Digital Systems and Microprocessors

Project 2: 7-Segment Display andSerial monitor

Enrique Almazán Sánchez
Víctor Miguel Álvarez Camarero

Table of Contents

Objetive

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 the 7-segment display as in practice 2, and a push button on pin 22 of the microcontroller. For this project, two programs must be carried out. Each program is independent and should have two different names.

Requirements

The serial monitor is an essential tool integrated within the Arduino platform, facilitating communication between a computer and the microcontroller, which will be introduced during this project. It enables the display of text messages on the computer screen, effectively establishing a means to interact with and control the microcontroller via the keyboard. The serial port's baud rate can be configured to various speeds ranging from 9600 to 115200 bits per second (bps).

In addition, the assembly presented the Lab Session 2 for a 7-Segment Display will be followed, adding any necessary part that the exercise asks. For this setup, we will require the following components:

- Arduino Mega 2560
- 7-segment display (used as an output device)
- Cables for connections
- 7 resistors with a resistance of 220 Ohms each
- Protoboard for circuit assembly

To ensure proper operation and prevent damage to the LEDs within the 7-segment display, it's crucial to address the voltage difference between the supply voltage (5V from the Arduino board) and the forward voltage of the LED. This can be achieved by connecting a resistor between each LED segment and the voltage source. These resistors serve to limit the current flowing through the LED, thereby setting the working voltage within safe limits and protecting each segment of the display.

In the setup, a common cathode 7-segment display will be used. In this configuration, all the cathodes of the LEDs or segments are internally connected to a common pin, which must be linked to a ground signal (GND). To illuminate each individual segment, a positive potential (high or logic level "1") needs to be applied to the corresponding pin. This is accomplished by connecting each segment pin to the microcontroller's digital output pins via a resistor to regulate current flow.

Each segment of the display is connected to a pin of the microcontroller that can be configured as a digital output. With this setup, we gain the ability to independently control the illumination of each segment of the display according to our requirements.

Exercise 1

a. Write the code for a counter from 0 to 9 which each time the push button is pressed the counter will increase by one unit. When the number is greater than 9 the counter is reset again.

The code for this exercise is designed to create a counter from 0 to 9, where each time a push button is pressed, the counter increases by one unit, resetting back to 0. When the number is greater than 9 and displaying it on a 7-segment display.

First of all, some variables are initialized, as well as the baud rate, the speed at which data bits are going to be transmitted, at 9600.

- 'cont' (integer): initialized to 0, keeps track of the current count.
- 'buttonState' (integer): initialized to 0, reads the state of the button.
- 'prevButtonState' (string): initialized to HIGH, reads the previous state of the button.
- 'button' (integer): set to 22, defines to which digital input (port) is connected.
- 'displaypin' (array): defines de pins connected to the 7-segment display.
- 'displaycode' (array): defines the values to be displayed on the 7-segment display.

```
- int cont = 0;
- int buttonState = 0;
- int prevButtonState = HIGH;
- int button = 22;
- int displaypin[] = {2,3,4,5,6,7,8};
- const int baudrate = 9600;
- int displaycode[10][7]=
-   {
-   {1,1,1,1,1,1,0}, //0
-   {0,1,1,0,0,0,0}, //1
-   {1,1,0,1,1,0,1}, //2
-   {1,1,1,1,0,0,1}, //3
-   {0,1,1,0,0,1,1}, //4
-   {1,0,1,1,0,1,1}, //5
-   {1,0,1,1,1,1,1}, //6
-   {1,1,1,0,0,0,0}, //7
-   {1,1,1,1,1,1,1}, //8
-   {1,1,1,1,0,1,1}, //9
-   };
```

