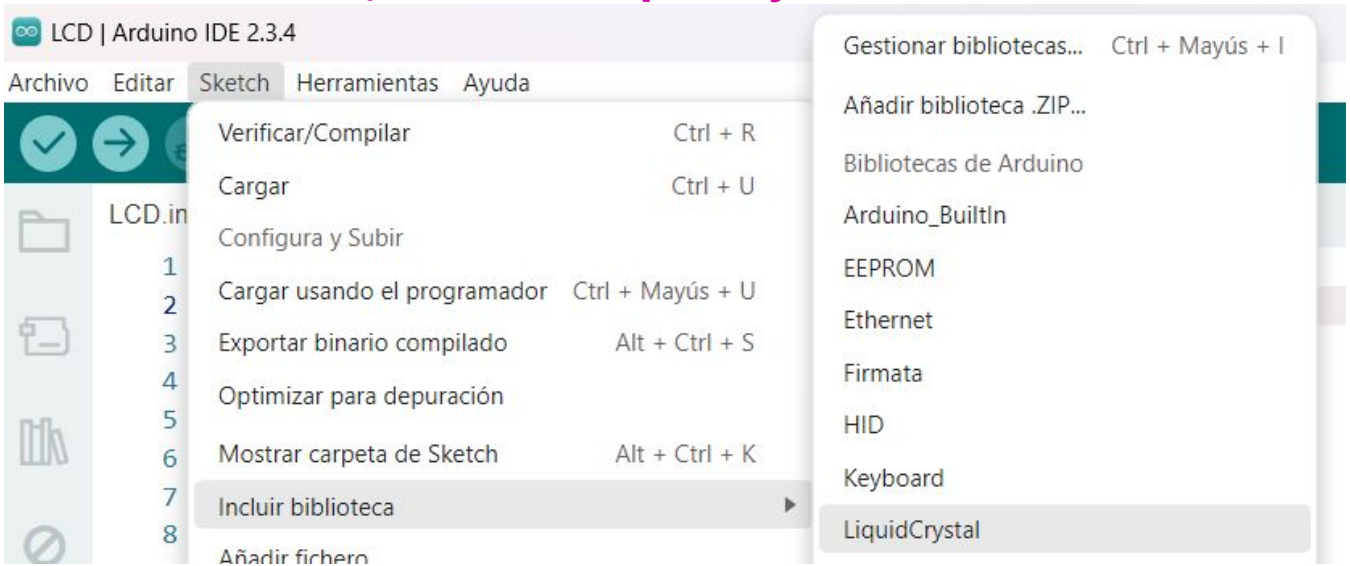


Pantalla LCD

1. Cargar librerías en el arduino:

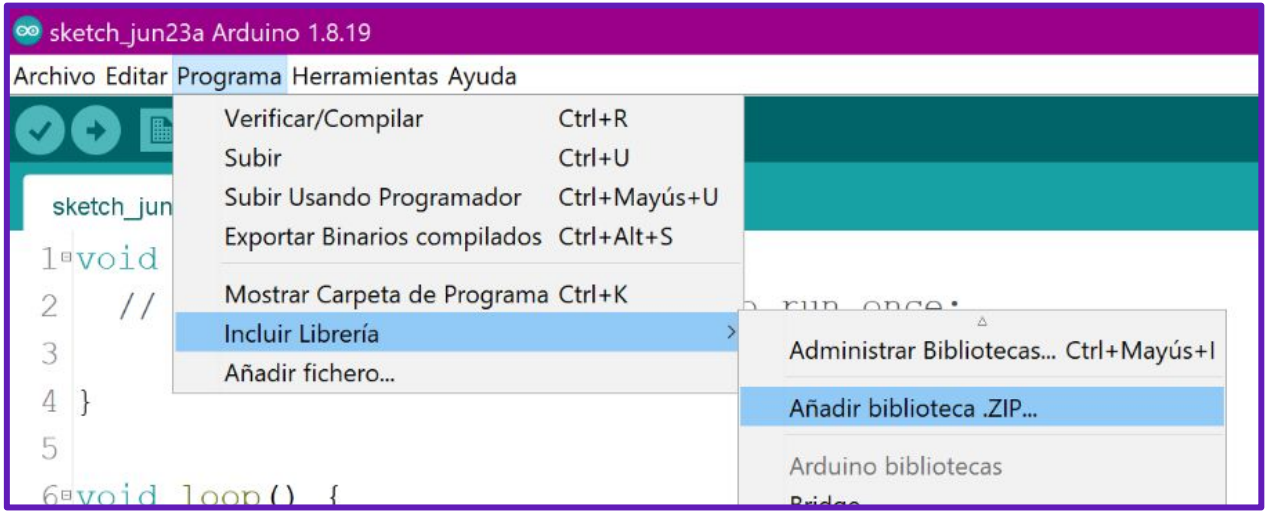
una vez añadida, para poder usarla hay que ir a:

