**Icono

Descripción generada automáticamente**

Enrique Almazán Sánchez Víctor Miguel Álvarez Camarero

Project 1: Introduction to Arduino hardware and circuit assembly

Digital Systems and Microprocessors

Table of Contents

[Objective 1](#_Toc163405799)

[Description 1](#_Toc163405800)

[Requirements 1](#_Toc163405801)

[Exercise 1 2](#_Toc163405802)

[Exercise 2 3](#_Toc163405803)

[Exercise 3 5](#_Toc163405804)

[Exercise 4 8](#_Toc163405805)

[Flowcharts 10](#_Toc163405806)

## Objective

**The general objective of this project is to demonstrate the use and skill using the Arduino System, and the management of input and output ports, timers, 7-segment display, data matrix, etc.**

## Description

**Connect 6 LEDs to 6 pins of a digital input/output port on the Arduino development board. The LEDs must be arranged and mounted on the breadboard, in addition, cables and six (220 ohm) resistors that are included in the individual kit must be used to connect each LED to the corresponding pin on the digital input/output port.**

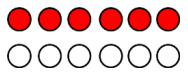
## Requirements

For this setup, we will require the following components:

* Arduino Mega 2560
* 6 LEDs
* 6 resistors with a resistance of 220Ω each
* Cables for connections
* Protoboard for circuit assembly

## Exercise 1

**Turn the 6 LEDs on and off every second as shown below.**



First, the code is presented, where the components of the circuit are defined as outputs using *void setup()*, as well as the pins to which they are connected. Moreover, a loop is created with *void loop()* as asked.

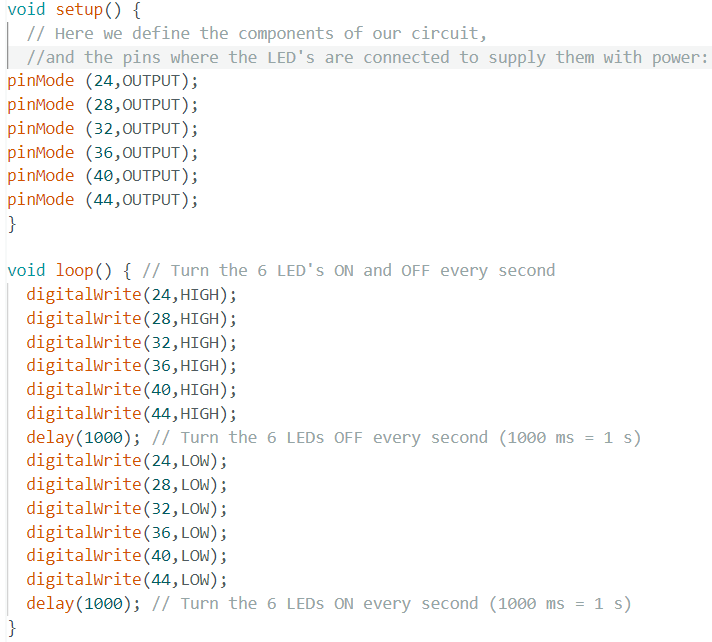


Figure 1: Exercise 1 code, for 6 LEDs.

Following with the circuit assembly with 6 LEDs turning on and off (*Figure 2*), with a delay of 1s (or 1000ms as seen in the code presented in *Figure 1* with *delay(1000)*).

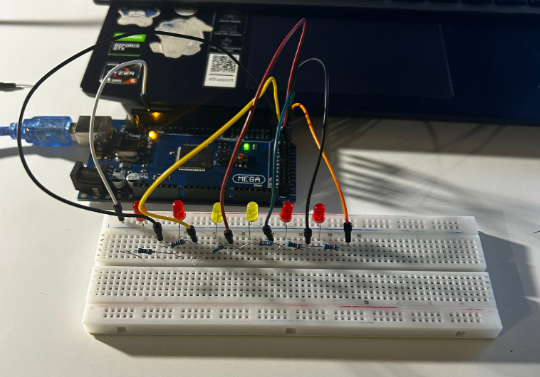
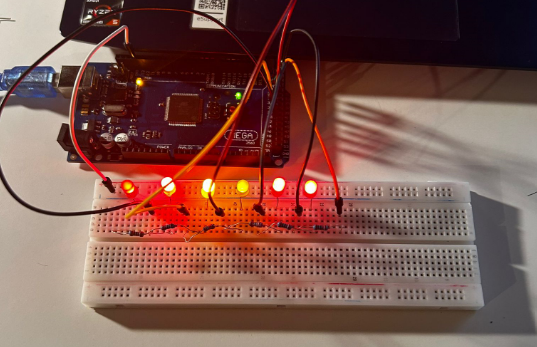
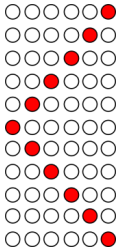


Figure 2: Implementation of the code on the circuit assembly, having 6 LEDs turning on (left) and off (right).