In the setup function, pins are configured, including setting the button pin as an input with an internal pull-up resistor ("*INPUT_PULLUP*" command), as shown in the previous lab session (number 1). Display pins are set as outputs ("*pinMode()*"), having the initial count displayed on the 7-segment display, as well as printed on the serial monitor. In addition, the serial communication is initialized ("*Serial.begin()*"), and a delay of 1s is set ("*delay(1000)*")).

```
void setup() {
  pinMode(button, INPUT_PULLUP);
  for(int i=0;i<=6;i++){
    pinMode(displaypin[i], OUTPUT);
  }
  Serial.begin(baudrate);
  delay(100);
  segmentdisplay(cont);
```

```
  Serial.print("Number = ");
  Serial.println(cont);
  delay(1000);
}
```

Furthermore, the loop function continuously checks the state of the button as well as it previous state. If the button is pressed ("*LOW*"), it increments the counter by one, without the possibility of a continuous increment in the 7-segment display. This means, that each time the button is pressed, the 'cont' variable will only increase by one, needing following pressings for continue counting. Then, the current count is displayed on the 7-segment display and printed to the serial monitor. If the counter passes 9, it resets to 0.

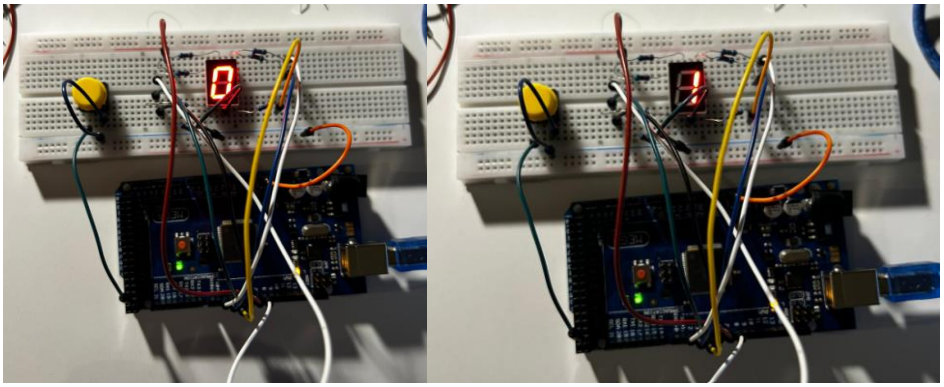
```
void loop() {
  buttonState = digitalRead(button);
  delay(50);
  if (buttonState == LOW && prevButtonState != LOW) {
    cont = (cont == 9) ? 0 : cont + 1;
  }
  segmentdisplay(cont);
  Serial.print("Number = ");
  Serial.println(cont);

  delay(300); //delay of 300ms before reading the state of the button
again
}
```

Finally, the segmentdisplay function takes an input parameter x (the count) and uses it to display the corresponding number on the 7-segment display by writing the appropriate values to the output pins based on the "displaycode" array, iterating over the "displaypin" array.

```
void segmentdisplay(int x){
  for(int i=0;i<=6;i++){
    digitalWrite(displaypin[i], displaycode[x][i]);
  }
}
```

b. Each time the push button is activated and the count increases by one unit, the number must be displayed through the 7-segment display and the serial monitor.

The implementation can be seen in the video 'Exercise_1'. However, below different images can be seen that mimic how the circuit assembly works.



