



MECHATRONICS SYSTEM INTEGRATION (MCTA 3203)

SEMESTER 1, 2025/2026

WEEK 2: DIGITAL LOGIC DESIGN

SECTION 1

GROUP 20

LECTURER: DR. WAHJU SEDIONO

DATE OF EXPERIMENT: Monday, 13th October 2025

DATE OF SUBMISSION: Monday, 20th October 2025

PREPARED BY:

NAME	MATRIC NUMBER
ALI RIDHA BIN MUHAMMAD AMINUDIN	2316137
SITI NORATHIRAH BINTI RAZALLY	2319788
NUR ALYA SAKINAH BINTI MOHD MOKHTAR	2319444

ABSTRACT

This experiment focuses on interfacing an Arduino Uno with a common cathode 7-segment display and two push buttons, one for increasing the count and another for resetting it. The main goal is to display digits from 0 to 9, where each press of the increment button advances the number by one, and the reset button brings the display back to 0. The process involved both hardware setup and software coding, allowing the Arduino to detect button inputs, update the displayed number, or reset it as required. The program utilized digital input readings and conditional statements to control the output display. The outcome verified that the connection between hardware and software functioned successfully, showcasing the capability of microcontrollers in digital counting tasks. Overall, this experiment emphasizes the significance of organized coding, proper circuit construction, and effective troubleshooting key skills in embedded systems applications such as digital counters, timers, and automation devices.

TABLE OF CONTENTS

1.0 INTRODUCTION	4
2.0 MATERIALS AND EQUIPMENTS	5
3.0 EXPERIMENTAL SETUP	6
4.0 METHODOLOGY	7
4.1 Circuit Assembly	8
4.2 Programming Logic	9
4.3 Code Used	10
4.4 Control Algorithm	11
5.0 DATA COLLECTION	12
6.0 DATA ANALYSIS	13
7.0 RESULT	14
8.0 DISCUSSION	16
9.0 CONCLUSION	18
10.0 RECOMMENDATIONS	19
11.0 REFERENCES	20
12.0 APPENDICES	21
ACKNOWLEDGEMENTS	23
Certificate of Originality and Authenticity	24

1.0 INTRODUCTION

This experiment aims to interface an Arduino Uno with a common cathode 7-segment display, operated using two push buttons one for incrementing the numerical value and the other for resetting the display. The objective is to create a simple digital counter capable of displaying digits from 0 to 9, where each press of the increment button increases the count by one, and the reset button returns it to zero. This project demonstrates the practical application of microcontroller-based display control, which is widely used in digital counters, clocks, and other embedded systems.

A 7-segment display serves as a visual indicator that represents numerical digits using seven individual LED segments arranged in a fixed pattern. Each segment lights up in specific combinations to display numbers from 0 to 9. In this experiment, a common cathode type is used, where all cathode pins are connected to ground (GND), and each segment is activated by applying a HIGH (5V) signal from the Arduino. The two push buttons function as input devices that allow user interaction. When pressed, the increment button increases the displayed digit, while the reset button clears the display to zero.

The experiment incorporates several key concepts in digital electronics and microcontroller programming:

1. **Binary and Digital Logic** — The Arduino processes HIGH and LOW signals to control each LED segment.
2. **Microcontroller Programming** — The system utilizes `digitalRead()` to detect button inputs and `digitalWrite()` to drive the 7-segment display.
3. **State Machines and Counters** — The Arduino maintains a counter variable that updates based on button inputs, similar to how counters function in real electronic devices.