The implementation of the code can be seen on the video ‘Exercise\_1’.

## Exercise 2

**Perform the following sequence, repeating indefinitely. The delay between each sequence will be 200ms.**



The code is presented:

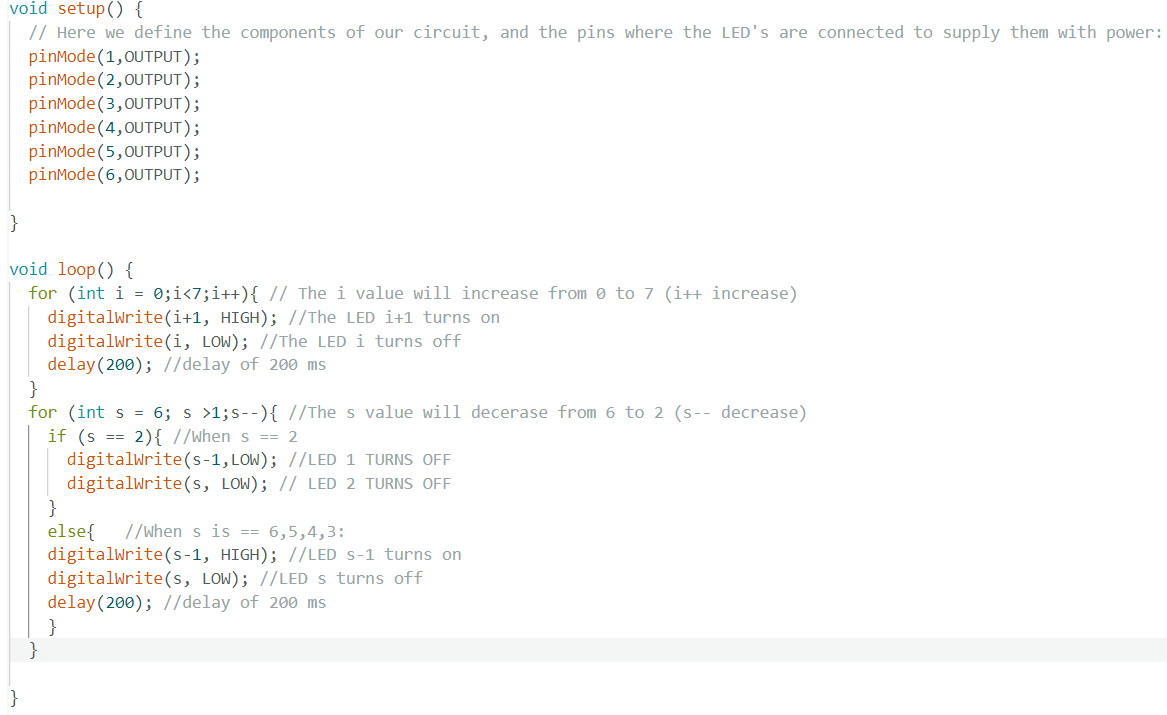
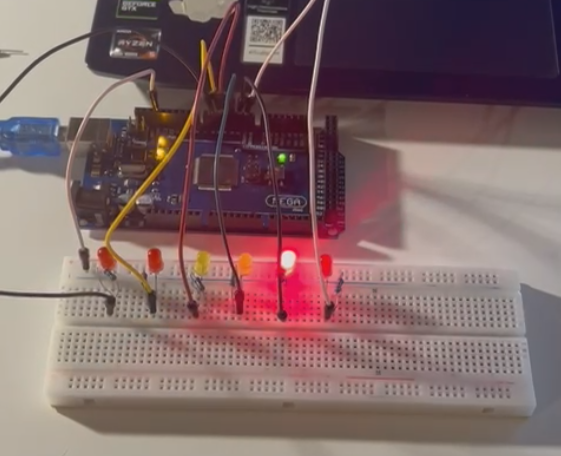
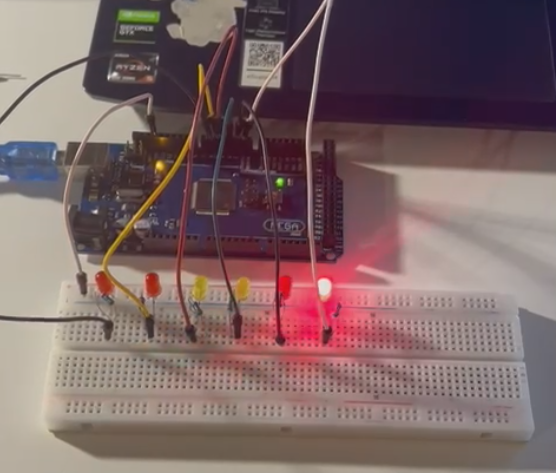


