



MECHATRONICS SYSTEM INTEGRATION (MCTA 3203)

SEMESTER 1 2025/2026

WEEK 2: DIGITAL LOGIC DESIGN

SECTION 1

GROUP 9

**LECTURER: ZULKIFLI BIN ZAINAL ABIDIN & WAHJU
SEDIONO**

NO.	GROUP MEMBERS	MATRIC NO.
1.	AHMAD FIRDAUS HUSAINI BIN HASAN AL-BANNA	2315377
2.	ADIB ZAFFRY BIN MOHD ABDUL RADZI	2315149
3.	UMAR FAROUQI BIN ABU HISHAM	2314995

Date of Submission: 22 October 2025

ABSTRACT

This experiment focuses on interfacing an Arduino Uno (R3) with a common cathode seven-segment display using two push buttons; one for incrementing the displayed number and another for resetting it to zero. The circuit was built by connecting each segment of the display to separate digital pins on the Arduino through appropriate current-limiting resistors to ensure safe operation. Two push buttons were also connected to the Arduino's digital input pins to provide manual control.

The system was programmed to automatically count from 0 to 5 upon startup before switching to manual increment mode. Afterward, each press of the increment button increased the displayed number sequentially from 0 to 9, while pressing the reset button returned the count to 0.

Through this experiment, we gained hands-on experience in interfacing electronic components with Arduino, controlling digital displays, and implementing troubleshooting skills

TABLE OF CONTENTS

NO	TOPIC	PAGE
1	INTRODUCTION	3
2	MATERIALS AND EQUIPMENTS	3
3	EXPERIMENTAL SETUP	4
4	METHODOLOGY	8
5	DATA COLLECTION	11
6	DATA ANALYSIS	12
7	RESULTS	14
8	DISCUSSION	18
9	CONCLUSION	21
10	RECOMMENDATIONS	22
11	REFERENCES	22
12	APPENDICES	23
13	ACKNOWLEDGEMENT	24

1.0 INTRODUCTION

This experiment focuses on interfacing a 7-segment display with push buttons using an Arduino Uno (R3) to create a simple digital counter that displays numbers from 0 to 9. The main objective is to understand the control mechanisms of basic logic systems and how manual inputs can be used to control digital outputs. By applying fundamental principles of digital logic and binary counting, the system enables users to visualize numeric changes on the 7-segment display through button interactions. Each press of the increment button increases the count, while the reset button returns the display to zero. This experiment provides a foundational understanding of digital interfacing, program control, and microcontroller-based system design, which are essential concepts for developing more complex automation and embedded systems.

2.0 MATERIALS AND EQUIPMENTS

1. Arduino Uno R3
2. Common cathode 7-segment display
3. 220-ohm resistor
4. 10k-ohm resistor
5. Push buttons
6. Breadboard
7. Jumper wires

3.0 EXPERIMENTAL SETUP

The Arduino Uno (R3) served as the main controller for the circuit. A common-cathode 7-segment display was interfaced to the Arduino so numeric values could be shown on the display. The wiring and assembly were performed on a solderless breadboard with jumper wires and $220\ \Omega$ current-limiting resistors for each segment.

1. Segment connections

- Each of the seven segment pins (A–G) of the 7-segment display was connected to an individual digital output on the Arduino Uno as follows:
 - Segment A → Digital pin 4
 - Segment B → Digital pin 7
 - Segment C → Digital pin 5
 - Segment D → Digital pin 3
 - Segment E → Digital pin 2
 - Segment F → Digital pin 6
 - Segment G → Digital pin 1
- A 220-ohm resistor was placed in series with each segment to limit the LED current and prevent damage.
- The common anode pins of the display were tied to the Arduino 5V supply.

2. Push-button inputs

- Two push buttons were installed to provide Increment and Reset controls.
 - Increment button → Digital pin 9
 - Reset button → Digital pin 10
- For each button, one leg was connected to the Arduino digital input pin and the other leg to GND.