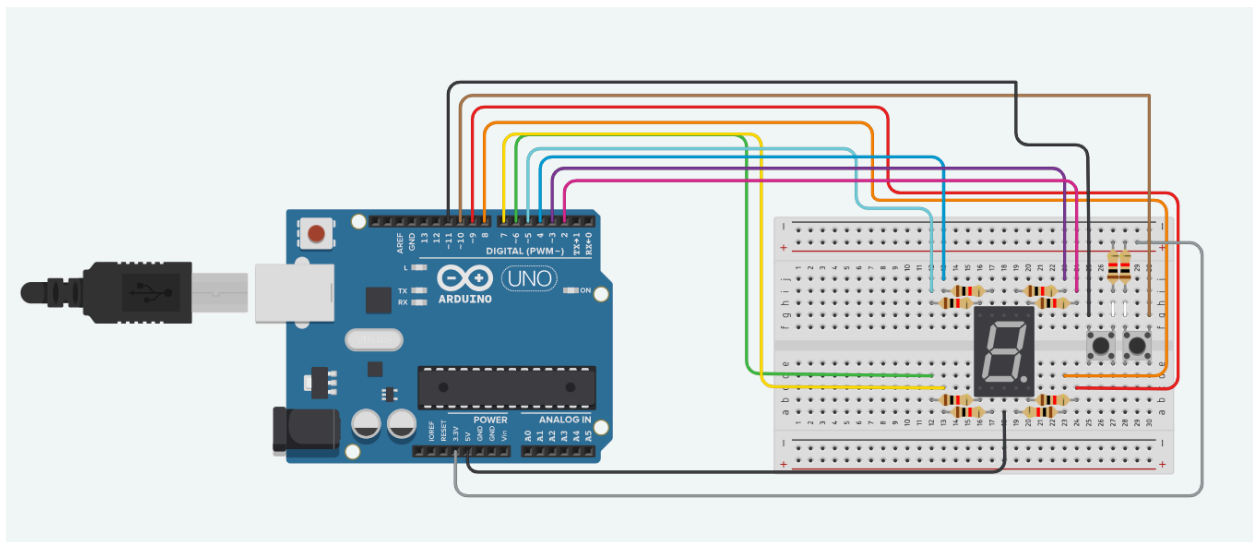
It is anticipated that the Arduino Uno will successfully control the 7-segment display, enabling sequential digit incrementation with each press of the increment button and a full reset to zero with the reset button. Additionally, the display should correctly represent each digit while implementing proper debouncing to prevent multiple unintended counts.

2.0 MATERIALS AND EQUIPMENTS

- Arduino Uno board
- Common cathode 7-segment display
- 10k-ohm resistor (2)
- 220-ohm resistor (7)
- push buttons (2)
- breadboard
- jumpers

3.0 EXPERIMENTAL SETUP

- Each pin on a common cathode 7-segment display is connected separately to digital pins from D0 to D7 on the Arduino Mega 2560 board.
- 220-ohm resistors are connected between Arduino Mega 2560 board and common cathode 7-segment display to limit the current.
- One leg of each Push button is connected to different digital pins on the Arduino Mega 2560 board, one connected to D8 as an increment button and another one is connected to D9 as a reset button.
- Another leg of the push buttons is connected to GND.
- Use a 10k-ohm pull up resistor for each push button. Connect one leg to digital pin and another one leg to 5V output on the Arduino Mega 2560.



(This circuit is just a simulation using TinkerCad. Please note that this is not the real circuit since TinkerCad does not have Arduino Mega 2560, we are using the Arduino Uno only.)

4.0 METHODOLOGY

1. The circuit was built according to the circuit setup instructions.
2. The Arduino code is uploaded into the Arduino Mega 2560.
3. Pressing the increment count button displayed numbers from 0 to 9 on the 7-segment display.
4. Press again and record the outcome until 9.
5. To see the outcome of the rest, the button was pressed and it turned to 0.

4.1 Circuit Assembly

- The 7-segment display is connected to the Arduino Uno with individual segment pins (D0 - D7) assigned to specific digital output pins.
- Two push buttons are used, one for incrementing the displayed number and another for resetting it.
- Internal pull-up resistors are enabled for the buttons to ensure stable readings.

4.2 Programming Logic

- The Arduino code reads button states to detect presses.
- A counter variable stores the displayed number, increasing upon a button press.
- When the reset button is pressed, the counter resets to zero.
- A function `displayDigit(int digit)` is used to control the 7-segment display output.

