INTERNATIONAL MULTI-AWARD WINNING INSTITUTION FOR SUSTAINABILITY

LAB REPORT 1

MECHATRONICS SYSTEM INTEGRATION

DR. ZULKIFLI BIN ZAINAL ABIDIN

SECTION 1

GROUP 6

MATRIC NUMBER	NAME
2312369	MUHAMMAD FAHIM BIN AHMAD NORISHAM
2312479	MUHAMMAD HAZIMUDDIN BIN FAIRUZ
2312451	MUHAMMAD ZAMARUL AZIM BIN ZULKHIBRI

DATE OF SUBMISSION

Wednesday, 22nd October 2025

Abstract

This experiment focuses on understanding digital logic systems and the practical interfacing of a 7-segment display using a Cytron CT-UNO microcontroller. The main objective was to learn how to display numeric sequences using push buttons and to understand how digital logic concepts are applied in hardware. A common cathode 7-segment display was connected to the CT-UNO through digital output pins and programmed to count and reset numbers based on user input. The circuit also included debounce handling to ensure accurate counting. Overall, the experiment successfully demonstrated how digital outputs can be used to control display devices in a simple microcontroller system.

Keywords: Digital Logic, 7-Segment Display, Cytron CT-UNO, Arduino, Push Button, Microcontroller, Debounce

Table of Contents

WEEK 2: DIGITAL LOGIC SYSTEM: BASIC LOGIC GATES, ELECTRONIC CIRCUIT INTERFACING, BASIC ALU, 7-SEGMENT DISPLAY, IC-BASED INTERFACING APPLICATIONS

1. Introduction

This week's lab focused on the topic of **Digital Logic Systems**, which are the foundation of all digital electronics such as computers. In digital systems, signals are either ON or OFF, represented by 1 (HIGH) and 0 (LOW). The experiment introduced us to basic logic gates, interfacing techniques, and display devices. Specifically, we used a **7-segment display** to show numbers using digital signals from an Arduino Uno board. The configuration of the 7-segment display pinout is shown in the Figure 1.0 below.

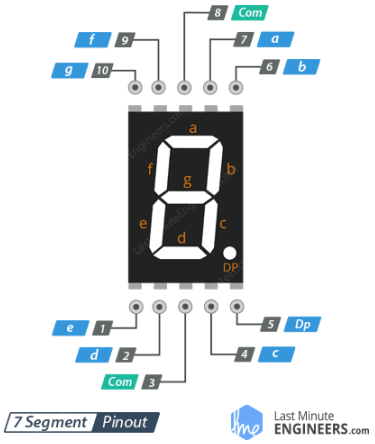


Figure 1.0 Shows the 7-segment display pinout

Our main objectives in this experiment were:

- To learn how a 7-segment display works and how to control each segment.
- To practice connecting electronic components on a breadboard.
- To understand how digital outputs can be programmed to display specific numbers.
- To control the increment sequence of the number display and reset it using push buttons.
- We expected that after connecting all segments properly and uploading the program, the display would show numbers from 0 to 9 correctly and the push button would increase and reset the number.

2. Materials and Equipments

Cytron CT-UNO	1
Micro USB Cable	1
Common Cathode 7-Segment Display (red)	1
220 Ω Resistors	8
10 kΩ Resistors	2
Pushbuttons	2
Breadboard	1
Jumper Wires	

3. Experimental Setup

The 7-segment display has 7 LEDs labeled A to G. Each LED lights up when current flows through it. By turning ON specific combinations of these LEDs, we can display numbers from 0 to 9.

The **common cathode of the 7-segment display** was connected to **5 V** instead of **GND**, which caused the logic to be **reversed**. In this configuration, the segments turn **ON** when the Cytron UNO output is **LOW** and **OFF** when it is **HIGH**. Segment pins A to G were connected to digital pins D2 to D9 through 220 Ω resistors for current protection.

Two pushbuttons were also connected:

- **Button 1 (Increment):** connected to pin D10.
- **Button 2 (Reset):** connected to pin D11.

Both buttons used **10 kΩ resistors** connected to the GND line to ensure stable logic levels.

Circuit Description:

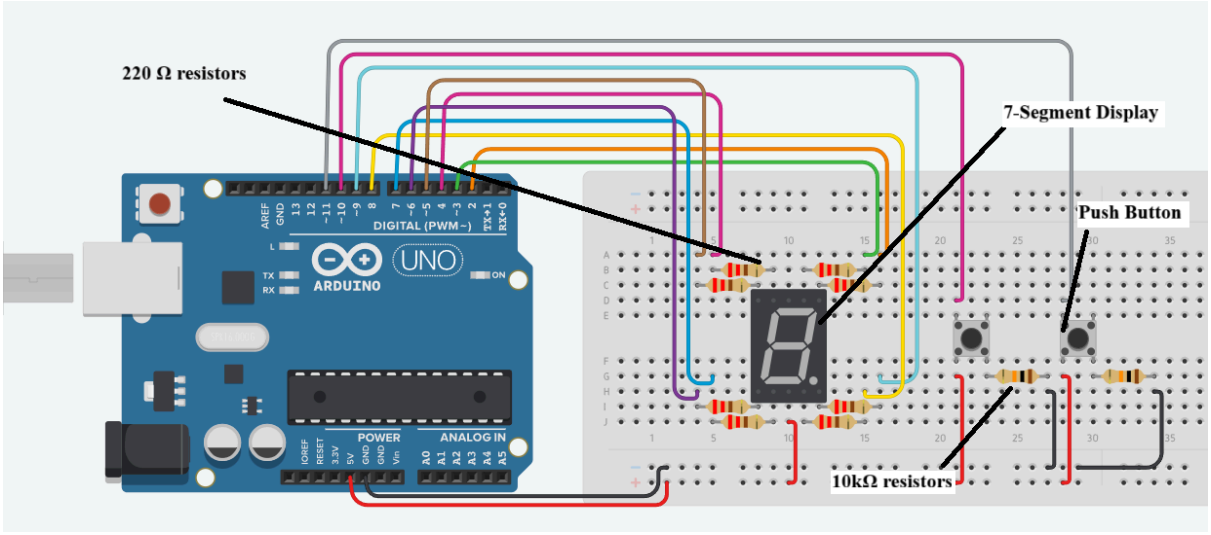


Figure 2.0 Shows the schematic diagram using Tinkercad

- When the Cytron UNO sends a HIGH signal to a pin, that segment lights OFF.
- When the signal is LOW, the segment turns ON.
- The combination of segments turned ON forms a number on the display.