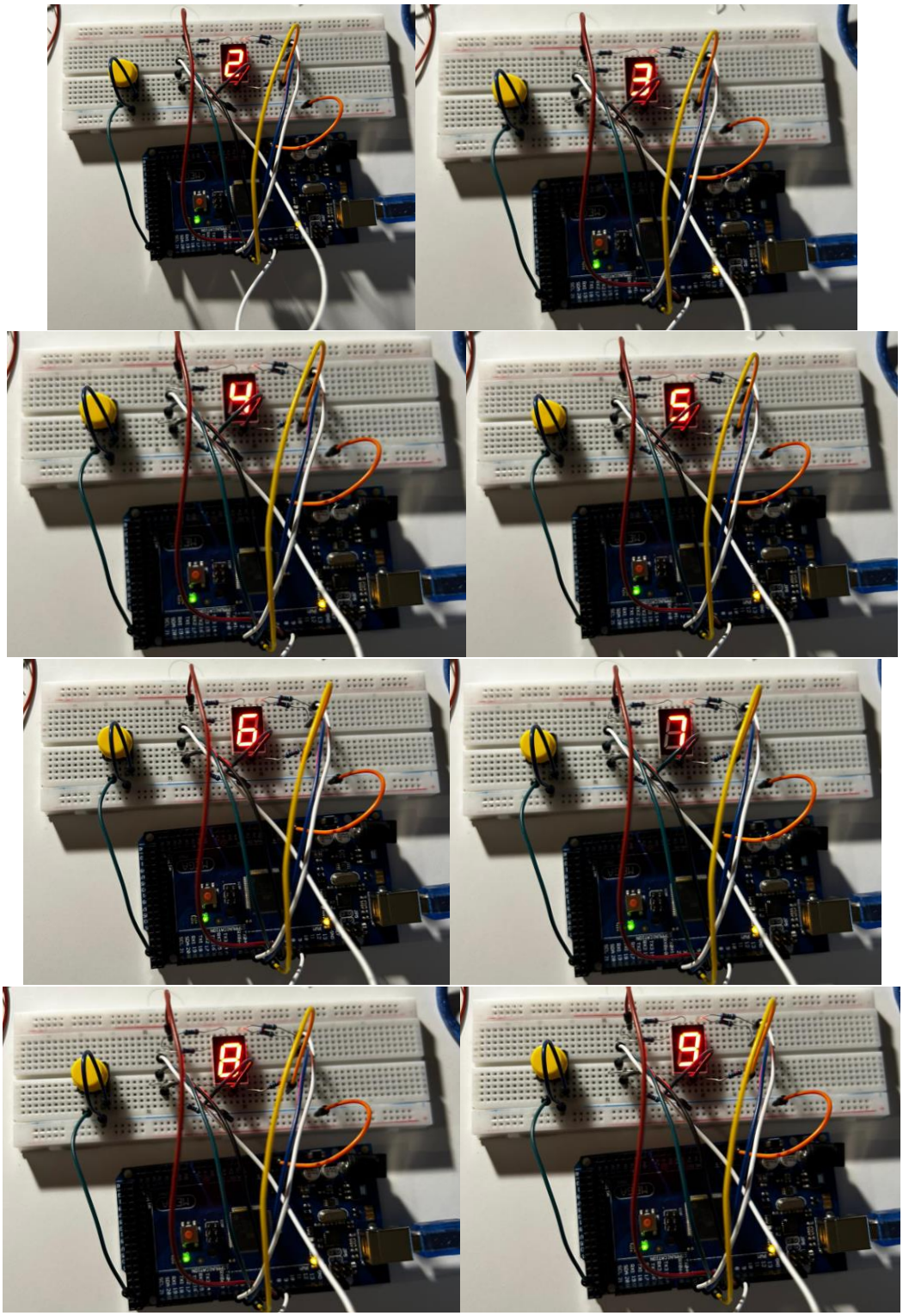


Figure 1: Circuit assembly of the 7-segment display with a push button for each state.

Exercise 2

Based on the previous exercise, write a code in which the serial monitor is used to send a number between 0 and 9. When the number is sent it should be displayed on the 7-segment display.

The code presented for this exercise is designed to receive a number between 0 and 9 from the serial monitor and display it on a 7-segment display. Its implementation can be seen in the video 'Exercise_2'.

First of all, as in the previous section, the variables 'cont' (integer), 'displaypin' (array) and 'displaycode' are initialized, as well as the baud rate at 9600. However, as no press button is added, its variables are supressed.

```
int cont;
int displaypin[] = {2,3,4,5,6,7,8};
const int baudrate = 9600;
int displaycode[10][7]=
  {
  {1,1,1,1,1,1,0}, //0
  {0,1,1,0,0,0,0}, //1
  {1,1,0,1,1,0,1}, //2
  {1,1,1,1,0,0,1}, //3
  {0,1,1,0,0,1,1}, //4
  {1,0,1,1,0,1,1}, //5
  {1,0,1,1,1,1,1}, //6
  {1,1,1,0,0,0,0}, //7
  {1,1,1,1,1,1,1}, //8
  {1,1,1,1,0,1,1}, //9
  };
```

