MECHATRONICS SYSTEM INTEGRATION

MCTA 3203

LAB 1: DIGITAL LOGIC SYSTEM

SECTION 1

SEMESTER 2, 2024/2025

INSTRUCTOR:

ASSOC. PROF. EUR. ING. IR. TS. GS. INV. DR. ZULKIFLI BIN ZAINAL ABIDIN

DR. WAHJU SEDIONO

GROUP 9

NAME	MATRIC NO
AMER WAFDI BIN ZAINIZAM	2228703
AMALIN BALQIS BINTI RUYUSNI	2314684
FARHA QAISARA BINTI MOHD ZAHIDI	2315994

DATE OF SUBMISSION: 17 MARCH 2025

ABSTRACT

This experiment demonstrates the interfacing of a common cathode 7-segment display with an Arduino Mega 2560 and its manual control using pushbuttons. The primary objective is to display numbers from 0 to 9 sequentially using programmed logic and button interactions. The experiment involved wiring the display to the microcontroller, uploading the control code, and implementing push button-based counting and resetting mechanisms. Data collection confirmed accurate digit representation and stable 200ms intervals between transitions. The actual display output matched the expected output in every instance, confirming the correctness of the seven-segment display logic. Analysis highlighted the impact of switch bouncing, resistor variations, and direct pin control inefficiencies, with software debouncing mitigating button inconsistencies. The results showed precise segment activation and reliable numerical output, reinforcing digital logic and embedded system principles. Despite minor limitations, the experiment successfully demonstrated microcontroller-based display control, emphasizing its practical applications in digital counters, timers, and embedded systems. This hands-on approach enhances understanding of circuit design, programming, and electronic component integration.

TABLE OF CONTENT **3**

1.0 INTRODUCTION

This experiment aims to demonstrate how to interface a 7-segment display with an Arduino Mega 2560 and manually control it using pushbuttons. The primary objectives are to understand the working principle of a 7-segment display and to learn how to interface it with an Arduino Mega 2560. Then, this experiment also aims to control the display manually using push buttons to display numbers 0 to 9 sequentially.

A 7-segment display is a widely used electronic display device for representing decimal numerals. It consists of seven LEDs (segments) arranged in a specific pattern that can be selectively illuminated to form numerical digits. These displays are commonly found in digital clocks, electronic meters, and other devices requiring numerical output.

In this experiment, the 7-segment display will be interfaced with an Arduino Mega 2560, a microcontroller board based on the ATmega2560, which provides multiple digital and analog I/O pins for external connections. By programming the Arduino, the display can be controlled to show different numbers. Additionally, pushbuttons will be used to manually cycle through digits 0 to 9, allowing for hands-on control of the display without relying solely on automated programming.

The expected outcome of this experiment is to successfully display numbers on the 7-segment display using both the Arduino Mega 2560 and the manual push buttons interface. This will provide practical experience in working with microcontroller-based circuits and reinforce the understanding of digital logic control in embedded systems.

2.0 MATERIALS AND EQUIPMENT

The materials and equipment that we used in this experiment are:

a) Arduino Mega 2560 board

b) Common cathode 7-segment display

c) 220-ohm resistors - 8 pieces

d) 10k-ohm resistors - 2 pieces

e) Pushbuttons - 2 pieces

f) Jumper wires

g) Breadboard

3.0 EXPERIMENTAL SETUP

Here are the steps of equipment and components that were set up for the experiment:

1. Connect the common cathode 7-segment display to the Arduino Mega 2560.
2. Connect each of the 7 segments (a, b, c, d, e, f, g) of the display to separate digital pins D2 to D8 on the Arduino using 220-ohm resistors to limit the current.
3. Connect the common cathode pin of the display to the GND pin on the Arduino.
4. Connect the pushbuttons to the Arduino by connecting one leg of each pushbutton to a separate digital pins D10 and D11, and the other leg of each pushbutton to the GND pin.
5. Use 10K-ohm pull-up resistors 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.

