(Company No.: 198301005860)

Garden of Knowledge and Virtue

MECHATRONICS SYSTEM INTEGRATION : MCTA 3203

LABORATORY REPORT

SECTION 1

TITLE: DIGITAL LOGIC SYSTEM. VER 2 (WEEK 2)

LECTURER'S NAME: ASSOC. PROF. IR. TS. DR. ZULKIFLI BIN ZAINAL ABIDIN

DATE OF EXPERIMENT: 15 OCTOBER 2025

DATE OF SUBMISSION: 22 OCTOBER 2025

GROUP: 4

NO.	NAME	MATRIC NO.
1	MOHAMMAD FARISH ISKANDAR BIN MOHD SYAHIDIN	2311779
2	AHMAD HARITH IMRAN BIN MOHD YUSOF	2311581
3	ARIFA AQILAH BINTI ABDUL HALIM	2311530

ACKNOWLEDGEMENTS

We would like to express our heartfelt appreciation to Dr. Zulkifli Bin Zainal Abidin for his continuous guidance, support, and insightful explanations throughout this experiment. His expertise greatly helped us understand the concepts of interfacing and digital control using Arduino. We would also like to thank our lab assistant for providing technical assistance and ensuring that the experiment ran smoothly. Finally, we extend our gratitude to our classmates for their cooperation and teamwork during the lab session.

ABSTRACT

This experiment involved connecting an Arduino Uno R3 board with a common cathode 7-segment display to observe and control numerical output. Each of the seven segments was connected to separate Arduino digital pins using appropriate resistors. Push buttons were also integrated into the setup to allow manual operations such as incrementing the displayed number and resetting it to zero. Once the Arduino code was uploaded, the circuit successfully displayed numbers from 0 to 9 as the increment button was pressed. The reset button functioned as expected, returning the display to zero when activated. Overall, the experiment provided valuable hands-on experience in hardware interfacing, particularly in managing a 7-segment display and incorporating user input through push buttons. The understanding gained from this exercise lays an essential groundwork for future projects involving digital control and Arduino-based system design.

TABLE OF CONTENT

1.0 INTRODUCTION

This project uses an Arduino Uno microcontroller to interface with a 7-segment display in order to comprehend and apply the concepts of digital logic systems. Using two push buttons, one for increasing the displayed amount and the other for returning it to zero, the goal is to create a basic counting system that shows digits 0 to 9. This project shows how microcontrollers analyse digital input signals and produce useful visual results.

In a 7-segment display, the numerical digits are represented by seven light-emitting diodes (LEDs) arranged in a particular way. This experiment uses a common cathode display, which means that each segment is controlled by a separate Arduino output pin and all cathode terminals are linked to ground (GND). Using push buttons as user inputs, the Arduino processes the HIGH and LOW digital signals to adjust the display output appropriately.

The primary goal of this experiment is to learn how microcontrollers and digital logic work. The Arduino uses the inputs it gets from the push buttons to drive the seven-segment display. The application determines which number should be shown on the screen by using basic conditions. We expected that when the button was pressed, the display would count from 0 to 9, increasing by one, and then returning to 0 when the reset button was pressed. Overall, this project demonstrates how to integrate code and hardware to produce a straightforward and useful counting system.

2.0 MATERIAL AND EQUIPMENT

APPARATUS	QUANTITY
Arduino UNO R3	1
Push button	2
1k resistor	7
Cathode 7 segment display	1