4.3 Code Used

```
// Segment pins
const int segmentA = 0;
const int segmentB = 1;
const int segmentC = 2;
const int segmentD = 3;
const int segmentE = 4;
const int segmentF = 5;
const int segmentG = 6;
const int segmentH = 7;

// Button pins
const int buttonIncrement = 8;
const int buttonReset = 9;
```

```

// State variables
int currentNumber = 0;
bool lastIncrementState = HIGH;
bool lastResetState = HIGH;

// Digit segment map: A-G for digits 0-9 (0 = ON, 1 = OFF for
common cathode)
const bool digitMap[10][8] = {
    {LOW, LOW, LOW, LOW, HIGH, LOW, LOW, LOW}, // 0
    {LOW, LOW, HIGH, HIGH, HIGH, HIGH, HIGH, LOW}, // 1
    {LOW, HIGH, LOW, LOW, LOW, HIGH, LOW, LOW}, // 2
    {LOW, LOW, LOW, HIGH, LOW, HIGH, LOW, LOW}, // 3
    {LOW, LOW, HIGH, HIGH, LOW, LOW, HIGH, LOW}, // 4
    {LOW, LOW, LOW, HIGH, LOW, LOW, LOW, HIGH}, // 5
    {LOW, LOW, LOW, LOW, LOW, LOW, LOW, HIGH}, // 6
    {LOW, LOW, HIGH, HIGH, HIGH, HIGH, LOW, LOW}, // 7
    {LOW, LOW, LOW, LOW, LOW, LOW, LOW, LOW}, // 8
    {LOW, LOW, LOW, HIGH, LOW, LOW, LOW, LOW} // 9
};

void setup() {
    // Set up segment pins
    pinMode(segmentA, OUTPUT);
    pinMode(segmentB, OUTPUT);
    pinMode(segmentC, OUTPUT);
    pinMode(segmentD, OUTPUT);
    pinMode(segmentE, OUTPUT);
    pinMode(segmentF, OUTPUT);
    pinMode(segmentG, OUTPUT);
    pinMode(segmentH, OUTPUT);

    // Set up button pins
    pinMode(buttonIncrement, INPUT_PULLUP); // Use pull-up
resistors
    pinMode(buttonReset, INPUT_PULLUP);

    // Display initial number
    displayDigit(currentNumber);
}

```

```

void loop() {
    // Read button states (LOW = pressed)
    bool incrementState = digitalRead(buttonIncrement);
    bool resetState = digitalRead(buttonReset);

    // Handle increment button press
    if (incrementState == LOW && lastIncrementState == HIGH) {
        currentNumber++;
        if (currentNumber > 9) currentNumber = 0;
        displayDigit(currentNumber);
        delay(200); // Simple debounce delay
    }

    // Handle reset button press
    if (resetState == LOW && lastResetState == HIGH) {
        currentNumber = 0;
        displayDigit(currentNumber);
        delay(200); // Simple debounce delay
    }

    // Save the current state for next loop
    lastIncrementState = incrementState;
    lastResetState = resetState;
}

// Function to display a digit on the 7-segment
void displayDigit(int digit) {
    digitalWrite(segmentA, digitMap[digit][0]);
    digitalWrite(segmentB, digitMap[digit][1]);
    digitalWrite(segmentC, digitMap[digit][2]);
    digitalWrite(segmentD, digitMap[digit][3]);
    digitalWrite(segmentE, digitMap[digit][4]);
    digitalWrite(segmentF, digitMap[digit][5]);
    digitalWrite(segmentG, digitMap[digit][6]);
    digitalWrite(segmentH, digitMap[digit][7]);
}