4. Methodology

4.1. Circuit Assembly

All the connections were made on a breadboard based on Figure 2.0. Each of the seven segment pins (segment a to segment g) was connected through a 220 Ω resistor to Cytron UNO pins D2–D9. The common cathode pin was supplied with 5V (logic will be reversed).

4.2. Button Connections

The increment button and reset button were connected to D10 and D11, respectively. Each button was connected to GND through a 10 kΩ resistor on one side and to 5V on the other side.

4.3. Code Upload

The Arduino IDE was used to write and upload the code to Cytron UNO. The program controlled which segments turned ON or OFF to form numbers 0–9.

4.3.1 Example Coding

Coding to display the number from 0 to 9 continuously while following the sequence. Delays were added to make each number stay visible for 1 second before changing.

```
// Define the pins for each segment (D0 to D6)

const int segmentA = 3; // D0
const int segmentB = 2; // D1
const int segmentC = 8; // D2
const int segmentD = 7; // D3
const int segmentE = 6; // D4
const int segmentF = 4; // D5
const int segmentG = 5; // D6

void setup() {
  // Initialize the digital pins as OUTPUTs
  pinMode(segmentA, OUTPUT);
```

```
  // Display 4
  digitalWrite(segmentA, HIGH);
  digitalWrite(segmentB, LOW);
  digitalWrite(segmentC, LOW);
  digitalWrite(segmentD, HIGH);
  digitalWrite(segmentE, HIGH);
  digitalWrite(segmentF, LOW);
  digitalWrite(segmentG, LOW);
  delay(1000);

  // Display 5
  digitalWrite(segmentA, LOW);
  digitalWrite(segmentB, HIGH);
  digitalWrite(segmentC, LOW);
```

```
  pinMode(segmentB, OUTPUT);
  pinMode(segmentC, OUTPUT);
  pinMode(segmentD, OUTPUT);
  pinMode(segmentE, OUTPUT);
  pinMode(segmentF, OUTPUT);
  pinMode(segmentG, OUTPUT);
}

void loop() {
  // Display 1
  digitalWrite(segmentA, HIGH);
  digitalWrite(segmentB, LOW);
  digitalWrite(segmentC, LOW);
  digitalWrite(segmentD, HIGH);
  digitalWrite(segmentE, HIGH);
  digitalWrite(segmentF, HIGH);
  digitalWrite(segmentG, HIGH);
  delay(1000);

  // Display 2
  digitalWrite(segmentA, LOW);
  digitalWrite(segmentB, LOW);
  digitalWrite(segmentC, HIGH);
  digitalWrite(segmentD, LOW);
  digitalWrite(segmentE, LOW);
  digitalWrite(segmentF, HIGH);
  digitalWrite(segmentG, LOW);
  delay(1000);

  // Display 3
  digitalWrite(segmentA, LOW);
  digitalWrite(segmentB, LOW);
  digitalWrite(segmentC, LOW);
  digitalWrite(segmentD, LOW);
  digitalWrite(segmentE, HIGH);
  digitalWrite(segmentF, HIGH);
  digitalWrite(segmentG, LOW);
  delay(1000);

  digitalWrite(segmentD, LOW);
  digitalWrite(segmentE, HIGH);
  digitalWrite(segmentF, LOW);
  digitalWrite(segmentG, LOW);
  delay(1000);

  // Display 6
  digitalWrite(segmentA, LOW);
  digitalWrite(segmentB, HIGH);
  digitalWrite(segmentC, LOW);
  digitalWrite(segmentD, LOW);
  digitalWrite(segmentE, LOW);
  digitalWrite(segmentF, LOW);
  digitalWrite(segmentG, LOW);
  delay(1000);

  // Display 7
  digitalWrite(segmentA, LOW);
  digitalWrite(segmentB, LOW);
  digitalWrite(segmentC, LOW);
  digitalWrite(segmentD, HIGH);
  digitalWrite(segmentE, HIGH);
  digitalWrite(segmentF, HIGH);
  digitalWrite(segmentG, HIGH);
  delay(1000);

  // Display 8
  digitalWrite(segmentA, LOW);
  digitalWrite(segmentB, LOW);
  digitalWrite(segmentC, LOW);
  digitalWrite(segmentD, LOW);
  digitalWrite(segmentE, LOW);
  digitalWrite(segmentF, LOW);
```

4.3.2 Task Coding

The coding was written to display a number on the 7-segment display, starting from **0** when the system is powered on. When the "**increase push button"** is pressed, the number on the display increases by **1**. If the number reaches **9**, it will return back to **0** on the next press. When the "**reset push button"** is pressed, the display will immediately reset and show the number **0** again. This operation allows the user to count numbers from 0 to 9 repeatedly using the increase button and reset the count at any time using the reset button.

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

// Push Button Pins
const int buttonUp = 10;
const int buttonDown = 11;

int currentNumber = 0;
// start from 0
int lastButtonUpState = HIGH;
int lastButtonDownState = HIGH;

void setup() {
  // Set 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);

  // Set button pins as inputs
with pull-ups
  pinMode(buttonUp, INPUT);
```

```
// Function to Display Numbers 0-9
void displayNumber(int num) {
  // LOW = ON, HIGH = OFF
switch (num) {
    case 0:
      setSegments(LOW, LOW, LOW,
LOW, LOW, LOW, HIGH); break;
    case 1:
      setSegments(HIGH, LOW, LOW,
HIGH, HIGH, HIGH, HIGH); break;
    case 2:
      setSegments(LOW, LOW, HIGH,
LOW, LOW, HIGH, LOW); break;
    case 3:
      setSegments(LOW, LOW, LOW,
LOW, HIGH, HIGH, LOW); break;
    case 4:
      setSegments(HIGH, LOW, LOW,
HIGH, HIGH, LOW, LOW); break;
    case 5:
      setSegments(LOW, HIGH, LOW,
LOW, HIGH, LOW, LOW); break;
    case 6:
      setSegments(LOW, HIGH, LOW,
LOW, LOW, LOW, LOW); break;
    case 7:
      setSegments(LOW, LOW, LOW,
HIGH, HIGH, HIGH, HIGH); break;
    case 8:
```