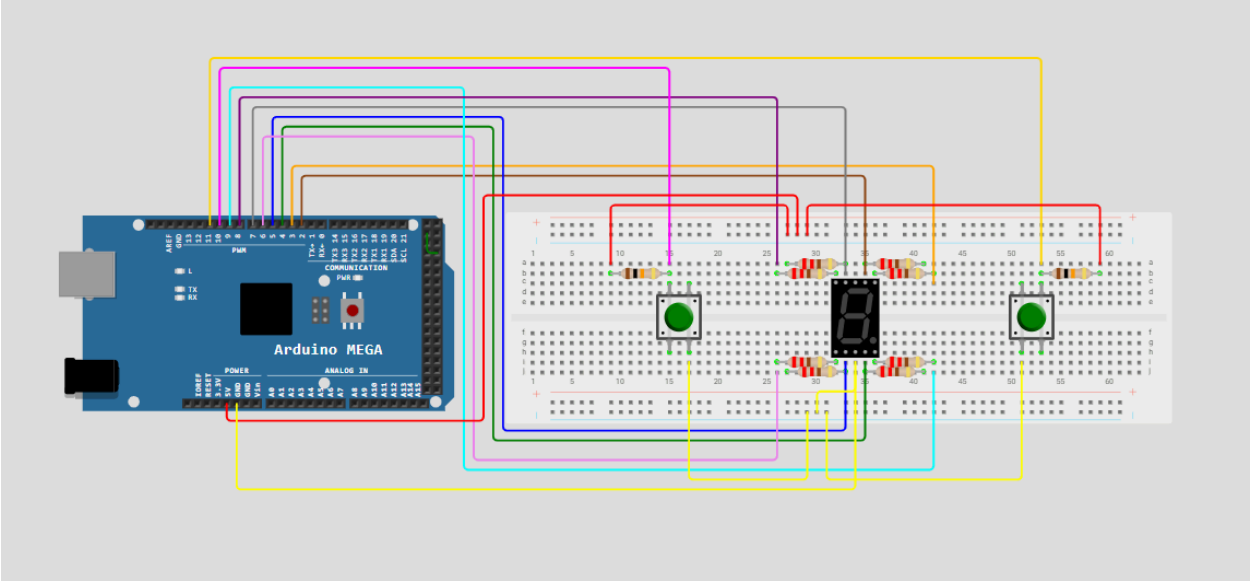


Figure 1: Schematic diagram

4.0 METHODOLOGY

Here are the steps followed during the experiment:

1. Build the circuit and set up the Arduino Mega 2560.
2. Set up the programming code in Arduino IDE and upload the code to Arduino Mega 2560.
3. Test the code by pressing the increment button to increase the count from 0 to 9 at 7-segment display and press the reset button to reset the count to 0.

