



الجامعة الإسلامية العالمية ماليزيا
INTERNATIONAL ISLAMIC UNIVERSITY MALAYSIA
يُونَيْبَرَسِيَّتِي إِسْلَامُ أَنْتَارَايَغْسِيَا مَلَيْسِيَا
Garden of Knowledge and Virtue

REPORT 1: DIGITAL LOGIC SYSTEM

GROUP 2

MCTA 3203

SEMESTER 1 2024/2025

MECHATRONICS SYSTEM INTEGRATION

DATE OF SUBMISSION: 23/10/2024

	NAME	MATRIC NUMBER
1.	AKMAL ZARIF BIN NAJIB	2211971
2.	AMIR FARHAN BIN GHAFAR	2115617
3.	AMIRAH HUDA BINTI JAMALULAIL ASRI	2210776
4.	AMIRUL AIZAD BIN AMIR	2211263
5.	ANAS BIN BADRULZAMAN	2219945

TABLE OF CONTENTS

Introduction.....	2
Abstract.....	2
Materials and Equipment.....	3
Experimental Setup.....	3
Methodology.....	3
Procedure.....	4
Results.....	6
Question.....	7
Discussion.....	7
Conclusion.....	11
Recommendation.....	11
References.....	12
Acknowledgement.....	12
Student's Declaration.....	12

INTRODUCTION

7-segment displays are widely used in digital devices to display numerical information. It consists of seven LEDs (called segments) arranged in such a way that they can form numbers when illuminated in different combinations. In this experiment, we will explore the process of interfacing a common cathode 7-segment display with an Arduino Uno. By building circuits and using push buttons to control counts, we aim to gain practical experience in connecting electronic components to the Arduino board. Through this, we will also develop a better understanding of how to manipulate digital outputs and inputs in Arduino using basic components such as displays and push buttons. The hands-on approach in this experiment reinforces key concepts in electronics and programming while highlighting the interaction between hardware and software in embedded systems.

ABSTRACT

This experiment aimed to interface a common cathode 7-segment display with an Arduino Uno board. The circuit setup involved connecting each of the 7 segments of the display to separate digital pins on the Arduino, along with an appropriate current-limiting resistor. Additionally, push buttons were connected to the Arduino to annually increase the count displayed on the 7-segment display and reset it to zero.

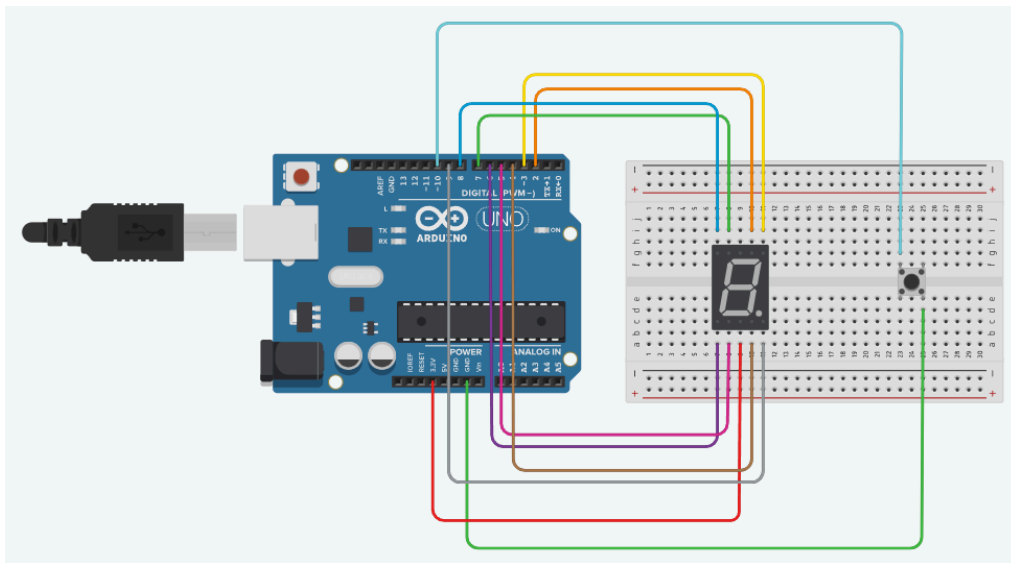
Upon uploading the provided Arduino code, observation was made on the sequential display of numbers from 0 to 9 of the 7-segment display by pressing the increment button. Furthermore, pressing the reset button will reset the count to zero, demonstrating basic control of the display using the push button.

Throughout this experiment; gaining practical insight into interfacing electronic components with Arduino, understanding the principles of controlling a 7-segment display and utilising push buttons for manual input are some of the few foundational knowledge that serves as a basis for more complex projects Arduino-based logic systems and interfacing applications.