```
  pinMode(buttonDown, INPUT);

  // Display the initial number
  displayNumber(currentNumber);
}

void loop() {
  int upState =
digitalRead(buttonUp);
  int resetstate =
digitalRead(buttonDown);

  // Increase button pressed
  if (upState == LOW &&
lastButtonUpState == HIGH) {
    currentNumber++;
    if (currentNumber > 9)
currentNumber = 0;
    displayNumber(currentNumber);
    delay(200); // debounce
  }
  // Reset button pressed
  if (resetstate == LOW &&
lastButtonDownState == HIGH) {
    currentNumber=0;
    if (currentNumber >= 0)
currentNumber = 0;
    displayNumber(currentNumber);
    delay(200); // debounce
  }
  lastButtonUpState = upState;
  lastButtonDownState =
resetstate;
}
```

```
      setSegments(LOW, LOW, LOW,
LOW, LOW, LOW, LOW); break;
    case 9:
      setSegments(LOW, LOW, LOW,
LOW, HIGH, LOW, LOW); break;
  }
}

// Function to Light Segments
void setSegments(int a, int b, int
c, int d, int e, int f, int g) {
  digitalWrite(segmentA, a);
  digitalWrite(segmentB, b);
  digitalWrite(segmentC, c);
  digitalWrite(segmentD, d);
  digitalWrite(segmentE, e);
  digitalWrite(segmentF, f);
  digitalWrite(segmentG, g);
}
```

4.4 Testing and Observation

After uploading, the display was observed to ensure correct numbers appeared in sequence. The increment and reset buttons were pressed to confirm functionality.

5. Data Collection

During the experiment, the Cytron CT-UNO (Arduino-compatible board) was used to control the circuit and the number displayed on the 7 segment-display. Two main tasks of testing were performed:

5.1. Automatic Display Mode:

- The 7-segment display was programmed to show numbers 1 to 9, followed by 0 in a continuous loop.
- Each number was displayed for approximately one second (1000 ms) before changing to the next number.
- The transition between numbers was smooth, with all relevant segments lighting up correctly for each digit.

5.2. Push Button Controlled Display Mode:

- Two push buttons were connected for increment and reset functions.
- The display started at 0.
- Pressing the increment button increased the number by 1 each time.
- After reaching 9, the next press cycled back to 0.
- The reset button immediately returned the display to 0 regardless of the current number.
- A short delay (200 ms) was used to prevent button bounce from causing double increments.

All results were observed in real time on the breadboard setup and verified through the Serial Monitor for debugging purposes.

6. Data Analysis

The experiment was designed to verify two types of control logic:

1. Automated sequential display using time delay.
2. Manual control using push buttons and conditional programming logic.

The coding for both setups demonstrated correct digital control principles:

- Digital Write Logic: Each segment (a–g) of the display was controlled individually using digitalWrite().
- Logic Reversal: Because the common cathode was mistakenly connected to 5V, the logic had to be reversed (LOW = ON, HIGH = OFF).
- Debounce Handling: The push button program used a short delay to avoid false triggering from mechanical bounce.
- Modular Programming: Functions displayNumber() and setSegments() simplified the code structure and made it reusable.

The performance of the circuit was stable after debugging. All numbers from 0–9 were displayed clearly without flickering. The buttons responded accurately with no skipped or repeated counts after debounce implementation.

7. Results

7.1. Automatic Display Mode:

The 7-segment display successfully showed numbers **0 to 9** in the correct order after supplying the power source to the Cytron UNO. The number is kept looping back to 0 when it reaches 9 until the power source is removed.

7.2. Push Button Controlled Display Mode:

The 7-segment display successfully showed numbers **0 to 9** in the correct order after an increasing push button was pressed and the display **reset to 0** when the reset button was pressed. The reversed wiring (common cathode to 5V) did not stop the circuit from working, but it **inverted the logic** in coding.