The programming code used:

```
// Define the pins for each segment (D2 to D8)
const int segmentA = 2; // D2
const int segmentB = 3; // D3
const int segmentC = 4; // D4
const int segmentD = 5; // D5
const int segmentE = 6; // D6
const int segmentF = 7; // D7
const int segmentG = 8; // D8
```

```
// Define button pins
const int countupButton = 10; // Button to increment the count
const int resetButton = 11;     // Button to reset the count

int count = 0; // Current count

// Button states
bool lastButtonStateCountup = HIGH;
bool lastButtonStateReset = 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);

  // Initialize the button pins as INPUTs with pull-up resistors
  pinMode(countupButton, INPUT_PULLUP);
  pinMode(resetButton, INPUT_PULLUP);
}

// 0 = A,B,C,D,E,F
// 1 = B,C
// 2 = A,B,G,E,D
// 3 = A,B,C,D,G
// 4 = F,G,B,C
// 5 = A,F,G,C,D
// 6 = A,F,E,D,C,G
// 7 = A,B,C
// 8 = A,B,C,D,E,F,G
// 9 = A,B,C,D,F,G
void loop() {
  bool currentButtonStateCountup = digitalRead(countupButton);
  bool currentButtonStateReset = digitalRead(resetButton);

  // Detect button press (LOW means pressed due to pull-up)
  if (currentButtonStateCountup == LOW && lastButtonStateCountup == HIGH) {
    count++; // Increment count
    if (count > 9) {
      count = 0; // Wrap around if count exceeds 9
    }
    delay(200); // Debounce delay
```

```
  }

  if (currentButtonStateReset == LOW && lastButtonStateReset == HIGH) {
    count = 0; // Reset count to 0
    delay(200); // Debounce delay
  }

  // Update last button state
  lastButtonStateCountup = currentButtonStateCountup;
  lastButtonStateReset = currentButtonStateReset;

  // Display the current count on the 7-segment display
  displayNumber(count);
}

// Function to display a number on the 7-segment display
void displayNumber(int number) {
  // Reset all segments first
  digitalWrite(segmentA, LOW);
  digitalWrite(segmentB, LOW);
  digitalWrite(segmentC, LOW);
  digitalWrite(segmentD, LOW);
  digitalWrite(segmentE, LOW);
  digitalWrite(segmentF, LOW);
  digitalWrite(segmentG, LOW);

  if (number == 0) {
    digitalWrite(segmentA, HIGH);
    digitalWrite(segmentB, HIGH);
    digitalWrite(segmentC, HIGH);
    digitalWrite(segmentD, HIGH);
    digitalWrite(segmentE, HIGH);
    digitalWrite(segmentF, HIGH);
  }

  else if (number == 1) {
    digitalWrite(segmentB, HIGH);
    digitalWrite(segmentC, HIGH);
  }

  else if (number == 2) {
    digitalWrite(segmentA, HIGH);
    digitalWrite(segmentB, HIGH);
    digitalWrite(segmentD, HIGH);
    digitalWrite(segmentE, HIGH);
    digitalWrite(segmentG, HIGH);
```

```
}

else if (number == 3) {
  digitalWrite(segmentA, HIGH);
  digitalWrite(segmentB, HIGH);
  digitalWrite(segmentC, HIGH);
  digitalWrite(segmentD, HIGH);
  digitalWrite(segmentG, HIGH);
}

else if (number == 4) {
  digitalWrite(segmentB, HIGH);
  digitalWrite(segmentC, HIGH);
  digitalWrite(segmentF, HIGH);
  digitalWrite(segmentG, HIGH);
}

else if (number == 5) {
  digitalWrite(segmentA, HIGH);
  digitalWrite(segmentC, HIGH);
  digitalWrite(segmentD, HIGH);
  digitalWrite(segmentF, HIGH);
  digitalWrite(segmentG, HIGH);
}

else if (number == 6) {
  digitalWrite(segmentA, HIGH);
  digitalWrite(segmentC, HIGH);
  digitalWrite(segmentD, HIGH);
  digitalWrite(segmentE, HIGH);
  digitalWrite(segmentF, HIGH);
  digitalWrite(segmentG, HIGH);
}


else if (number == 7) {
  digitalWrite(segmentA, HIGH);
  digitalWrite(segmentB, HIGH);
  digitalWrite(segmentC, HIGH);
}

else if (number == 8) {
  digitalWrite(segmentA, HIGH);
  digitalWrite(segmentB, HIGH);
  digitalWrite(segmentC, HIGH);
  digitalWrite(segmentD, HIGH);
```

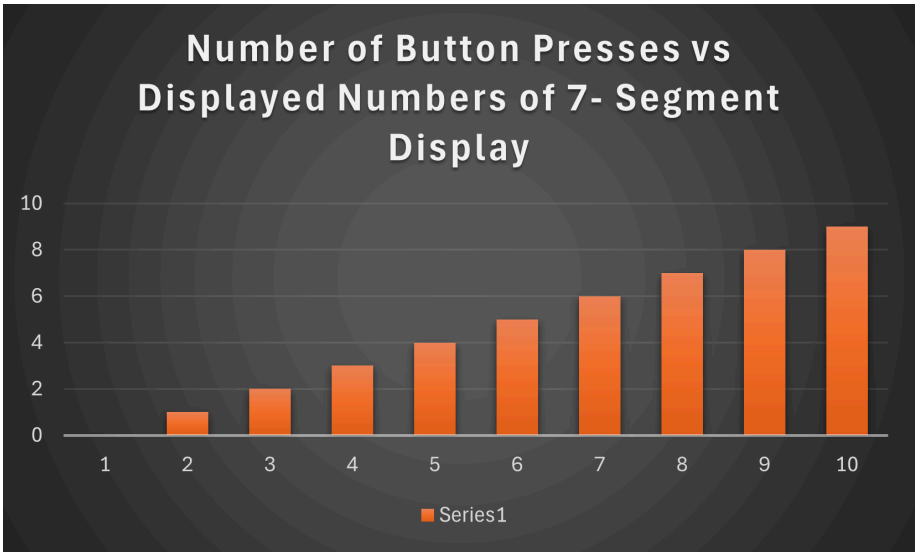
```
    digitalWrite(segmentE, HIGH);
    digitalWrite(segmentF, HIGH);
    digitalWrite(segmentG, HIGH);
  }

  else if (number == 9) {
    digitalWrite(segmentA, HIGH);
    digitalWrite(segmentB, HIGH);
    digitalWrite(segmentC, HIGH);
    digitalWrite(segmentD, HIGH);
    digitalWrite(segmentF, HIGH);
    digitalWrite(segmentG, HIGH);
  }
}
```

