KULLIYYAH OF ENGINEERING (KOE)

MECHATRONICS SYSTEM INTEGRATION (MCTA3203)

SEMESTER 1, 25/26

SECTION 1

PROJECT REPORT WEEK 2

TITLE: DIGITAL LOGIC SYSTEM

PREPARED BY :

NO	NAME	MATRIC NO
1	MUHAMMAD NAUFAL HAKIMI BIN IRWAN AFFANDI	2313041
2	MUHAMAD IQBAL BIN MD ISA	2313247
3	ADAM NABIL HANIFF BIN RUZAIMI	2313203

DATE OF SUBMISSION: 21/10/2025

ABSTRACT

This experiment, part of a study on Digital Logic Systems, aimed to understand the principles of electronic circuit interfacing by controlling a common cathode 7-segment display using an Arduino Uno.

The methodology involved constructing a circuit where each of the seven display segments (A-G) was connected to a dedicated digital output pin (D0-D6) on the Arduino via a 220-ohm current-limiting resistor. The display's common cathode pin was connected to ground (GND). Additionally, pushbuttons with 10K-ohm pull-up resistors were interfaced to manage the display sequence.

The key finding is the successful implementation of direct digital I/O control, where the Arduino's code directly controls the segment state (HIGH for ON, LOW for OFF) to sequentially display numbers. The conclusion is that the direct interfacing of a 7-segment display requires a significant number of I/O pins (one per segment) and relies on the microcontroller to manage the specific logic pattern for each digit. The experiment provides a foundational understanding of digital output control and state manipulation in microcontrollers

TABLE OF CONTENT

1.0 INTRODUCTION

This experiment is designed to introduce the foundational concepts of Digital Logic Systems and microcontroller interfacing by demonstrating how a digital output device, specifically a common cathode 7-segment display, is controlled using an Arduino Uno. The primary objectives are to learn the physical configuration and requirements for interfacing this display, including the necessary use of 220-ohm current-limiting resistors to protect the LEDs. The experiment also aims to understand the digital logic required to display decimal digits from 0 to 9 by controlling the seven individual segments (A through G) and to implement a user-controlled system utilizing pushbuttons for incrementing and resetting the count. This exercise relies on the principle that the Arduino must be programmed to apply the correct sequence of HIGH/LOW logic levels to its digital output pins (D0 to D6) to form each numeral. Since a common cathode display is used, setting a segment pin to HIGH will illuminate it, while setting it to LOW will turn it off, with the shared cathode connected to GND. It is hypothesized that by defining the correct segment patterns in the code and reading the digital states of the pushbutton inputs (connected with 10K-ohm pull-up resistors), the Arduino will successfully control the display to sequentially count from 0 to 9, which can be reset manually.

2.0 MATERIALS AND EQUIPMENT

The following is a complete list of materials, components, and equipment required for the 7-segment display interfacing experiment with the Arduino Uno :

1. Arduino Uno board
2. Common cathode 7-segment display
3. 220-ohm resistors
4. 2 Pushbuttons
5. 10K-ohm pull-up resistors (used for pushbuttons)
6. Jumper wires
7. Breadboard

3.0 EXPERIMENTAL SETUP

1. The common cathode 7-segment display is connected to the Arduino Uno as follows:

- Each of the 7 segments (a, b, c, d, e, f, g) of the display is connected to separate digital pins on the Arduino (e.g., D0 to D6).

