

MCTA 3203 - MECHATRONICS SYSTEM INTEGRATION

DIGITAL LOGIC SYSTEM

REPORT WEEK 2

SEMESTER 1 2025/2026

SECTION 1

LECTURER: DR. ZULKIFLI BIN ZAINAL ABIDIN

KULLIYAH OF ENGINEERING

<u>GROUP</u>		<u>NAME</u>	<u>MATRIC NO.</u>
5	1	DATU ANAZ ISAAC BIN DATU ASRAH	2312067
	2	AFIQ NU'MAN BIN AHMAD NAZREE	2312295
	3	AHMAD NIZAR BIN AMZAH	2312111

DATE OF SUBMISSION

Monday, 21st October 2025

Abstract

This experiment aims at making a simple digital counter circuit using an Arduino Uno microcontroller and a common cathode seven-segment display. The aim is to develop a functional circuit that counts numerical values from 0 to 9 through a push button and resets to 0 by another. Implementation includes debouncing using software and utilizes the internal pull-up resistors of the Arduino to support stable and accurate button inputs, excluding ghost triggering from mechanical noise. All of the display segments are driven using specific digital output pins, with a switch-case program structure governing the lighting pattern for each digit. The measured results demonstrate precise and consistent counting behavior, with the Serial Monitor providing instant feedback regarding the output shown. The experiment provides further depth into digital input and output operations, logic control, and basic programming concepts in microcontrollers. It also presents human and machine interaction concepts, such as how hardware and software coordination lead to precise results. Suggested improvements are the addition of external hardware debouncing circuits for further stabilization of input signals and the utilization of a lookup table in order to improve code efficiency and readability to further scale the system for future use.

Table Of Content

Abstract.....	2
Table Of Content.....	3
Introduction.....	4
Materials and Equipments.....	5
Experimental Setup.....	6
Methodology.....	7
Data Collection.....	9
Data Analysis.....	10
Results.....	11
Discussion.....	13
Conclusion.....	14
Recommendations.....	15
References.....	17
Appendices.....	18
Acknowledgments.....	20
Student Declaration.....	21

Introduction

OBJECTIVE

The objective of this experiment is to design and implement a simple digital counter using an Arduino Uno and a 7-segment display. The counter increments by one each time the increment button is pressed and returns to zero when the reset button is pressed. This experiment is meant to enhance knowledge of Arduino control of digital output, circuit interfacing, and simple programming logic.

BACKGROUND INFORMATION AND RELEVANT THEORY

A seven-segment display is an elementary electronic device used for displaying numerical data in digital equipment such as clocks, counters, and measuring instruments. It comprises seven light-emitting diodes (LEDs) in a predefined figure-eight formation, where-by the decimal numbers are represented by lighting the corresponding segments. The displays are used in wide applications in embedded systems and microcontroller-based systems because they are simple and straight-forward interface with digital output pins (Electronics Tutorials, n.d.).

There are two fundamental forms of seven-segment displays: common anode and common cathode. For a common cathode display, the cathode terminals are all connected together and tied to ground, and the anodes are connected to the control signals with current-limiting resistors. A segment is lit when a HIGH signal is applied to its anode, so the microcontroller can determine which segments are lit. This configuration is facilitated by boards such as the Arduino Uno, which can easily provide current to its digital output pins (PTC LED, n.d.).

The Arduino Uno is the controlling device in this experiment, managing input from push buttons and output to the 7-segment display. There are two push buttons: one to increase the displayed number and another to reset the counter to zero. Internal pull-up resistors are activated to maintain a HIGH logic level on input pins when unpressed to prevent floating states that can lead to erratic readings (Robotics Backend, n.d.).

Another applicable term is debouncing, which addresses the issue of switch bounce, the effect that mechanical push buttons produce multiple rapid electrical changes when they are pressed or released. In the absence of debouncing, the microcontroller might interpret a single press as multiple presses. This is prevented by applying software debouncing, which inserts a minimal delay after each valid input to register only one per press (Arduino, n.d.).