3. Assembly and wiring checks

- All components (display, resistors, buttons, and Arduino) were assembled on a breadboard using jumper wires to make connections tidy and reworkable.
- After wiring, visual and continuity checks were performed to confirm correct pin connections and to avoid loose contacts that could cause intermittent operation.

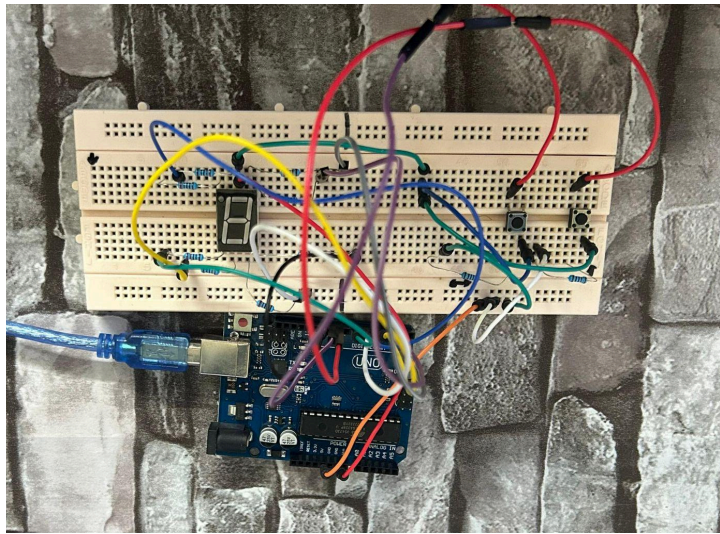
4. Software and initial test

- The Arduino was programmed so the system automatically counts from 0 to 5 on power-up, then switches to manual mode where the Increment button advances the displayed number by one (wrapping from 9 back to 0) and the Reset button returns the display to 0.

- Initial validation steps included:
 - Uploading the sketch and observing the automatic 0→5 startup sequence.
 - Pressing the Increment button repeatedly to verify the display advances sequentially through 0–9.
 - Pressing the Reset button at several stages to verify the display returns to 0.

5. Troubleshooting

- If a segment failed to illuminate, the wiring and resistor for that segment were rechecked and continuity verified.
- Button faults were checked by ensuring correct orientation on the breadboard and verifying the internal pull-ups produced a stable HIGH when the button was not pressed.
- All connections were rechecked after any code change to ensure hardware and software matched (pin definitions in code versus physical wiring).



4.0 METHODOLOGY

1. The circuit was assembled based on the given circuit setup and wiring instructions.

2. The Arduino Uno (R3) was connected to the computer using a USB cable.

The programmed Arduino code was uploaded to the board successfully.

Once powered, the system was observed to ensure the initial display was functioning correctly.

The increment button was pressed once, and the number 1 appeared on the 7-segment display.

3. The increment button was pressed repeatedly to display the next numbers sequentially until 9 was shown.
4. After reaching 9, the reset button was pressed to return the display value to 0.
5. The test was repeated several times to confirm consistent counting and reset functionality.
6. Observations were recorded to verify that both push buttons and the 7-segment display operated as expected.