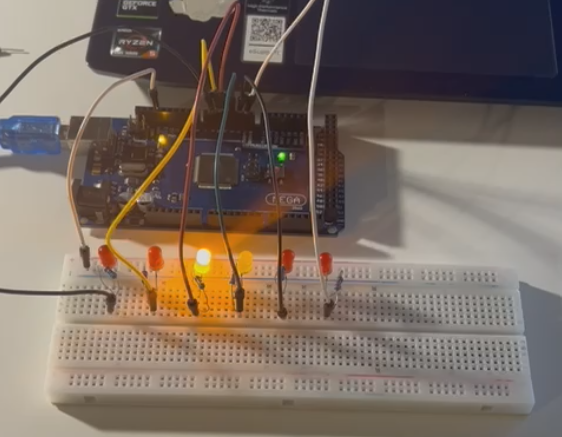
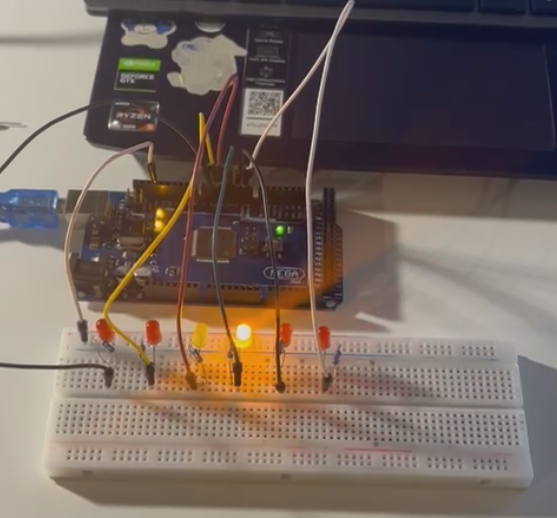
Figure 3: Exercise 2 code, for 6 LEDs with a cascade visual effect.

For the code presented above in the setup function, the code initializes the pins connected to the LEDs as output pins, as in the previous exercise. This prepares the microcontroller to send signals to those pins to control the LEDs.

On the other hand, in the loop function, there are two ‘for’ loops. The first loop turns on each LED in sequence from 1 to 6, while turning off the previous LED. It has a 200ms delay before moving to the next LED, having:

* i = 0 LED(1) turns on.
* i = 1 LED(2) turns on and LED(1) turns off.
* i = 2 LED(3) turns on and LED(2) turns off.
* i = 3 LED(4) turns on and LED(3) turns off.
* i = 4 LED(5) turns on and LED(4) turns off.
* i = 5 LED(6) turns on and LED(5) turns off. JUMPING





Un circuito electrónico

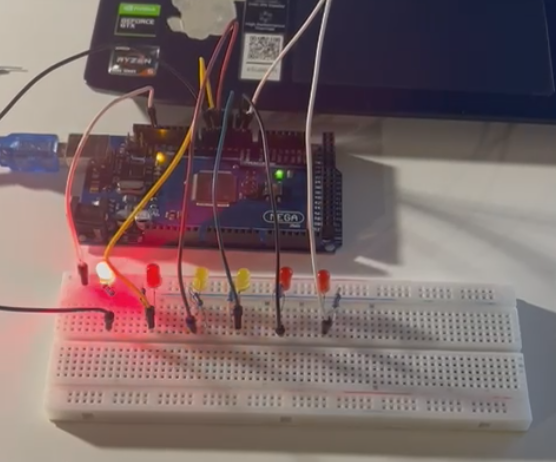
Descripción generada automáticamente con confianza baja 

Figure 4: Cascade visual effect of the circuit assembly.

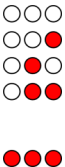
The second loop is activated with i = 6, turning off the LEDs in sequence from 6 to 2, while turning on the next LED (s-1), with a 200ms delay. It also has a condition to turn off the first two LEDs separately when the loop reaches LED 2 (s=2), jumping to the previous loop, in order to avoid having LED 1 on twice in this second cycle, having:.

* s = 6 LED(5) turns on and LED(6) turns off.
* s = 5 LED(4) turns on and LED(5) turns off.
* s = 4 LED(3) turns on and LED(4) turns off.
* s = 3 LED(2) turns on and LED(3) turns off.
* s = 2 LED(1) turns off and LED(2) turns off. JUMPING

Thus, the code creates a visual effect where the LEDs light up in a cascade pattern from one end to the other and then turn off in the reverse order. This implementation can be seen in the video ‘Exercise\_2’.