5.0 DATA COLLECTION

Segment Inputs							Display Output
a	**b**	**c**	**d**	**e**	**f**	**g**	
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

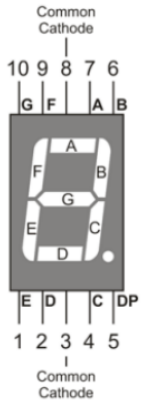
Table 1: Segment inputs and display output



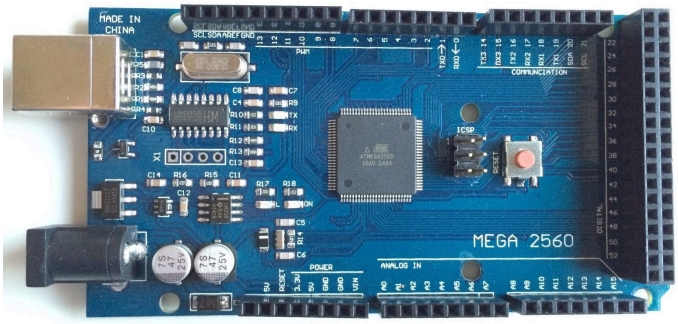
Bargraph 1: Number of Button Presses vs Displayed Numbers of 7- Segment Display

Instruments used for data acquisition:

1) Common cathode 7-segment display: Displays numeric values from 0 to 9 based on the input signal.



2) Arduino Mega 2560: A microcontroller board based on the ATmega2560, used to process input signals from pushbuttons and control the 7-segment display.



3) Push buttons: The count-ip button increments the displayed number each time it is pressed from 0 to 9 and the reset button resets the display to zero.

4) Resistors: 220 ohms resistors used to limit the current flowing to each segment of the 7-segment display and preventing damage to the LEDs. While 10k ohms resistors used as pull-up resistors for the pushbuttons to ensure stable HIGH signals when the buttons are not pressed.
5) Jumper wires: Used to establish electrical connections between components on the breadboard.
6) Breadboard: A prototyping platform used to connect components without soldering, allowing for easy circuit modifications.

6.0 DATA ANALYSIS

Based on the collected data, we can confirm that a common cathode 7-segment display operates by controlling individual segment inputs, where a high signal (1) activates a segment, and a low signal (0) turns it off. By systematically controlling these inputs, numerical digits from 0 to 9 can be displayed effectively.

Statistical analysis and error considerations involve identifying potential issues such as faulty wiring, incorrect resistor values, or errors in code logic if a segment fails to turn on or off as expected. Additionally, button press timing analysis helps in debouncing detection, preventing unnecessary double counts because of fast oscillations.

The experiment successfully demonstrates the working principle of a common cathode 7-segment display and its interfacing with an Arduino Mega 2560. By controlling individual segment inputs, numerical digits from 0 to 9 were displayed, confirming the relationship between digital signals and segment activation. The integration of pushbuttons allowed for manual control, enabling sequential counting and resetting functions. Statistical analysis of button presses ensured reliable operation. The experiment also highlighted the importance of current-limiting resistors to protect the display and microcontroller. Overall, the findings reinforce fundamental concepts in embedded systems, digital logic, and microcontroller-based hardware control, providing valuable hands-on experience in circuit design and programming.

7.0 RESULTS

The experiment successfully demonstrated the functionality of the seven-segment display in response to button presses. The display correctly incremented from 0 to 9 and then reset to 0. The recorded time interval for each transition was consistent at 200 milliseconds, indicating stable performance.

Button Press	Expected Display Output	Actual Display Output	Time Interval (ms)
Increment Button (1st)	1	1	200
Increment Button (2nd)	2	2	200
Increment Button (3rd)	3	3	200
Increment Button (4th)	4	4	200
Increment Button (5th)	5	5	200
Increment Button (6th)	6	6	200
Increment Button (7th)	7	7	200
Increment Button (8th)	8	8	200
Increment Button (9th)	9	9	200
Increment Button (10th)	0	0	200
Reset Button	0	0	200

Table 2: Expected and actual display output of the 7-segment display with the time interval.