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



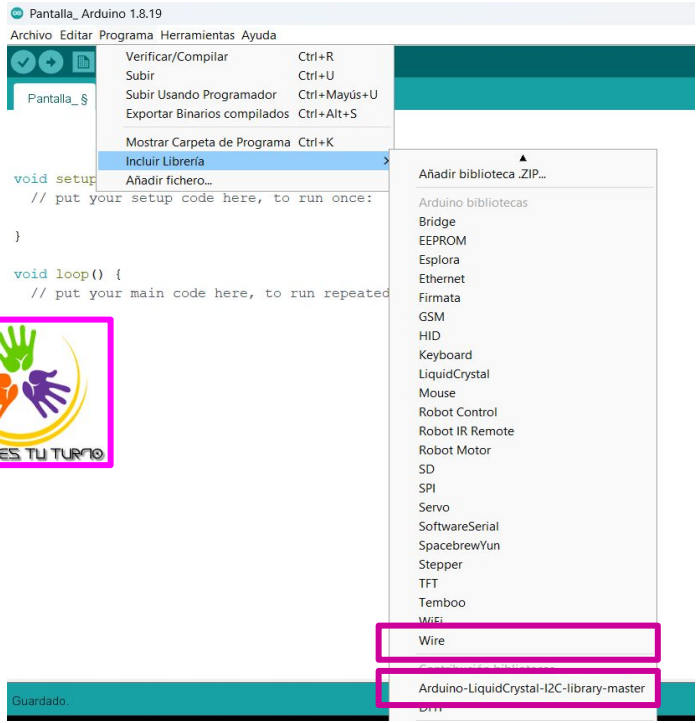
Pantalla LCD

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

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

¿Os acordáis de los vectores y matrices?

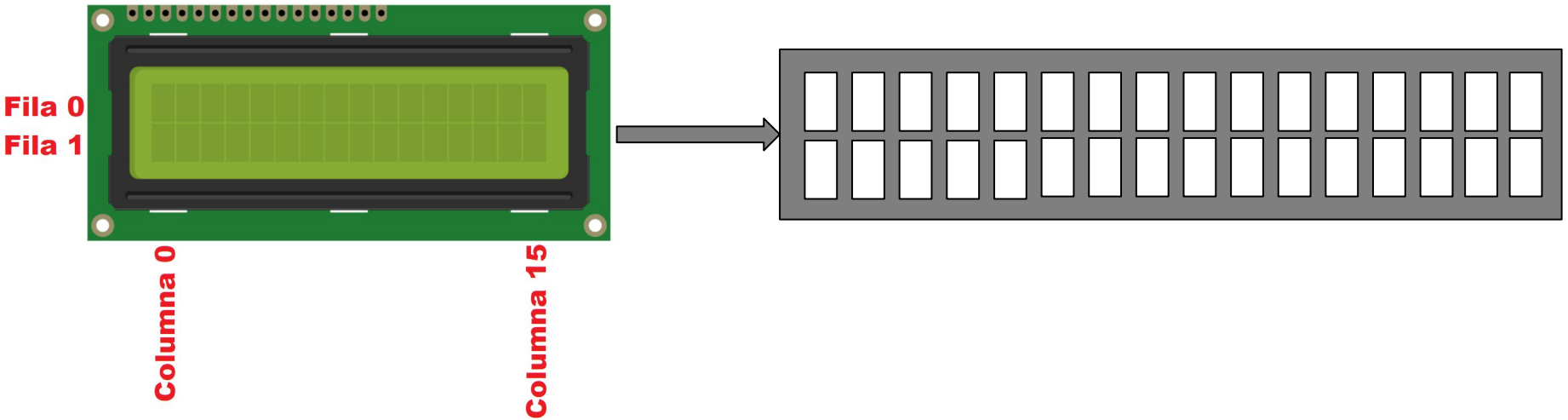
¿Creéis que podríais hacer que la pantalla muestre en la primera línea “Hola”?