3.0 EXPERIMENTAL SETUP

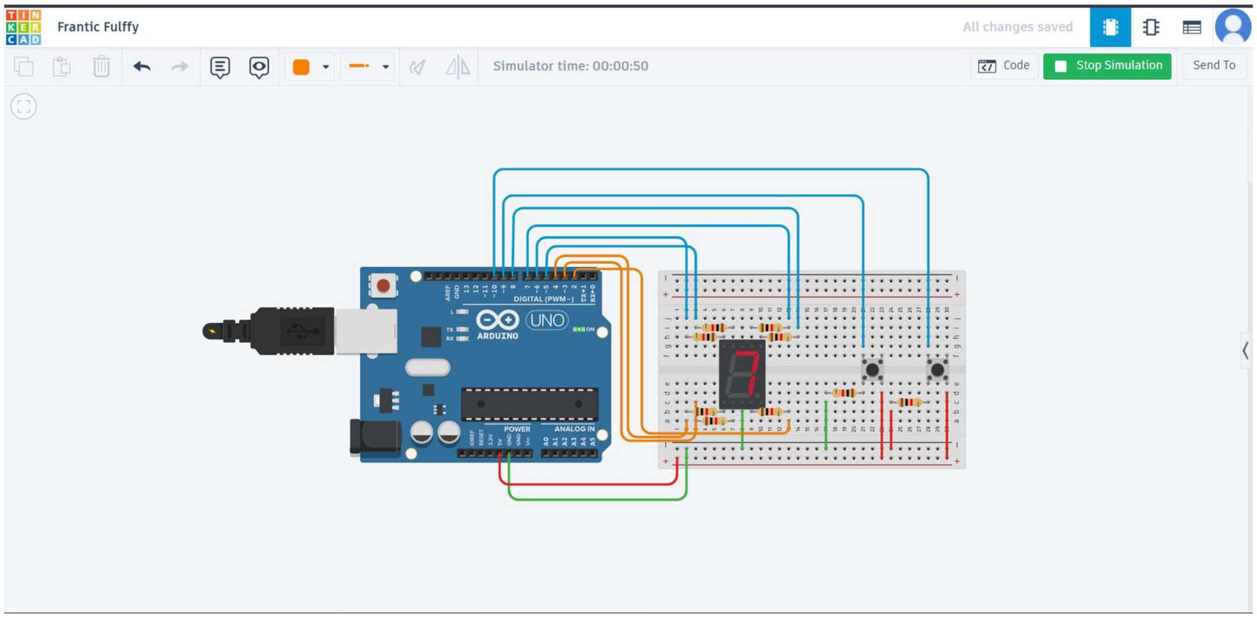


Figure A: Components connected to arduino uno in Tinkercad

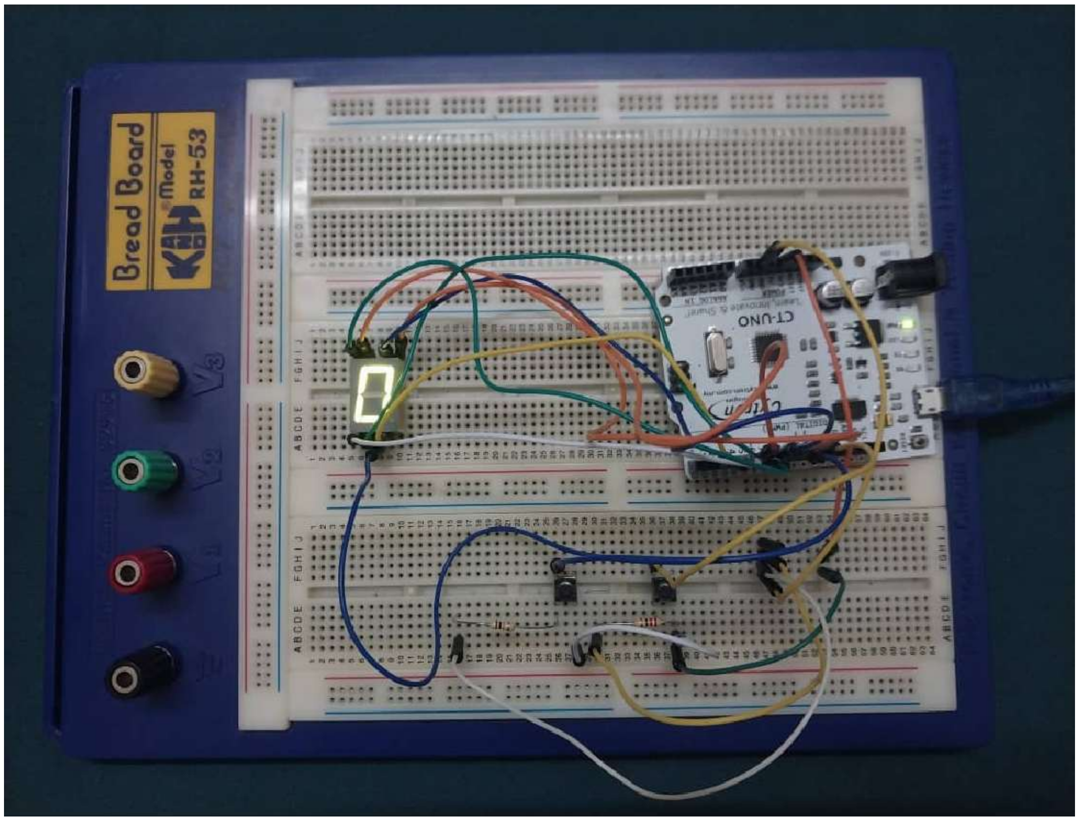


Figure B: Components connected to Arduino uno physically

4.0 METHODOLOGY

1. The circuit was set up according to the instructions for circuit arrangement.
2. The Arduino Uno receives the uploaded Arduino code.
3. The following number appeared on the 7-segment display when the increment count button was pressed.
4. The process was repeated several times to verify that the numbers increased correctly from 0 to 9.
5. The button was hit and it turned to 0 to view the result of rest.

Circuit Assembly

- Each segment of the 7-segment display was attached to a different digital output pin on the Arduino Uno.
- The circuit has two push buttons. One for setting the number and another for increasing it.
- The button signals were stabilised and incorrect triggers were avoided by using internal pull-up resistors.
- Double-checking the power and ground connections ensured the display operated steadily.

Programming Logic

- The purpose of the Arduino code was to continuously read the current states of both buttons.
- The current number displayed on the 7-segment display was stored in a counter variable.
- The counter increased by one when the increment button was pressed, and the display was updated to show the new value.
- The display was reset when the reset button was pressed, bringing the counter value back to zero.
- To create the proper number pattern for every count, a custom function controlled which display segments were activated.