```

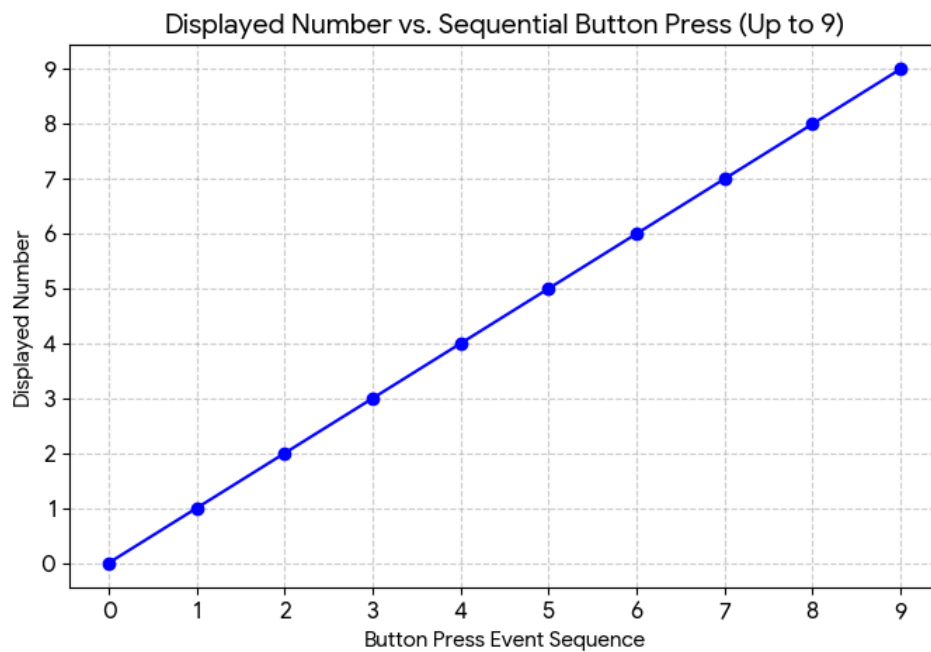

4.4 Control Algorithm

1. Defining pins
 - a. Initialize the LEDs to light up on the 7-segment display by defining Segment A - G. Example: `const int segmentA = 0;`
 - b. Button Increments (D8) and Button Rest (D9) are connected to digital pins.
2. Global variables
 - a. `int currentNumber` stores the number shown on the 7-segment display.
 - b. `bool lastIncrementState` and `bool lastResetState` are used for edge detection, which is detecting a new button press.
3. Setup function
 - a. 7- segment display Pins to be configured as `OUTPUT`
 - b. Buttons to be configured as `INPUT_PULLUP`
*`INPUT_PULLUP` makes the pin HIGH by default (button pressed and goes LOW when pressed).
4. Main loop:
 - a. `bool incrementState = digitalRead(buttonIncrement)` and `bool resetState = digitalRead(buttonReset)` are used to read the current state of both buttons.
 - b. Increment button logic
 - i. To check button press and the count will increase, it will debounce a delay for 200ms to prevent false triggering.
 - ii. It keeps increasing until 9 and turns back to 0
 - c. Reset button logic
 - If the button rest is pressed, the count value is set to 0.
 - d. Update button state
 - Updates previous button states for edge detection
 - e. Display number
 - i. Calls the function to update the displayed count on the 7-segment display.
`displayDigit(currentNumber);`
 - All segments turn off first before displaying a new number
 - Each number is mapped to the corresponding segments
 - The necessary segments are set to LOW to form the shape of the number

5.0 DATA COLLECTION

Observations:

Button Press	Displayed Number
Increment	1
Increment	2
Increment	3
Increment	4
Increment	5
Increment	6
Increment	7
Increment	8
Increment	9
Increment	0 (Wraps around)
Reset	0



6.0 DATA ANALYSIS

- The number increases sequentially as expected when the button is pressed
- It's showing the number of times pressed will show the same number of display on the LCD 7-segment display
- A debounce delay of 200 ms prevents unintended multiple triggers
- The reset button sets the display back to 0

7.0 RESULT

The system successfully displayed numbers from 0 to 9 on the seven-segment display using button control. Each press of the increment button increased the displayed digit by one, and after reaching 9, the display returned to 0. The reset button functioned accurately by resetting the display to zero at any point. The implemented debouncing delay effectively minimized false triggering and ensured smooth transitions between digits.

How to Interface an I2C LCD with Arduino

Interfacing an I2C LCD with Arduino is more efficient compared to the 7-segment display because it requires only two communication lines to transmit data which are SDA (Serial Data) and SCL (Serial Clock). This reduces the number of I/O pins used and simplifies the circuit wiring.

Steps to interface an I2C LCD with Arduino

1. Connect VCC of the LCD to 5V on the Arduino.
2. Connect SDA and SCL pins to the corresponding A4 and A5 analog pins (for most Arduino boards).
3. Connect the GND of the LCD to Arduino GND.
4. Install the LiquidCrystal_I2C library in the Arduino IDE.
5. Write the Arduino code to initialize and display text on the LCD.