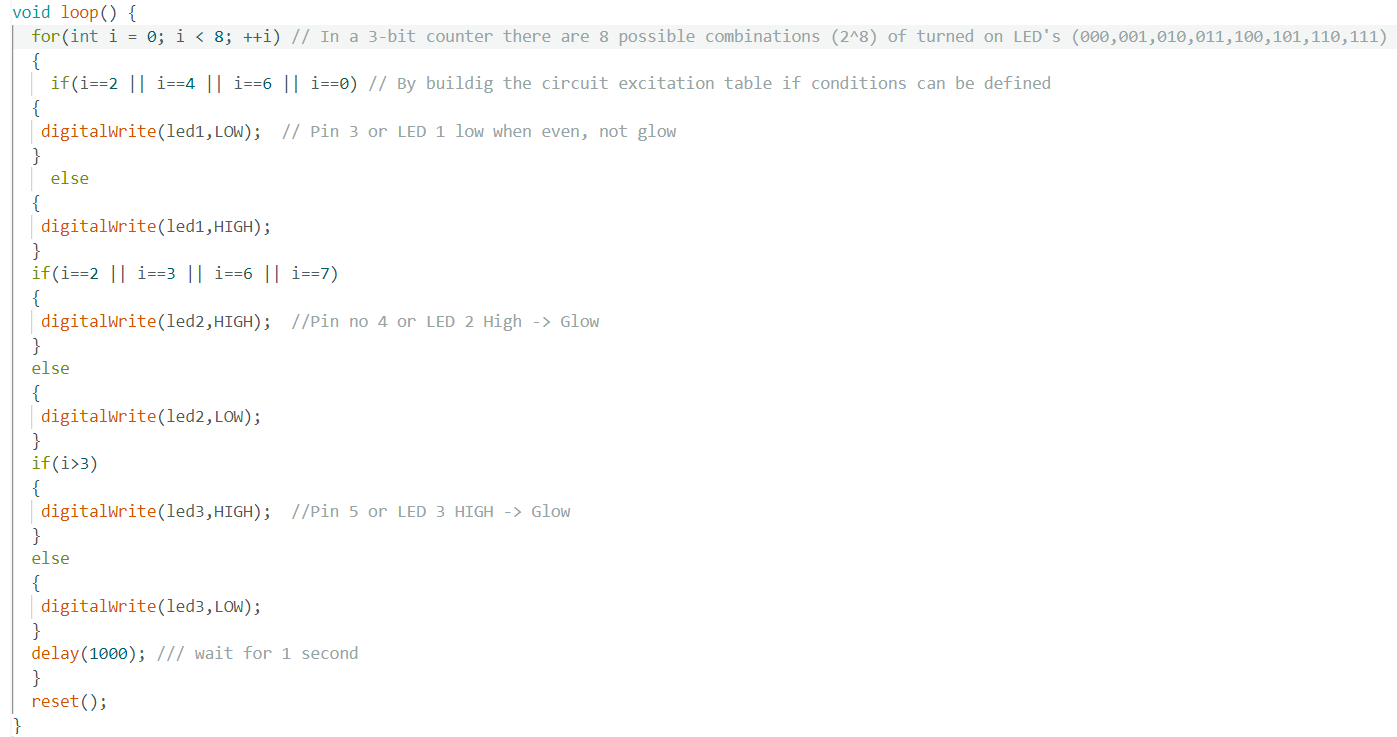
## Exercise 3

**With the 3 LEDs, simulate a 3-bit counter, which counts and restarts again. The delay between each binary code is 1s.**



First of all, the code is presented:





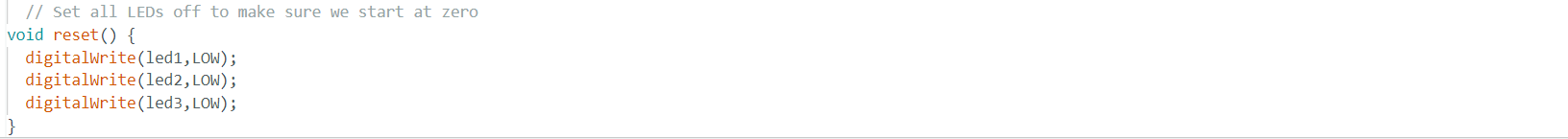


Figure 5: Exercise 3 code for 3-bit counter.