Code Used

```
// Segment pins
const int segA = 7;
const int segB = 8;
const int segC = 2;
const int segD = 3;
const int segE = 4;
const int segF = 5;
const int segG = 6;

// Button pins
const int RESET = 9;
const int START = 10;

// State variables
int STARTBUT = 0;
int RESETBUT = 0;
int lastSTARTBUT = LOW;
int lastRESETBUT = LOW;
int count = 0;

//0= ABCDEF
//1= BC
//2= ABGED
//3=ABCDG
//4=BCFG
//5=AFGCD
//6=AFGEDC
//7=ABC
//8=ABCDEFG
//9=ABCFG

// Setup function - runs once when the program starts
void setup() {
 // Set up segment pins as output
  pinMode(segA, OUTPUT);
  pinMode(segB, OUTPUT);
  pinMode(segC, OUTPUT);
  pinMode(segD, OUTPUT);
  pinMode(segE, OUTPUT);
  pinMode(segF, OUTPUT);
  pinMode(segG, OUTPUT);

// Set button pins as input
 pinMode(RESET, INPUT);
  pinMode(START, INPUT);

// Start serial communication for debugging
  Serial.begin(9600);
}

// Loop function
void loop()
{
  STARTBUT = digitalRead(START);
  RESETBUT = digitalRead(RESET);

  // Detect increment button press
  if(STARTBUT == HIGH && lastSTARTBUT == LOW)
  {
    count++;
    if(count>9)
    {
      count = 0;
    }
    Serial.print("COUNT=");
    Serial.println(count);
  }

  // Detect reset button press
  if(RESETBUT == HIGH && lastRESETBUT == LOW)
  {
    count = 0;
    Serial.print("COUNT=");
    Serial.println(count);
```

```
}

// Small delay to avoid bouncing effect
delay(250);

// Store the current button states for next loop iteration
lastSTARTBUT = STARTBUT;
lastRESETBUT = RESETBUT;

// Display the current count on the 7-segment display
switch(count)
{
 case 0:
 digitalWrite(segA,HIGH);
 digitalWrite(segB,HIGH);
 digitalWrite(segC,HIGH);
 digitalWrite(segD,HIGH);
 digitalWrite(segE,HIGH);
 digitalWrite(segF,HIGH);
 digitalWrite(segG,LOW);
 break;

 case 1:
 digitalWrite(segA,LOW);
 digitalWrite(segB,HIGH);
 digitalWrite(segC,HIGH);
 digitalWrite(segD,LOW);
 digitalWrite(segE,LOW);
 digitalWrite(segF,LOW);
 digitalWrite(segG,LOW);
 break;

 case 2:
 digitalWrite(segA,HIGH);
 digitalWrite(segB,HIGH);
 digitalWrite(segC,LOW);
 digitalWrite(segD,HIGH);
 digitalWrite(segE,HIGH);
digitalWrite(segF,LOW);
digitalWrite(segG,HIGH);
break;

case 3:
digitalWrite(segA,HIGH);
digitalWrite(segB,HIGH);
digitalWrite(segC,HIGH);
digitalWrite(segD,HIGH);
digitalWrite(segE,LOW);
digitalWrite(segF,LOW);
digitalWrite(segG,HIGH);
break;

case 4:
digitalWrite(segA,LOW);
digitalWrite(segB,HIGH);
digitalWrite(segC,HIGH);
digitalWrite(segD,LOW);
digitalWrite(segE,LOW);
digitalWrite(segF,HIGH);
digitalWrite(segG,HIGH);
break;

case 5:
digitalWrite(segA,HIGH);
digitalWrite(segB,LOW);
digitalWrite(segC,HIGH);
digitalWrite(segD,HIGH);
digitalWrite(segE,LOW);
digitalWrite(segF,HIGH);
digitalWrite(segG,HIGH);
break;

case 6:
digitalWrite(segA,HIGH);
digitalWrite(segB,LOW);
digitalWrite(segC,HIGH);
```

```
    digitalWrite(segD,HIGH);
    digitalWrite(segE,HIGH);
    digitalWrite(segF,HIGH);
    digitalWrite(segG,HIGH);
    break;

    case 7:
    digitalWrite(segA,HIGH);
    digitalWrite(segB,HIGH);
    digitalWrite(segC,HIGH);
    digitalWrite(segD,LOW);
    digitalWrite(segE,LOW);
    digitalWrite(segF,LOW);
    digitalWrite(segG,LOW);
    break;

    case 8:
    digitalWrite(segA,HIGH);
    digitalWrite(segB,HIGH);
    digitalWrite(segC,HIGH);
    digitalWrite(segD,HIGH);
    digitalWrite(segE,HIGH);
    digitalWrite(segF,HIGH);
    digitalWrite(segG,HIGH);
    break;

    case 9:
    digitalWrite(segA,HIGH);
    digitalWrite(segB,HIGH);
    digitalWrite(segC,HIGH);
    digitalWrite(segD,HIGH);
    digitalWrite(segE,LOW);
    digitalWrite(segF,HIGH);
    digitalWrite(segG,HIGH);
    break;
  }
}
```

Control Algorithm

1. Pin Setup
    - Each segment of the 7-segment display (A–G) is assigned to separate digital pins on the Arduino.
    - Two buttons are connected to the Arduino:
        - Increment button on pin D10
        - Reset button on pin D9
2. Variable Declaration
    - count stores the current number shown on the display.
    - STARTBUT and RESETBUT hold the current button readings.
    - lastSTARTBUT and lastRESETBUT keep the previous button states for edge detection (to detect when a new press occurs).
3. Setup function
    - All segment pins are set as output, while both buttons are configured as input.
    - Serial communication is started using Serial.begin(9600) to monitor the counter value.
4. Main Loop
    - The Arduino continuously reads both button states.

    - When the increment button is pressed, the counter increases by one.
        - If the count reaches above 9, it returns to 0 automatically.
        - A short delay of 250 ms is used to avoid false multiple counts (button debounce).

- When the reset button is pressed, the counter resets to 0 immediately.
- The previous button readings are updated each time to ensure proper press detection in the next cycle.

5. Display Function

- The display updates every time the count changes.
- Before showing a new number, all segments are turned OFF.
- A switch(count) statement determines which segments should light up to form the correct number pattern (0–9).

5.0 DATA COLLECTION

This lab experiment involves components like Arduino uno, resistor, button and 7 segment display. This combination of components is used to demonstrate a digital logic system. Below is the table showing the wiring details for each component.