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

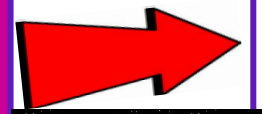


Pantalla LDC

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

- **lcd.begin()** 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 nuestra hucha.
- Lo primero será añadir lo necesario para que funcione (bibliotecas, declaración del lcd, setup…) Igual que los ejemplos anteriores.
- Vamos a crear una función que sea así:

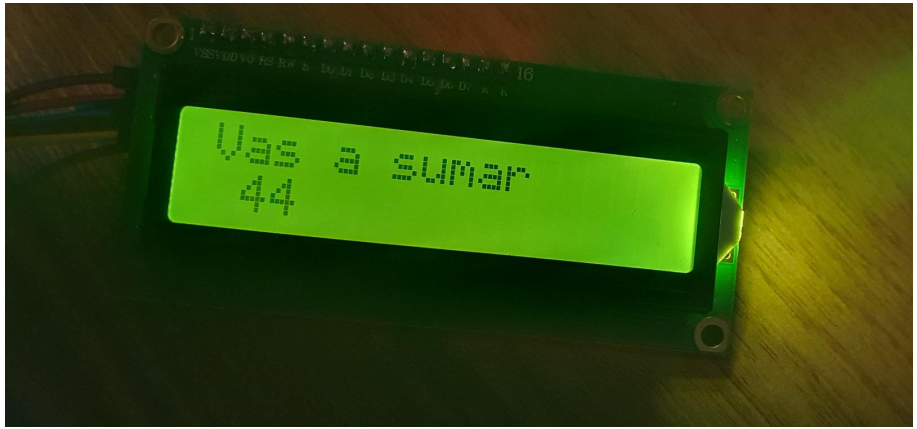
Por cada serial.println, podemos cambiarlo por un write del lcd o usar esta función, según lo que necesitemos, para ver todos nuestros mensajes por pantalla.

```
void escribir_lcd(String mensaje1, String mensaje2){
  // limpiar pantalla

  // poner cursor en posición 0,0

  // escribir primer mensaje

  // poner cursor en posicón 0 de la segunda fila

  // escribir segundo mensaje

}
```

Fase 4 - LCD

¿Qué nos debería salir?

Nuestra hucha funciona igual, pero toda la información la muestra en el lcd



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

Fase 4 - Led RGB

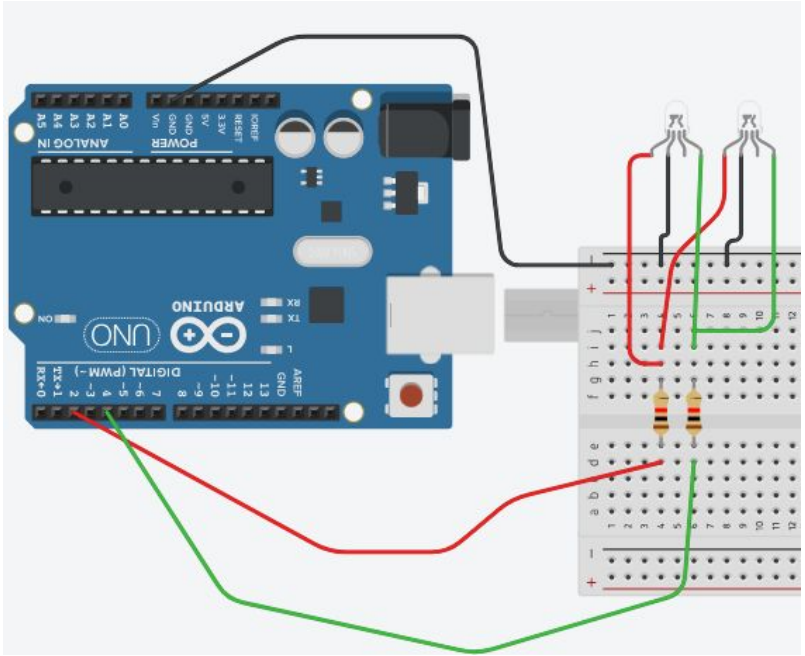
Vamos a añadir leds rgb para indicar que nos fue mal o bien poniendo la contraseña. Yo los utilicé para los ojos del cocodrilo.

Vamos a necesitar un pin por color que queramos mezclar, y el común va a ir a GND.
Necesitamos también **una resistencia** entre los colores y sus pines para que el led no se funda.