The final result proved that the code logic was correct and the system functioned properly.

Mode	Input Method	Expected Output	Actual Output	Remarks
Automatic	Delay-based loop	1 → 2 → … → 9 → 0	0 → 1 → … → 9 → 0	Works correctly, smooth transition
Manual (Increment)	Push Button 1 (D10)	Increase by +1	Works correctly	Stable with debounce
Manual (Reset)	Push Button 2 (D11)	Reset to 0	Works correctly	Immediate response
Logic Configuration	LOW = ON	Inverted Logic	Corrected in code	Functioned properly

Table 1.0 Shows the result of the lab example and task

Both modes successfully demonstrated digital logic control using Arduino programming. No overheating or circuit failure occurred during the test.

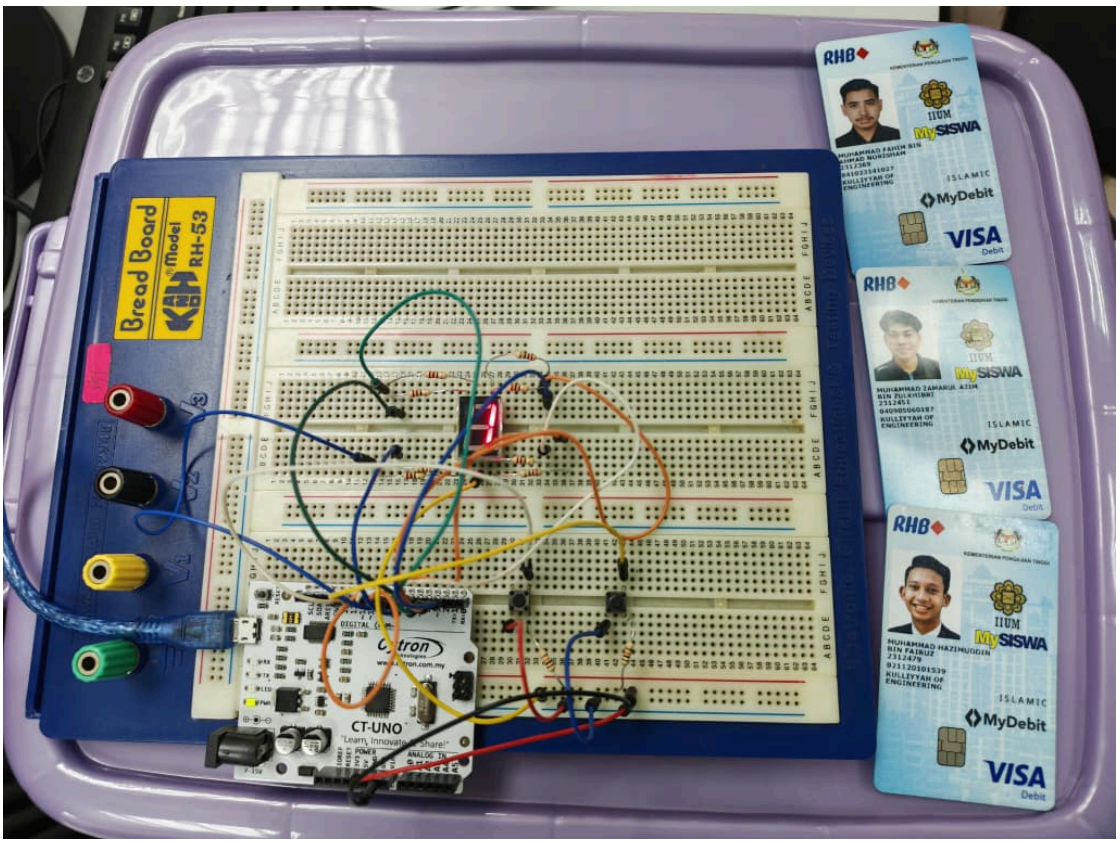


Figure 7.1 SHows the outputs of 1 from 7-segment display

The Cytron UNO successfully controlled the 7-segment display to show numeric outputs from 0 to 9, as demonstrated in Figure 4.1.

Additional images of the display showing all digits (0–9) are provided in *Appendix C* for reference.

8. Discussion

This experiment showed how a Cytron CT-UNO can be used to control a 7-segment display with the help of push buttons and digital output pins. The main idea was to make the display show numbers that increase when one button is pressed and reset when another button is pressed.

In this experiment, the 7-segment display was accidentally connected in the wrong way which is why the common cathode pin was connected to 5V instead of GND. Normally, for a common cathode display, the common pin should be connected to ground. Because of this mistake, the logic became opposite. The display turned ON when the signal was LOW instead of HIGH. This issue was fixed through coding by reversing the logic in the program, showing how flexible microcontrollers can be.

This experiment also proves that coding can sometimes fix minor hardware wiring mistakes. For future experiments, it is recommended to connect the common cathode pin to GND to follow the standard setup and make troubleshooting easier.

Possible sources of error include button debounce, which can cause the number to increase more than once with a single press if there is no proper delay in the code. Loose jumper wires on the breadboard might also cause some segments not to light up properly.

Overall, the experiment successfully achieved its purpose which was to display numbers on a 7-segment display and control them using push buttons while also demonstrating how both hardware and software work together in electronics.

Questions:

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.

In this experiment involving the Cytron CT-UNO and a 7-segment display, the use of an I2C LCD could provide another way to visualize digital outputs while demonstrating more advanced communication methods. Unlike the 7-segment display, which requires multiple digital pins to manually control each LED segment, the I2C LCD uses only two communication lines (SDA and SCL) through serial data transfer. This greatly eases the wiring process as well as uses less pins than a 7-segment display.

In terms of programming, the difference lies in how the data is handled. For the 7-segment display, each segment's state (ON/OFF) or (HIGH/LOW) is directly controlled by digital logic outputs, making the process highly dependent on hardware wiring. In contrast, the I2C LCD uses the I2C protocol, where the microcontroller sends encoded data to the LCD's internal controller. With the help of libraries such as LiquidCrystal_I2C, the user can easily display characters using simple commands like lcd.print(). This approach abstracts the low-level control, focusing more on data display rather than bit-level manipulation.