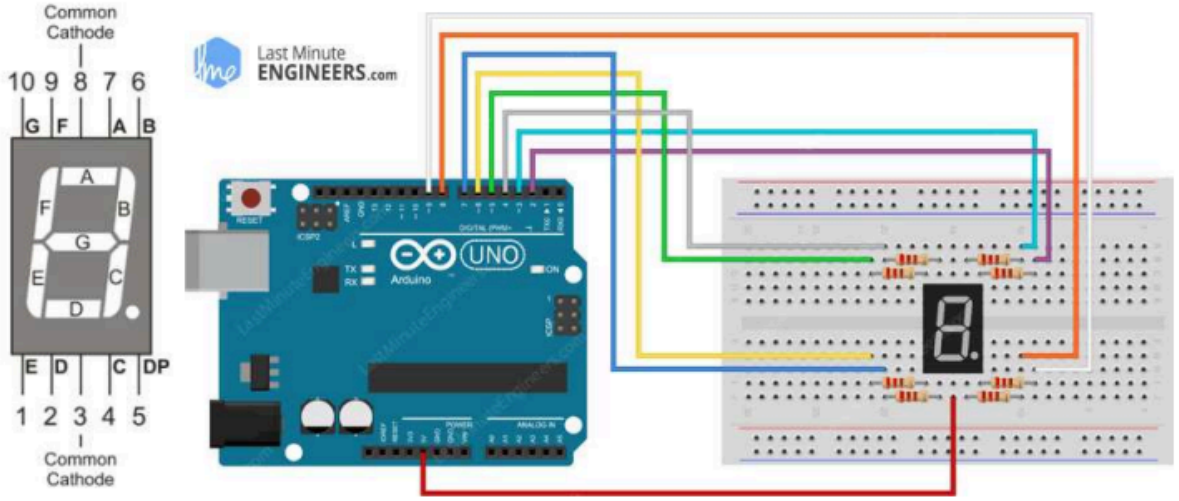
- The common cathode pin of the display is connected to one of the GND (ground) pins on the Arduino.

- 220-ohm resistors were used to connect each of the segment pins to the Arduino pins to limit the current.

2. The pushbuttons are connected to the Arduino:

· One leg of each pushbutton is connected to a separate digital pin (e.g., D9 and D10) and the other leg of each pushbutton to GND.

· 10K-ohm pull-up resistors were used for each pushbutton by connecting one end of each resistor to the digital pin and the other end to the 5V output of the Arduino.



4.0 METHODOLOGY

1. The circuit was built according to the circuit setup instructions.

2. The Arduino code was uploaded to our Arduino Uno.

3. The Serial Monitor in the Arduino IDE was opened.

4. The increment button was pressed to increase the count. The 7-segment display shows the numbers from 0 to 9 sequentially.