Fase 4 - Led RGB

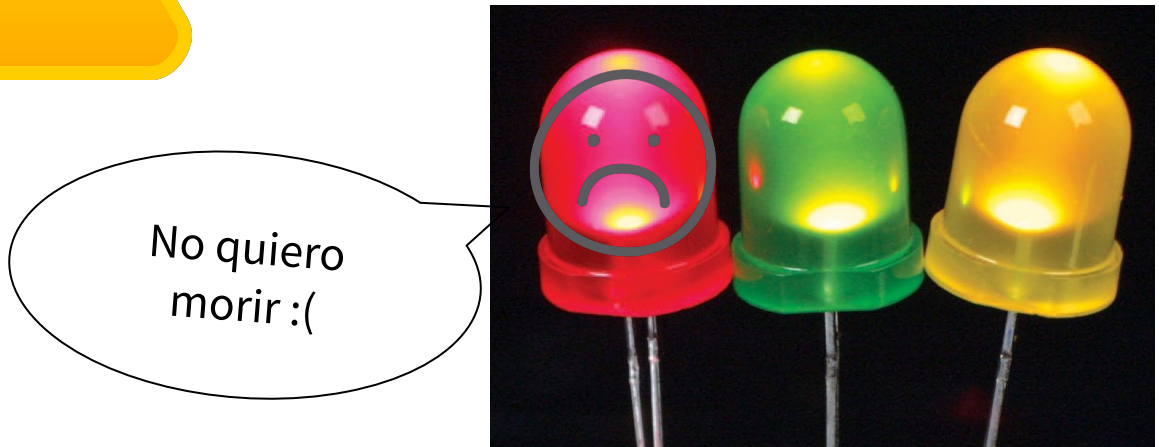
Para conectar un led, necesitamos **asociar los colores a los pines que queramos**, yo usé 4 para el verde y 2 para el rojo, porque los tenía libres y estaban cerca de la cabeza del cocodrilo.

Como podemos observar en este esquemático *súper claro y bien dibujado*, los pines rojos irán a través de una resistencia al 2, los pines verdes a través de una resistencia al 3 y **el pin largo** (el cátodo) **irá a GND,** así que no necesita una resistencia.



LED RGB

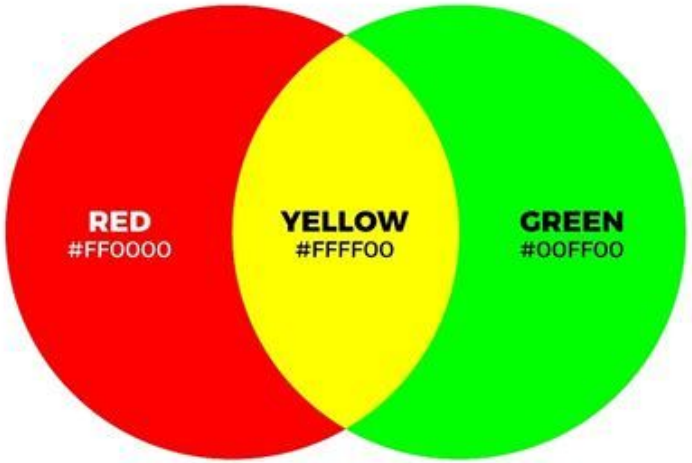
NO CONECTÉIS UN PIN DEL LED SIN RESISTENCIA AL ARDUINO Y PONGÁIS UN HIGH, PORQUE LO QUEMARÉIS.



Fase 4 - Led RGB

Solo con esta construcción y solo utilizando dos pines, podríamos pensar en conectar todos los leds rgb que queramos, pero en realidad, si pusiéramos muchos, no les llegaría la potencia que necesitan a todos y se verían apagados.

Solo con dos pines, podemos conseguir **Rojo**, **verde** y **amarillo**:



Fase 4 - Led RGB

Para añadir los leds, seguiremos los pasos que hemos aprendido durante el proyecto:

- **Definir el pin del led rojo y verde** (o azul, si lo preferís)
- **Declararlo como OUTPUT en el setup** (puesto que queremos mandarle información al led desde el arduino)
- Cuando queramos encender un color, usaremos **digitalWrite(PIN, HIGH);**, y eso mandará un 1 lógico al pin que queramos encender. Esperaremos un poco (delay(1000), por ejemplo) y lo volveremos a poner a 0 con **digitalWrite(PIN, LOW);**
- Escribiremos ese código cada vez que queramos encender los leds, por ejemplo, cuando la contraseña sea correcta.