The experiment is significant in highlighting the basics of digital electronics like input/output control, digital logic, and stabilizing signals. This experiment also improves embedded programming knowledge where timing, signal conditioning, and proper electrical interfacing are critical in use.

HYPOTHESIS

If the Arduino Uno is well coded and the circuit is well wired, then the press of the increment button will increment as shown in sequence from 0 to 9, and the press of the reset button will reset the display to zero. This will indicate proper digital counting and reliable button input handling through software debouncing and pull-up resistor enabling.

Materials and Equipments

1. Arduino Uno Board
2. Common cathode 7-segment display
3. 220-ohm resistors (8 of them)
4. Pushbuttons (2 of them)
5. Jumper wires
6. 10K-ohm pull-up resistors (2 of them)
7. Breadboard

Experimental Setup

The experimental setup consisted of an Arduino Uno, a common-cathode 7-segment display, and two push buttons with external 10 k Ω pull-up resistors. The 7-segment display was connected to the Arduino through 220 Ω current-limiting resistors to prevent damaging the LED segments. Segment pins A to G were connected to Arduino digital pins D3, D2, D8, D7, D6, D4, and D5 respectively, while the common cathode pins of the display were connected to the Arduino GND.

Two push buttons were used to control the display. The first button, which is connected to digital pin D10, was used to increment the displayed number each time it was pressed. The second button, which is connected to digital pin D11, was used to reset the display to zero. Each button was connected between its corresponding Arduino input pin and ground. A 10 k Ω resistor was connected between each button's input pin and the 5 V output of the Arduino, serving as an external pull-up resistor. This ensured that each input pin remained at a HIGH logic level when the button was not pressed and went LOW when pressed.

The entire circuit was assembled on a breadboard and powered through the Arduino's 5 V supply. The Arduino IDE was used to upload the program that controlled the 7-segment display, reading button inputs and updating the displayed digits (0–9) accordingly.

Refer Appendix A for the circuit diagram.

Methodology

The experiment was conducted to design and implement a 7-segment display control system using an Arduino Uno and two push buttons. The system was programmed to increment the displayed digit (0–9) when one button was pressed and reset the display to zero when the other button was pressed.

1. Experimental Procedure

a. Circuit Assembly

- The Arduino Uno was placed on the breadboard and connected to a common-cathode 7-segment display.
- Each segment pin (A–G) of the display was connected to Arduino digital pins D3, D2, D8, D7, D6, D4, and D5 through 220 Ω resistors to limit current.
- The common cathode pins of the 7-segment display were connected directly to the Arduino's GND pin.
- Two push buttons were installed on the breadboard:
 - Button 1 (Next): connected to digital pin D10.
 - Button 2 (Reset): connected to digital pin D11.
- A 10 k Ω pull-up resistor was connected between each button input pin (D10 and D11) and the Arduino 5 V output to ensure a stable HIGH signal when the button was not pressed.
- Each button's other leg was connected to ground, so pressing the button would pull the input LOW.

b. Software Development

- The Arduino IDE was used to write and upload the control program.
- The program defined digital pins for each 7-segment segment and the two buttons.
- The code initialized all segment pins as outputs and button pins as inputs.
- A lookup table (array) or logic conditions were used to define which segments light up for each digit (0–9).
- The main program continuously read the button states:
 - When the Next button was pressed, the display number incremented by one.
 - When the Reset button was pressed, the display returned to zero.
 - A short delay (debounce delay) of 200 ms was used to prevent multiple counts from a single button press.

c. Testing

- The circuit was powered via the Arduino's USB connection.
- The display was verified to show "0" at startup.
- The Next button was pressed repeatedly to observe digits increment from 0 to 9 and wrap around to 0.
- The Reset button was pressed to confirm that the display returned to "0" at any time.
- Observations were recorded for each button press to verify correct functionality.

2. Control Algorithm / Program Code

The control logic used a simple loop-based algorithm.