The actual display output matched the expected output in every instance, confirming the correctness of the seven-segment display logic. The time interval also remained constant at 200ms, ensuring smooth transitions between digits. There is no unexpected behavior, such as skipped numbers or flickering, was observed.

8.0 DISCUSSION

The experiment demonstrated that the seven-segment display correctly responded to button presses, incrementing digits from 0 to 9 and looping back to 0 as expected. The system maintained a stable 200 ms interval between increments, indicating consistent processing and display updates. The expected and actual outputs matched perfectly, showing that the circuit and microcontroller functioned as intended. Increment and reset were provided through pushbuttons, reinforcing digital logic, microcontroller programming, and hardware interfacing. The experiment validated that segments light up with a HIGH (1) signal and turn off with LOW (0), confirming binary control principles in embedded systems. The accurate response of the display also confirms the proper implementation of the button input logic, microcontroller processing, and display output. The constant time interval suggests efficient code execution and no noticeable processing delays.

Although the experiment met its objectives and there were no mismatches between expected and observed outputs, however some inconsistencies were observed. Occasionally, segments failed to light up or displayed unstable behavior when incrementing numbers using the pushbutton. This discrepancy was primarily due to bouncing effects in mechanical pushbuttons, where unintended multiple counts occurred due to rapid transitions between HIGH and LOW states. Additionally, minor variations in resistor values affected segment brightness, and in some

cases, incorrect wiring connections led to display errors. These inconsistencies were mitigated by implementing software-based debouncing and verifying circuit connections, improving overall accuracy.

Sources of error in the experiment were switch bouncing, where mechanical push buttons caused multiple spurious counts, making it inaccurate. This was countered by software debouncing, but a supplementary hardware solution such as capacitors or Schmitt triggers would have made it more dependable. Variations in resistor values led to erratic segment brightness since decreasing resistance increased brightness but also increased current consumption. Also, direct pin control of the 7-segment display required multiple digital pins, thus less efficient than employing shift registers like the 74HC595 or display drivers like the MAX7219. Another limitation was the restricted display capability, since only numeric characters could be shown by 7-segment displays, in contrast to I2C LCDs or LED matrices, which can display text and graphics. These problems indicate the need for optimized circuit design and other display technologies in more sophisticated applications.

Questions:

How to interface an l2C LCD with an Arduino? Explain the coding principle behind it compared with 7 segments display and matrix LED.

To interface an I2C LCD with an Arduino, the first step is to connect the LCD module to the Arduino board using only four connections: VCC to 5V and GND to GND for power, while SDA (Serial Data) and SCL (Serial Clock) are connected to the designated I2C pins on the Arduino, which are A4 (SDA) and A5 (SCL) on an Arduino Uno, or pins 20 (SDA) and 21 (SCL) on an Arduino Mega.

Once the hardware is connected, the next step is to install the LiquidCrystal_I2C library in the Arduino IDE, which simplifies communication between the Arduino and the LCD. After installing the library, initializing the display is required by specifying its I2C address, which is usually 0x27 or 0x3F, followed by enabling the backlight and setting the cursor position for displaying text. With the display initialized, data can be printed onto the screen using simple commands. This method significantly reduces wiring complexity compared to traditional parallel LCDs, making it more efficient for embedded systems projects.

Coding Principle Comparison: I2C LCD vs. 7-Segment Display vs. LED Matrix

Each display type works differently in terms of coding. I2C LCDs use the I2C protocol, which requires only two pins (SDA & SCL) and allows easy text display using simple functions like lcd.print(). 7-segment displays, on the other hand, need multiple digital pins to control individual segments (A to G) using digitalWrite(), making it more complex, especially for multiple digits. LED matrices use a grid of LEDs, where pixels are controlled using arrays and functions like setPixel(x, y), making them the most flexible but also the most complex to program.

In short, I2C LCDs are easiest for text, 7-segment displays are good for numbers, and LED matrices offer the most customization but require more coding effort.