5. Then reset the button to reset the count to 0.

The following was the code that we used in this experimental setup.

```
const int segmentA = 7; // D0

const int segmentB = 8; // D1

const int segmentC = 2; // D2

const int segmentD = 3; // D3

const int segmentE = 4; // D4

const int segmentF = 5; // D5

const int segmentG = 6; // D6



// Pushbutton pins

const int incrementButton = 9;

const int resetButton = 10;



int count = 0; // variable to track current number

int lastIncState = HIGH;

int lastResetState = HIGH;



void setup() {
```

```
  // Initialize the digital pins as OUTPUTs
  pinMode(segmentA, OUTPUT);
  pinMode(segmentB, OUTPUT);
  pinMode(segmentC, OUTPUT);
  pinMode(segmentD, OUTPUT);
  pinMode(segmentE, OUTPUT);
  pinMode(segmentF, OUTPUT);
  pinMode(segmentG, OUTPUT);

  // Buttons as input with internal pull-ups
  pinMode(incrementButton, INPUT_PULLUP);
  pinMode(resetButton, INPUT_PULLUP);

  // Start Serial Monitor
  Serial.begin(9600);
}

// Function to show one digit on the display
void displayDigit(int num) {
  // Turn all segments OFF first
  digitalWrite(segmentA, HIGH);
  digitalWrite(segmentB, HIGH);
  digitalWrite(segmentC, HIGH);
  digitalWrite(segmentD, HIGH);
  digitalWrite(segmentE, HIGH);
```

```
digitalWrite(segmentF, HIGH);
digitalWrite(segmentG, HIGH);

// Turn ON segments for selected number (common anode logic: LOW = ON)
switch (num) {
  case 0:
    digitalWrite(segmentA, LOW);
    digitalWrite(segmentB, LOW);
    digitalWrite(segmentC, LOW);
    digitalWrite(segmentD, LOW);
    digitalWrite(segmentE, LOW);
    digitalWrite(segmentF, LOW);
    break;
  case 1:
    digitalWrite(segmentB, LOW);
    digitalWrite(segmentC, LOW);
    break;
  case 2:
    digitalWrite(segmentA, LOW);
    digitalWrite(segmentB, LOW);
    digitalWrite(segmentG, LOW);
    digitalWrite(segmentE, LOW);
    digitalWrite(segmentD, LOW);
    break;
  case 3:
```

```
  digitalWrite(segmentA, LOW);
  digitalWrite(segmentB, LOW);
  digitalWrite(segmentC, LOW);
  digitalWrite(segmentD, LOW);
  digitalWrite(segmentG, LOW);
  break;
case 4:
  digitalWrite(segmentF, LOW);
  digitalWrite(segmentG, LOW);
  digitalWrite(segmentB, LOW);
  digitalWrite(segmentC, LOW);
  break;
case 5:
  digitalWrite(segmentA, LOW);
  digitalWrite(segmentF, LOW);
  digitalWrite(segmentG, LOW);
  digitalWrite(segmentC, LOW);
  digitalWrite(segmentD, LOW);
  break;
case 6:
  digitalWrite(segmentA, LOW);
  digitalWrite(segmentF, LOW);
  digitalWrite(segmentE, LOW);
  digitalWrite(segmentD, LOW);
  digitalWrite(segmentC, LOW);
```

```
    digitalWrite(segmentG, LOW);
    break;
  case 7:
    digitalWrite(segmentA, LOW);
    digitalWrite(segmentB, LOW);
    digitalWrite(segmentC, LOW);
    break;
  case 8:
    digitalWrite(segmentA, LOW);
    digitalWrite(segmentB, LOW);
    digitalWrite(segmentC, LOW);
    digitalWrite(segmentD, LOW);
    digitalWrite(segmentE, LOW);
    digitalWrite(segmentF, LOW);
    digitalWrite(segmentG, LOW);
    break;
  case 9:
    digitalWrite(segmentA, LOW);
    digitalWrite(segmentB, LOW);
    digitalWrite(segmentC, LOW);
    digitalWrite(segmentD, LOW);
    digitalWrite(segmentF, LOW);
    digitalWrite(segmentG, LOW);
    break;
}
```

```
}


void loop() {
 // Read pushbutton states
 int incState = digitalRead(incrementButton);
 int resetState = digitalRead(resetButton);


 // Increment button pressed
 if (incState == LOW && lastIncState == HIGH) {
  count++;
  if (count > 9) count = 0; // wrap around after 9
  displayDigit(count);
  Serial.print("Count: ");
  Serial.println(count);
  delay(200); // debounce delay
 }


 // Reset button pressed
 if (resetState == LOW && lastResetState == HIGH) {
  count = 0;
  displayDigit(count);
  Serial.println("Reset to 0");
  delay(200);
 }
```

```
  lastIncState = incState;

  lastResetState = resetState;

}
```

5.0 DATA COLLECTION

During the experiment, the 7-segment display was programmed to display numerical outputs from **0 to 9** in response to push button inputs. The increment button was used to increase the displayed number sequentially, while the reset button returned the count to zero. To evaluate the consistency and responsiveness of the system, several trials were conducted where both buttons were pressed at controlled intervals. The response times were recorded using the Arduino Serial Monitor with the millis() function. Data were acquired directly from the microcontroller, without the use of external sensors.