CODING

```
// Define segment pins (D0-D6)
const int segmentA = 4;
const int segmentB = 7;
const int segmentC = 5;
const int segmentD = 3;
const int segmentE = 2;
const int segmentF = 6;
const int segmentG = 11;

// Pushbutton pins
const int buttonIncrement = 9;
const int buttonReset = 10;

// Variable to store count value (0-9)
int count = 0;

// Digit patterns for 0-9
const int digits[10][7] = {
  {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
};

const int segments[7] = {segmentA, segmentB, segmentC, segmentD, segmentE, segmentF, segmentG};

// Variables to store previous button states
bool lastIncState = HIGH;
bool lastResetState = HIGH;

void setup() {
  for (int i = 0; i < 7; i++) pinMode(segments[i], OUTPUT);

  pinMode(buttonIncrement, INPUT_PULLUP);
  pinMode(buttonReset, INPUT_PULLUP);

  Serial.begin(9600);
  displayDigit(count);
}
```

```

}

void loop() {
    bool currentIncState = digitalRead(buttonIncrement);
    bool currentResetState = digitalRead(buttonReset);

    // Detect when button is just pressed (transition from HIGH to LOW)
    if (lastIncState == HIGH && currentIncState == LOW) {
        delay(50); // debounce delay
        if (digitalRead(buttonIncrement) == LOW) {
            count++;
            if (count > 9) count = 0;
            displayDigit(count);
            Serial.print("Count: ");
            Serial.println(count);
        }
    }

    if (lastResetState == HIGH && currentResetState == LOW) {
        delay(50);
        if (digitalRead(buttonReset) == LOW) {
            count = 0;
            displayDigit(count);
            Serial.println("Count reset to 0");
        }
    }

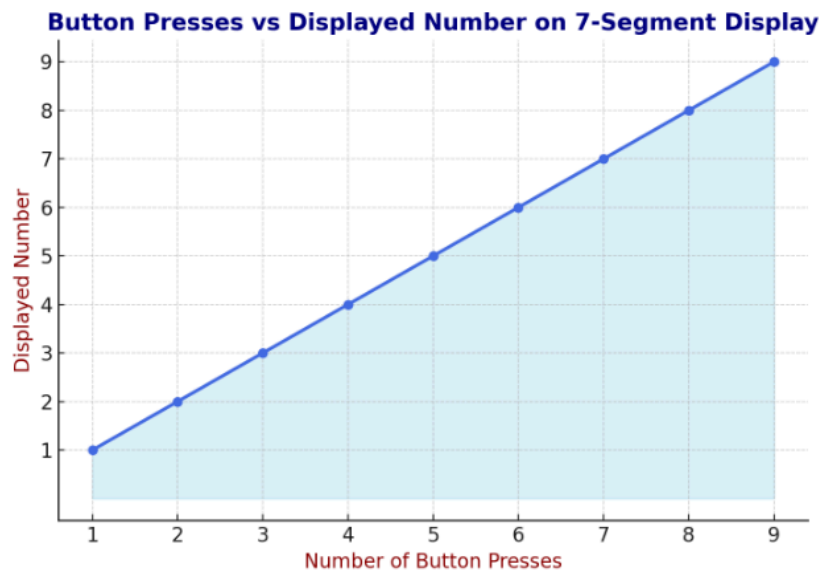
    // update states
    lastIncState = currentIncState;
    lastResetState = currentResetState;
}

void displayDigit(int num) {
    for (int i = 0; i < 7; i++) {
        digitalWrite(segments[i], digits[num][i] ? HIGH : LOW);
    }
}

```

5.0 DATA COLLECTION

Button	
INCREMENT	1
INCREMENT	2
INCREMENT	3
INCREMENT	4
INCREMENT	5
INCREMENT	6
INCREMENT	7
INCREMENT	8
INCREMENT	9
INCREMENT	0
RESET	0



6.0 DATA ANALYSIS

The data gathered indicates a clear and constant correlation between the number on the 7-segment display and the number of button presses. Every time the INCREMENT button is pressed, the displayed value rises by one, from 0 to 9, and then falls back to 0. By successfully resetting the display to 0 at any time, the RESET button verifies that the hardware and code are operating correctly.

Button pushes and display increments correlate one to one, as seen by the straight, linear trend formed by the plotted button presses vs. displayed number graph. This connection may be stated mathematically as $X = Y$

X = Number of button presses

Y = Displayed number on 7-segment display

This mathematical relationship illustrates how the system resets to 0 to resume the counting cycle after reaching 9.

From the data, we can observe:

- The increment function doesn't skip values and operates accurately and consistently..