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

Hucha avanzada

¡Fase superada!

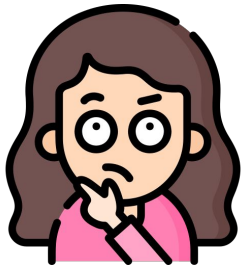
Este proyecto está súper avanzado, ¡enhorabuena!

De aquí en adelante hay algunas sugerencias de expansión, podéis elegir la que más os guste, o centraros en Decorar vuestro proyecto y hacer el vídeo.

¡Haced vídeos! **¡¡Guardadlo todo!!**

Tips Montaje

- Tiene que ser fácil pulsar los botones
  - Deben estar en su posición correcta
  - Cuando el usuario pulse, hará presión y tiene que ser estable
- Los LEDs se deben ver bien
- La pantalla se tiene que ver bien
- El cableado debe estar lo más ordenado posible
- El teclado numérico debe ser accesible y estable.



Fase 5: Hucha Extra

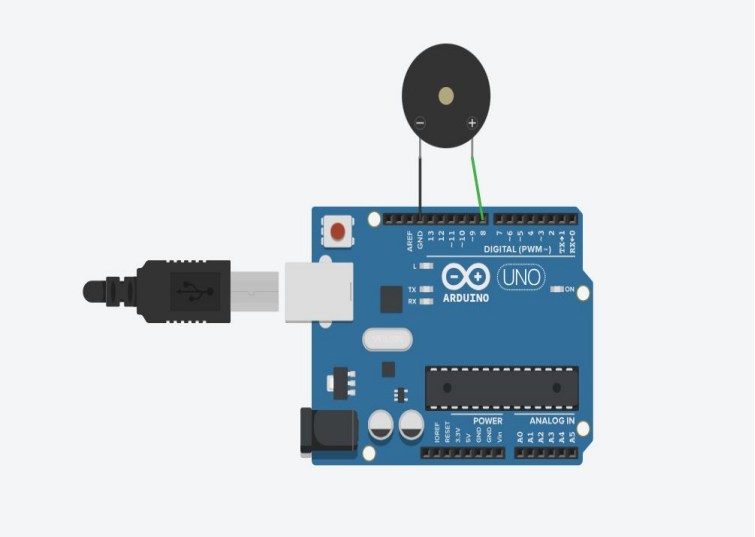
- Ofreceremos ampliaciones para el proyecto final.

Ampliación Buzzer

Añade un buzzer para emitir sonidos cuando la contraseña está bien o mal

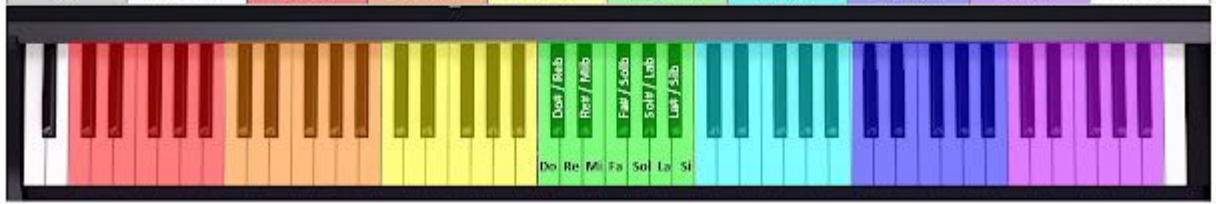
Buzzer

- Conexión
  - GND
  - Pin 5
- Para generar sonidos utilizamos la función tones() junto con un delay() para dar tiempo a que suene la melodía
- Para definir una melodía utilizamos las frecuencias. La frecuencia de sonido va desde 31 Hz a 65535 Hz. Cada nota musical se corresponde con una frecuencia