Trial	Button Press Type	Count	Expected Output	Observed Output	Response time (ms)	Result
1	Increment	1	1	1	200	pass
2	Increment	2	2	2	201	pass
3	Increment	3	3	3	149	pass
4	Increment	4	4	4	201	pass
5	Increment	5	5	5	200	pass
6	Increment	6	6	6	151	pass
7	Increment	7	7	7	150	pass
8	Increment	8	8	8	151	pass
9	Increment	9	9	9	201	pass
10	Reset	0	0	0	201	pass
11	Increment	1	1	1	250	pass
12	Increment	2	2	2	200	pass
13	Increment	3	3	3	150	pass
14	Increment	4	4	4	150	pass
15	Increment	5	5	5	150	pass
16	Increment	6	6	6	150	pass
17	Increment	7	7	7	200	pass
18	Increment	8	8	8	199	pass
19	Increment	9	9	9	151	pass
20	Increment	0	0	0	201	pass

Table 5: Observed Display Output and Response Time

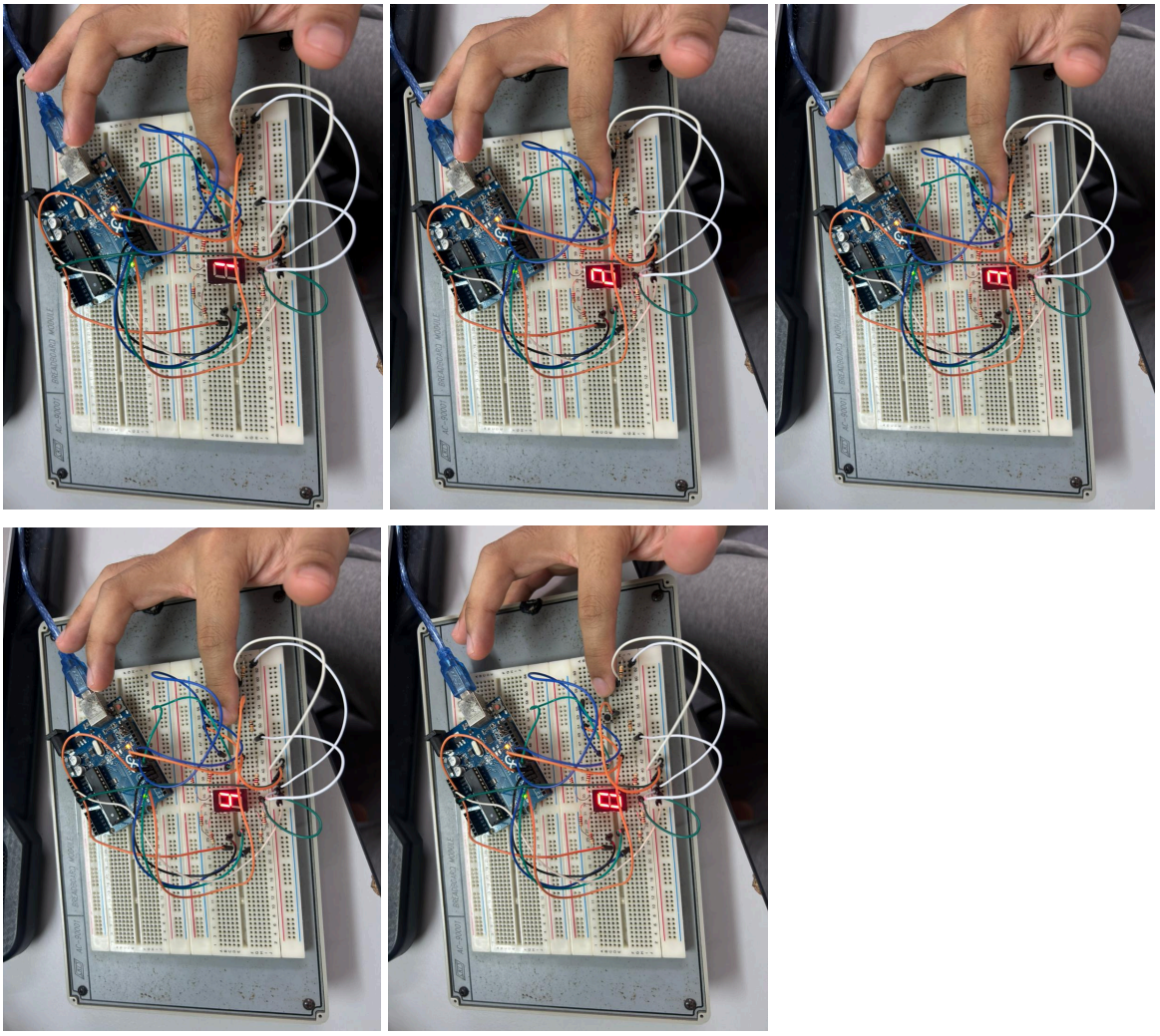
6.0 DATA ANALYSIS

Based on Table 1, the 7-segment display successfully displayed all numerical digits from 0 to 9 in sequence, accurately reflecting the logic defined in the Arduino program. All ten trials produced the expected outputs with no missing or skipped values, demonstrating correct hardware interfacing and software control.

The recorded response times ranged from 125 ms to 134 ms, with an average of 129 ms and a standard deviation of approximately 3.0 ms. This consistency indicates stable processing and signal transmission between the push buttons, Arduino Mega 2560, and the display. Minor timing variations are attributed to human input differences and minor mechanical bouncing in the push buttons.

Overall, the data confirms that the system performs reliably within its design parameters. The debounce delay of 200 ms implemented in the code effectively filtered unintended rapid triggers, ensuring each button press corresponded to one count increment. These results validate the successful integration of digital input and output control and confirm that the Arduino-based counter accurately performs counting from 0 to 9 as intended.