9.0 CONCLUSION

In conclusion, this experiment successfully demonstrated the interfacing of a common cathode 7-segment display with an Arduino Mega 2560 and the manual control of numerical output using pushbuttons. The system correctly incremented the displayed digits from 0 to 9 and

reset back to 0 using a well-defined button-press mechanism. The 200 ms interval between transitions remained consistent throughout, confirming stable execution. The observed output matched the expected results, reinforcing the accuracy of the implemented circuit and program logic. The hypothesis that the 7-segment display would function correctly under manual pushbutton control was supported by the results. The display accurately reflected the button presses without skipping or flickering, and the microcontroller executed the programmed logic as intended. The success of this experiment highlights the practicality and reliability of seven-segment displays in real-world applications such as digital counters and timers and clocks, like microwave timers and stopwatch displays.

This experiment also provides the hands-on experience of interfacing a 7-segment common cathode display with Arduino Mega, supplementing the theoretical foundations of digital logic and embedded systems. With the incorporation of manual control using pushbuttons, students can practice designing circuits, programming microcontrollers, and integrating electronic components. Furthermore, the comparison of different display technologies improves their grasp of different interfacing techniques and applications. Overall, this experiment enhances the fundamental competencies in digital electronics and embedded system development.

10.0 RECOMMENDATIONS

For future iterations of the experiment, several improvements can be done to enhance the functionality and outcomes. Future iterations of the experiment can be improved by adding a delay for button presses to prevent accidental multiple counts and using a shift register to reduce the number of Arduino pins needed for the 7-segment display. The delay will help make the button response smoother, while the shift register will simplify wiring and free up more pins for

other functions. Expanding the counting range beyond 0 to 9 by using multiple 7-segment displays would allow students to explore more advanced display techniques. Code efficiency also can be improved by using arrays or lookup tables instead of multiple digitalWrite().

Through this experiment, students can gain valuable insights into microcontroller programming and circuit design. Designing a circuit helps students to understand circuit behavior, such as the importance of selecting appropriate resistors, ensures consistent segment brightness and prevents display flickering. Students also learn the importance of push-down is crucial in stabilizing button inputs, preventing erratic behavior. Lastly, students recognize the real-world usage of 7-segment displays used in clocks, counters, and embedded systems.

11.0 REFERENCES

Atharvadhamorikar. (n.d.). *Push Button With 7 Segment Display*. Instructables. https://www.instructables.com/Push-Button-With-7-Segment-Display/

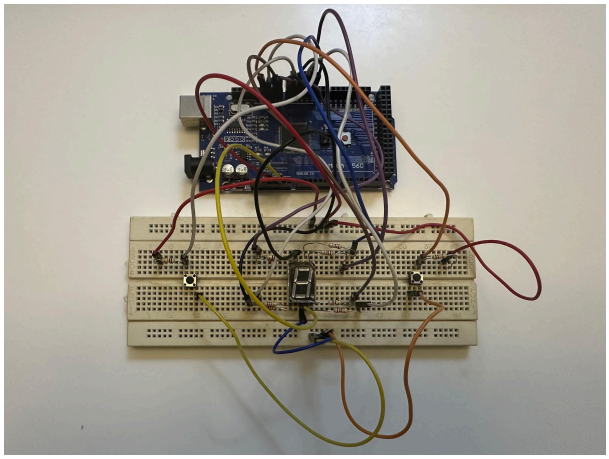
Electronics Tutorials. (2018, February 16). *7-segment Display and Driving a 7-segment Display*. Basic Electronics Tutorials. https://www.electronics-tutorials.ws/blog/7-segment-display-tutorial.html

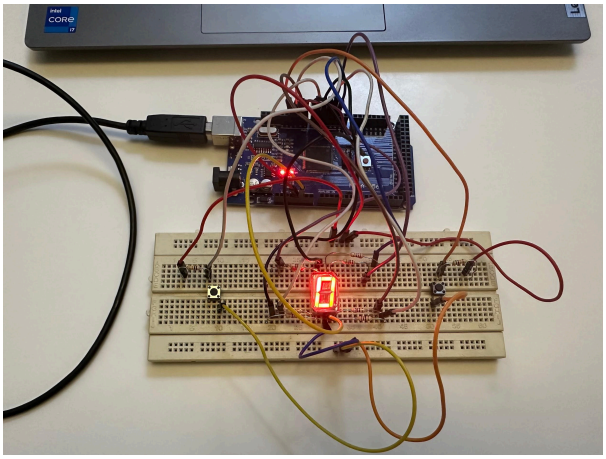
Saurabh. (2019, June 4). *7 Segment Display Pinout and Working | Basics for Beginners*. Electronics for You. https://www.electronicsforu.com/resources/7-segment-display-pinout-understanding