COMPONENT	**WIRING DETAILS**
Resistor	In the actual experiment, the resistor was not used due to the faulty 7 segment display which requires direct intake voltage to function properly.
Push Button	The push button was connected to pin D9 (Reset button) and pin D10 (Start button).
7 segment display	The components have 10 pins where 2 of them represent cathode

	pins (See appendix A for pin types). The G, F, A, B, E, D, C and common cathode pins are connected to pin D6, D5, D7, D8, D4, D3, D2 and ground pin respectively. The decimal point pin was not connected in the setup as it plays no role for the experiment.

The table below represents the correct representation to form numbers based on the pin that are HIGH ;

DISPLAY	**PIN**
0	D7, D8, D2, D3, D4, D5
1	D8, D2
2	D7, D8, D6, D4, D3
3	D7, D8, D2, D3, D6
4	D8, D2, D5, D6
5	D7, D5, D6, D2, D3
6	D7, D5, D6, D2, D3, D4
7	D7, D8, D2
8	D7, D5, D6, D2, D3, D4, D8
9	D7, D5, D6, D2, D8

6.0 DATA ANALYSIS

From the collected data in the experiment, we analyzed the function of the code from the task given. (See appendix B to see the code and serial monitor output)

ACTION	DESCRIPTION	OUTPUT
Increment count	In the code there is a loop that uses the if statement for the variable “StartBUT” that controls the start button. When it is pressed, the count will increase until it reaches 9 and will count back to 0.	Count until 9 and reach back 0.
Reset count	Reset button is used to reset the count back to 0 regardless of the current number.	Reset the number to 0.
LastStartBUT and LastResetBUT	These 2 variables control the flow of the button and ensure the count only increases once even when the button is held. If this variable is not added, then the count will not be accurate.	Allow the number to increase once.
Switch case	The switch loop used in the code represents all the segments turning ON based on the case. For example, if the count on the Serial	Turning ON the segment.

	Monitor displays 3 then the code will run case 3.	

7.0 RESULTS

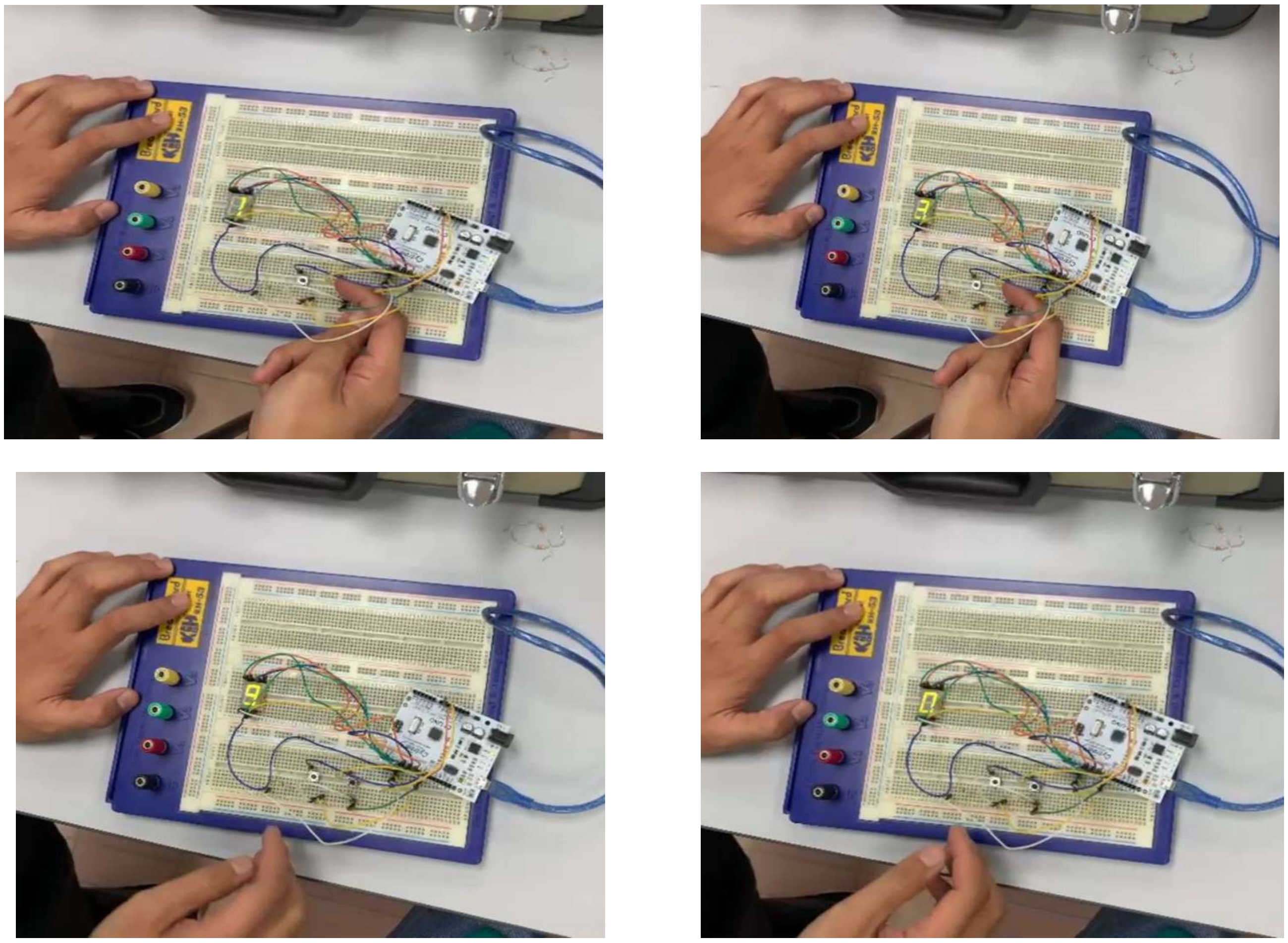


Figure C: The segment display turning on based on the button pushed

The experiment successfully demonstrated the operation of a 7-segment display controlled by an Arduino Uno R3 using push buttons as manual inputs. The circuit was built and programmed according to the experiment steps provided. When the increment button was pressed, the LED segments on the display increased the number shown sequentially from 0 up to 9. When the reset button was pressed, the counter reset to 0, confirming the proper functionality of the reset feature.

