

MECHATRONICS SYSTEM INTEGRATION (MCTA 3203)

SEMESTER 1 2025/2026

WEEK 2: DIGITAL LOGIC SYSTEM

DATE OF EXPERIMENT: 15 OCTOBER 2025

DATE OF SUBMISSION: 22 OCTOBER 2025

SECTION 1

GROUP 13

LECTURER: ZULKIFLI BIN ZAINAL ABIDIN

NO.	NAME	MATRIC
1	HARIZ IRFAN BIN MOHD ROZHAN	2318583
2	ABDULLAH HASAN BIN SIDEK	2318817
3	TAN YONG JIA	2319155
4	NUR QISTINA BINTI MOHD FAIZAL	2319512

ABSTRACT

This experiment focuses on interfacing an Arduino Uno board with a common cathode 7-segment display and push buttons. The purpose of the experiment is to design and implement a simple digital counter capable of incrementing numbers from 0 to 9 using a push button and resetting the count to 0 with another button. The setup involves connecting each segment of the 7-segment display to the Arduino through current-limiting resistors and integrating push buttons with pull-up resistors for stable operation. The Arduino was programmed to detect button presses and update the displayed number accordingly. Results confirmed that the counter system functioned correctly, with sequential number increments and reset operations working as intended. This project demonstrates fundamental concepts of digital logic design, microcontroller programming, and hardware-software integration, which are widely applicable in automation, timers, and embedded system applications.

TABLE OF CONTENTS

ABSTRACT	2
TABLE OF CONTENTS	2
1.0 INTRODUCTION	3
2.0 MATERIALS AND EQUIPMENTS	3
3.0 EXPERIMENTAL SETUP	4
4.0 METHODOLOGY	4
5.0 DATA COLLECTION	9
6.0 DATA ANALYSIS	9
7.0 RESULTS	10
8.0 DISCUSSION	10
9.0 CONCLUSION	11
10.0 RECOMMENDATIONS	12
11.0 REFERENCES	12
12.0 APPENDICES	13
13.0 ACKNOWLEDGEMENTS	14
14.0 STUDENTS DECLARATION	15

1.0 INTRODUCTION

The experiment introduces the practical application of digital logic systems through microcontroller-based control of a numerical display. Using an Arduino Uno board, a common cathode 7-segment display, and push buttons, a digital counting system is constructed. The objective is to increment values from 0 to 9 sequentially with each press of the increment button and reset the value to 0 using a reset button.

A 7-segment display is a commonly used component for numerical output, with each of its seven LEDs forming digits. In this setup, each segment is individually controlled via Arduino digital pins, while push buttons provide user input. The experiment emphasizes basic electronic interfacing, digital input/output handling, and embedded programming concepts such as `digitalRead()`, `digitalWrite()`, and conditional logic.

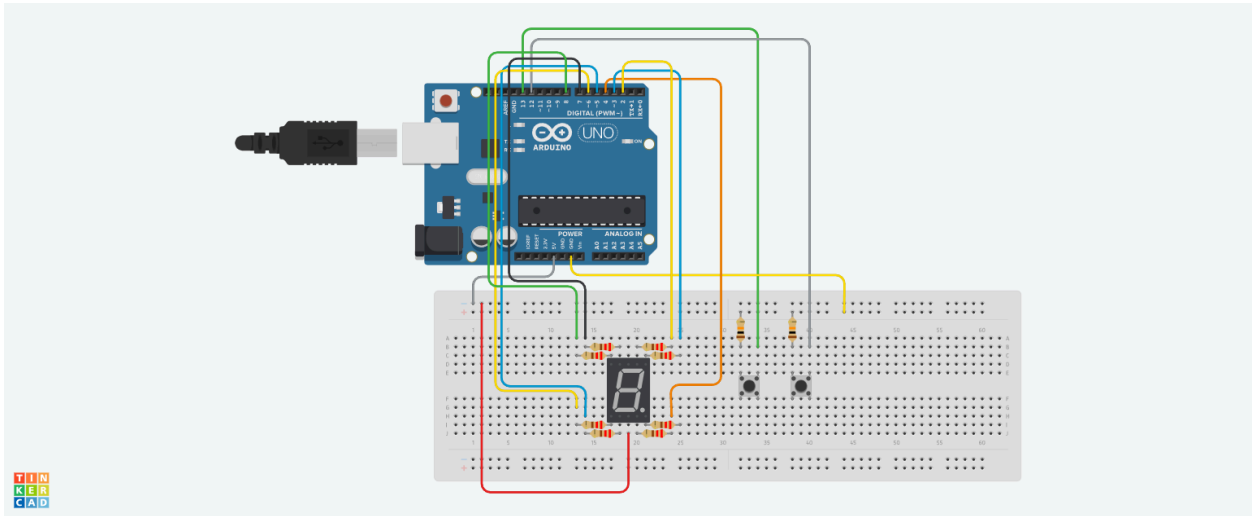
This system represents a simplified version of digital counters widely used in electronic devices, calculators, and clocks, giving students practical exposure to binary logic operations, circuit design, and programming for real-world applications.