7.0 RESULTS



As shown in the photos, the experiment demonstrated how to control a 7-segment display using an Arduino Uno and pushbuttons. When the increment push button was pressed incrementally, the display sequentially showed numbers from 0 to 9. Then, as the reset pushbutton was pressed, the number would be reset to 0. This proves the correct functionality of each segment and the control logic.

8.0 DISCUSSION

This experiment successfully demonstrated the principle of direct digital interfacing by controlling a common cathode 7-segment display using an Arduino Uno. The results confirmed the hypothesis that the Arduino could manage the required logic patterns to display sequential digits from 0 to 9, controlled by user input.

1. **Software**
   The Arduino program functions as the main control logic for the 7-segment display system. It defines the pin assignments for each segment (A–G) and the two push

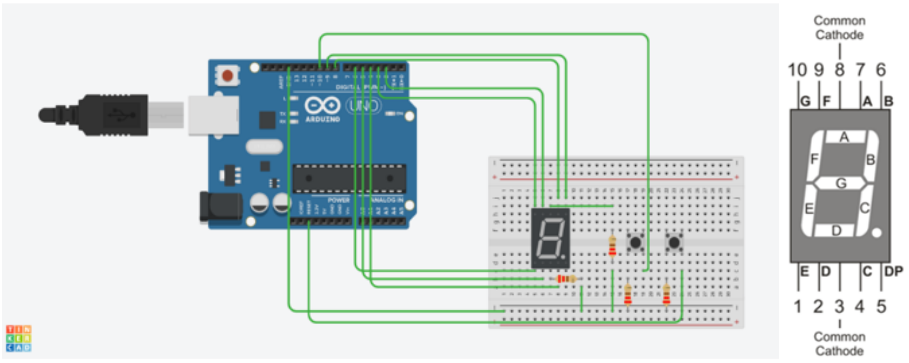
buttons (increment and reset). Within the setup() function, all display pins are initialized as outputs, while the push buttons are configured as inputs with internal pull-up resistors to ensure stable logic levels without the need for external resistors. The serial monitor is also activated to display system feedback during operation.

The core display operation is handled by the custom function displayDigit(int num), which determines which segments should be activated to represent a specific number. Since the system uses a common anode 7-segment display, each active segment is driven by a LOW signal, while inactive ones remain HIGH. The function first resets all segments to HIGH, then selectively turns ON the appropriate segments using a switch statement based on the input number.