The attached photos (Figure C) show the sequential changes on the 7-segment display as the increment button was pressed, as well as the reset state when the count returned to zero. These images provide clear evidence that both the hardware and the programmed logic operated correctly, demonstrating the successful interfacing between the Arduino and the 7-segment display.

8.0 DISCUSSION

This experiment's primary objective was to see whether an Arduino Uno and push buttons could be used to control a 7-segment display. As can be seen from the results, the system functioned as planned, in which pressing the increment button caused the numbers to rise from 0 to 9, and pressing the reset button went back to 0. This proves the proper integration of the coding and hardware.

Sources of Error and Limitations:

1. Component Malfunction
    - During the testing stage, one of the components stopped working properly, which caused the display to show incomplete or unclear numbers. After checking the circuit, it was confirmed that the faulty component affected the overall performance of the system, resulting in an unstable display output
2. Unstable Wiring

- The 7-segment display had some segments that did not light up correctly during testing. This was caused by poor breadboard connections or loose jumper wires.

Improvement:

1. The circuit was retested after the damaged part was changed for a new one. The display functioned normally and displayed the proper brightness after being replaced. Additionally, an effective power connection from the Arduino was used to guarantee that all components received enough power.
2. The circuit layout was made neater, and all connections were double-checked and tightened. After that, all segments are displayed correctly.

The process of connecting an Arduino to an I2C LCD

Since an I2C LCD just requires two signal pins (SDA for data and SCL for clock), connecting it to an Arduino is simple. Compared to a 7-segment display, which needs one pin for each segment, this helps minimize the amount of wires required.

Steps to connect:

1. Connect VCC to the 5V pin on the Arduino
2. Connect GND to the GND pin on the Arduino
3. Connect SDA pin of the LCD to A4 and the SCL pin to A5 on the Arduino Uno
4. Install and include the LiquidCrystal_I2C library in the Arduino IDE
5. Write a simple program to turn on the display and show text.

Code to display “Hello World” on the I2C LCD:

- Include the library

Use the line #include <LiquidCrystal_I2C.h> at the top of your code to allow Arduino to communicate with the I2C LCD.

- Set up the LCD address and size

  Create an LCD object with its I2C address and display size.

  Example: LiquidCrystal_I2C lcd(0x27, 16, 2);

  0x27 is the I2C address and 16, 2 means the display has 16 columns and 2 rows.

- Initialize and turn on the LCD

  Call these functions inside setup() to get the LCD ready:

- lcd.init(); → initialize the LCD
- lcd.clear(); → clear the screen and reset the cursor
- lcd.backlight(); → turn on the backlight

- Set the cursor position

  Example: lcd.setCursor(0, 1);

  0 is the column position and 1 is the row position

- Print the message

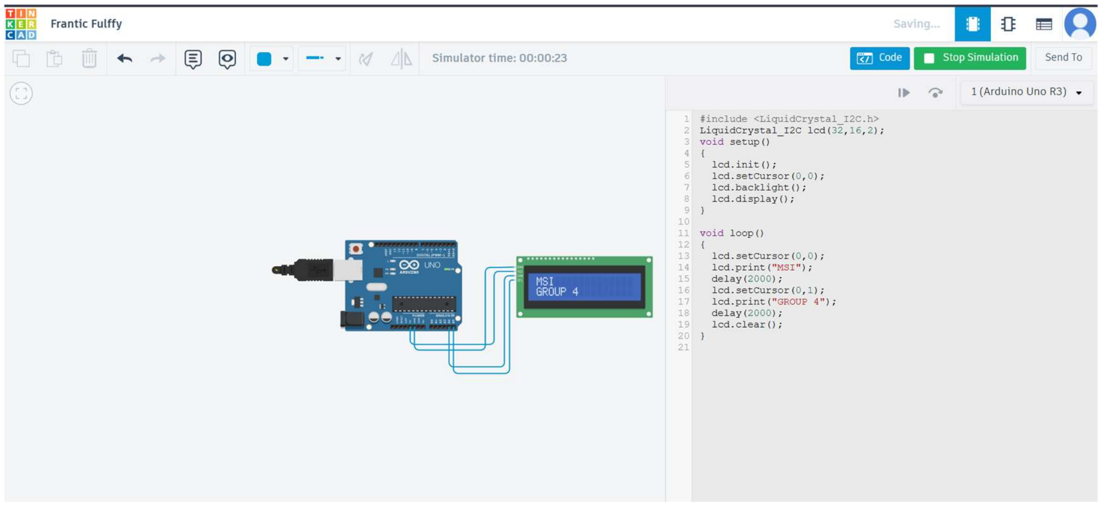
  Example: lcd.print("Hello World");

Explain the coding concept that drives it in comparison to matrix LED and 7-segment displays

Feature	I2C LCD	7-Segment Display	LED Matrix
Communication Type	Serial (uses 2 lines: SDA and SCL)	Parallel (each segment controlled separately)	Multiplexed (rows & columns)

Programming Method	Use LiquidCrystal_I2C.h library with simple commands	Controlled manually by code	Use special libraries for patterns or text
Pins Required	2 pins only	7 or more	8 or more
Display Capability	Can show text, numbers and symbols	Numbers only (0-9)	Can show text, shapes or animations
Coding Complexity	Easy	Medium	Hard