Compared to matrix LED or 7-segment displays, the I2C LCD offers more flexibility, easier coding, and better readability for multi-character output. It also highlights how digital logic principles extend from direct pin control to serial communication systems, reinforcing the understanding of both hardware interfacing and software abstraction.

Overall, the I2C LCD interface demonstrates a practical evolution from basic digital outputs to more efficient and intelligent display systems, aligning with the experiment's goal of exploring how hardware and software integration enhances control in microcontroller-based designs.

9. Conclusion

The experiment was successful because the Cytron CT-UNO microcontroller was able to control the 7-segment display effectively using simple push-button inputs. The display showed the correct numbers as expected, proving that the system worked as planned.

Although there was a wiring mistake the common cathode of the 7-segment display was connected to 5V instead of GND the problem was fixed through coding. By reversing the logic in the program (making LOW turn the segment ON and HIGH turn it OFF), the display still worked correctly. This shows how powerful and flexible programming can be in solving small hardware issues.

During the experiment, it was also discovered that button **debouncing** can cause problems. When a push button is pressed, it may send multiple signals instead of one, causing the display to count more than once. To reduce this problem, a short delay or debounce code can be added so that each button press is only counted once.

From this experiment, several things were learned:

- How a 7-segment display works and how each segment lights up to form numbers.
- How to use digital inputs (push buttons) and digital outputs (to control the display) in Arduino programming.
- How code logic can correct wiring mistakes and handle issues like button bouncing.

In conclusion, the hypothesis that “a microcontroller can control a 7-segment display using push buttons” is supported by the results. The experiment successfully demonstrated how hardware and software can work together to create a functioning digital display system.

10. Recommendations

1. Correct Hardware Connection:
   Always ensure the common cathode is connected to GND to avoid reversed logic, which can cause confusion during debugging.

2. Use Built-in Pull-up Resistors:
   The pinMode(button, INPUT_PULLUP) configuration can simplify the setup and eliminate the need for external 10 kΩ resistors.

3. Add a Display Counter via Serial Monitor:
   Displaying the current number on the Serial Monitor alongside the 7-segment can help verify real-time operation and debugging.

4. Improve Code Efficiency:
   Using arrays or binary mappings for each digit pattern would make the code shorter and easier to modify.

5. Expand Application:
   Future work can integrate two 7-segment displays to show double-digit numbers or even implement a stopwatch or reaction timer using similar logic.

11. References

All About Circuits. (2019). *All About Circuits - Electrical Engineering & Electronics Community*. Allaboutcircuits.com. https://www.allaboutcircuits.com/

Arduino. (n.d.). *Arduino Docs | Arduino Documentation*. Docs.arduino.cc. https://docs.arduino.cc/

Cytron Technologies. (2025). *Cytron UNO - Arduino UNO Compatible*. Cytron Technologies Singapore. https://sg.cytron.io/p-cytron-uno-arduino-uno-compatible?r=1

International Islamic University Malaysia. (2022). *Mechatronics system integration lab manual: Week 2 – Digital logic system*.

Kuphaldt, T. R. (2015, February 18). *7-segment Display*. Allaboutcircuits.com; All About Circuits. https://www.allaboutcircuits.com/textbook/experiments/chpt-7/7-segment-display/

Appendix A - Circuit Diagram

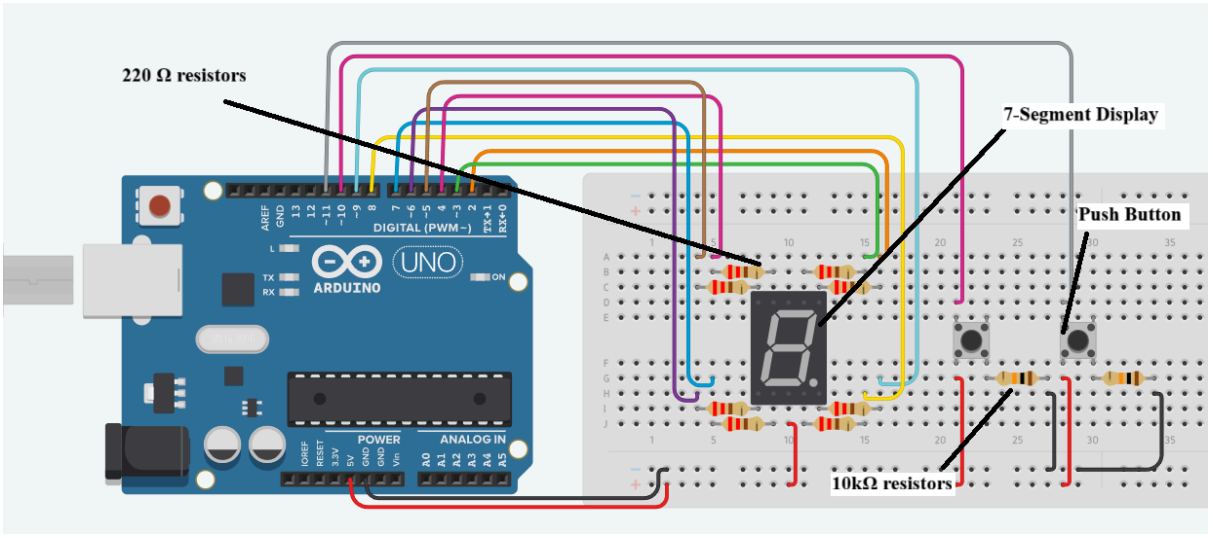


Figure A1 Shows the connection of circuit configuration by using Tinkercad

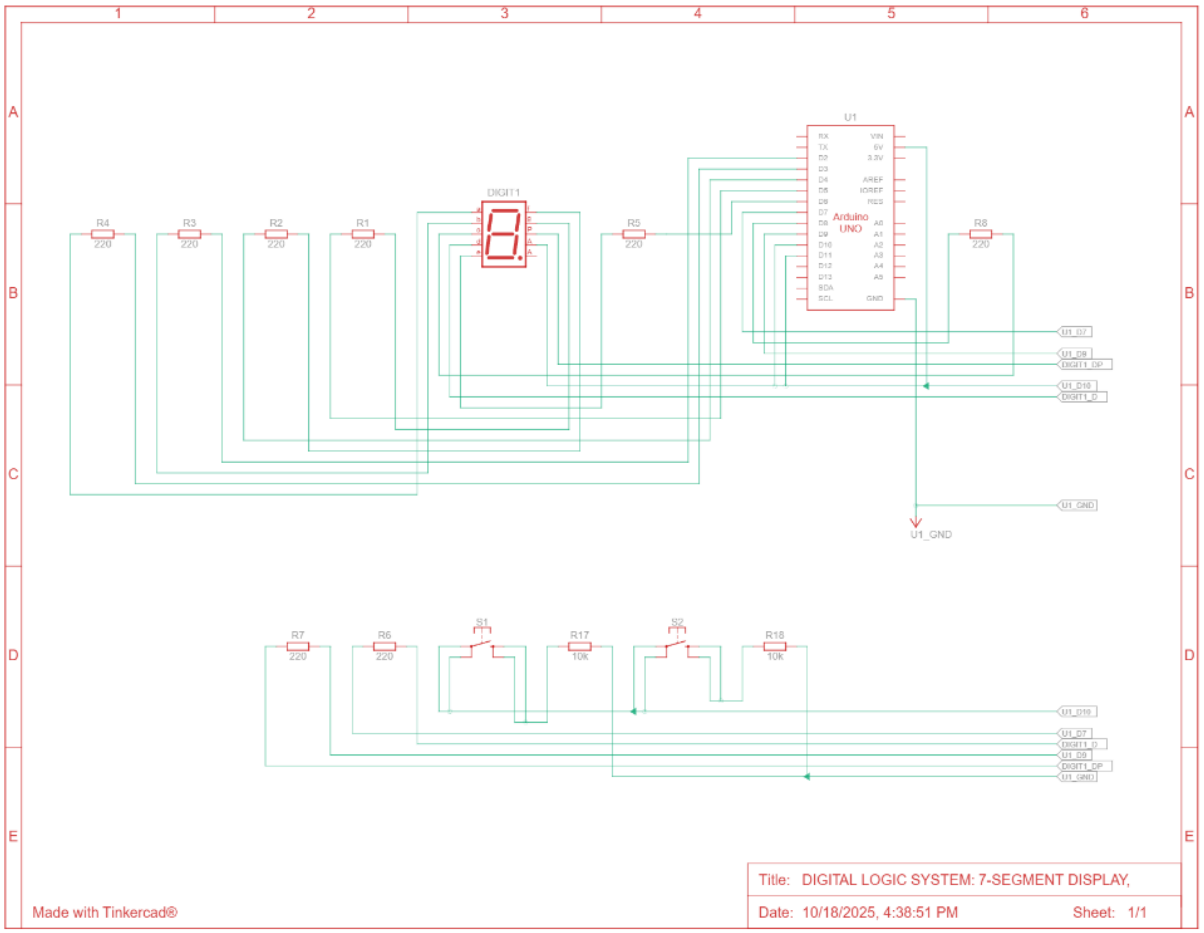


Figure A2 Shows the circuit schematic diagram by using Tinkercad

Appendix B - Coding Snippets

B1 Example Coding