- The counter restarts at 0 thanks to the reset feature working as planned.
- The graph's linearity shows that each input pulse is appropriately followed by the digital logic system's sequential logic and state evolution.

The results of this experiment highlight the successful implementation of a simple digital counting system using an Arduino and a 7-segment display. The consistent and linear relationship between button presses and displayed numbers demonstrates the proper functioning of sequential logic and the reliability of the system's response to user input. This experiment is significant because it shows how basic digital components can work together to perform logical operations and real-time output display, which are fundamental concepts in digital electronics. Furthermore, it illustrates the practical application of concepts such as modular counting, input signal processing, and microcontroller interfacing. Overall, the experiment reinforces the understanding of how digital systems translate user actions into numerical output through controlled logic states, forming the foundation for more complex digital design and automation systems.

7.0 RESULTS

Using an increment button, the system was able to correctly display numbers 0 through 9 on the 7-segment display. The display was accurately reset to 0 at any time by pressing the reset button. The code's debounce function successfully stopped several incorrect counts from occurring from a single button click, guaranteeing a precise and seamless counting process.

How to Interface a 7-Segment Display with Arduino

Interfacing a 7-segment display with an Arduino allows numeric values to be displayed visually using LEDs. Each segment (labeled A–G) is connected to an individual output pin on the Arduino, which controls whether the segment is ON or OFF. For this experiment, a common-cathode 7-segment display was used.

Steps to Interface:

Connect the Segment Pins:

Connect each of the seven segment pins (A–G) of the 7-segment display to the Arduino digital output pins as follows:

- A → D4
- B → D7
- C → D5

- $D \rightarrow D3$
- $E \rightarrow D2$
- $F \rightarrow D6$
- $G \rightarrow D1$

2. Add Current-Limiting Resistors:

Insert a $220\ \Omega$ resistor in series with each segment connection to protect the LEDs from excessive current.

3. Connect the Common Cathode:

Tie the display's common cathode pin(s) directly to **GND** of the Arduino.

4. Wire the Push Buttons:

- **Increment button:** connect one leg to D9 and the other to GND.
- **Reset button:** connect one leg to D10 and the other to GND.

Both buttons use Arduino's internal pull-up resistors for stable input readings.

5. Upload the Program:

In the Arduino IDE, upload the counter sketch. When powered, the system counts from 0 to 5 automatically, then switches to manual mode where:

- Pressing *Increment* advances the displayed number by 1 (looping 9 \rightarrow 0).
- Pressing *Reset* clears the count to 0.

6. Verify the Operation:

Observe that the 7-segment display updates correctly with each button press. Ensure that debounce delays are implemented in code to avoid multiple counts from a single press.

7. Troubleshoot if Necessary:

- If a segment does not light, recheck its wiring and resistor.
- Verify button placement and ensure correct orientation on the breadboard.
- Confirm that pin definitions in the code match the actual wiring.

Display Type	Pins Required	Data Representation	Coding Complexity	Best Use Case
7-Segment Display	7–8 digital pins	Manual control of each segment	Moderate	Displaying digits (0–9)
LED Matrix	8–16 pins (depends on size)	Control of rows and columns	High	Displaying symbols or scrolling text
I2C LCD Display	2 pins (SDA, SCL)	Text-based control using library	Low	Multi-line text or messages

8.0 DISCUSSION

Expected vs observed outcomes:

Function	Expected Outcomes	Observed Outcomes
Increment button	Displayed number increases by 1 each time the button is pressed	Worked as expected after debouncing
Reset button	Display resets to 0 when pressed	Worked as expected
Wrap around	Display resets from 9 back to 0	Worked as expected
Display stability	Segments light correctly with no flicker	Minor flicker observed due to loose wiring or noise

Sources of Error:

1. Wiring Issues:

Loose or short jumper wires sometimes caused incorrect segment lighting or brief display flickers. This affected the circuit's stability and caused temporary display errors.

2. Mechanical Button Bounce:

The increment button initially produced multiple counts with a single press because of contact bounce. This issue was minimized by adding software delays for debouncing, but minor inconsistencies still occurred.