A 3-bit counter can be summarized in the following table:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **i** | **Button** | **Q0** | **Q1** | **Q2** | **Q(0+1)** | **Q(1+1)** | **Q(2+1)** |
| **1** | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| **2** | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| **3** | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| **4** | 1 | 0 | 1 | 1 | 1 | 0 | 0 |
| **5** | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| **6** | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| **7** | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| **8** | 1 | 1 | 1 | 1 | 0 | 0 | 0 |

It can be seen that the output of the 3-bit counter will follow the next pattern:

000 001 010 011 100 101 110 111 000

Regarding the code, first the variables representing the pins to which the LEDs are connected are defined. In the setup() function, it sets these pins as output pins, indicating that the Arduino will be sending signals to these pins to control the LEDs.

Furthermore, a ‘for’ loop is used to iterates through 8 states (0 to 7) representing all possible combinations of the 3-bit counter. Depending on the value of i, different LEDs are turned on or off according to the conditions defined in the comments. After each state, there's a delay of 1 second (delay(1000)) to hold the current state for a moment.

* LED 1 turns on when i takes odd values.
* LED 2 turns on when i = 2, 3, 6, 7.
* LED 3 turns on when i > 3.

Finally, the reset() function turns off all LEDs to ensure they start from a clean state after cycling through all combinations, obtaining.