The I2C LCD requires fewer pins and pre-built functions from its library, making it the easiest to program and connect. The LED matrix provides greater display versatility but involves complex programming and timing management, whereas the 7-segment display requires additional wiring and human coding for every number.



```
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(32,16,2);
void setup()
{
  lcd.init();
  lcd.setCursor(0,0);
  lcd.backlight();
  lcd.display();
}

void loop()
{
  lcd.setCursor(0,0);
  lcd.print("MSI");
  delay(2000);
  lcd.setCursor(0,1);
  lcd.print("GROUP 4");
  delay(2000);
  lcd.clear();
}
```

Figure D: Arduino connected to LCD I2C using TinkerCad

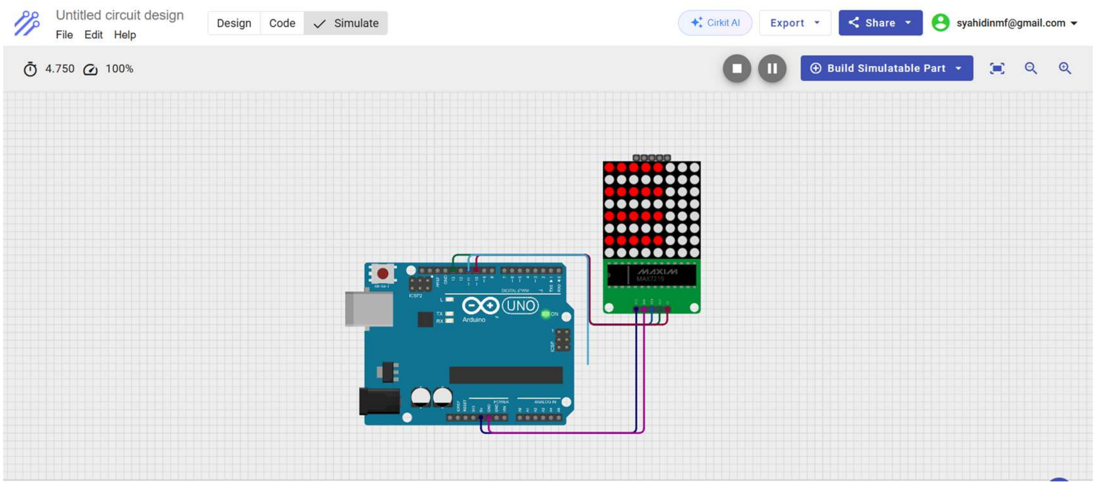


Figure E: Arduino connected to Matrix LED using Cirkit Designer

9.0 CONCLUSION

This experiment successfully achieved its main goal, which was to build a simple counting system using an Arduino Uno and a 7-segment display. With one button raising the number range from 0 to 9 and the other button bringing the display back to 0, the circuit operated as planned.

The outcomes confirmed our initial hypothesis that the Arduino could use programmed logic to accurately control the display. This demonstrates the efficient processing of inputs and control of electronic components by microcontrollers.

This experiment taught us how coding and hardware interact in digital systems. Additionally, the experiment enhanced our comprehension of fundamental circuit connections and digital logic. This idea can be used in the future for more complex applications.

10.0 RECOMMENDATIONS

1. Enhance display functionality by expanding the setup to include multiple 7-segment displays capable of showing two-digit numbers or functioning as simple digital clocks and timers. This improvement will allow students to explore multiplexing and more advanced display control techniques.

2. Relate the experiment concepts to real-world applications by highlighting how 7-segment displays and digital counters are commonly used in technologies such as elevators, calculators, and electronic meters. This connection can help future students understand the practical relevance of digital logic systems.

11.0 REFERENCES

Dziubym. (2020, November 29). *Controlling 8x8 dot matrix with MAX7219 and Arduino*. Arduino Project Hub. https://projecthub.arduino.cc/Dziubym/controlling-8x8-dot-matrix-with-max7219- and-arduino-0c417a

Jehankandt. (2021, September 3). Arduino 16x2 LCD display with I2C—Hello world. Arduino Project Hub. https://projecthub.arduino.cc/jehankandt/arduino-16x2-lcd-display-with-i2c-hello -world-8a3af4

Stannano. (2021, August 18). One digit 7-segment LED display. Arduino Project Hub. https://projecthub.arduino.cc/stannano/one-digit-7-segment-led-display-819bcd