3. Display Running Automatically:

The display occasionally increased its count without any button input. This was likely caused by electrical interference or floating input pins not properly pulled LOW.

Improvements:

- Utilize an interrupt-driven input detection method to enhance response speed and ensure every button press is accurately registered.
- Employ longer, well-secured wiring connections to maintain consistent electrical contact and minimize signal interruptions.
- Incorporate pull-down resistors or hardware debounce circuits to reduce false triggers and stabilize input signals.
- Implement software-based signal verification or short timing delays to filter out electrical noise and prevent unintended counting.
- Upgrade the design to support multiple-digit displays through multiplexing techniques, allowing for extended counting ranges and broader applications.

Question:

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.

To begin with, an I2C LCD can be connected to an Arduino using only two main communication lines, SDA (Serial Data) and SCL (Serial Clock), along with the power connections VCC and GND. In this setup, the LiquidCrystal_I2C library is used, where the LCD is initialized with its specific I2C address (for example, 0x27) and display size. After that, functions such as `lcd.begin()`, `lcd.setCursor()`, and `lcd.print()` are used to display text. Through the I2C interface, the Arduino sends data serially, and the module converts it into parallel signals to operate the LCD. On the other hand, a 7-segment display requires multiple pins and direct control of each segment (a to g) using HIGH or LOW signals, which makes it less efficient. Moreover, a matrix LED display operates through multiplexing, where the Arduino rapidly switches between rows and columns to display patterns or characters. In conclusion, the I2C LCD offers a simpler and more efficient way to handle wiring and programming by using serial communication instead of manual or multiplexed control.

9.0 CONCLUSION

In this experiment, an Arduino Mega was used to control a common anode 7-segment display with two push buttons. The counter worked as expected, where one button increased the displayed number by one, and the other button reset the count to zero. This shows that the Arduino was able to read input signals correctly and send the right outputs to display the numbers.

Some small issues were observed, such as multiple counts from a single press, loose wiring, and random counting without pressing the button. These problems were mainly caused by button bounce, unstable connections, and electrical noise. By applying software debouncing and checking the wiring, the system became more stable and accurate.

Overall, the experiment was successful because it achieved its main goal and helped us understand how digital inputs and outputs work together in a microcontroller system. This project also showed how software and hardware can be combined to build a simple digital counter, which can be applied to other systems like digital clocks or automatic counting devices.

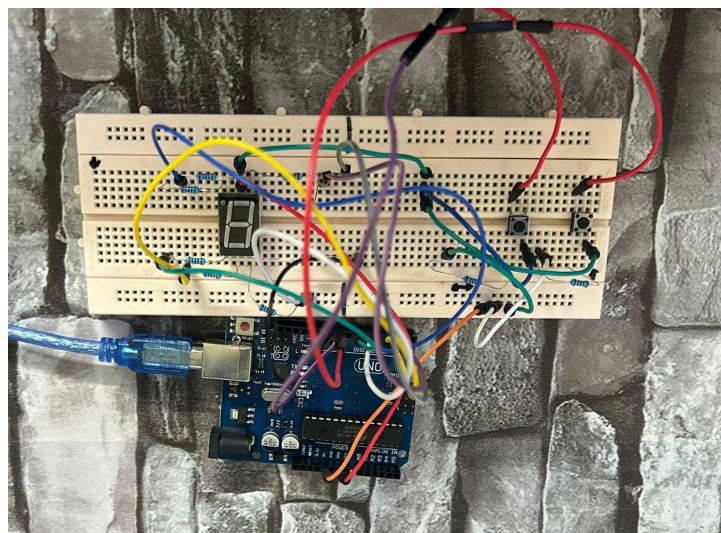
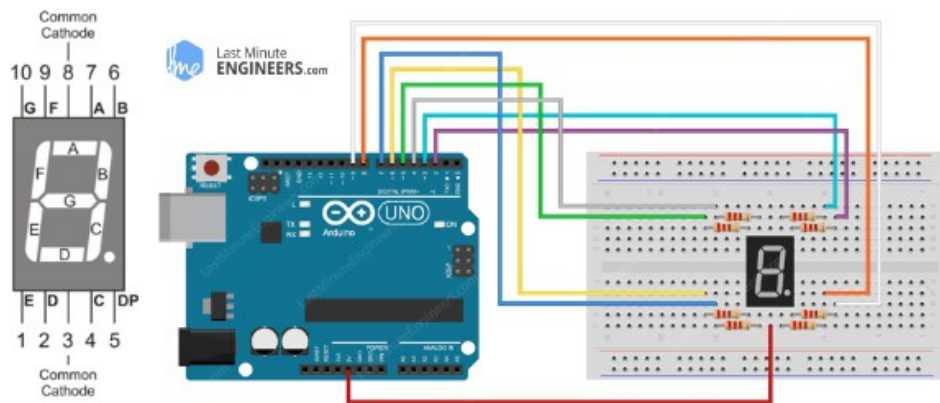
10.0 RECOMMENDATIONS