Code to sketch 'Hello World' on the I2C LCD

- Call the library
`#include <LiquidCrystal_I2C.h>` : This library allows the Arduino to communicate with the LCD through the I2C interface.
- Define the I2C address and display size
Assign the I2C address and dimensions (e.g., 16 columns × 2 rows) to the display:
`LiquidCrystal_I2C lcd(0x3F, 16, 2);`
- Initialize and prepare the LCD
Use the following functions to start the LCD and prepare it for displaying text:
 - `lcd.init()` → Initializes the LCD module.
 - `lcd.clear()` → Clears the screen and resets the cursor position.
 - `lcd.backlight()` → Turns on the LCD backlight for better visibility.

- **Set cursor position and display message**

Position the text by using `lcd.setCursor(column, row)`.

For example: `lcd.setCursor(2, 0)`; sets the cursor at column 2, row 0.

- **Print text on the display**

Use the print function to show the desired message:

`lcd.print("Hello World");`

Explain the coding principle behind it compared with 7-Segments display and LED Matrix

Feature	I2C LCD	7-Segments Display	LED Matrix
Communication	I2C (2 wires: SDA & SCL)	Parallel (multiple digital pins)	Multiplexed (row/column scanning)
Control Library	LiquidCrystal_I2C.h	Manual segment control	Depend on type of LED Matrix board <ul style="list-style-type: none"> • MD_Parola.h • Arduino_LED_Matrix.h
Pins Required	2	7+ (for single digit)	8+ (for 8×8 matrix)
Text Capability	Yes (custom text)	No (numeric only 0–9)	High (text and graphics)

The I2C LCD provides the easiest and most effective way to display text since it uses serial communication and a dedicated library, which simplifies the programming process. It requires fewer pins and can display text along with symbols. On the other hand, the 7-segment display is only suitable for numeric output because each digit will have to be driven manually using HIGH and LOW values. The LED matrix is capable of scrolling messages or animation but is more complicated when it comes to connections and coding.

Overall, the I2C LCD is the easiest in programs requiring readable text output, while the 7-segment display is suitable for simple numeric displays.

8.0 DISCUSSION

Expected vs. observed outcomes:

Function	Expected Outcomes	Observed Outcomes
Increment button	Number increases by 1	Works as expected
Reset button	Display resets to 0	Works as expected
Wrap around	9 to 0	Works as expected

Sources of error:

- Button Debouncing Issues
 - When the push button was pressed, multiple signals might have been detected due to contact bounce.
 - Although a short delay (debouncing) was added in the code, insufficient delay time could still cause the display to skip or repeat numbers unintentionally.
- Loose or Faulty Connections
 - Inaccurate readings may occur if jumper wires or breadboard connections were loose.
 - Poor contact between the Arduino pins and the display module could lead to some segments not lighting up properly.
- Incorrect Wiring or Pin Mapping
 - A mismatch between the defined pin numbers in the code and the actual wiring on the breadboard may cause wrong segments to illuminate or the LCD to fail to display.
- Code Logic or Timing Errors
 - The delay used for debouncing might not have been fully optimized, causing lag or delayed response when pressing buttons.
 - Incorrect use of array indexing in the 7-segment digit mapping could display wrong digits if not handled properly.

Improvements:

- Implement an interrupt-based button reading system for improved responsiveness.
- Ensure all jumper wires and connections are properly secured to avoid unstable signals and replace them with the long wire to make an effective structure of wiring.
- Verify wiring and pin mapping to match the program configuration and display correctly.
- Optimize program logic and delay timing for accurate and responsive output.

9.0 CONCLUSION

From this experiment, we have successfully demonstrated the practical implementation of digital logic systems through the interfacing of a 7-segment display with an Arduino. The push buttons allowed us to control the numbers shown on the display, which helped us understand how digital signals interact with electronic components. This project also showed several important aspects of circuit wiring, debouncing, and microcontroller programming. One of the key lessons we discovered was the importance of proper pin mapping and stable connections to ensure the display works accurately. This experiment also provided us with hands-on experience in troubleshooting, identifying wiring issues, and improving circuit performance. By the end of this experiment, we successfully developed a strong foundation for more advanced digital applications such as multi-digit counters or automated display systems. We can conclude that this experiment strengthened both the theoretical concepts learned in class and the practical skills gained in the lab, making it a valuable learning experience.