12.0 APPENDICES

Appendix A

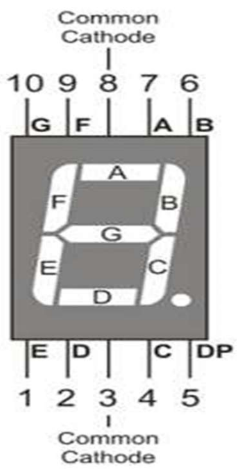
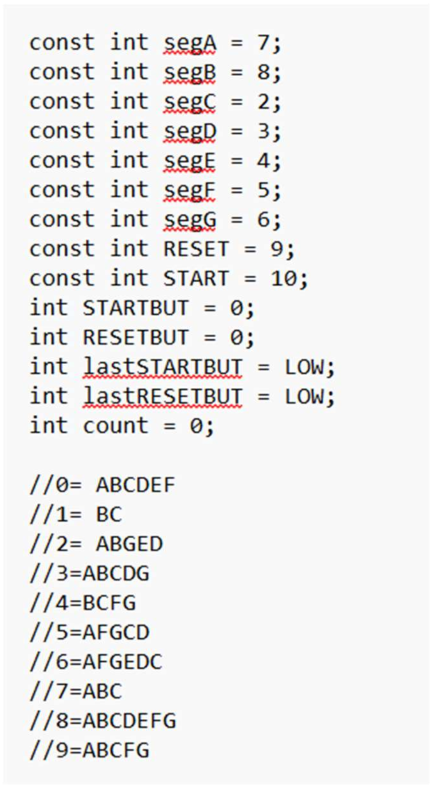
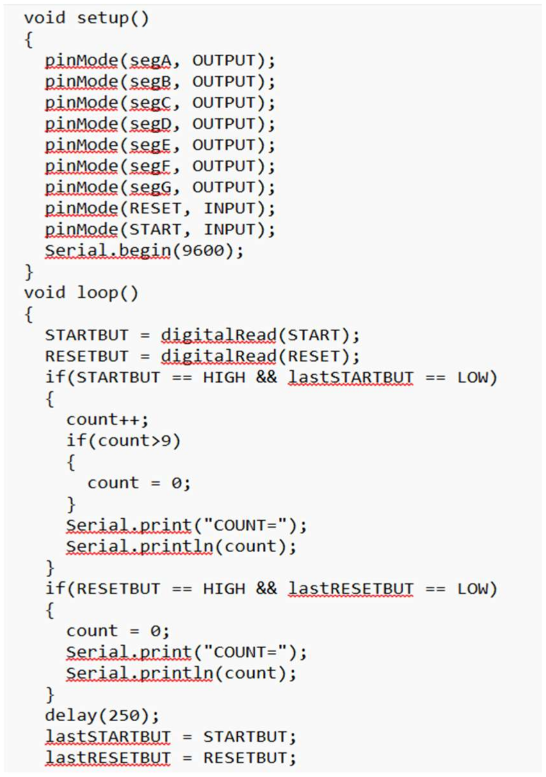


Figure F : Pin on the 7 segment display

Appendix B



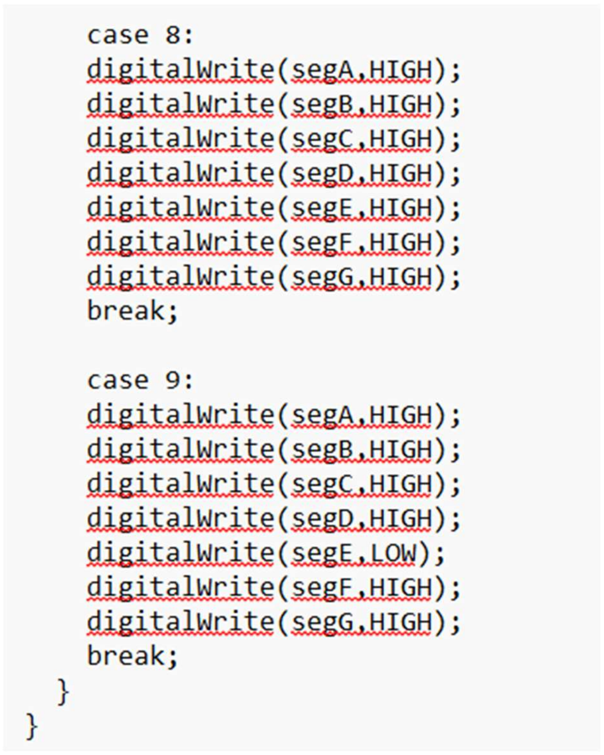
```
const int segA = 7;
const int segB = 8;
const int segC = 2;
const int segD = 3;
const int segE = 4;
const int segF = 5;
const int segG = 6;
const int RESET = 9;
const int START = 10;
int STARTBUT = 0;
int RESETBUT = 0;
int lastSTARTBUT = LOW;
int lastRESETBUT = LOW;
int count = 0;

//0= ABCDEF
//1= BC
//2= ABGED
//3=ABCDG
//4=BCFG
//5=AFGCD
//6=AFGEDC
//7=ABC
//8=ABCDEFG
//9=ABCFG
```