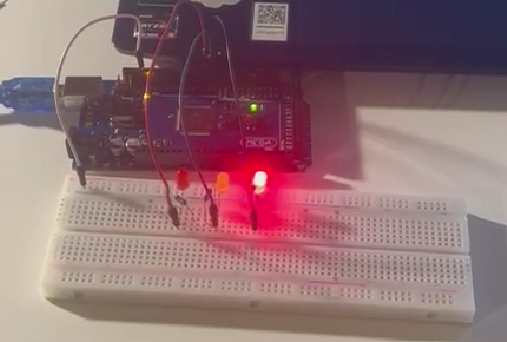


Figure 6: 001

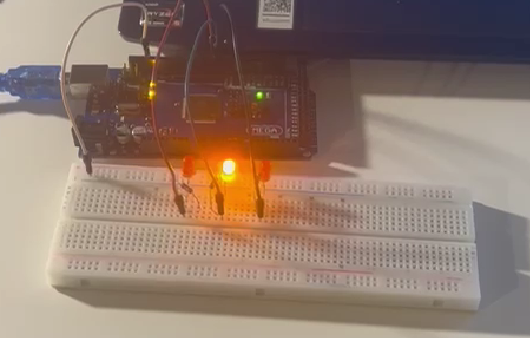


Figure 7: 010

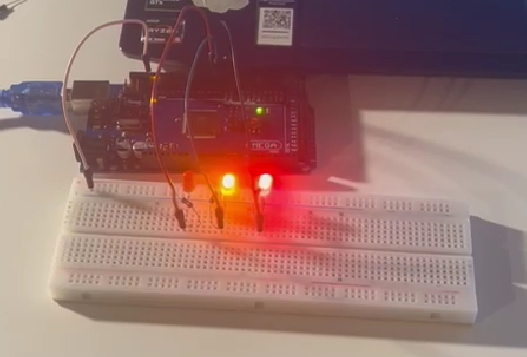


Figure 8: 011

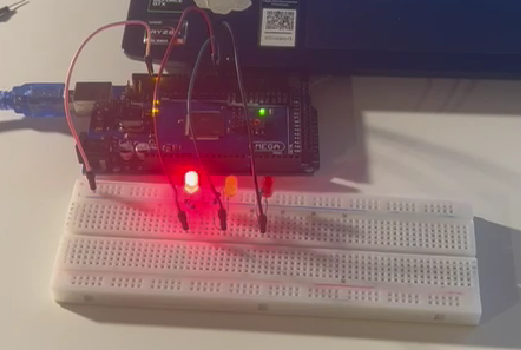


Figure 9: 100

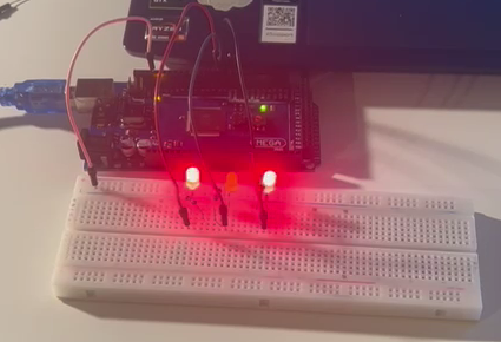


Figure 10: 101

Un control de video juego

Descripción generada automáticamente con confianza baja

Figure 11: 110

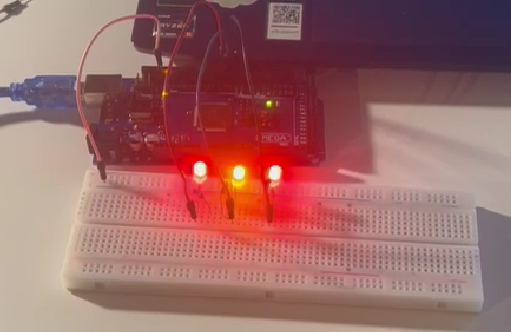
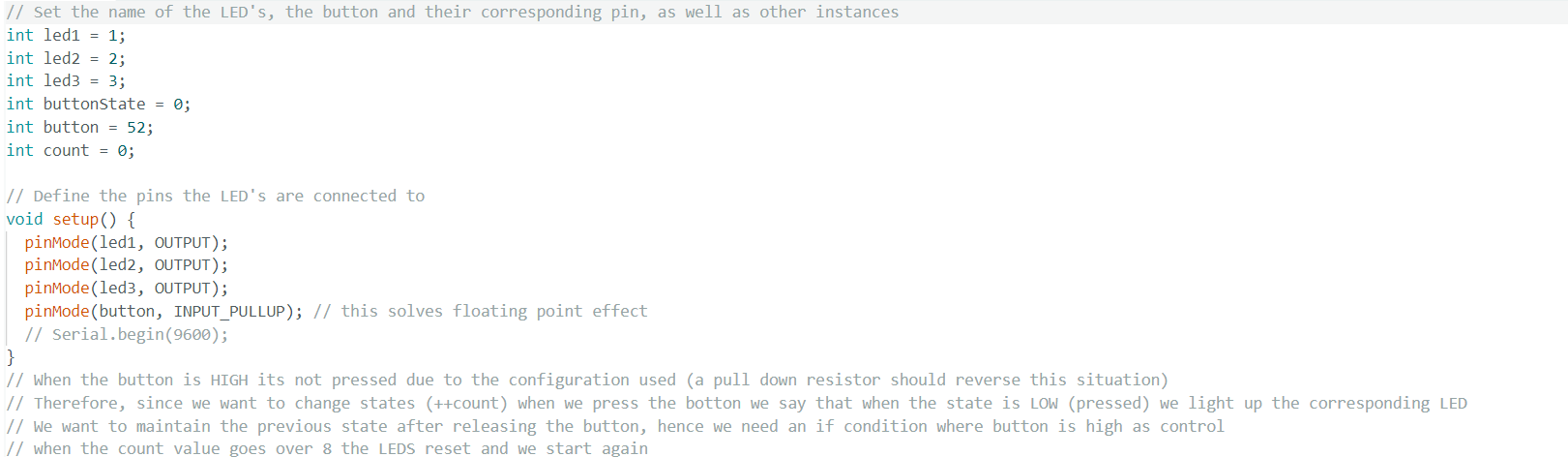


