Digital Systems and Microprocessors

Project 1: Introduction to Arduino hardware and circuit assembly

Enrique Almazán Sánchez
Víctor Miguel Álvarez Camarero

Table of Contents

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.

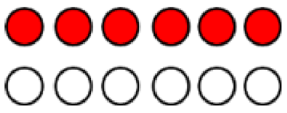
Requirements

For this setup, we will require the following components:

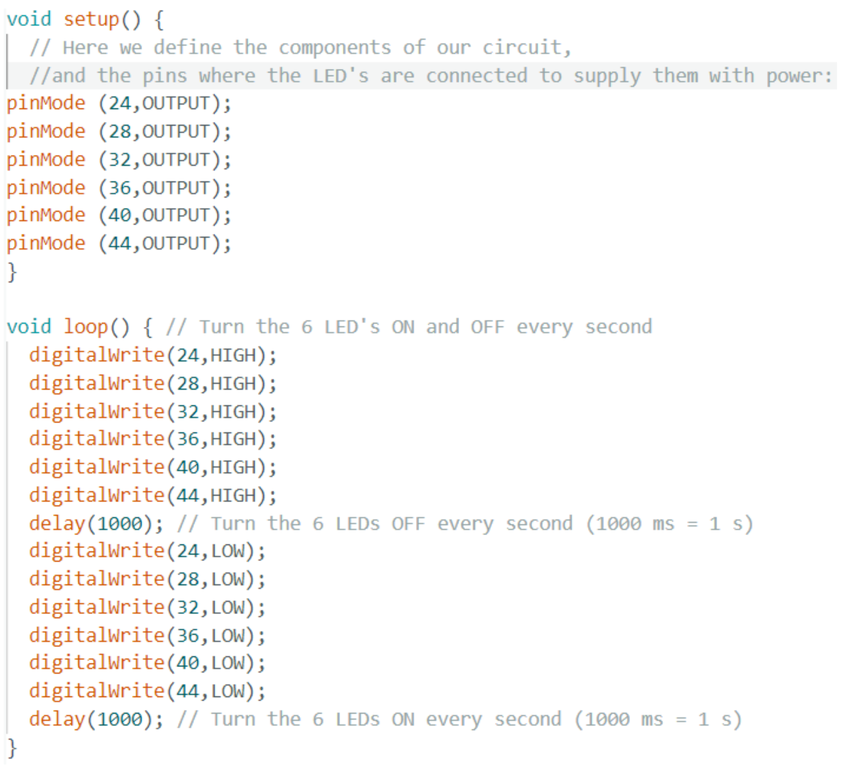
- Arduino Mega 2560
- 6 LEDs
- 6 resistors with a resistance of 220Ω each
- Cables for connections
- Protoboard for circuit assembly

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.



```
void setup() {
  // Here we define the components of our circuit,
  //and the pins where the LED's are connected to supply them with power:
pinMode (24,OUTPUT);
pinMode (28,OUTPUT);
pinMode (32,OUTPUT);
pinMode (36,OUTPUT);
pinMode (40,OUTPUT);
pinMode (44,OUTPUT);
}

void loop() { // Turn the 6 LED's ON and OFF every second
  digitalWrite(24,HIGH);
  digitalWrite(28,HIGH);
  digitalWrite(32,HIGH);
  digitalWrite(36,HIGH);
  digitalWrite(40,HIGH);
  digitalWrite(44,HIGH);
  delay(1000); // Turn the 6 LEDs OFF every second (1000 ms = 1 s)
  digitalWrite(24,LOW);
  digitalWrite(28,LOW);
  digitalWrite(32,LOW);
  digitalWrite(36,LOW);
  digitalWrite(40,LOW);
  digitalWrite(44,LOW);
  delay(1000); // Turn the 6 LEDs ON every second (1000 ms = 1 s)
}
```

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)*).

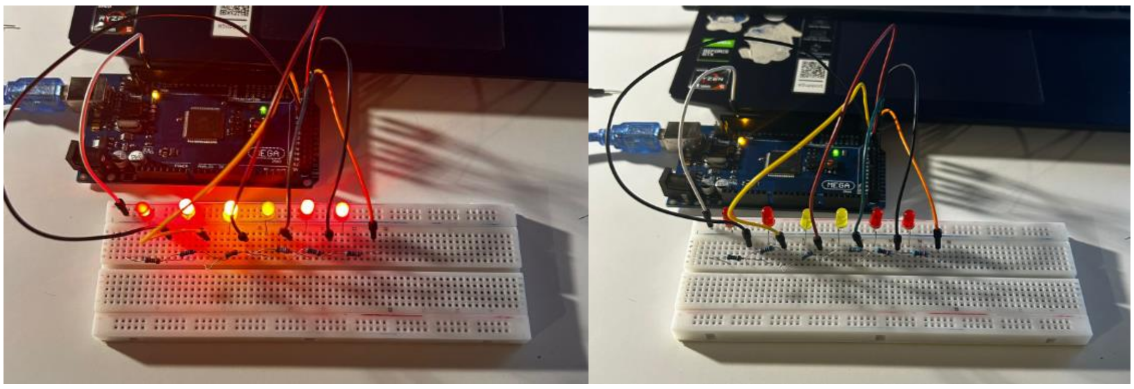
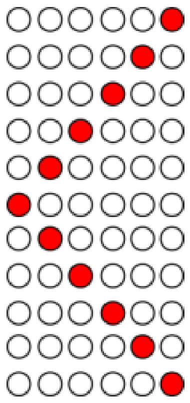


Figure 2: Implementation of the code on the circuit assembly, having 6 LEDs turning on (left) and off (right).