Buzzer

FRECUENCIA DE LAS NOTAS MUSICALES EN HERCIOS (Hz)									
	OCTAVA 0	OCTAVA 1	OCTAVA 2	OCTAVA 3	OCTAVA 4	OCTAVA 5	OCTAVA 6	OCTAVA 7	OCTAVA 8
Do	16,3516	32,7032	65,4064	130,813	261,626	523,251	1046,50	2093,00	4186,01
Do# / Reb	17,3239	34,6479	69,2957	138,591	277,183	554,365	1108,73	2217,46	4434,92
Re	18,3540	36,7081	73,4162	146,832	293,665	587,330	1174,66	2349,32	4698,64
Re# / Mib	19,4454	38,8909	77,7817	155,563	311,127	622,254	1244,51	2489,02	4978,04
Mi	20,6017	41,2035	82,4069	164,814	329,628	659,255	1318,51	2637,02	5274,04
Fa	21,8268	43,6536	87,3071	174,614	349,228	698,456	1396,91	2793,83	5587,66
Fa# / Solb	23,1246	46,2493	92,4986	184,997	369,994	739,989	1479,98	2959,96	5919,92
Sol	24,4997	48,9995	97,9989	195,998	391,995	783,991	1567,98	3135,96	6271,92
Sol# / Lab	25,9565	51,9130	103,826	207,652	415,305	830,609	1661,22	3322,44	6644,88
La	27,5000	55,0000	110,000	220,000	440,000	880,000	1760,00	3520,00	7040,00
La# / Sib	29,1353	58,2705	116,541	233,082	466,164	932,328	1864,66	3729,31	7458,62
Si	30,8677	61,7354	123,471	246,942	493,883	987,767	1975,53	3951,07	7902,14



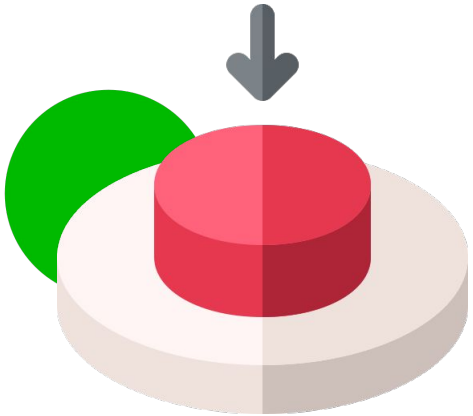
OCTAVA 1 OCTAVA 2 OCTAVA 3 OCTAVA 4 OCTAVA 5 OCTAVA 6 OCTAVA 7

Buzzer

1. En el setup
   - Añade el buzzer como salida
2. En el void loop
   - Añade un tono
   - Delay de 2 segundos
   - Parar el tono

Añade una melodía para cuando acierte la contraseña y otra cuando falle

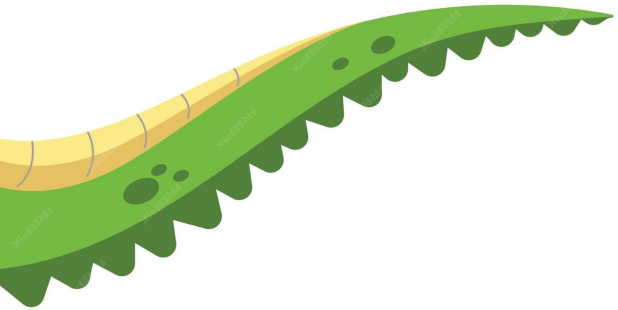
```
int pinBuzzer = 5;

void setup() {
pinMode (______,______)
}

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

  //detener tono
  noTone(pinBuzzer);

}
```



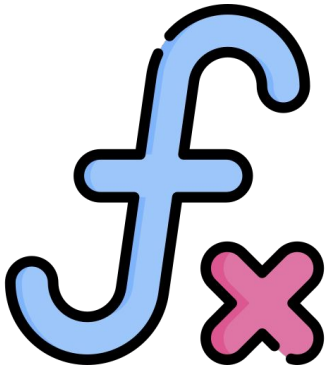
Ampliación

Añade un botón secreto, que cuando lo pulsas, se abre la hucha sin contraseña.

Ampliación



Añade un servomotor que sirva como cola, yo lo he añadido en el pin 11, se mueve cuando abres correctamente la hucha.

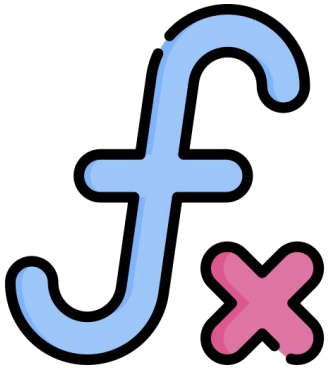


Ampliación

Crea funciones para mejorar el código

Función encender led()

- Pon pin en HIGH
- espera un segundo
- Pon el pin en LOW



Ampliación

Crea funciones para mejorar el código

Función mover_cola()
Mueve la cola a un lado, delay de medio segundo, y muévela al otro, varias veces.