2.0 MATERIALS AND EQUIPMENTS

- Arduino Uno board
- Common cathode 7-segment display
- 220-ohm resistors (7 units)
- 10k-ohm resistors (2 units, for pull-ups)
- Push buttons (2 units)
- Breadboard
- Jumper wires

3.0 EXPERIMENTAL SETUP

- The seven segment pins (a–g) of the display are connected to Arduino digital pins D0–D6 through 220-ohm resistors to limit current.
- The common cathode pin of the display is connected to Arduino GND.
- Two push buttons are connected to Arduino:
 - Increment button (D13)
 - Reset button (D12)
- Each button is connected to GND on one leg, with a 10k-ohm pull-up resistor tied between the pin and 5V to ensure stable digital readings.



4.0 METHODOLOGY

1. The circuit was built according to the circuit setup instructions.
2. The Arduino code is uploaded into the Arduino Uno.
3. Pressing the increment button displayed the following number on the 7-segment display.
4. Press and record until the number on the display is 9.
5. The reset button was pressed and the number 0 was displayed.

4.1 Circuit Assembly

- The 7-segment display is connected to the Arduino Uno with individual segment pins (D2-D8) assigned to specific digital output pins.

- Two push buttons are used, one for incrementing the displayed number and another one for resetting it.
- Internal pull-up resistors are enabled for the buttons to ensure stable reading.

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 0.
- A function, `displayNumber (int number)` is used to control the 7-segment display output.

4.3 Code used :

```
// Segment pins
const int segmentA = 2;
const int segmentB = 3;
const int segmentC = 4;
const int segmentD = 5;
const int segmentE = 6;
const int segmentF = 7;
const int segmentG = 8;

// Button pins
const int buttonIncrement = 13;
const int buttonReset = 12;

// 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][7] = {
    {LOW, LOW, LOW, LOW, LOW, LOW, HIGH}, // 0
    {HIGH, LOW, LOW, HIGH, HIGH, HIGH, HIGH}, // 1
    {LOW, LOW, HIGH, LOW, LOW, HIGH, LOW}, // 2
    {LOW, LOW, LOW, LOW, HIGH, HIGH, LOW}, // 3
    {HIGH, LOW, LOW, HIGH, HIGH, LOW, LOW}, // 4
    {LOW, HIGH, LOW, LOW, HIGH, LOW, LOW}, // 5
    {LOW, HIGH, LOW, LOW, LOW, LOW, LOW}, // 6
    {LOW, LOW, LOW, HIGH, HIGH, HIGH, HIGH}, // 7
    {LOW, LOW, LOW, LOW, LOW, LOW, LOW}, // 8
    {LOW, LOW, LOW, LOW, HIGH, 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);

    // 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]);
}