In the setup function, pins are configured, including setting the button pin as an input with an internal pull-up resistor ("*INPUT_PULLUP*" command). Display pins are set as outputs ("*pinMode()*"), and the serial communication is initialized ("*Serial.begin()*"). Also a delay of 1s is set ("*delay(1000)*")). A message is then printed on the serial monitor asking the user to type a number between 0 and 9.

```
void setup() {
  for(int i=0;i<=6;i++){
    pinMode(displaypin[i], OUTPUT);
  }
  Serial.begin(baudrate);
  delay(1000);
  Serial.print("Type a number between 0 and 9: \n");
  delay(300);
}
```

The loop function continuously checks if there are any characters available in the serial buffer ("*Serial.available()*"). If characters are present, the function reads the entire string until a newline character ("*\n*") is detected ("*Serial.readStringUntil()*"). It then converts the string to an integer ("*toInt()*"), showing an error if the condition (between 0 and 9) is not met.

```
void loop() {
  if (Serial.available()>0){
    String read = Serial.readStringUntil('\n');
    Serial.print("You typed: ");
    Serial.println(read);
    cont = read.toInt();
    // Conditions for correct numbers between 0 and 9
    if (cont >= 0 && cont <= 9){
      segmentdisplay(cont);
      delay(300);
    }
    else{
      Serial.print("Choose another number! ");
    }
  }
}
```

If the integer falls within the range of 0 to 9, the segmentdisplay function is invoked to display the corresponding digit on the 7-segment display using the binary codes from the "displaycode" matrix. It accepts an integer argument `x`, representing the digit to be displayed. It sets the output pins connected to the 7-segment display to the corresponding binary codes ("*digitalWrite()*").

```
void segmentdisplay(int x){
  for(int i=0;i<=6;i++){
    digitalWrite(displaypin[i], displaycode[x][i]);
  }
}
```

A delay of 300ms is introduced to allow for a brief pause in the display. If the integer is outside this valid range, an error message is printed to the serial monitor. To send information from the microcontroller and display it on the serial monitor ("*Serial.print*" or "*Serial.println*").

Among various methods available for reading characters from the serial monitor, "*Serial.readStringUntil("\n")*" proved most effective as it enabled reading of the entire message and discarding of numbers consisting of 2 or more digits. Other options like "*Serial.read()*" were not employed since they read characters in ASCII code and necessitated further conversion, processing digits separately.