To enhance the performance and reliability of the digital counter system, several improvements can be implemented. The responsiveness of the push buttons can be increased by using an interrupt-based input system, allowing the Arduino to detect button presses more efficiently and accurately. Incorporating hardware debouncing with resistors or capacitors would help prevent multiple counts from a single press and ensure consistent input readings. Securing all wiring connections properly and using pull-down resistors on the input pins would reduce false triggers caused by electrical noise and create a more stable circuit. Additionally, upgrading the design to a multi-digit counter using multiplexing would allow larger numbers to be displayed. Finally, constructing the circuit on a printed circuit board (PCB) instead of a breadboard would improve durability and reliability, resulting in a more robust and user-friendly experimental setup.

11.0 REFERENCE

1. Staf, L. E. (2023, May 10). In-depth: How seven segment display works & interface with Arduino. Last Minute Engineers.
<https://lastminuteengineers.com/seven-segment-arduino-tutorial/>
2. Storr, W. (2024, December 27). 7-segment display and driving a 7-segment display. Basic. Electronics Tutorials.
<https://www.electronics-tutorials.ws/blog/7-segment-display-tutorial.html>
3. I2C LCD - Seeed Wiki. (n.d.). Wiki.seeedstudio.com.
https://wiki.seeedstudio.com/I2C_LCD/

12.0 APPENDICES



13.0 ACKNOWLEDGEMENT

We would like to express our sincere gratitude to Dr. Zulkifli Bin Zainal Abidin and Dr. Wahyu Sediono for their continuous support and for providing the essential resources and facilities required to carry out this project. We also extend our heartfelt appreciation to everyone who contributed to the success of this lab report through their valuable insights, assistance, and feedback.

14.0 Certificate of Originality and Authenticity

This is to certify that we are responsible for the work submitted in this report, that the original work is our own except as specified in the references and acknowledgement, and that the original work contained herein have not been untaken or done by unspecified sources or persons. We hereby certify that this report has not been done by only one individual and all of us have contributed to the report. The length of contribution to the reports by each individual is noted within this certificate. We also hereby certify that we have read and understand the content of the total report and that no further improvement on the reports is needed from any of the individual contributors to the report.

Signature:



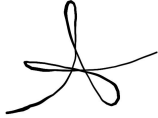
Name: AHMAD FIRDAUS HUSAINI BIN HASAN AL-BANNA
Matric Number: 2315377

Read [/]

Understand [/]

Agree [/]

Signature:



Read [/]

Name: ADIB ZAFFRY BIN MOHD ABDUL RADZI

Matric Number: 2315149

Understand [/]

Agree [/]

Signature:



Read [/]

Name: UMAR FAROUQI BIN ABU HISHAM

Matric Number: 2314995

Understand [/]

Agree [/]