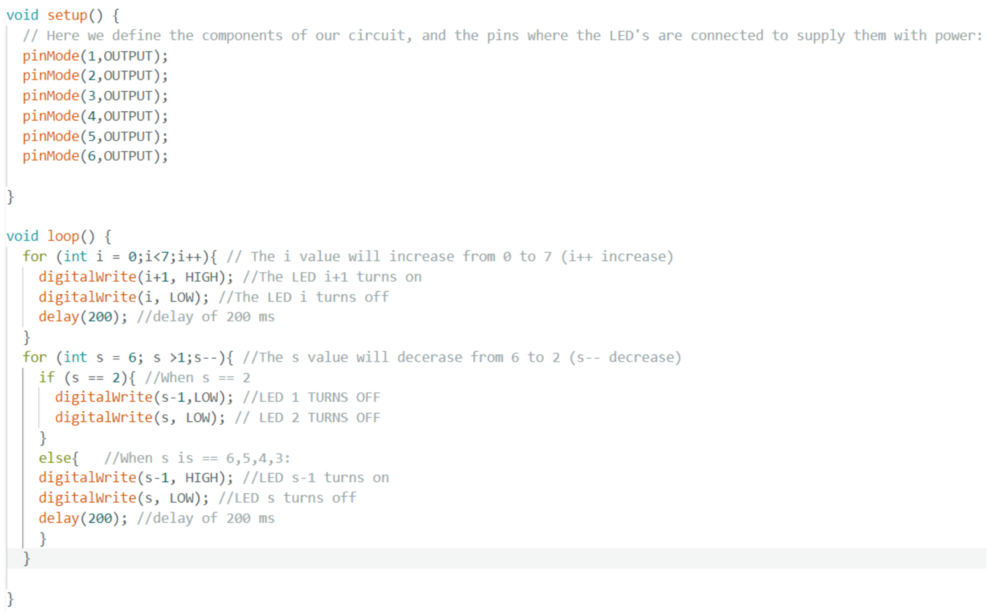
The implementation of the code can be seen on the video 'Exercise_1'.

Exercise 2

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



The code is presented:



```
void setup() {
  // Here we define the components of our circuit, and the pins where the LED's are connected to supply them with power:
  pinMode(1,OUTPUT);
  pinMode(2,OUTPUT);
  pinMode(3,OUTPUT);
  pinMode(4,OUTPUT);
  pinMode(5,OUTPUT);
  pinMode(6,OUTPUT);

}

void loop() {
  for (int i = 0;i<7;i++){ // The i value will increase from 0 to 7 (i++ increase)
    digitalWrite(i+1, HIGH); //The LED i+1 turns on
    digitalWrite(i, LOW); //The LED i turns off
    delay(200); //delay of 200 ms
  }
  for (int s = 6; s >1;s--){ //The s value will decerase from 6 to 2 (s-- decrease)
    if (s == 2){ //When s == 2
      digitalWrite(s-1,LOW); //LED 1 TURNS OFF
      digitalWrite(s, LOW); // LED 2 TURNS OFF
    }
    else{   //When s is == 6,5,4,3:
    digitalWrite(s-1, HIGH); //LED s-1 turns on
    digitalWrite(s, LOW); //LED s turns off
    delay(200); //delay of 200 ms
    }
  }

}
```