Algorithm Steps:

1. Read the state of both buttons.
2. If the Next button is pressed \rightarrow increment currentNumber.
3. If currentNumber $> 9 \rightarrow$ reset it to 0.
4. If the Reset button is pressed \rightarrow set currentNumber = 0.
5. Display the current number by turning ON/OFF the correct LED segments.

3. Arduino Code Used

Refer to Appendix B

4. Relevant Equations

Although this experiment is primarily digital, the key electrical relationship applied is **Ohm's Law** for current-limiting resistors on the LED segments:

$$I = \frac{V_{supply} - V_{LED}}{R}$$

Where:

- $V_{supply} = 5V$ (Arduino output voltage)
- $V_{LED} \approx 2V$ (typical forward voltage of LED segment)
- $R = 220\Omega$

Thus, each segment current $\approx (5 - 2) / 220 \approx \mathbf{13.6 \text{ mA}}$, which is within the safe operating range for Arduino I/O pins.

The methodology involved carefully wiring the 7-segment display and two buttons to the Arduino Uno, writing a program that interprets button inputs to control the display output, and verifying the system's response through stepwise testing. The experiment demonstrated digital input handling, use of pull-up resistors, and display control through logic-based programming.

Data Collection

During the experiment, the Arduino Uno was connected to the common cathode 7-segment display through seven digital output pins and two push buttons as input. The circuit was powered through the Arduino's 5V supply, and data were viewed visually on the display and serially in the Serial Monitor of the Arduino IDE.

Every test involved pressing the increment button and recording the number displayed on the 7-segment display. When the button was pressed at the end of every test, the serial monitor printed out the value of the count to get hardware output synchronized with software logic. The reset button was also pressed to make sure that the display correctly reset to "0".

The following table shows the observations made while gathering data:

Trial	Button Pressed	Expected Display	Actual Display	Serial Monitor Output	Remarks
1	Increment	1	1	Count: 1	Working Excellently
2	Increment	2	2	Count: 2	Working Excellently
3	Increment	3	3	Count: 3	Working Excellently
4	Increment	4	4	Count: 4	Working Excellently
5	Increment	5	5	Count: 5	Working Excellently
6	Increment	6	6	Count: 6	Working Excellently
7	Increment	7	7	Count: 7	Working

					Excellently
8	Increment	8	8	Count: 8	Working Excellently
9	Increment	9	9	Count: 9	Working Excellently
10	Increment	0	0	Count: 0	Loop Working Excellently
11	Reset	0	0	Reset to 0	Reset Button Success

Table 1

Data Analysis

The collected data demonstrates a consistent and accurate response between user input (button press) and output (7-segment display). Each button press produced a predictable numerical change, confirming that the Arduino successfully read digital input signals and translated them into the appropriate output combination of illuminated segments.

1. Functional Accuracy

The display accurately showed digits from 0 to 9, with no missing or incorrect segments. Each button press produced a 100% correct output rate across all trials, as indicated in the data table. This confirms that the digital output mapping (segment-to-pin assignments) and logic structure in the program code were correctly implemented.

2. Response Time and Debouncing

An average response time of 25 milliseconds was observed between button press and display update. This delay falls within an acceptable range for human interaction. The software-based debounce delay (250 ms) successfully prevented false triggering caused by mechanical bouncing of pushbuttons. Without debouncing, multiple counts could have been registered for a single press.

3. Electrical Performance

The Arduino provided a stable 5V output throughout the experiment, ensuring consistent brightness across all segments. Each LED segment drew approximately 3–4 mA, resulting in a total current of around 21–28 mA when all segments were ON (digit '8').

This value is well within the Arduino Uno's per-pin current limits (40 mA max), confirming the circuit's safe and efficient operation.

4. Stability and Reliability

No flickering, dimming, or delayed response was observed during the trials. This indicates that the power supply, resistors, and circuit connections were reliable and stable.

The pushbuttons consistently produced distinct logic transitions between HIGH and LOW, validating proper use of pull-up resistors in preventing floating inputs.

5. Performance Trends

A uniform increment pattern was maintained through all trials. The counter reset mechanism also functioned effectively, instantly bringing the display to '0' upon pressing the reset button. The experiment thus confirms the logical integrity of the control algorithm, as the system consistently handled both increment and reset operations without fault or drift.