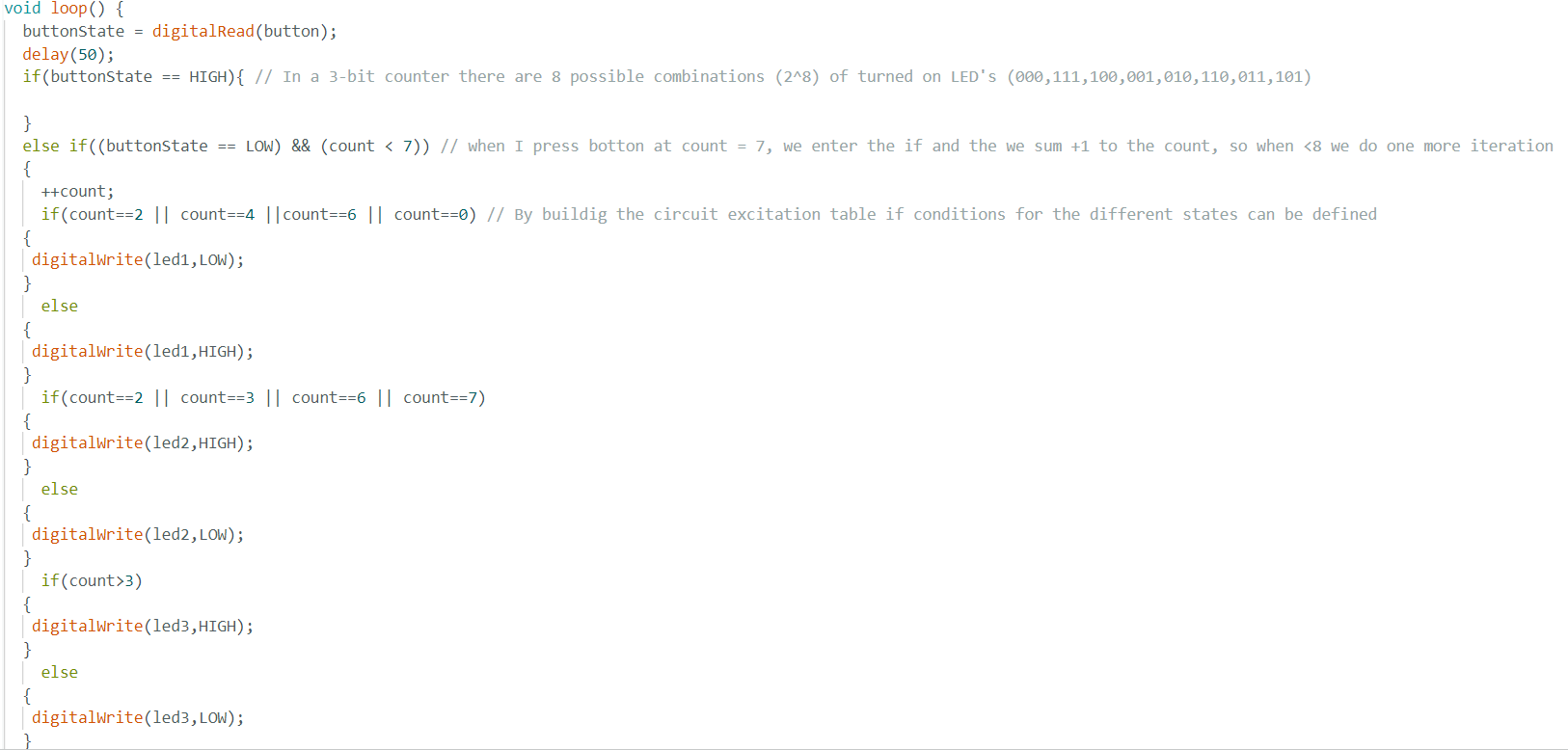
Figure 12: 111

## Exercise 4

**Using the previous point, include a push button in the system and have the counter increment the count each time you push the button. The program starts with the counter at position 000, and when it reaches count 111, the counter must be reset.**

First of all, the code is presented below:





Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

Figure 13: Exercise 4 code for 3-bit counter with a push button.

The only difference with the previous exercise is the adding of a push in order for the 3-bit counter state to change manually, as it can be seen in the video ‘Exercise\_4’. Thus, in the setup function, an instance for the state of the counter is set, as well as for the button is included, as it will change every time it is pressed.

In the loop function, the button functionality is handled, reading its state. If the button is not pressed (buttonState == HIGH), it does nothing and waits for the button to be pressed. If the button is pressed (buttonState == LOW), it increments the count variable to change the LED states accordingly.

The reset function is called when the count takes values over 8. It resets the count to 0 and turns off all the LEDs, providing a visual indication that the counter has reset.

It is also worth mentioning that the input pin of the microcontroller might not read the input logic value correctly due to floating pin issues caused by electromagnetic interference. To address this, a pull-up resistor is needed to ensure a stable high voltage state when the button is not pressed. This can be achieved physically by connecting a 10K Ohm resistor to 5 volts and connecting the other side of the button to ground. Alternatively, it can be implemented digitally through the Arduino code using the `INPUT\_PULLUP` mode for the pin connected to the button. When using `INPUT\_PULLUP`, the microcontroller's internal pull-up resistor is activated, ensuring stable high voltage when the button is not pressed. This configuration eliminates the need for an external pull-up resistor.

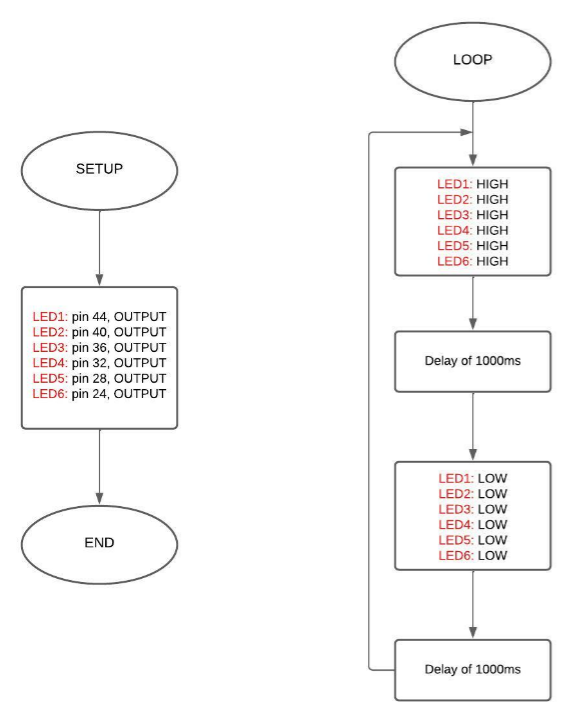
Imagen que contiene interior, electrónica, tabla, sostener