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

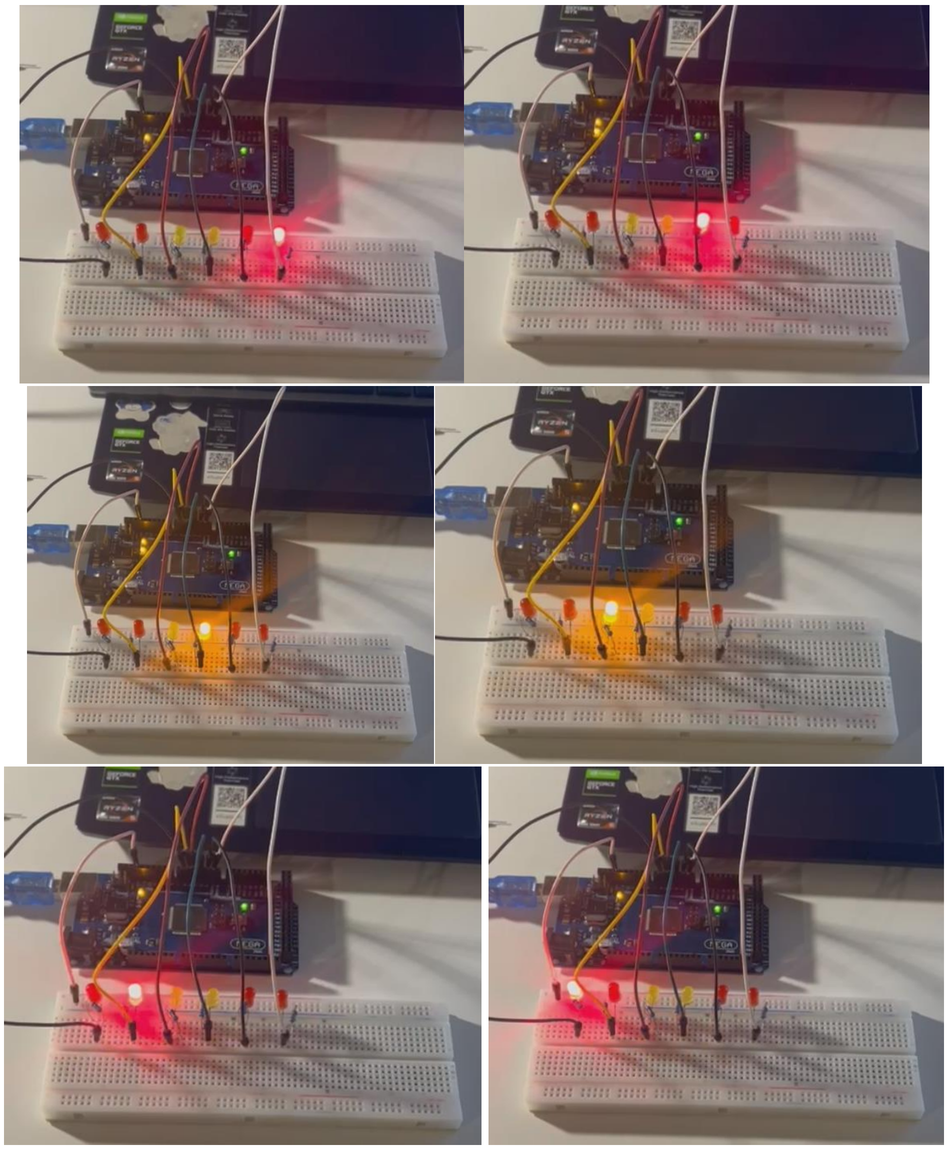


Figure 4: Cascade visual effect of the circuit assembly.

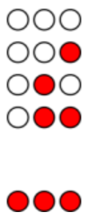
The second loop is activated with i = 6, turning off the LEDs in sequence from 6 to 2, while turning on the next LED (s-1), with a 200ms delay. It also has a condition to turn off the first two LEDs separately when the loop reaches LED 2 (s=2), jumping to the previous loop, in order to avoid having LED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

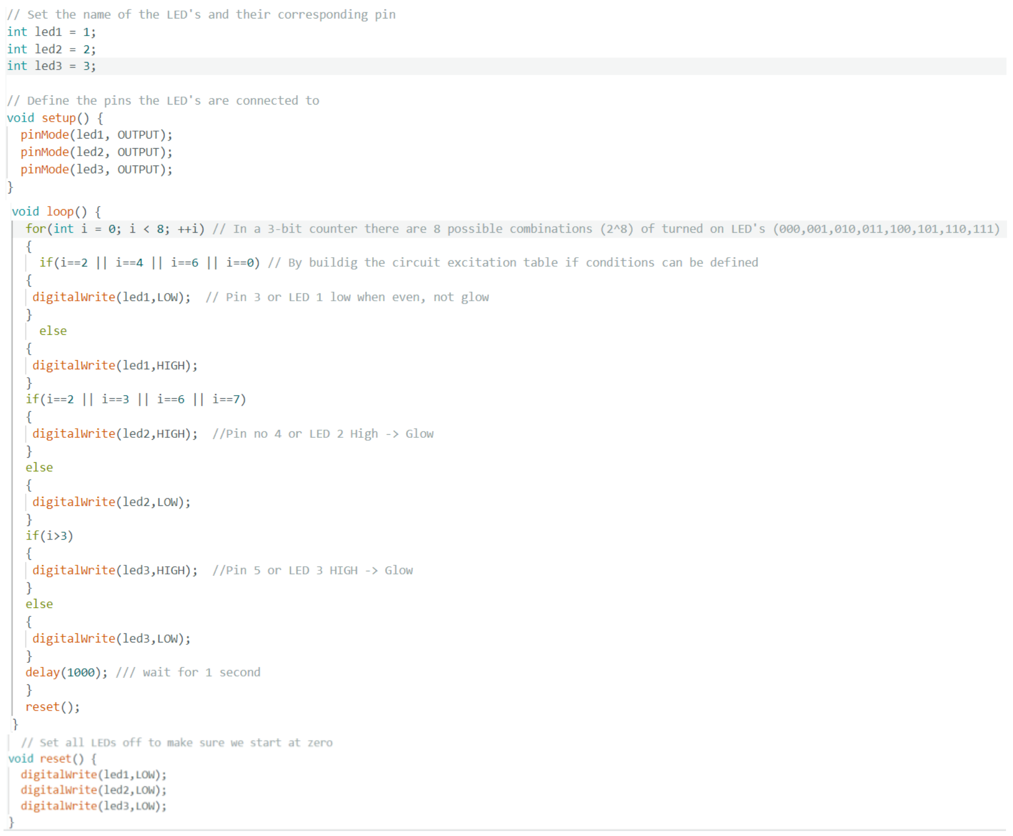
Thus, the code creates a visual effect where the LEDs light up in a cascade pattern from one end to the other and then turn off in the reverse order. This implementation can be seen in the video ‘Exercise_2’.

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:



```
// Set the name of the LED's and their corresponding pin
int led1 = 1;
int led2 = 2;
int led3 = 3;

// Define the pins the LED's are connected to
void setup() {
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(led3, OUTPUT);
}
 void loop() {
   for(int i = 0; i < 8; ++i) // In a 3-bit counter there are 8 possible combinations (2^8) of turned on LED's (000,001,010,011,100,101,110,111)
   {
     if(i==2 || i==4 || i==6 || i==0) // By buildig the circuit excitation table if conditions can be defined
   {
    digitalWrite(led1,LOW);  // Pin 3 or LED 1 low when even, not glow
   }
     else
   {
    digitalWrite(led1,HIGH);
   }
   if(i==2 || i==3 || i==6 || i==7)
   {
    digitalWrite(led2,HIGH);  //Pin no 4 or LED 2 High -> Glow
   }
   else
   {
    digitalWrite(led2,LOW);
   }
   if(i>3)
   {
    digitalWrite(led3,HIGH);  //Pin 5 or LED 3 HIGH -> Glow
   }
   else
   {
    digitalWrite(led3,LOW);
   }
   delay(1000); /// wait for 1 second
   }
   reset();
 }
  // Set all LEDs off to make sure we start at zero
void reset() {
  digitalWrite(led1,LOW);
  digitalWrite(led2,LOW);
  digitalWrite(led3,LOW);
}
```

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.