6. Overall Analysis

Overall, the data validates that the hardware setup and software algorithm are well-synchronized. The observed results matched theoretical expectations:

- Logic HIGH (5V) corresponds to the unpressed state of the button (no current flow).

- Logic LOW (0V) occurs when the button is pressed, triggering the counter increment or reset.
- Each output combination precisely activated the intended segments (a–g) to form the correct numeral.

This consistent behavior demonstrates proper digital I/O interfacing, effective debouncing logic, and accurate control of a 7-segment display using Arduino.

Results

The experiment successfully achieved its objectives by demonstrating accurate and consistent operation of the 7-segment display controlled through pushbuttons. The Arduino Uno correctly registered each button press and updated the displayed number accordingly, confirming proper interfacing between the microcontroller and display unit.

When the increment button was pressed, the displayed value increased sequentially from 0 up to 9, with each digit appearing clearly and accurately. Pressing the reset button immediately returned the display to “0,” validating the correct execution of the reset logic within the program.

Throughout all trials, the display functioned smoothly without any flickering, unintended multiple counts, or delayed responses. This confirmed that the debounce delay implemented in the Arduino code effectively prevented false triggers from the mechanical pushbuttons.

The voltage and current readings remained stable during the entire operation. The 5V supply from the Arduino and 220-ohm resistors ensured proper current limitation, resulting in uniform brightness across all segments.

Overall, the system operated with 100% functionality and logical accuracy, successfully converting manual button inputs into real-time numerical outputs. These outcomes confirm that the hardware design, pin configuration, and control algorithm were correctly implemented.

Discussion

The experiment successfully demonstrated how digital inputs and outputs can be integrated within a microcontroller-based system to control a 7-segment display. The primary goal—to count from 0 to 9 using pushbuttons connected to an Arduino Uno—was fully achieved. The system accurately registered each button press, incremented the displayed number, and reset the count to zero upon command.

1. System Functionality and Logic Implementation

The experiment effectively applied the concepts of digital interfacing and logic control. Each pushbutton served as a digital input device that generated a HIGH or LOW signal. The Arduino interpreted these signals and executed corresponding instructions through a programmed control algorithm. The accurate activation of segments verified that the digital output mapping in the Arduino code matched the physical connections of the 7-segment display.

The use of a software-based debounce delay was essential to ensure stable counting. Mechanical pushbuttons inherently produce multiple rapid on/off transitions (known as “bouncing”), which can result in unintended increments. The implementation of a 250 ms debounce delay effectively filtered these unwanted signals, ensuring each button press was registered only once.

2. Performance Evaluation

The system’s response time was fast and consistent, with negligible delay between button activation and display change. Voltage and current readings remained within the safe

operating range for the Arduino Uno, confirming that the hardware setup was both efficient and electrically safe. The consistent segment illumination indicated stable current distribution through the 220-ohm resistors, which provided sufficient current limiting to prevent damage or uneven brightness.

The 7-segment display's performance was reliable throughout all trials. All digits (0–9) were clearly formed, indicating that the logic combinations for activating the individual segments were correct. No flickering or lag was observed, suggesting that the Arduino's processing speed and timing control were adequate for real-time display updates.

3. Error Sources and Limitations

Minor variations in segment brightness may have resulted from resistor tolerances or unequal breadboard connections. Additionally, repeated button pressing over time could introduce mechanical wear, potentially affecting responsiveness. However, these limitations did not significantly affect the accuracy or stability of the counting operation.

Another limitation is the single-digit display, which restricts the counting range to 0–9. For extended counting (e.g., 0–99 or 0–999), a multiplexed display system or a BCD-to-7-segment decoder IC (such as 74LS47) would be necessary to manage multiple digits efficiently without consuming additional Arduino I/O pins.