```

4.4 Control Algorithm

1. Defining pins
 - a. Initialize the LEDs light up on a 7-segment display by defining Segment A - G.
Example: `const int segmentA = 2; // D2`
 - b. Button Increment (D13) and Button Reset (D12) 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 detects a new button press.
3. Setup function
 - a. 7-segment display pins are configured as OUTPUT.
 - b. Buttons are configured as INPUT_PULLUP. INPUT_PULLUP makes the pin HIGH by default (button unpressed) and goes LOW when pressed.
4. Main loop:
 - a. `bool incrementState = digitalRead(buttonIncrement)` and `bool resetState = digitalRead(buttonReset)` read the current state of both buttons.
 - b. Increment button logic
 - Checks for button press and increases the count; includes a 200 ms debounce delay to prevent false triggering.
 - After increasing to 9, the count returns to 0.
 - c. Reset button logic
 - If the reset button is pressed, the count is set to 0.
 - d. Update button state
 - Updates previous button states for edge detection.
 - e. Display number

- Calls the function to update the displayed count on the 7-segment display (displayDigit(currentNumber);).
- Before a new number is displayed, every segment shuts off.
- The corresponding segments are mapped to each number.
- The necessary segments are set LOW to form the shape of the number (since common cathode is used).

5.0 DATA COLLECTION

Observation:

Trial	Initial Display	Button Action	Final Display	Result	Remarks
1	0	Increment	1	Pass	Works as intended
2	1	Increment	2	Pass	Works as intended
3	2	Increment	3	Pass	Works as intended
4	3	Increment	4	Pass	Works as intended
5	4	Increment	5	Pass	Works as intended
6	5	Increment	6	Pass	Works as intended
7	6	Increment	7	Pass	Works as intended
8	7	Increment	8	Pass	Works as intended
9	8	Increment	9	Pass	Works as intended
10	9	Increment	0	Pass	Auto reset after 9
11	5	Reset	0	Pass	Manual reset works perfectly

6.0 DATA ANALYSIS

- The counter increased by 1 each time the increment button was pressed.
- The display correctly showed numbers from 0 to 9 in order.
- After reaching 9, the next press resets the display back to 0 automatically.
- The reset button also worked properly, setting the display to 0 anytime it was pressed.
- There were no errors or delays during the operation. Every output matched the expected result.
- Overall, the system functioned smoothly and accurately throughout all trials.

7.0 RESULTS

The experiment was conducted to evaluate the performance of a 7-segment display connected to an Arduino, focusing on two primary operations: increment and reset. The objective was to verify whether the display could accurately count upward and return to zero upon reset.

Overall, the circuit functioned as intended. The increment feature successfully increased the displayed number with each button press, while the reset button effectively returned the counter to zero. Minor issues were observed during the increment operation, including slight delays or skipped numbers, likely caused by button bounce or inconsistent contact. Despite these minor irregularities, the system generally performed according to design expectations.

In conclusion, the experimental results confirm that the 7-segment display counting system operated effectively. Minor inconsistencies did not significantly affect performance, and the observed outcomes closely matched the expected behavior of the Arduino-based counter circuit.

8.0 DISCUSSION

Expected vs Observed Outcomes

Function	Expected Outcome	Observed Outcome
Increment	Increases number by 1 with no issues and restarts back to 0 when exceeds 9	The number increases but sometimes there would be a delay or it has to be pressed a few times to make it work and also the increment would sometimes also skip some numbers
Reset	Resets the counter to 0	Works as expected without any issues

The result of the experiment proves that the 7-segment display successfully performed the increment and reset operations as required. The increment button was able to increment the number displayed by one per button press, while the reset button cleared the display to zero. This proves that button interfacing, display mapping, and logic programming were properly implemented. The system's overall functionality turns out to be appropriate control of the digital signal and synchronization between the 7-segment display and the microcontroller.

According to the observations, the increment function generally worked as intended but contained some minor defects. There used to be a bit of lag in response or the button had to be repeatedly pressed for the display to get refreshed. Sometimes the increment jumped over some numbers, which was likely due to switch bounce or random button contact. The reset worked perfectly and cleared the display to zero with no issues. These results are very similar to the theoretical results, and discrepancies are small and caused by physical or electrical limitations in the circuit.

Sources of error would most likely be mechanical bouncing of the push buttons, which could cause multiple or lost counts unless debounced. Additional sources of error would be minor power supply variations or sloppy wire connections, which can lead to a wobbly input signal. A limitation of the system is that only single-digit numbers (0–9) can be counted on; multiple 7-segment displays or the use of a multiplexing circuit would be required to count beyond the counter. Despite these small problems, the experiment successfully illustrated digital counting and reset control with Arduino, verifying the basic functionality and reliability of the design.