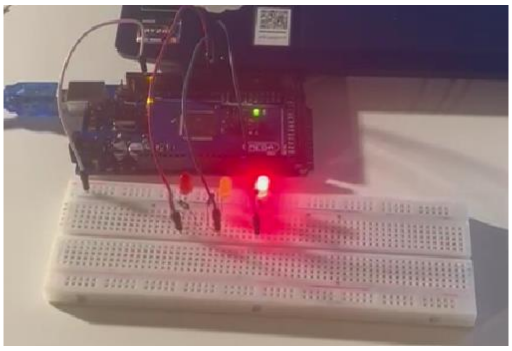


Figure 6: 001

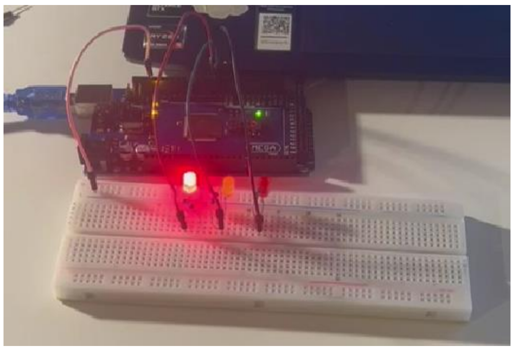


Figure 9: 100

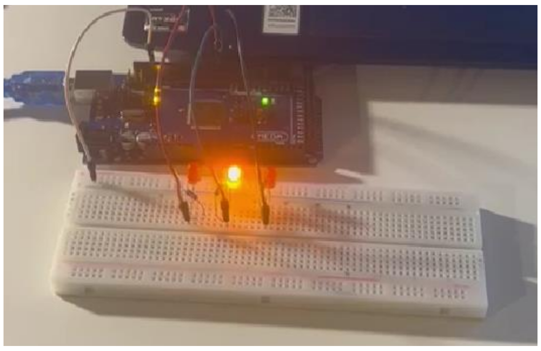


Figure 7: 010

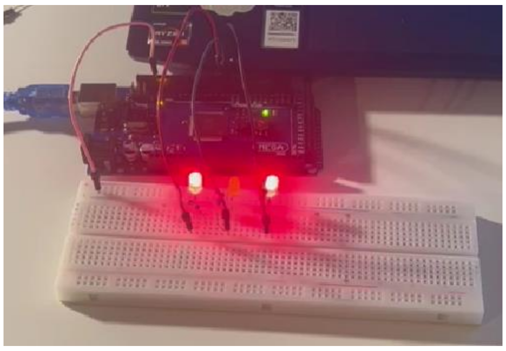


Figure 10: 101

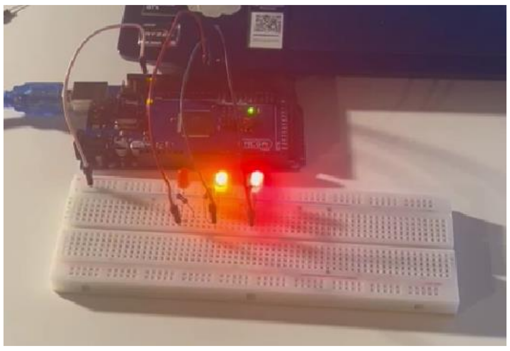


Figure 8: 011

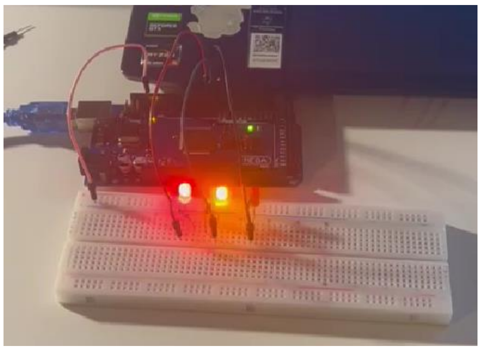


Figure 11: 110

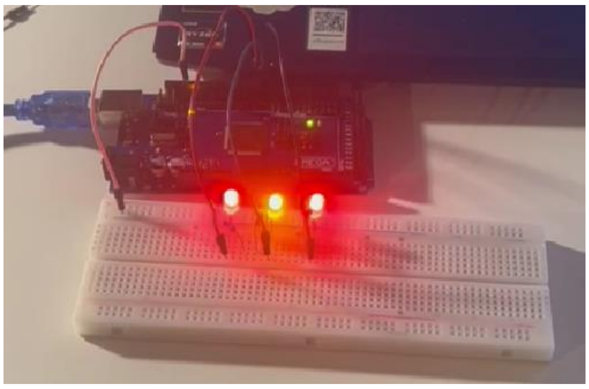
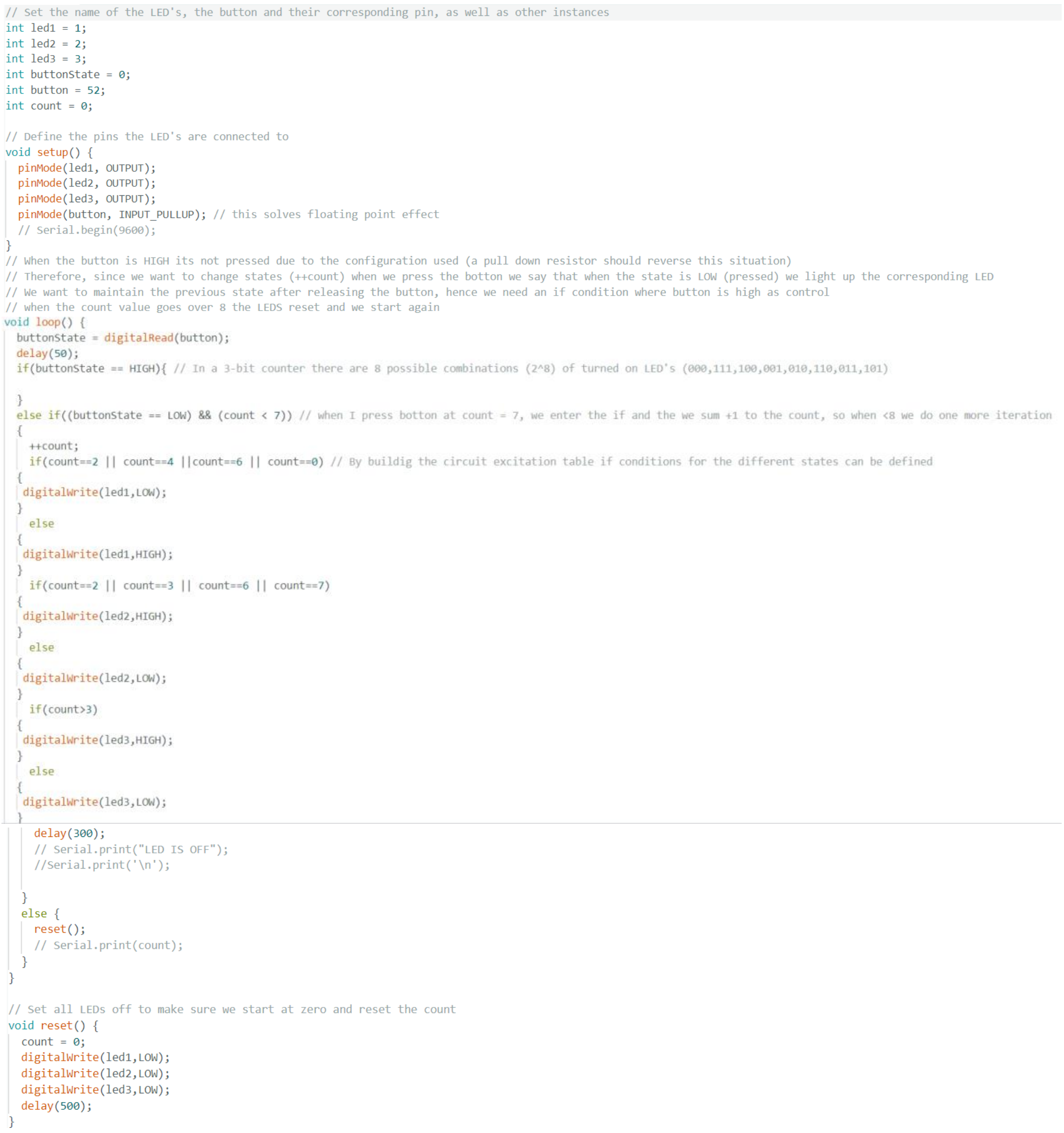


Figure 12: 111

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:



```
// Set the name of the LED's, the button and their corresponding pin, as well as other instances
int led1 = 1;
int led2 = 2;
int led3 = 3;
int buttonState = 0;
int button = 52;
int count = 0;

// Define the pins the LED's are connected to
void setup() {
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(led3, OUTPUT);
  pinMode(button, INPUT_PULLUP); // this solves floating point effect
  // Serial.begin(9600);
}
// When the button is HIGH its not pressed due to the configuration used (a pull down resistor should reverse this situation)
// Therefore, since we want to change states (++count) when we press the botton we say that when the state is LOW (pressed) we light up the corresponding LED
// We want to maintain the previous state after releasing the button, hence we need an if condition where button is high as control