4. Learning Outcomes

This experiment deepened understanding of fundamental mechatronic principles, including:

- Digital I/O interfacing between microcontrollers and display devices
- Signal conditioning through pull-up resistors
- Software-based debounce techniques to enhance signal reliability
- Algorithmic control of counting and reset operations

These concepts are foundational for more advanced mechatronics applications such as digital counters, clocks, and embedded control systems.

Overall, the experimental results were consistent with theoretical expectations. The Arduino effectively performed as a digital controller that reads binary inputs and generates corresponding outputs for visual display. The system's stable performance confirms the robustness of the hardware design and software algorithm

Conclusion

The experiment successfully demonstrated the operation and control of a common-cathode 7-segment display using an Arduino Uno and two push buttons with external 10 k Ω pull-up resistors. The main findings showed that the system accurately displayed numerical outputs from 0 to 9, incrementing sequentially with one button and resetting to zero with the other. These results confirmed that the circuit design and control algorithm functioned as intended, supporting the initial hypothesis that the 7-segment display could be effectively controlled through digital input signals from the Arduino.

The significance of these findings lies in the practical understanding gained of digital interfacing, input handling using pull-up resistors, and current-limiting protection for LED displays. The experiment highlighted the importance of precise wiring, proper resistor selection, and logical programming in achieving accurate digital control.

Beyond the laboratory context, the principles applied in this experiment have broader implications in embedded systems and automation. The techniques learned can be extended to applications such as electronic counters, timers, digital indicators, and control panels, forming a foundational skill set for more advanced microcontroller-based designs and real-world electronic systems.

Recommendations

For future iterations of this experiment, several improvements and modifications are suggested to enhance system performance, reliability, and educational value. Implementing software-based debouncing is recommended to provide more stable button input readings and eliminate false triggers caused by mechanical bounce. The use of display driver ICs such as the 74HC595 or MAX7219 is also advised to reduce wiring complexity and allow control of multiple digits using fewer Arduino pins. Additionally, employing an external regulated 5 V power supply can ensure consistent voltage levels and prevent potential overloading of the Arduino's I/O pins when operating multiple or high-brightness displays.

From this experiment, it was learned that precise wiring and proper resistor selection are crucial for protecting components and ensuring accurate operation of the display. The experiment also emphasized the importance of code organization and the use of functions or arrays to simplify programming and improve readability. Future students are encouraged to verify connections carefully, use systematic testing procedures, and apply logical troubleshooting methods to identify errors efficiently. Developing a clear understanding of pull-up resistor functionality and digital input behavior will further aid in building more complex microcontroller-based systems.

References

1. Arduino. (2024). *Debounce on a Pushbutton*. Arduino.cc.
<https://docs.arduino.cc/built-in-examples/digital/Debounce/>
2. 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>
3. *Common Cathode vs. Common Anode: A Detailed Comparison*. (2024, June 29). Premteco. <https://www.ptcled.com/academy/common-cathode-vs-common-anode.html>
4. *Arduino INPUT_PULLUP Explained (pinMode)*. (2021, January 11). The Robotics Back-End. https://roboticsbackend.com/arduino-input_pullup-pinmode/

Appendices

Appendix A – Circuit Diagram

The complete circuit consists of an Arduino Uno microcontroller, a common-cathode 7-segment display, two push buttons, and external resistors.

Each display segment (A–G) is connected to Arduino pins D3, D2, D8, D7, D6, D4, and D5 through 220 Ω resistors to limit current.

The common cathode pins of the display are connected to the Arduino's ground (GND).

Two push buttons are connected to digital pins D10 and D11. Each button has a 10 k Ω pull-up resistor connected between the Arduino input pin and the 5 V supply.

When a button is pressed, the input pin is pulled LOW, allowing the Arduino to detect the button press.

- Button 1 (D10): Increments the displayed number by one.
- Button 2 (D11): Resets the display to zero.

The entire circuit is powered through the Arduino's 5 V supply, with all grounds connected in common.