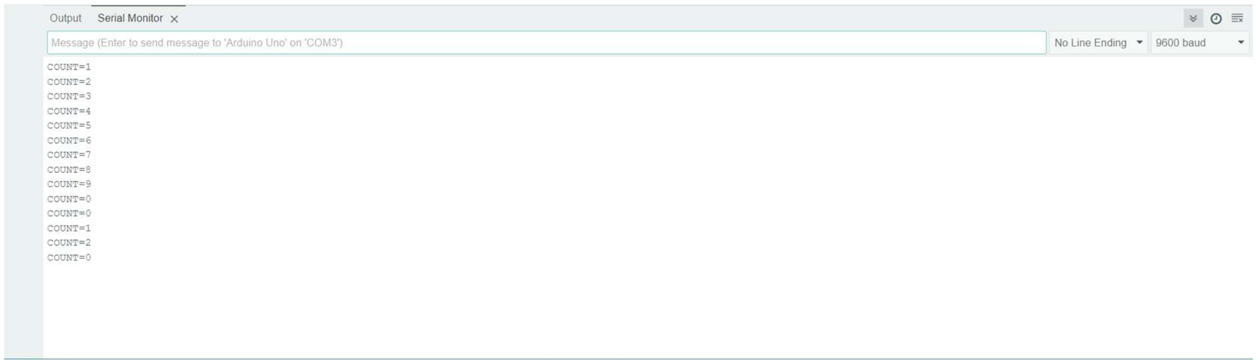
```
void setup()
{
  pinMode(segA, OUTPUT);
  pinMode(segB, OUTPUT);
  pinMode(segC, OUTPUT);
  pinMode(segD, OUTPUT);
  pinMode(segE, OUTPUT);
  pinMode(segF, OUTPUT);
  pinMode(segG, OUTPUT);
  pinMode(RESET, INPUT);
  pinMode(START, INPUT);
  Serial.begin(9600);
}
void loop()
{
  STARTBUT = digitalRead(START);
  RESETBUT = digitalRead(RESET);
  if(STARTBUT == HIGH && lastSTARTBUT == LOW)
  {
    count++;
    if(count>9)
    {
      count = 0;
    }
    Serial.print("COUNT=");
    Serial.println(count);
  }
  if(RESETBUT == HIGH && lastRESETBUT == LOW)
  {
    count = 0;
    Serial.print("COUNT=");
    Serial.println(count);
  }
  delay(250);
  lastSTARTBUT = STARTBUT;
  lastRESETBUT = RESETBUT;
```

```
switch(count)
{
  case 0:
  digitalWrite(segA,HIGH);
  digitalWrite(segB,HIGH);
  digitalWrite(segC,HIGH);
  digitalWrite(segD,HIGH);
  digitalWrite(segE,HIGH);
  digitalWrite(segF,HIGH);
  digitalWrite(segG,LOW);
  break;

  case 1:
  digitalWrite(segA,LOW);
  digitalWrite(segB,HIGH);
  digitalWrite(segC,HIGH);
  digitalWrite(segD,LOW);
  digitalWrite(segE,LOW);
  digitalWrite(segF,LOW);
  digitalWrite(segG,LOW);
  break;

  case 2:
  digitalWrite(segA,HIGH);
  digitalWrite(segB,HIGH);
  digitalWrite(segC,LOW);
  digitalWrite(segD,HIGH);
  digitalWrite(segE,HIGH);
  digitalWrite(segF,LOW);
  digitalWrite(segG,HIGH);
  break;

  case 3:
  digitalWrite(segA,HIGH);
  digitalWrite(segB,HIGH);
  digitalWrite(segC,HIGH);
  digitalWrite(segD,HIGH);
  digitalWrite(segE,LOW);
  digitalWrite(segF,LOW);
  digitalWrite(segG,HIGH);
  break;
```

```
  case 4:
  digitalWrite(segA,LOW);
  digitalWrite(segB,HIGH);
  digitalWrite(segC,HIGH);
  digitalWrite(segD,LOW);
  digitalWrite(segE,LOW);
  digitalWrite(segF,HIGH);
  digitalWrite(segG,HIGH);
  break;

  case 5:
  digitalWrite(segA,HIGH);
  digitalWrite(segB,LOW);
  digitalWrite(segC,HIGH);
  digitalWrite(segD,HIGH);
  digitalWrite(segE,LOW);
  digitalWrite(segF,HIGH);
  digitalWrite(segG,HIGH);
  break;

  case 6:
  digitalWrite(segA,HIGH);
  digitalWrite(segB,LOW);
  digitalWrite(segC,HIGH);
  digitalWrite(segD,HIGH);
  digitalWrite(segE,HIGH);
  digitalWrite(segF,HIGH);
  digitalWrite(segG,HIGH);
  break;

  case 7:
  digitalWrite(segA,HIGH);
  digitalWrite(segB,HIGH);
  digitalWrite(segC,HIGH);
  digitalWrite(segD,LOW);
  digitalWrite(segE,LOW);
  digitalWrite(segF,LOW);
  digitalWrite(segG,LOW);
  break;
```



```
    case 8:
    digitalWrite(segA,HIGH);
    digitalWrite(segB,HIGH);
    digitalWrite(segC,HIGH);
    digitalWrite(segD,HIGH);
    digitalWrite(segE,HIGH);
    digitalWrite(segF,HIGH);
    digitalWrite(segG,HIGH);
    break;

    case 9:
    digitalWrite(segA,HIGH);
    digitalWrite(segB,HIGH);
    digitalWrite(segC,HIGH);
    digitalWrite(segD,HIGH);
    digitalWrite(segE,LOW);
    digitalWrite(segF,HIGH);
    digitalWrite(segG,HIGH);
    break;
  }
}
```



```
COUNT=1
COUNT=2
COUNT=3
COUNT=4
COUNT=5
COUNT=6
COUNT=7
COUNT=8
COUNT=9
COUNT=0
COUNT=0
COUNT=1
COUNT=2
COUNT=0
```

Figure G: The output serial monitors for 7 segment display experiment

13.0 STUDENT’S DECLARATION

Certificate of Originality and Authenticity

This is to certify that we are responsible for the work submitted in this report, that **the original work** is our own except as specified in the references and acknowledgement, and that the original work contained herein has not been untaken or done by unspecified sources or persons.

We hereby certify that this report has **not been done by only one individual** and **all of us have contributed to the report**. The length of contribution to the reports by each individual is noted within this certificate.

We also hereby certify that we have **read** and **understand** the content of the total report and that no further improvement on the reports is needed from any of the individual contributors to the report.

Signature: [illegible]	Read	/
Name: AHMAD HARITH IMRAN BIN MOHD YUSOF	Understand	/
Matric No.: 2311581	Agree	/

Contribution: Assembler, abstract, material and equipment, results, recommendation and student declaration.

Signature: [illegible]	Read	/
Name: MOHAMMAD FARISH ISKANDAR BIN MOHD SYAHIDIN	Understand	/
Matric No.: 2311779	Agree	/
Contribution: Coder, table of content, experimental setup, data collection, data analysis, references and appendices.		

Signature: [illegible]	Read	/
Name: ARIFA AQILAH BINTI ABDUL HALIM	Understand	/
Matric No.: 2311530	Agree	/
Contribution: Assembler, introduction, methodology, discussion and conclusion.		