// when the count value goes over 8 the LEDS reset and we start again
void loop() {
  buttonState = digitalRead(button);
  delay(50);
  if(buttonState == HIGH){ // In a 3-bit counter there are 8 possible combinations (2^8) of turned on LED's (000,111,100,001,010,110,011,101)

  }
  else if((buttonState == LOW) && (count < 7)) // when I press botton at count = 7, we enter the if and the we sum +1 to the count, so when <8 we do one more iteration
  {
    ++count;
    if(count==2 || count==4 ||count==6 || count==0) // By buildig the circuit excitation table if conditions for the different states can be defined
  {
   digitalWrite(led1,LOW);
  }
    else
  {
   digitalWrite(led1,HIGH);
  }
    if(count==2 || count==3 || count==6 || count==7)
  {
   digitalWrite(led2,HIGH);
  }
    else
  {
   digitalWrite(led2,LOW);
  }
    if(count>3)
  {
   digitalWrite(led3,HIGH);
  }
    else
  {
   digitalWrite(led3,LOW);
  }
    delay(300);
    // Serial.print("LED IS OFF");
    //Serial.print('\n');

  }
  else {
    reset();
    // Serial.print(count);
  }
}

// Set all LEDs off to make sure we start at zero and reset the count
void reset() {
  count = 0;
  digitalWrite(led1,LOW);
  digitalWrite(led2,LOW);
  digitalWrite(led3,LOW);
  delay(500);
}
```

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.