10.0 RECOMMENDATIONS

The improvement that we can make to improvise this experiment is to use a 12C LCD instead of a 7-segment display, which can help in displaying the number more efficiently. Other than that, by adding a debounce system for push buttons will prevent accidental multiple presses, this will make the counting more accurate. We can also use a multiplexing method, which can allow controlling multiple 7-segment displays with fewer pins, which will then make it easier for us. Lastly, the simplest but one of the most important things to improve is the length of the wire we use. The shorter the length of the wire, the easier it is to make mistakes. We believe these improvements will make the system easier and more reliable to use while learning to understand the advanced digital logic concepts.

11.0 REFERENCES

Arduino | 36, K. P. |. (2017, May 23). *How to Set up Seven Segment Displays on the Arduino*. Circuit Basics.

<https://www.circuitbasics.com/arduino-7-segment-display-tutorial/>

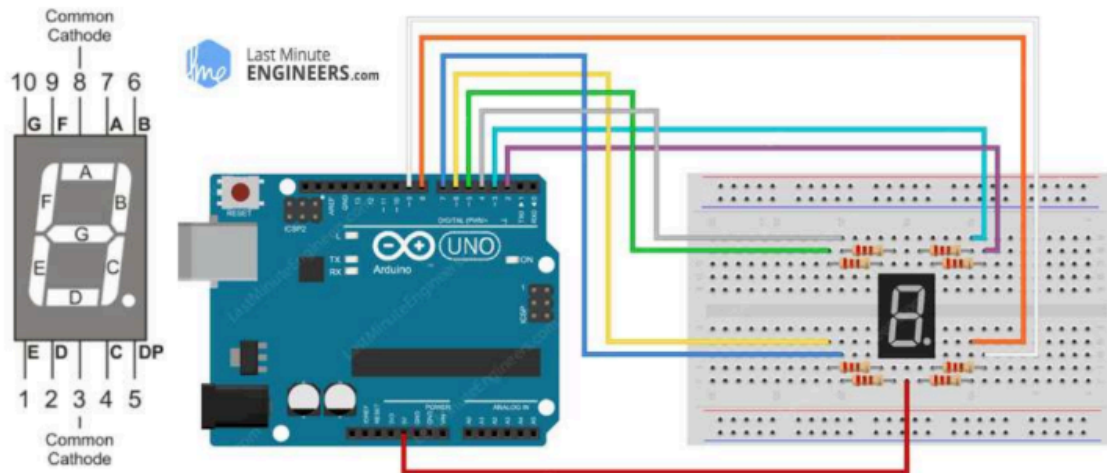
7-segment Display Counter Tutorial. (2020, March 16). Basic Electronics Tutorials.

<https://www.electronics-tutorials.ws/counter/7-segment-display.html>

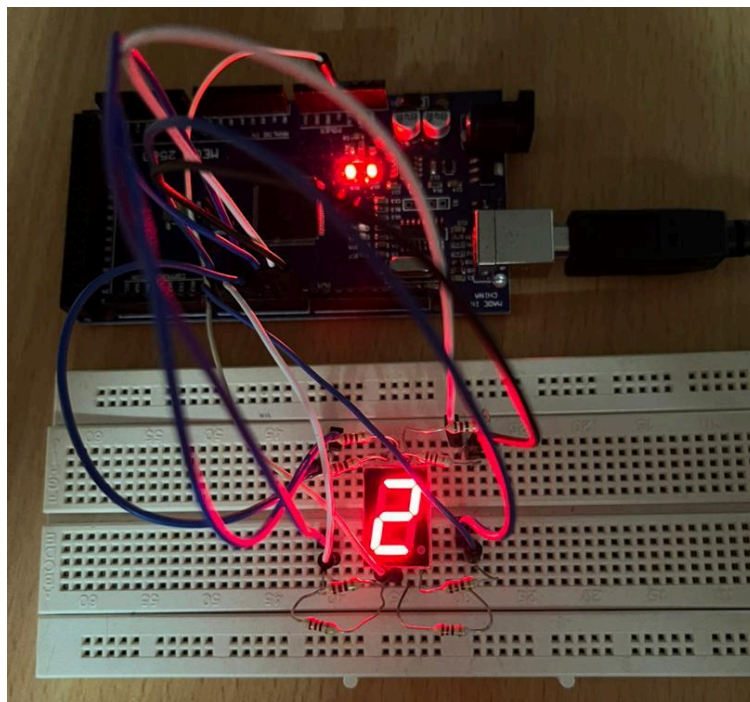
stannano. (2021, August 18). *One Digit 7-Segment LED Display*. Projecthub.arduino.cc.