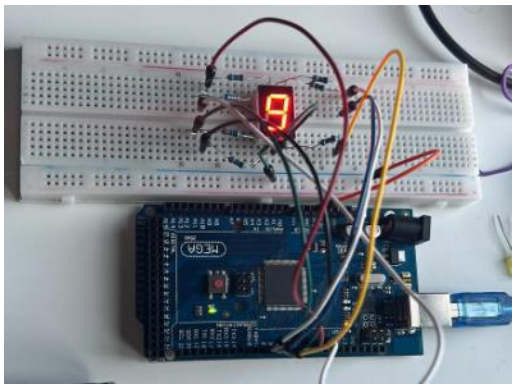
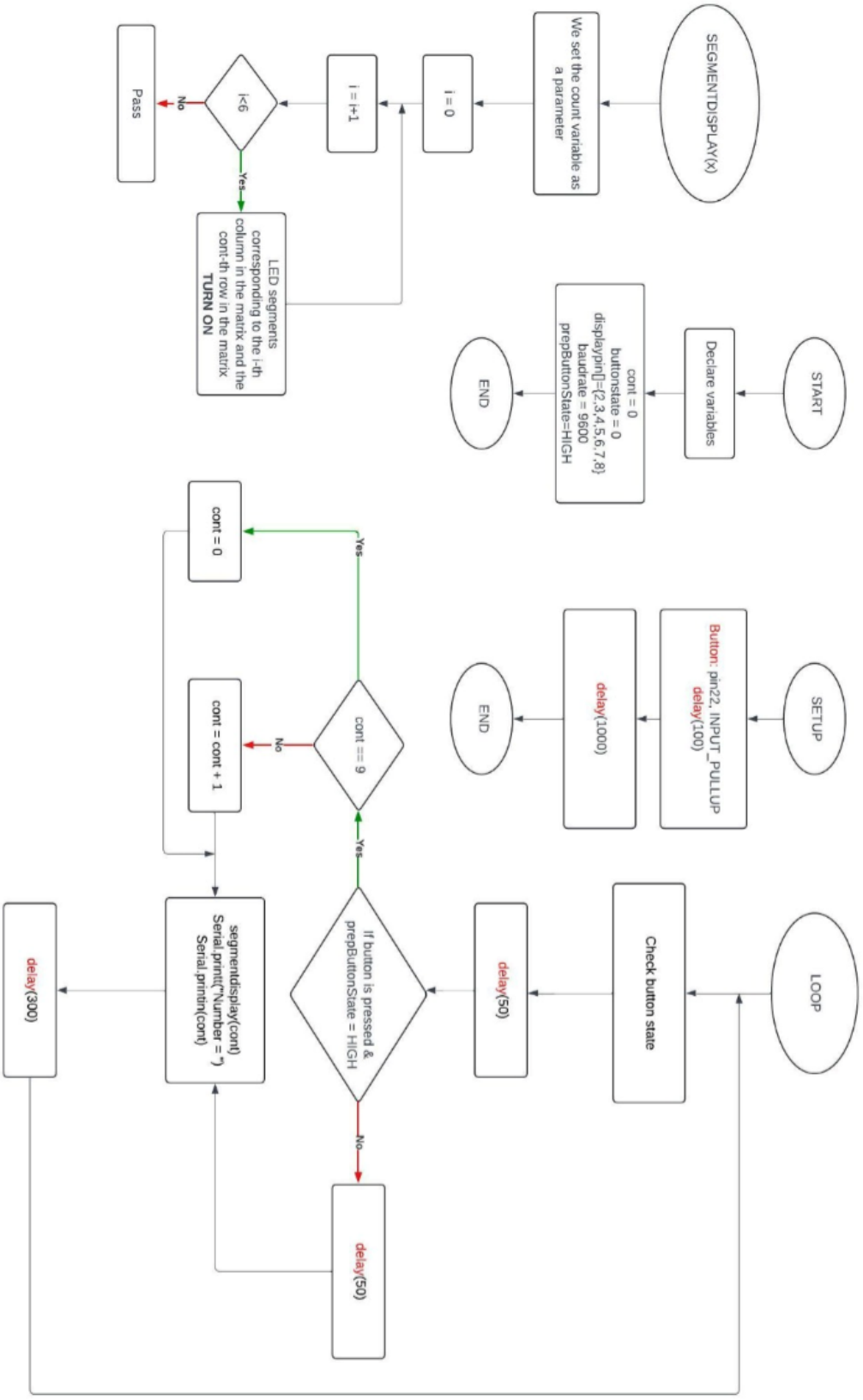