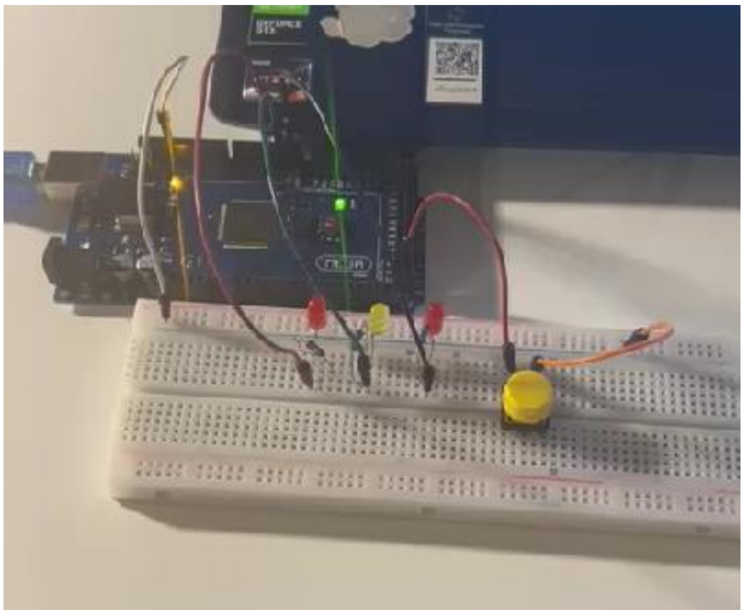
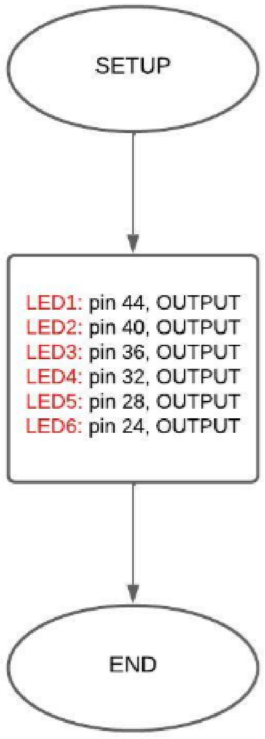
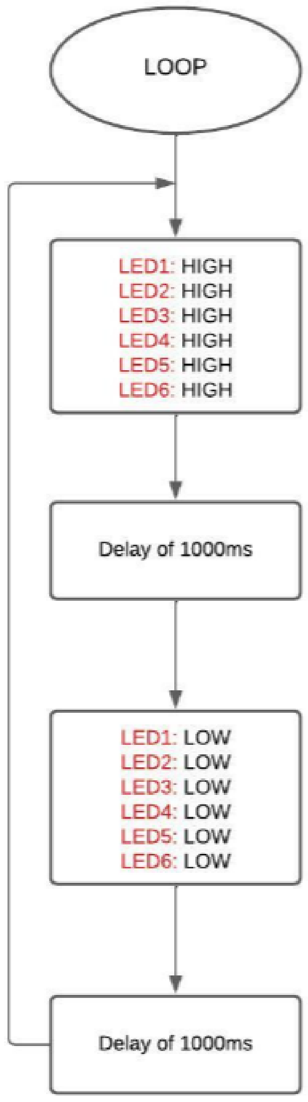


Figure 14: Circuit assembly representing a 3-bit counter with a push button.