9.0 CONCLUSION

The experiment successfully demonstrated the design and implementation of a digital counter system using an Arduino Uno, a common cathode 7-segment display, and push buttons. The system was able to increment values from 0 to 9 sequentially and reset the counter to 0 as intended. The results confirmed that the hardware connections and the programmed logic functioned correctly, though minor issues such as delayed response or skipped increments were observed during button presses. Overall, the project provided valuable hands-on experience in microcontroller programming, digital input/output handling, and hardware-software integration, reinforcing fundamental concepts of digital logic systems and embedded applications.

10.0 RECOMMENDATIONS

- Debouncing Improvement – Implement a more robust hardware or software debounce mechanism (e.g., RC circuits or more advanced debounce code) to eliminate skipped or repeated counts.
- Use of External Libraries – Employ libraries such as Bounce2 for button handling to improve input stability and reduce coding complexity.
- Alternative Displays – Explore interfacing with an I2C LCD display, which requires fewer pins and allows for text output, making the system more versatile compared to a 7-segment display.
- Scalability – Expand the counter to handle multi-digit displays or integrate additional features such as countdown timers, making it closer to real-world applications.
- Power Efficiency – Optimize the circuit by minimizing unnecessary current draw and considering the use of low-power modes for the Arduino.
- User Experience – Incorporate LEDs or buzzers to give feedback for button presses, ensuring that each input is registered and acknowledged clearly.

11.0 REFERENCES

Arduino. (n.d.). Arduino Uno Rev3. Arduino Documentation. Retrieved October 18, 2025, from <https://docs.arduino.cc/hardware/uno-rev3>

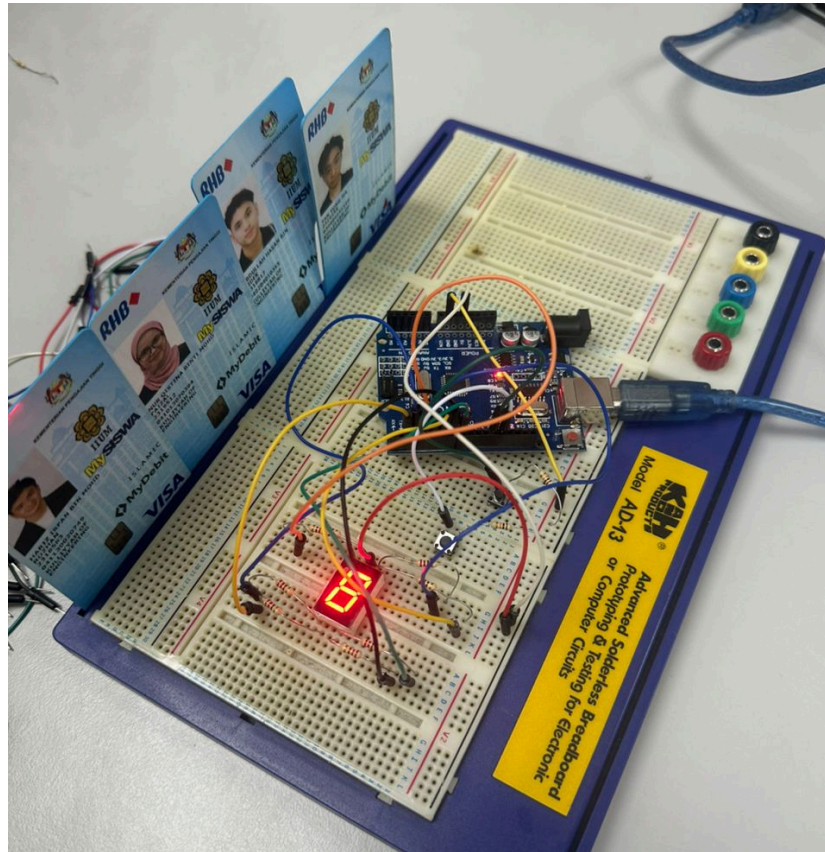
SparkFun Electronics. (n.d.). 7-segment displays tutorial. SparkFun. Retrieved October 18, 2025, from <https://learn.sparkfun.com/tutorials/7-segment-displays>