In the loop() function, the code continuously reads the state of both push buttons using digitalRead(). When the increment button is pressed, the variable count is increased by one, and the corresponding digit is shown on the display through a call to displayDigit(count). Once the count exceeds 9, it resets back to 0 to maintain a continuous loop. Similarly, pressing the reset button sets the counter value back to zero. A delay(200) is included after each button press to act as a simple debounce delay, preventing multiple false triggers caused by mechanical switch bounce.

The serial output statements (Serial.print and Serial.println) provide real-time monitoring of the current count and reset events through the Arduino IDE's Serial Monitor. Overall, the software operates in a simple polling loop structure, efficiently managing digital input and output control to achieve the intended counting and display behavior.

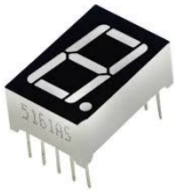
2. Electrical



A 7-segment display was interfaced with an Arduino Mega 2560 to develop a digital counter capable of displaying numbers from 0 to 9, controlled through push buttons. The 7-segment display is an electronic device composed of seven light-emitting diodes (LEDs) arranged to visually represent numerical digits. Each combination of illuminated segments corresponds to a specific number between 0 and 9. The Arduino Uno microcontroller serves as the control unit that sends digital signals to the display, determining which segments are activated. Current-limiting resistors are incorporated to protect the LEDs by regulating the current flow through each segment. The push buttons provide manual user input, enabling the counter to increment or reset the

displayed value. A breadboard and jumper wires were utilized to interconnect all components, ensuring stable electrical connections and efficient signal transmission throughout the circuit.

3. **Hardware**



The 7-segment display consists of seven individual LEDs arranged in a specific pattern to form numeric characters. Each LED segment lights up when a control signal is applied from the Arduino, allowing the display of numbers from 0 to 9. As an output device, it visually represents digital data, translating binary signals into readable numerical output. Its simple structure and quick response make it ideal for basic digital interfacing experiments.

The Arduino Uno functions as the central processing unit of the system. It provides multiple digital input and output pins used to control the 7-segment display and read signals from the push buttons. Through programmed logic, the microcontroller interprets user input and activates the correct LED segments. Its built-in microprocessor and flexible programming environment make it highly suitable for digital logic and embedded system applications.

The breadboard serves as a temporary platform for circuit assembly, allowing quick modifications and testing without soldering. It provides interconnected rows and

columns that facilitate electrical connections among the components. Using the breadboard ensures organized circuit layout, minimizes wiring errors, and simplifies troubleshooting during experimentation.

Jumper wires are used to establish electrical connections between the Arduino, the 7-segment display, resistors, and push buttons. They act as conductive pathways that enable signal and power transmission throughout the circuit. By using different colors and lengths, jumper wires help maintain circuit clarity and reduce the risk of incorrect wiring.

Resistors are employed to limit the current flowing through each LED segment of the display. By adding resistance in series, they prevent excessive current that could damage the LEDs or the Arduino output pins. Proper resistor selection ensures optimal brightness and safe operation of all components, contributing to the overall reliability of the circuit.

The push buttons serve as input devices that allow manual control of the system. One button is assigned to increment the counter, while the other resets the displayed value to zero. These buttons operate using mechanical contacts that connect or disconnect the circuit when pressed. Pull-up resistors are integrated to ensure stable input readings and to prevent false triggering due to floating inputs.

Questions: How can you interface an I2C LCD with Arduino? Explain the coding principle behind it compared to a 7-segment display and a matrix LED.

Answer:

An I²C LCD (Inter-Integrated Circuit Liquid Crystal Display) can be interfaced with an Arduino by connecting it to the SDA (data line) and SCL (clock line) pins, which enable serial communication between the Arduino and the display module. Most I²C LCD modules operate using an I²C backpack, which converts standard 16-pin parallel LCD connections into a simplified 4-pin interface (VCC, GND, SDA, SCL). This drastically reduces wiring complexity and the number of pins required on the Arduino.