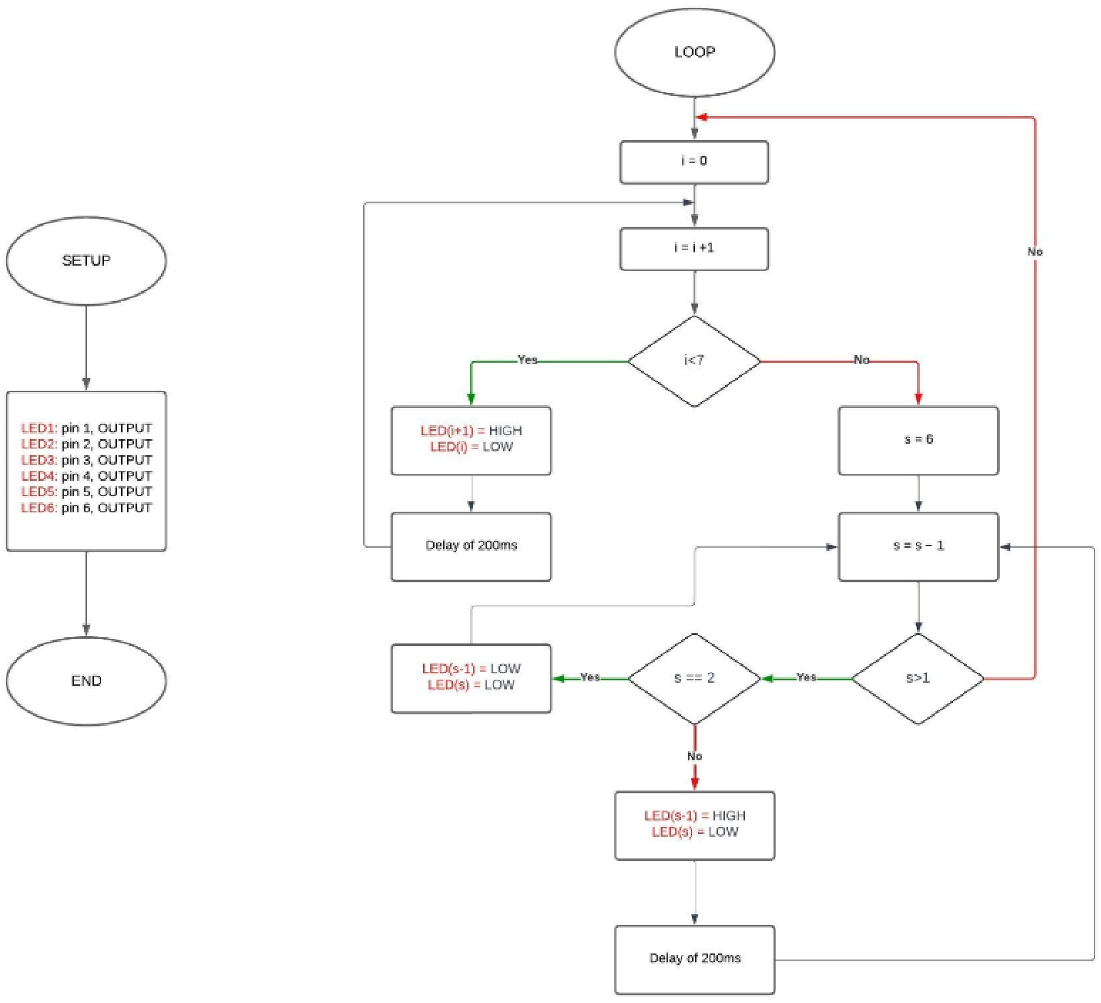
Flowcharts

Exercise 1

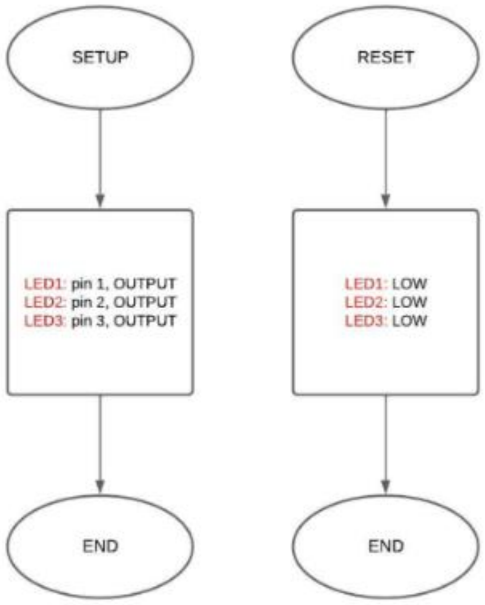


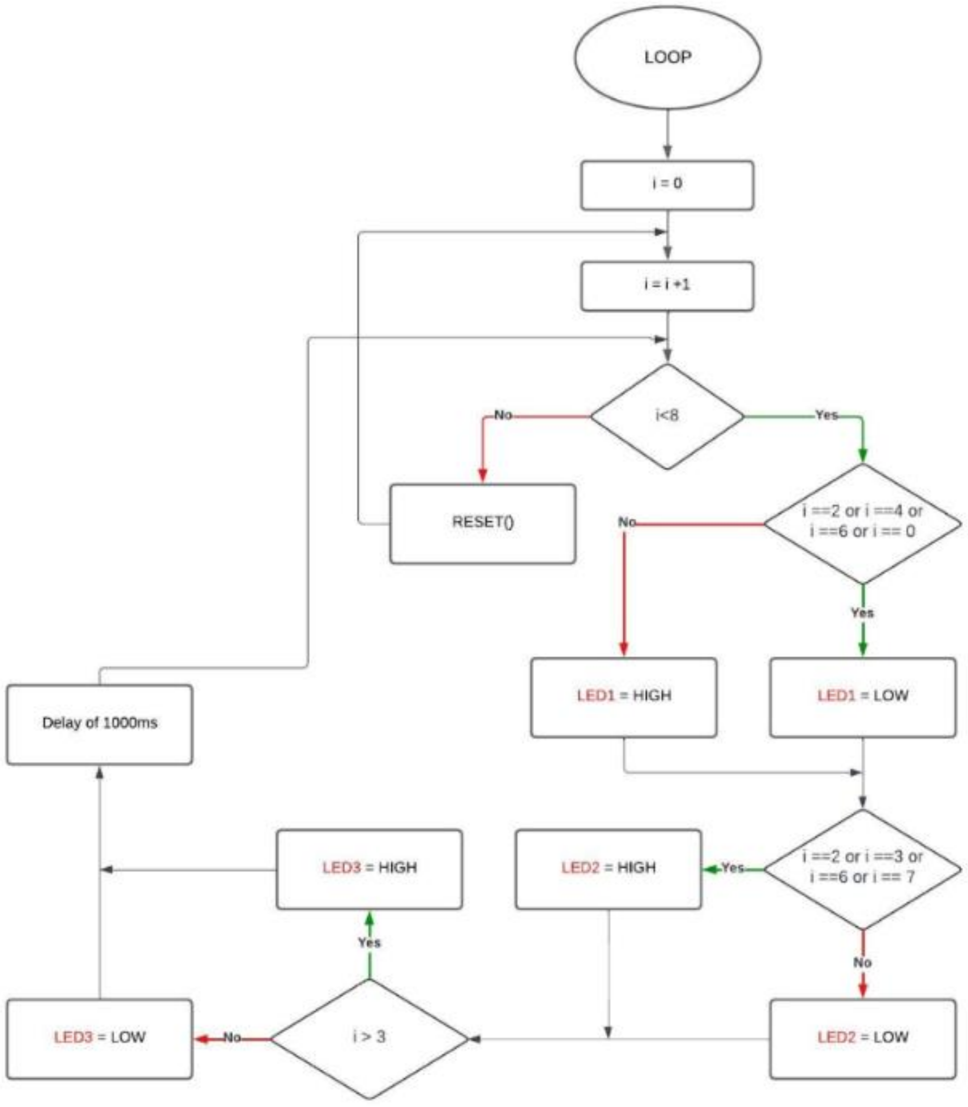


Exercise 2

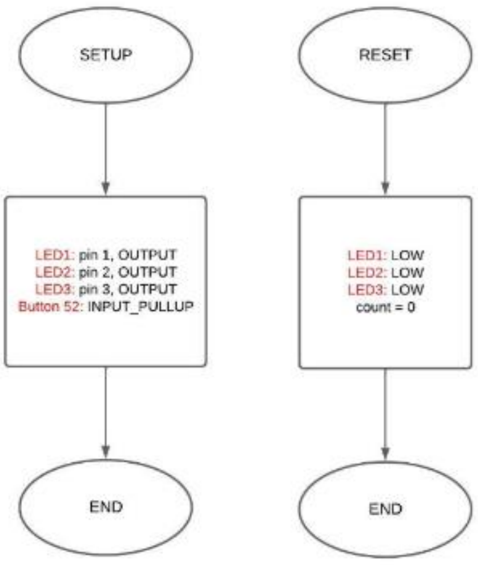

SETUP
LED1: pin 1, OUTPUT
LED2: pin 2, OUTPUT
LED3: pin 3, OUTPUT
LED4: pin 4, OUTPUT
LED5: pin 5, OUTPUT
LED6: pin 6, OUTPUT
END
LOOP
i = 0
i = i +1
i<7
Yes
No
LED(i+1) = HIGH
LED(i) = LOW
Delay of 200ms
s = 6
s = s - 1
s>1
No
Yes
s == 2
Yes
LED(s-1) = LOW
LED(s) = LOW
No
LED(s-1) = HIGH
LED(s) = LOW
Delay of 200ms

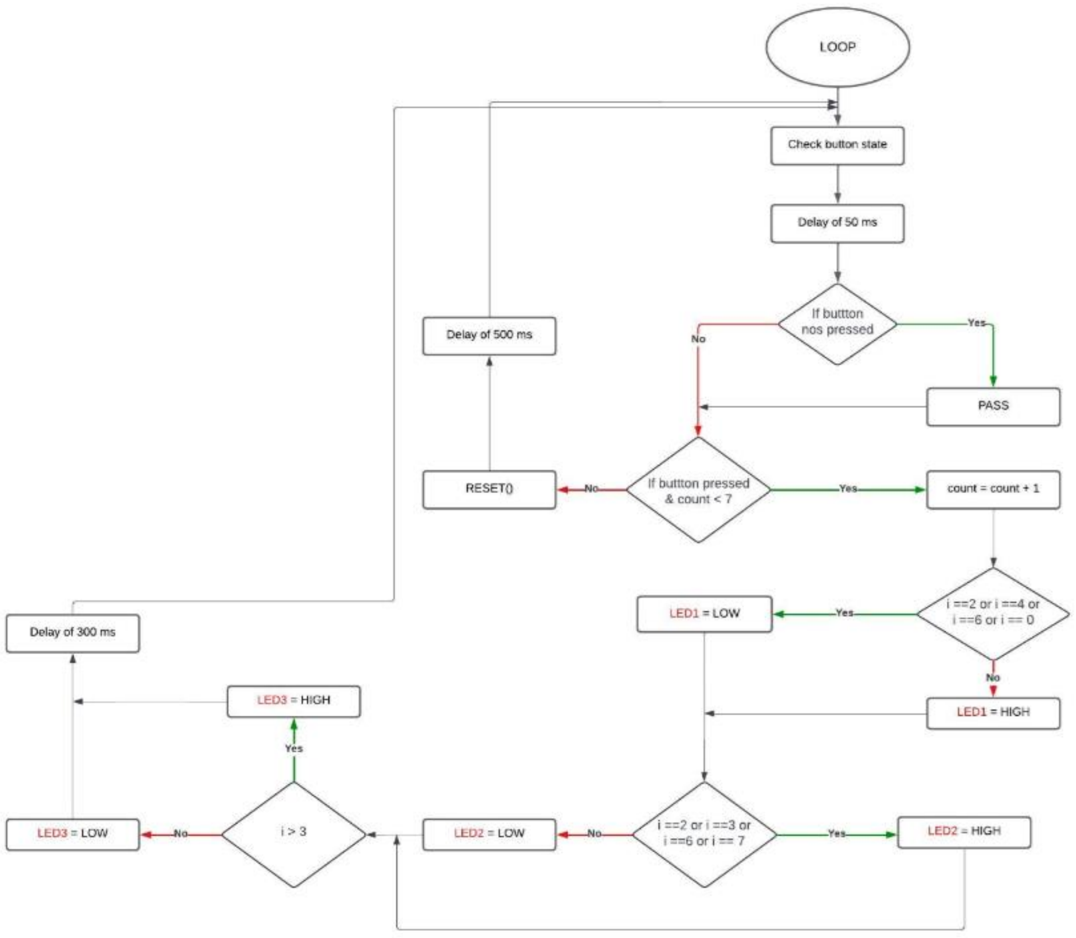
Exercise 3


SETUP
LED1: pin 1, OUTPUT
LED2: pin 2, OUTPUT
LED3: pin 3, OUTPUT
END
RESET
LED1: LOW
LED2: LOW
LED3: LOW
END


LOOP
i = 0
i = i +1
i<8
No
Yes
RESET()
i ==2 or i ==4 or i ==6 or i == 0
No
Yes
LED1 = HIGH
LED1 = LOW
i ==2 or i ==3 or i ==6 or i == 7
Yes
No
LED2 = HIGH
LED2 = LOW
i > 3
Yes
No
LED3 = HIGH
LED3 = LOW
Delay of 1000ms

Exercise 4


SETUP
LED1: pin 1, OUTPUT
LED2: pin 2, OUTPUT
LED3: pin 3, OUTPUT
Button 52: INPUT_PULLUP
END
RESET
LED1: LOW
LED2: LOW
LED3: LOW
count = 0
END


LOOP
Check button state
Delay of 50 ms
If buttton nos pressed
Yes
No
PASS
If buttton pressed & count < 7
Yes
No
count = count + 1
RESET()
Delay of 500 ms
i ==2 or i ==4 or i ==6 or i == 0
Yes
No
LED1 = LOW
LED1 = HIGH
i ==2 or i ==3 or i ==6 or i == 7
Yes
No
LED2 = HIGH
LED2 = LOW
i > 3
Yes
No
LED3 = HIGH
LED3 = LOW
Delay of 300 ms