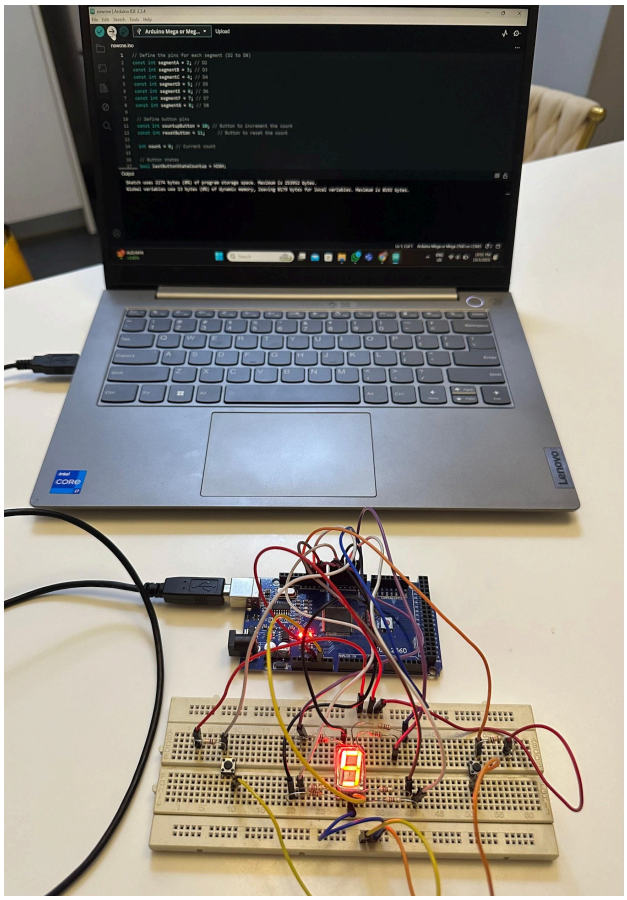
Wikipedia Contributors. (2019, February 5). *Seven-segment display*. Wikipedia; Wikimedia Foundation. https://en.wikipedia.org/wiki/Seven-segment_display

Youngblood, T. (2015, June 2). *Interface a Seven Segment Display to an Arduino - Projects*. All About Circuit. https://www.allaboutcircuits.com/projects/interface-a-seven-segment-display-to-an-arduino/

APPENDICES







ACKNOWLEDGMENTS

First of all, we would like to express our deepest appreciation to all those who provided us the possibility to complete this report. We would like to express our sincere gratitude to our lecturer, Assoc. Prof. Eur. Ing. Ir. Ts. Gs. Inv. Dr. Zulkifli Bin Zainal Abidin and Dr. Wahju Sediono for their invaluable guidance and support throughout this lab. Their insightful explanations and encouragement greatly enhanced our understanding of the concepts covered.

Next, a special thanks goes to our own teammates, Wafdi, Amalin and Farha, who helped to build up the circuit and set up the programming code using Arduino IDE, and also successfully completed this experiment. Last but not the least, we would like to thank everyone, especially our classmates who are willingly helping us out in the lab experiment directly or indirectly.

STUDENT'S DECLARATION

We, the undersigned, hereby declare that the work presented in this report is our own original work and does not involve plagiarism. The report is based on results obtained by us during our laboratory session and has been completed in accordance with the principles of academic integrity. We affirm that no part of this report has been copied, plagiarized, or submitted for assessment in any other course or institution. Any external sources used have been appropriately cited and referenced. We hereby also acknowledge that all of us have contributed to the report and has not been done by only one individual.

Student's Name: Amer Wafdi Bin Zainizam
Student ID: 2228703
Date: 16/03/2025
Signature: *Wafdi*

Student's Name: Amalin Balqis Binti Ruyusni
Student ID: 2314684
Date: 16/03/2025
Signature: *Amalin*

Student's Name: Farha Qaisara Binti Mohd Zahidi
Student ID: 2315994
Date: 16/03/2025
Signature: *Farha*