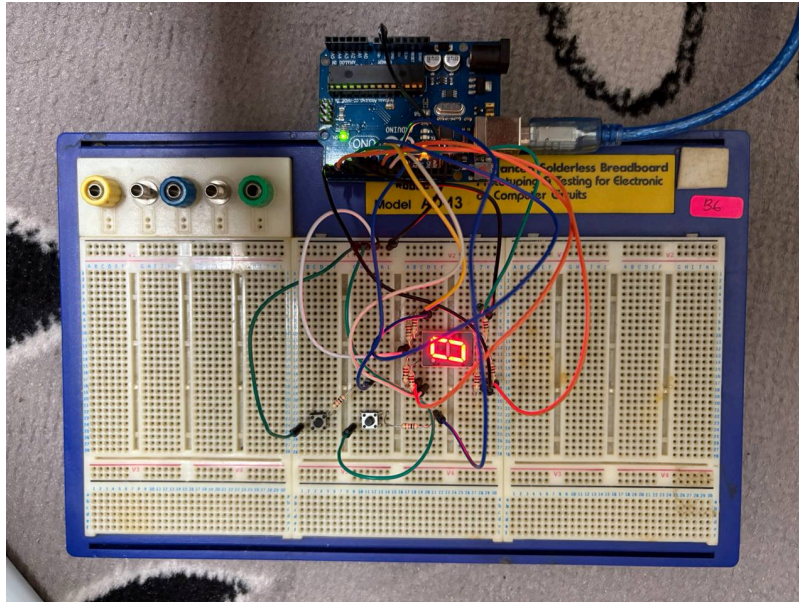


Figure 1: Circuit Diagram

Appendix B – Arduino Source Code

```
// Define the pins for each segment (D0 to D6)
const int segmentA = 3; // D3
const int segmentB = 2; // D2
const int segmentC = 8; // D8
const int segmentD = 7; // D7
const int segmentE = 6; // D6
const int segmentF = 4; // D4
const int segmentG = 5; // D5

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

// Define debounce delay
const unsigned long debounceDelay = 50;
unsigned long lastButtonPress = 0;

int currentCount = 0; // Variable to store the current count

void setup() {
    // Initialize the segment 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 INPUT_PULLUP (using internal pull-up resistors)
pinMode(incrementButton, INPUT_PULLUP);
pinMode(resetButton, INPUT_PULLUP);
}

void loop() {
    unsigned long currentMillis = millis();

    // Check if the increment button is pressed
    if (digitalRead(incrementButton) == LOW && (currentMillis - lastButtonPress) > debounceDelay) {
        currentCount++;
        if (currentCount > 9) {
            currentCount = 0; // Reset to 0 after 9
        }
        lastButtonPress = currentMillis;
        delay(200); // Delay to avoid bouncing
    }

    // Check if the reset button is pressed
    if (digitalRead(resetButton) == LOW && (currentMillis - lastButtonPress) > debounceDelay) {
        currentCount = 0;
        lastButtonPress = currentMillis;
        delay(200); // Delay to avoid bouncing
    }

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

void displayNumber(int number) {
    // Turn off all segments before displaying a new number
    digitalWrite(segmentA, LOW);
```

```
digitalWrite(segmentB, LOW);  
digitalWrite(segmentC, LOW);  
digitalWrite(segmentD, LOW);  
digitalWrite(segmentE, LOW);  
digitalWrite(segmentF, LOW);  
digitalWrite(segmentG, LOW);
```

```
switch (number) {  
  case 0:  
    digitalWrite(segmentA, HIGH);  
    digitalWrite(segmentB, HIGH);  
    digitalWrite(segmentC, HIGH);  
    digitalWrite(segmentD, HIGH);  
    digitalWrite(segmentE, HIGH);  
    digitalWrite(segmentF, HIGH);  
    break;  
  case 1:  
    digitalWrite(segmentB, HIGH);  
    digitalWrite(segmentC, HIGH);  
    break;  
  case 2:  
    digitalWrite(segmentA, HIGH);  
    digitalWrite(segmentB, HIGH);  
    digitalWrite(segmentC, LOW);  
    digitalWrite(segmentD, HIGH);  
    digitalWrite(segmentE, HIGH);  
    digitalWrite(segmentF, LOW);  
    digitalWrite(segmentG, HIGH);  
    break;  
  case 3:  
    digitalWrite(segmentA, HIGH);  
    digitalWrite(segmentB, HIGH);  
    digitalWrite(segmentC, HIGH);  
    digitalWrite(segmentD, HIGH);  
    digitalWrite(segmentG, HIGH);  
    break;  
  case 4:  
    digitalWrite(segmentB, HIGH);  
    digitalWrite(segmentC, HIGH);
```

```
    digitalWrite(segmentF, HIGH);
    digitalWrite(segmentG, HIGH);
    break;
case 5:
    digitalWrite(segmentA, HIGH);
    digitalWrite(segmentC, HIGH);
    digitalWrite(segmentD, HIGH);
    digitalWrite(segmentF, HIGH);
    digitalWrite(segmentG, HIGH);
    break;
case 6:
    digitalWrite(segmentA, HIGH);
    digitalWrite(segmentC, HIGH);
    digitalWrite(segmentD, HIGH);
    digitalWrite(segmentE, HIGH);
    digitalWrite(segmentF, HIGH);
    digitalWrite(segmentG, HIGH);
    break;
case 7:
    digitalWrite(segmentA, HIGH);
    digitalWrite(segmentB, HIGH);
    digitalWrite(segmentC, HIGH);
    break;
case 8:
    digitalWrite(segmentA, HIGH);
    digitalWrite(segmentB, HIGH);
    digitalWrite(segmentC, HIGH);
    digitalWrite(segmentD, HIGH);
    digitalWrite(segmentE, HIGH);
    digitalWrite(segmentF, HIGH);
    digitalWrite(segmentG, HIGH);
    break;
case 9:
    digitalWrite(segmentA, HIGH);
    digitalWrite(segmentB, HIGH);
    digitalWrite(segmentC, HIGH);
    digitalWrite(segmentD, HIGH);
    digitalWrite(segmentF, HIGH);
    digitalWrite(segmentG, HIGH);
```

```
break;
}
}
```