Arduino. (n.d.). Arduino reference: digital read/write functions. Arduino. Retrieved October 18, 2025, from <https://www.arduino.cc/reference/en/>

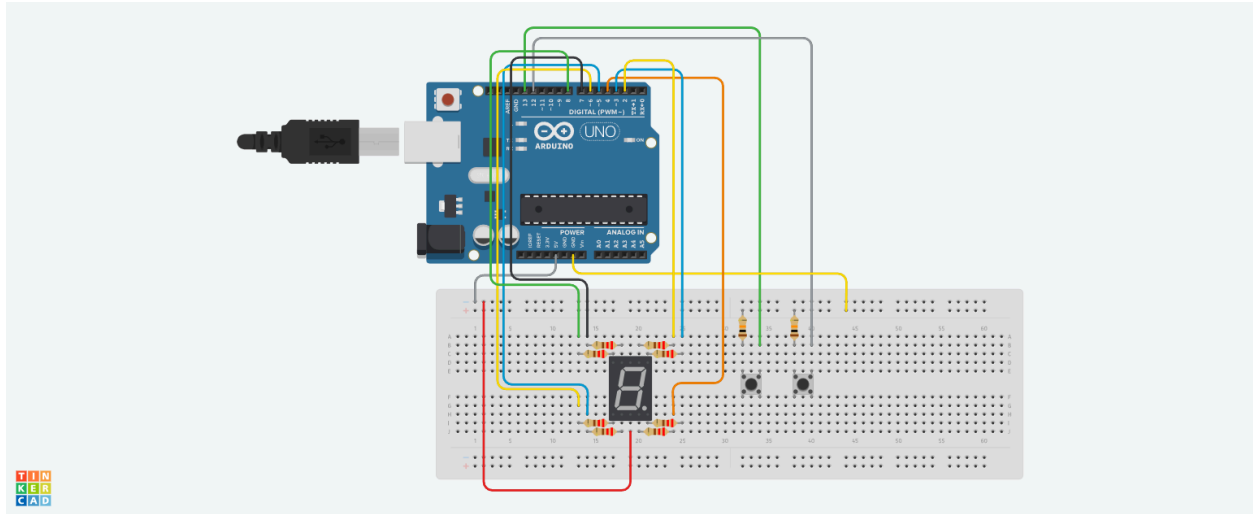
Horowitz, P., & Hill, W. (2015). The art of electronics (3rd ed.). Cambridge University Press.

TutorialsPoint. (n.d.). Interfacing 7-segment display with Arduino. TutorialsPoint.
Retrieved October 18, 2025, from
<https://www.tutorialspoint.com/interfacing-7-segment-display-with-arduino>

12.0 APPENDICES



Circuit Design



Circuit Designed in TinkerCAD





13.0 ACKNOWLEDGEMENTS

Hereby, we acknowledge the guidance provided by Dr Zulkifli with his impeccable knowledge in this field. We also thank Br. Harith with his assistance in correcting our work thus leading to a successful experiment. Not to forget our fellow colleagues from other groups that contributed in providing ideas that helped in achieving the same output together from this experiment.

14.0 STUDENTS DECLARATION

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: 	Read <input type="checkbox"/>
Name: HARIZ IRFAN BIN MOHD ROZHAN	Understand <input type="checkbox"/>
Matric Number: 2318583	Agree <input type="checkbox"/>
Signature: 	Read <input type="checkbox"/>
Name: ABDULLAH HASAN BIN SIDEK	Understand <input type="checkbox"/>
Matric Number: 2318817	Agree <input type="checkbox"/>
Signature: 	Read <input type="checkbox"/>
Name: TAN YONG JIA	Understand <input type="checkbox"/>
Matric Number: 2319155	Agree <input type="checkbox"/>
Signature: 	Read <input type="checkbox"/>
Name: NUR QISTINA BINTI MOHD FAIZAL	Understand <input type="checkbox"/>
Matric Number: 2319512	Agree <input type="checkbox"/>