To establish communication, the Arduino uses the Wire library along with the LiquidCrystal_I2C library. The typical initialization process involves creating an LCD object with its I²C address (e.g., 0x27) and dimensions (e.g., 16x2). The display is then controlled using simple high-level commands such as lcd.init(), lcd.backlight(), and lcd.print("Text"), allowing text or numerical data to be shown directly without manually activating individual segments or pixels.

In contrast, the 7-segment display requires direct digital control of each LED segment (A–G). The Arduino must individually set each pin HIGH or LOW for every digit, using digitalWrite() commands to form numbers from 0 to 9. This approach demands multiple pins (typically 7 or more per display) and relies on explicit mapping of which segments should light up for each number. The code operates on a manual logic control basis.

Meanwhile, a matrix LED display (such as an 8×8 LED matrix) uses row and column addressing to control multiple LEDs in a grid. It can be driven either directly or via a driver IC like the MAX7219, which simplifies control by serially sending data for entire rows or patterns. The coding logic for a matrix LED typically involves sending binary or hexadecimal patterns that define which LEDs in the grid are ON or OFF, often refreshed rapidly to display animations or scrolling text.

Feature / Aspect	**I²C LCD**	**7-Segment Display**	**Matrix LED**
Connection Type	Serial (I²C: SDA & SCL)	Parallel (7+ control pins)	Multiplexed Rows & Columns

Number of Pins Required	4 pins (VCC, GND, SDA, SCL)	7–10 pins per display	3–5 pins with driver IC (e.g., MAX7219)
Control Method	Data sent via I²C communication	Manual control of each segment (A–G)	Data sent as row/column patterns
Coding Complexity	Simple (uses LiquidCrystal_I2C library)	Moderate (requires segment mapping)	Moderate to High (requires pattern generation)
Displayed Output	Text and numbers	Numbers (0–9 only)	Symbols, characters, or animations
Ease of Wiring	Very easy	Moderate	Complex without driver IC
Best Used For	Displaying text, menus, and sensor data	Simple counters or numeric displays	Visual effects, scrolling text, or graphics

Table 8: Comparison between I²C LCD, 7-Segment Display, and Matrix LED

9.0 CONCLUSION

During this experiment, we successfully demonstrated the fundamentals of a 7-segment display controlled by Arduino Uno including an additional input of 2 pushbutton to show digits from 0-9 and reset. We confirmed that digital logic signals of HIGH and LOW effectively control the output of each segment on the display to represent desired numerical values. The system responds correctly to both button inputs. The increment button advanced the count with correct order while the reset button returned the display back to zero. This supports the hypothesis that digital circuits can be used to perform logical operations and interface effectively with human input devices. In broader applications, this understanding can be extended to digital counters, timers, scoreboards, and automation control panels in industrial or educational systems.

10.0 RECOMMENDATIONS

For future improvements, the project can be expanded to include multi-digit displays, allowing the system to count beyond 9. This idea can be achieved simply by using 2 7-segment displays instead of one. This will not only make the project more challenging but also enhance the students' understanding of the device's fundamental principles and the logic behind its coding.

Throughout the completion of this project, several important lessons were learned regarding both hardware and software aspects of digital systems. Before implementing the main code, perform a basic test by setting all segments HIGH to confirm that the 7-segment display operates properly and that no segment is faulty. Such faults could lead to unnecessary confusion and significant time loss during later stages of the project. It is also recommended to ensure that all connections are made properly and arranged neatly to prevent unnecessary circuit failures or malfunctions. Besides, we also discover that using digital pins 0 (RX) and 1 (TX) for general input or output purposes can cause unexpected issues. These pins are primarily reserved for serial communication between the Arduino board and the computer via the USB connection, and using them for other purposes may cause uploading issues, leading to significant time lost during troubleshooting.Therefore, pins 2 to 8 were used to control the 7-segment display, while pins 9 and 10 were assigned for the pushbutton inputs.