```
// Define the pins for each
segment (D0 to D6)

const int segmentA = 3; // D0
const int segmentB = 2; // D1
const int segmentC = 8; // D2
const int segmentD = 7; // D3
const int segmentE = 6; // D4
const int segmentF = 4; // D5
const int segmentG = 5; // D6

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);
}

void loop() {
  // Display 1
  digitalWrite(segmentA, HIGH);
  digitalWrite(segmentB, LOW);
  digitalWrite(segmentC, LOW);
  digitalWrite(segmentD, HIGH);
  digitalWrite(segmentE, HIGH);
  digitalWrite(segmentF, HIGH);
  digitalWrite(segmentG, HIGH);
  delay(1000);

  // Display 2
  digitalWrite(segmentA, LOW);
  digitalWrite(segmentB, LOW);
```

```
  // Display 4
  digitalWrite(segmentA, HIGH);
  digitalWrite(segmentB, LOW);
  digitalWrite(segmentC, LOW);
  digitalWrite(segmentD, HIGH);
  digitalWrite(segmentE, HIGH);
  digitalWrite(segmentF, LOW);
  digitalWrite(segmentG, LOW);
  delay(1000);

  // Display 5
  digitalWrite(segmentA, LOW);
  digitalWrite(segmentB, HIGH);
  digitalWrite(segmentC, LOW);
  digitalWrite(segmentD, LOW);
  digitalWrite(segmentE, HIGH);
  digitalWrite(segmentF, LOW);
  digitalWrite(segmentG, LOW);
  delay(1000);

  // Display 6
  digitalWrite(segmentA, LOW);
  digitalWrite(segmentB, HIGH);
  digitalWrite(segmentC, LOW);
  digitalWrite(segmentD, LOW);
  digitalWrite(segmentE, LOW);
  digitalWrite(segmentF, LOW);
  digitalWrite(segmentG, LOW);
  delay(1000);

  // Display 7
  digitalWrite(segmentA, LOW);
  digitalWrite(segmentB, LOW);
  digitalWrite(segmentC, LOW);
  digitalWrite(segmentD, HIGH);
  digitalWrite(segmentE, HIGH);
```

```
  digitalWrite(segmentC, HIGH);
  digitalWrite(segmentD, LOW);
  digitalWrite(segmentE, LOW);
  digitalWrite(segmentF, HIGH);
  digitalWrite(segmentG, LOW);
  delay(1000);

  // Display 3
  digitalWrite(segmentA, LOW);
  digitalWrite(segmentB, LOW);
  digitalWrite(segmentC, LOW);
  digitalWrite(segmentD, LOW);
  digitalWrite(segmentE, HIGH);
  digitalWrite(segmentF, HIGH);
  digitalWrite(segmentG, LOW);
  delay(1000);
```

```
  digitalWrite(segmentF, HIGH);
  digitalWrite(segmentG, HIGH);
  delay(1000);

  // Display 8
  digitalWrite(segmentA, LOW);
  digitalWrite(segmentB, LOW);
  digitalWrite(segmentC, LOW);
  digitalWrite(segmentD, LOW);
  digitalWrite(segmentE, LOW);
  digitalWrite(segmentF, LOW);
```

B2 Task Coding

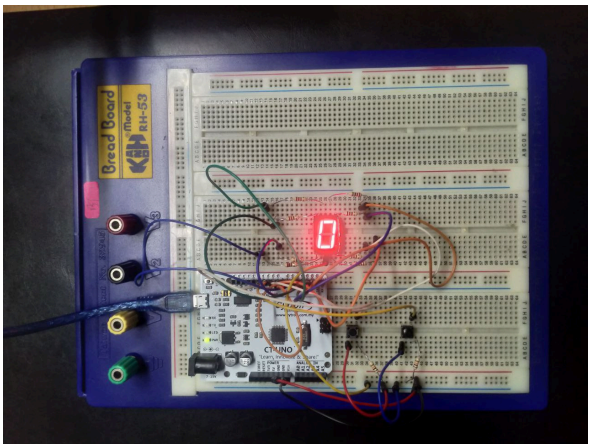
```
const int segmentA = 3;
const int segmentB = 2;
const int segmentC = 8;
const int segmentD = 7;
const int segmentE = 6;
const int segmentF = 4;
const int segmentG = 5;

// Push Button Pins
const int buttonUp = 10;
const int buttonDown = 11;

int currentNumber = 0;
// start from 0
int lastButtonUpState = HIGH;
int lastButtonDownState = HIGH;

void setup() {
  // Set 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);

  // Set button pins as inputs
with pull-ups
  pinMode(buttonUp, INPUT);
  pinMode(buttonDown, INPUT);