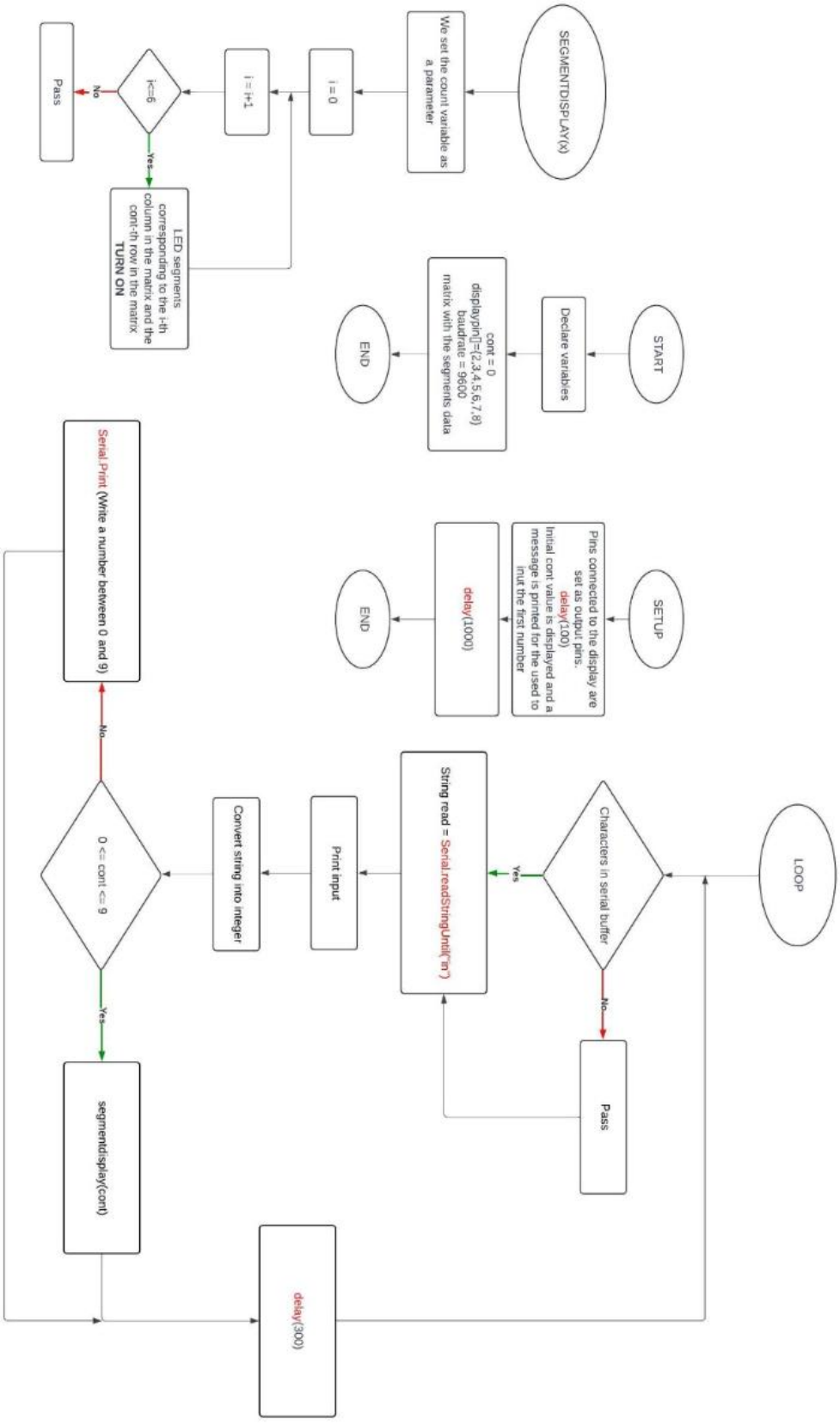
Figure 2: Assembly circuit for a 7-Segement display.

Flowcharts

Exercise 1



Exercise 2


SEGMENTDISPLAY(x)
We set the count variable as a parameter
i = 0
i = i+1
i<=6
No
Yes
Pass
LED segments corresponding to the i-th column in the matrix and the cont-th row in the matrix TURN ON
START
Declare variables
cont = 0
displaypin[]={2,3,4,5,6,7,8}
baudrate = 9600
matrix with the segments data
END
SETUP
Pins connected to the display are set as output pins.
delay(100)
Initial cont value is displayed and a message is printed for the used to inut the first number
delay(1000)
END
LOOP
Characters in serial buffer
Yes
No
String read = Serial.readStringUntil('\n')
Pass
Print input
Convert string into integer
0 <= cont <= 9
No
Yes
Serial.Print (Write a number between 0 and 9)
segmentdisplay(cont)
delay(300)