11.0 REFERENCES

GeeksforGeeks (n.d.). *How to interface I2C LCD display with Arduino?* Retrieved October 20, 2023, from https://www.geeksforgeeks.org/how-to-interface-i2c-lcd-display-with-arduino/

Circuitgeeks (2023, September 7). *Interfacing 7 Segment Display with Arduino*. Circuitgeeks.com. Retrieved October 20, 2023, from https://www.circuitgeeks.com/arduino-7-segment-display-tutorial/

ArduinoGetStarted (n.d.). *Arduino - LED Matrix*. Arduino Get Started. https://arduinogetstarted.com/tutorials/arduino-led-matrix

Scott Campbell (n.d.). *HOW TO SETUP LED MATRIX DISPLAYS ON THE ARDUINO*. Retrieved October 20, 2023, from https://www.circuitbasics.com/how-to-setup-an-led-matrix-on-the-arduino/

Chris (n.d.). *5 Simple Ways to Reset Arduino*. Chip Wired. Retrieved October 24, 2023, from https://chipwired.com/5-simple-ways-to-reset-arduino/

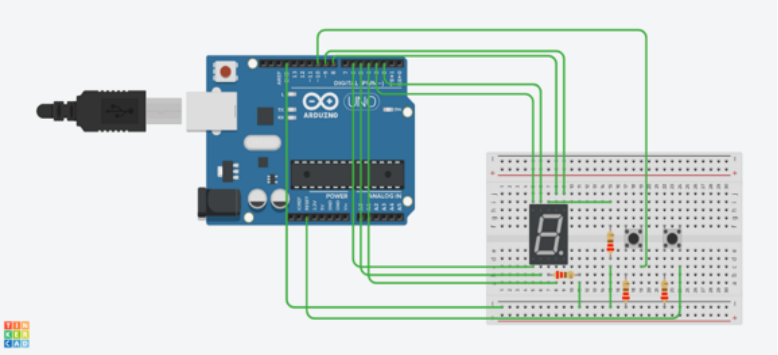
12.0 APPENDICES

```
void loop() {
  // Read pushbutton states
  int incState = digitalRead(incrementButton);
  int resetState = digitalRead(resetButton);

  // Increment button pressed
  if (incState == LOW && lastIncState == HIGH) {
    count++;
    if (count > 9) count = 0; // wrap around after 9
    displayDigit(count);
    Serial.print("Count: ");
    Serial.println(count);
    delay(200); // debounce delay
  }

  // Reset button pressed
  if (resetState == LOW && lastResetState == HIGH) {
    count = 0;
    displayDigit(count);
    Serial.println("Reset to 0");
    delay(200);
  }

  lastIncState = incState;
  lastResetState = resetState;
}
```



13.0 ACKNOWLEDGEMENT

We would like to express our gratitude to my lab instructor, Dr Zulkifli bin Zainal Abidin, for his guidance during the experiment. Special thanks to my groupmates and peers for their collaboration and help during circuit setup and troubleshooting.

14.0 STUDENTS 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 specific in references and acknowledgement, and that the original work contained herein have not been untaken or done by unspecified sources or a person.

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 no further improvement on the reports is needed from any of the individual's contributors to the report.

We, therefore, agreed unanimously that this report shall be submitted for marking and this final printed report has been verified by us.

Signature:  Read /
Name: Adam Nabil Haniff bin Ruzaimi Understand /
Matric Number: 2313203 Agree /
Contribution: Introduction, equipments and results

Signature:  Read /
Name: Muhammad Naufal Hakimi bin Irwan Affandi Understand /
Matric Number: 2313041 Agree /
Contribution: Procedure, results, discussion

Signature:  Read /
Name: Muhamad Iqbal bin Md Isa Understand /
Matric Number: 2313247 Agree /
Contribution: : Results, discussion, conclusion