  // Display the initial number
  displayNumber(currentNumber);
}

void loop() {
  int upState =
digitalRead(buttonUp);
```

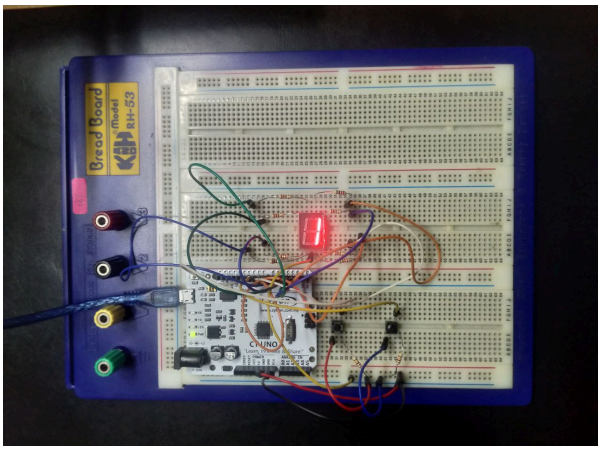
```
// Function to Display Numbers 0-9
void displayNumber(int num) {
  // LOW = ON, HIGH = OFF
switch (num) {
    case 0:
      setSegments(LOW, LOW, LOW,
LOW, LOW, LOW, HIGH); break;
    case 1:
      setSegments(HIGH, LOW, LOW,
HIGH, HIGH, HIGH, HIGH); break;
    case 2:
      setSegments(LOW, LOW, HIGH,
LOW, LOW, HIGH, LOW); break;
    case 3:
      setSegments(LOW, LOW, LOW,
LOW, HIGH, HIGH, LOW); break;
    case 4:
      setSegments(HIGH, LOW, LOW,
HIGH, HIGH, LOW, LOW); break;
    case 5:
      setSegments(LOW, HIGH, LOW,
LOW, HIGH, LOW, LOW); break;
    case 6:
      setSegments(LOW, HIGH, LOW,
LOW, LOW, LOW, LOW); break;
    case 7:
      setSegments(LOW, LOW, LOW,
HIGH, HIGH, HIGH, HIGH); break;
    case 8:
      setSegments(LOW, LOW, LOW,
LOW, LOW, LOW, LOW); break;
    case 9:
      setSegments(LOW, LOW, LOW,
LOW, HIGH, LOW, LOW); break;
  }
}

// Function to Light Segments
```

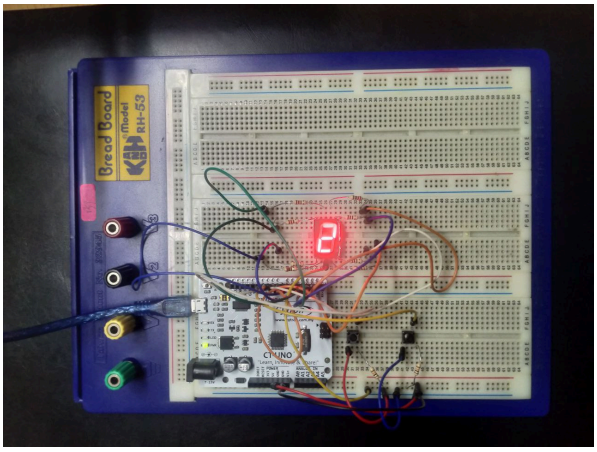
```
  int resetstate =
digitalRead(buttonDown);