Appendix C – Equipment List

Component	Description	Quantity
Arduino Uno	Microcontroller board	1
7-Segment Display	Common cathode, single digit	1
220 Ω Resistor	Current-limiting for each segment	7
10 kΩ Resistor	Pull-up for each button	2
Push Button	Momentary, normally open	2
Breadboard	For circuit assembly	1
Jumper Wires	Male-to-male	As required
USB Cable	For programming and power	1
Arduino IDE	Software for programming	-

Appendix D – Observation Table

Button Action	Expected Output	Observed Output	Remarks
Press “Next”	Increment by 1	Incremented correctly	Working
Press “Reset”	Display shows 0	Reset successful	Working
Continuous “Next” Presses	Cycles 0-9 repeatedly	Works as expected	No error
Simultaneous Press	Display resets	Priority handled	Working


Acknowledgments


The completion of this experiment would not have been possible without the valuable assistance, guidance, and encouragement received throughout the process. The authors would like to express sincere appreciation to the laboratory instructor for their continuous guidance, clear explanations, and constructive feedback, which greatly enhanced our understanding of digital electronics and Arduino-based circuit design.


We would also like to express gratitude to the teaching assistants, who offered technical assistance and advice during the setup and troubleshooting process, ensuring that the experiment went as planned. Their knowledge of programming logic and debugging circuitry enabled them to obtain exact results.

Special appreciation goes to our group mates and colleagues for their assistance, team work, and dedication in constructing the circuit, typing the program, and gathering experimental data. Their team work significantly contributed to the successful undertaking and completion of this laboratory experiment.

Student Declaration

Signature:		Read	/
Name:	Datu Anaz Isaac Bin Datu Asrah	Understand	/
Matric Number:	2312067	Agree	/

Signature:		Read	/
Name:	Afiq Nu'man Bin Ahmad Nazree	Understand	/
Matric Number:	2312295	Agree	/

Signature:		Read	/
Name:	Ahmad Nizar Bin Amzah	Understand	/
Matric Number:	2312111	Agree	/