<https://projecthub.arduino.cc/stannano/one-digit-7-segment-led-display-819bcd>

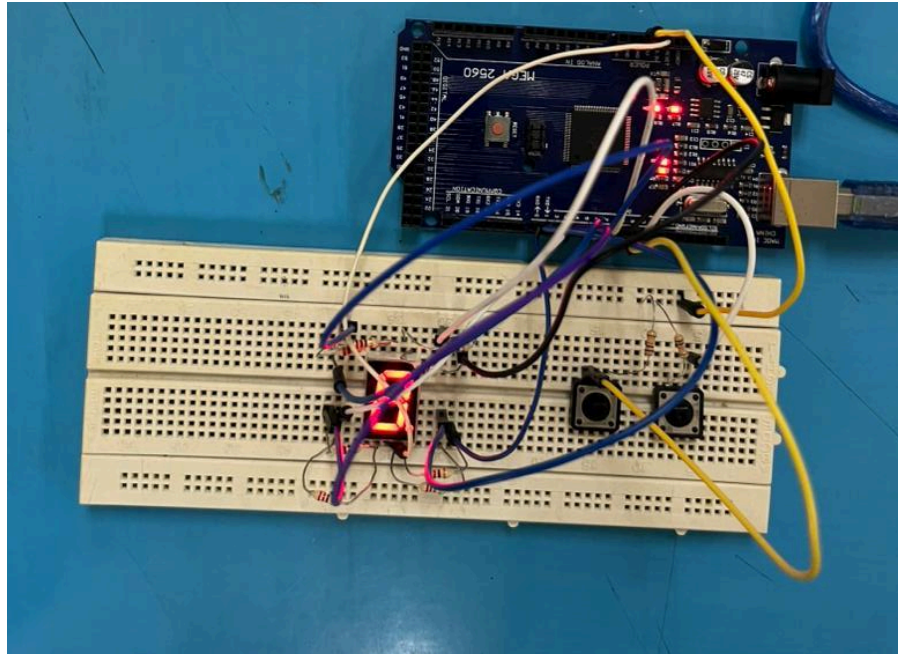
12.0 APPENDICES



Part A: 7-Segments without push button



Part B: 7-Segments with push buttons (Increment and reset push buttons)




ACKNOWLEDGEMENTS

The completion of this project is made possible through the guidance and support of the lecturer (DR. WAHJU SEDIONO), laboratory instructor, and teammates. Appreciation is extended to the institution for providing the necessary equipment and facilities to conduct the experiment successfully

Certificate of Originality and Authenticity

We hereby certify that we are responsible for the work presented in this report. The content is our original work, except where proper references and acknowledgements are made. We confirm that no part of this report has been completed by anyone not listed as a contributor. We also certify that this report is the result of group collaboration and not the effort of a single individual. The level of contribution by each member is stated in this certificate. Furthermore, we have read and understood the entire report, and we agree that no further revisions are required. We collectively approve this final report for submission and confirm that it has been reviewed and verified by all group members.

Signature: 
Name: ALI RIDHA BIN MUHAMMAD AMINUDIN
Matric Number: 2316137

Read [/]
Understand [/]
Agree [/]

Signature: 
Name: SITI NORATHIRAH BINTI RAZALLY
Matric Number: 2319788

Read [/]
Understand [/]
Agree [/]

Signature: 
Name: NUR ALYA SAKINAH BINTI MOHD MOKHTAR
Matric Number: 2319444

Read [/]
Understand [/]
Agree [/]