  // Increase button pressed
  if (upState == LOW &&
lastButtonUpState == HIGH) {
    currentNumber++;
    if (currentNumber > 9)
currentNumber = 0;
    displayNumber(currentNumber);
    delay(200); // debounce
  }
  // Reset button pressed
  if (resetstate == LOW &&
lastButtonDownState == HIGH) {
    currentNumber=0;
    if (currentNumber >= 0)
currentNumber = 0;
    displayNumber(currentNumber);
    delay(200); // debounce
  }
  lastButtonUpState = upState;
  lastButtonDownState =
resetstate;
}
```

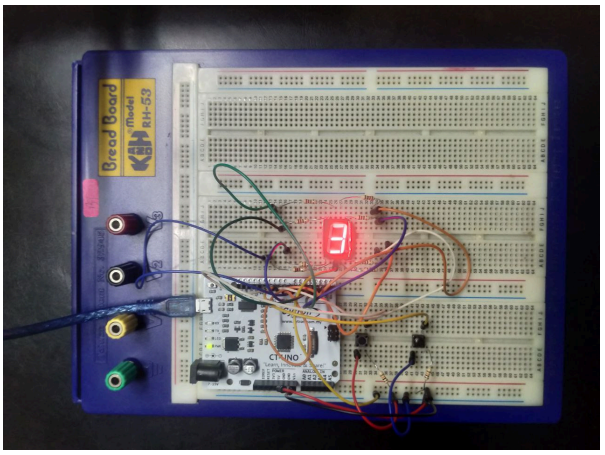
```
void setSegments(int a, int b, int
c, int d, int e, int f, int g) {
  digitalWrite(segmentA, a);
  digitalWrite(segmentB, b);
  digitalWrite(segmentC, c);
  digitalWrite(segmentD, d);
  digitalWrite(segmentE, e);
  digitalWrite(segmentF, f);
  digitalWrite(segmentG, g);
}
```

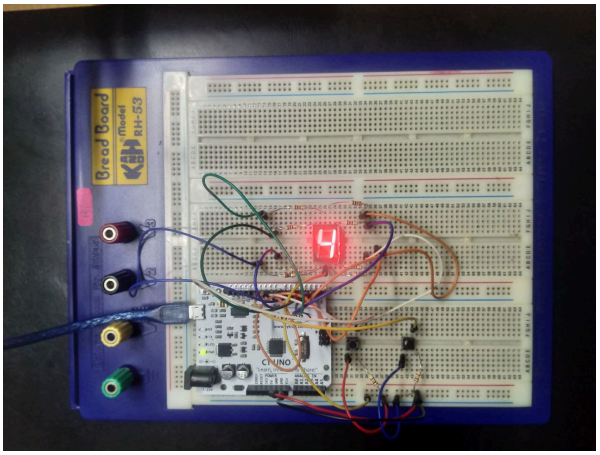
Appendix C - 7-Segment Display Results

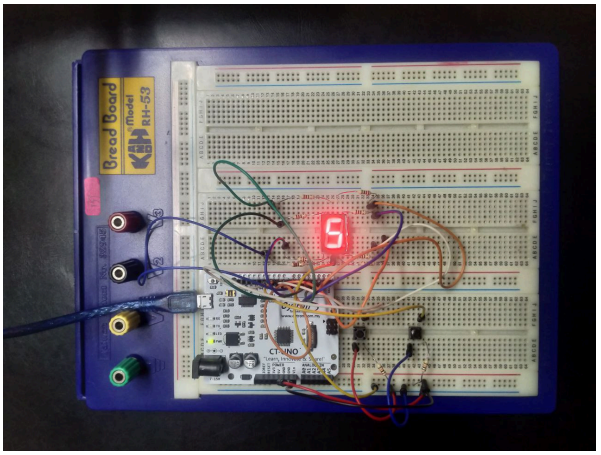












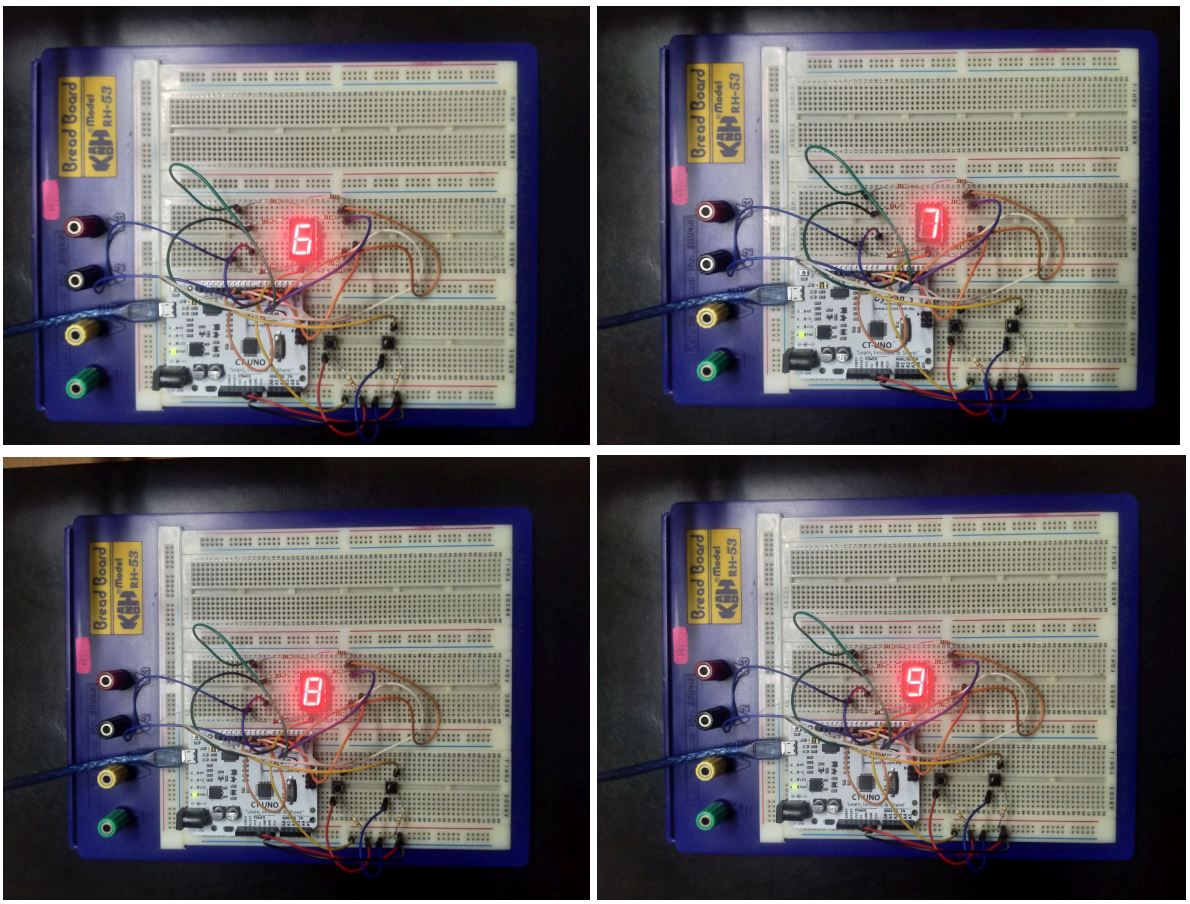


Figure C Shows the results of the 7-Segment Display

Acknowledgments

We would like to express my sincere gratitude to my lecturer and lab instructor for their continuous guidance, clear explanations, and assistance with experimental setups throughout this experiment. Their detailed instructions with the several needed files attached have helped us understand how to properly set up the Cytron CT-UNO microcontroller, connect the 7-segment display, and program the system using Arduino IDE.

I would also like to thank the teaching assistants for their patience and assistance during the lab session. Their help in identifying wiring mistakes and explaining how to correct the logic in the program was very valuable.

In addition, I truly appreciate my classmates and lab partners for their teamwork, cooperation, and ideas during the experiment. We worked together to troubleshoot connection issues, share components, and test the code to make sure everything functioned as expected.

Overall, the success of this experiment would not have been possible without the guidance and support from both the instructors and my lab partners.

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 specific in references and acknowledgement, and that the original work contained herein have not been untaken or done by unspecified sources or a person.

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 no further improvement on the reports is needed from any of the individual's contributors to the report.

We, therefore, agreed unanimously that this report shall be submitted for **marking** and this **final printed report** has been **verified by us**.

Signature: 	Read /
	Understand /
Name: Muhammad Fahim Bin Ahmad Norisham	Agree /
Matric Number: 2312369	
Signature: 	Read /
Name: Muhammad Zamarul Azim Bin Zulkhibri	Understand /
Matric Number: 2312451	Agree /
Signature: 	Read /
Name: Muhammad Hazimuddin Bin Fairuz	Understand /
Matric Number: 2312479	Agree /