Descripción generada automáticamente

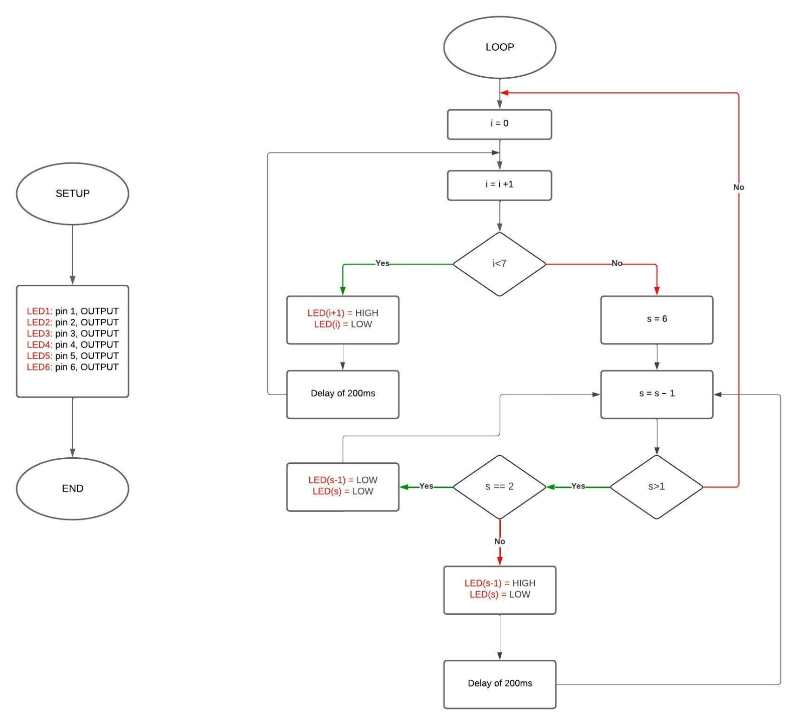
Figure 14: Circuit assembly representing a 3-bit counter with a push button.

## Flowcharts

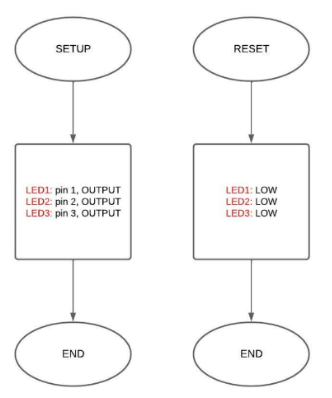
**Exercise 1**

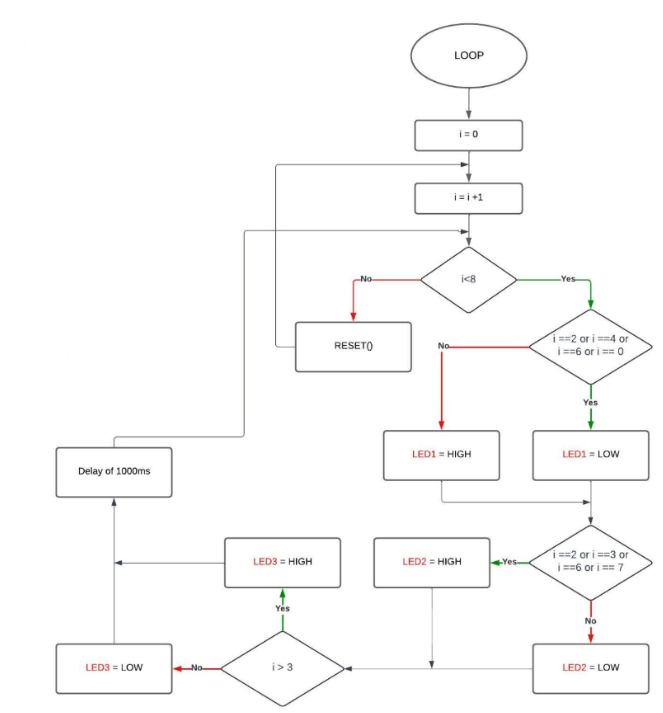
****

**Exercise 2**

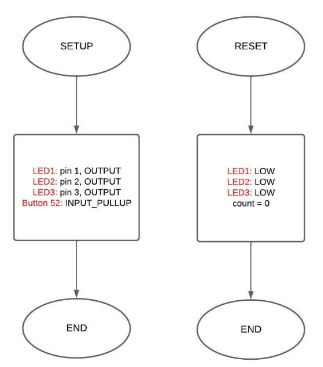
****

**Exercise 3**

****

****

**Exercise 4**

****

**Diagrama

Descripción generada automáticamente**