MATERIALS AND EQUIPMENT

- Arduino Uno board
- Common cathode 7-segment display
- Pushbuttons (2 or more)
- Jumper wires
- Breadboard

EXPERIMENTAL SETUP



METHODOLOGY

The methodology of this experiment focuses on interfacing a common cathode 7-segment display with an Arduino Uno board. The display was used to show numbers, and two push buttons were employed to control the counting and reset functions. The steps involve wiring, coding, and testing the hardware components, with a particular focus on understanding digital input/output control.

1. Arduino and 7-segment Display Setup:

- The 7-segment display was connected to the Arduino Uno using jumper wires. Each segment (A, B, C, D, E, F, G) was connected to a digital pin on the Arduino.
- Push buttons were used for the input of incrementing and resetting the display count.

2. Programming:

- A program was developed in the Arduino IDE to control the segments of the display. The code was designed to increment the displayed number when the push button was pressed and reset it when another button was pressed.
- The logic was written in C/C++ and uploaded to the Arduino to control the display output.

3. Testing and Observation:

- The system was tested to ensure that each segment of the display worked correctly. The increment and reset buttons were tested to ensure proper functionality.
- A minor issue was encountered with the segment E of the display, likely due to a hardware failure (LED burnout).

PROCEDURE

1. Wiring the 7-Segment Display:

- Connect each pin of the 7-segment display to digital pins on the Arduino Uno. The following pin assignments were used:

Segment A →	Pin 2
Segment B →	Pin 3
Segment C →	Pin 4
Segment D →	Pin 5
Segment E →	Pin 6
Segment F →	Pin 7
Segment G →	Pin 8
Segment Decimal Point (DP) →	Pin 9

- Use a breadboard to organize connections and connect the common cathode of the display to the ground.

Connecting the Push Buttons:

- One push button is connected to a digital pin (Pin 10) and GND, configured to increment the displayed number.
- Another button is connected similarly to reset the display.

2. Programming the Arduino:

- Write a program to control the display. The code initializes the pins for each segment and reads input from the buttons.
- The increment button increases the displayed number, and the reset button clears the display.

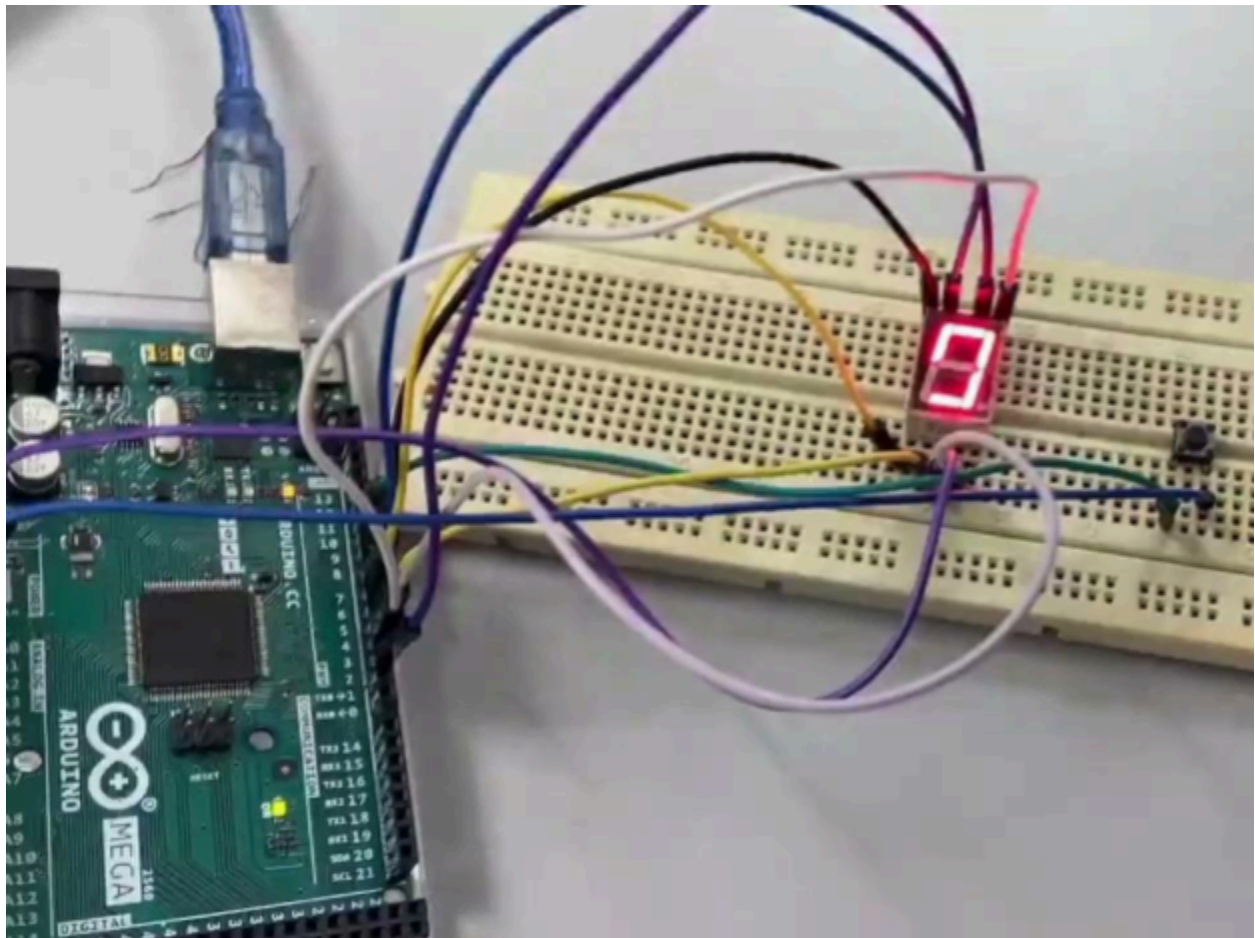
3. Testing the Setup:

- Upload the code to the Arduino and test the display by pressing the increment and reset buttons.
- Observe the display counting from 0 to 9 and resetting to 0 when the reset button is pressed.

4. Troubleshooting:

- During the testing phase, if any segment does not function as expected, check the wiring and the code. In this experiment, Segment E was found to be non-functional, possibly due to a burned-out LED.

RESULTS



QUESTION

1) How to interface an I2C LCD with Arduino? Explain the coding principle behind it compared with 7 segments display and matrix LED.

Attach the Arduino board and I2C LCD. In general, connect the LCD's SDA pin to the Arduino's A4 pin and the SCL pin to the A5 pin; make sure the LCD VCC and GND pins are linked to the appropriate power and ground pins on the Arduino. Due to coding principles, each segment of the 7-segment display is controlled independently by connecting it directly to a specific pin on the Arduino. To display a digit, the corresponding pin must be set to HIGH or LOW, depending on the segment you want to light up. However, in the case of matrix LEDs, they are arranged in rows and columns. By Setting the matching row pin to HIGH and column pin to LOW, a given LED can be turned on. Multiplexing is necessary to quickly cycle between rows and columns to give the appearance of a continuous display. On The other hand, regardless of the size of the display, the I2C LCD requires only two pins(SDA and SCL)for communication. This reduces the number of pins required by the Arduino and simplifies wiring.

DISCUSSION

- **Software**

In this experiment, Arduino software is used to program the microcontroller. Arduino offers an open-source platform, which makes it easy to write, build and upload codes to Arduino boards. It has a user-friendly interface which features a text editor, a toolbar with common actions, and various menus providing access to examples, libraries, and tools. To manage the behaviour of the microcontroller, the C/C++ programming language is used. Attached below is the coding of the project :


```

1  // Define pins for each segment of the 7-segment display
2  const int segmentA = 2;
3  const int segmentB = 3;
4  const int segmentC = 4;
5  const int segmentD = 5;
6  const int segmentE = 6;
7  const int segmentF = 7;
8  const int segmentG = 8;
9  const int segmentDP = 9; // Decimal point (optional)
10
11 // Define pin for the push button
12 const int buttonPin = 10; // Button connected to pin 10
13
14 // Array to hold segment states for digits 0-9 (for common anode)
15 const int digitArray[10][7] = {
16     {0, 0, 0, 0, 0, 0, 1}, // 0
17     {1, 0, 0, 1, 1, 1, 1}, // 1
18     {0, 0, 1, 0, 0, 1, 0}, // 2
19     {0, 0, 0, 0, 1, 1, 0}, // 3
20     {1, 0, 0, 1, 1, 0, 0}, // 4
21     {0, 1, 0, 0, 1, 0, 0}, // 5
22     {0, 1, 0, 0, 0, 0, 0}, // 6
23     {0, 0, 0, 1, 1, 1, 1}, // 7
24     {0, 0, 0, 0, 0, 0, 0}, // 8
25     {0, 0, 0, 0, 1, 0, 0} // 9
26 };
27
28 int buttonState; // Variable to hold the button state
29 int previousButtonState = HIGH; // Previous button state to detect changes
30 int currentDigit = 0; // Variable to keep track of the current digit (0-9)
31
32 void setup() {
33     // Set segment pins as outputs
34     pinMode(segmentA, OUTPUT);
35     pinMode(segmentB, OUTPUT);
36     pinMode(segmentC, OUTPUT);
37     pinMode(segmentD, OUTPUT);
38     pinMode(segmentE, OUTPUT);
39     pinMode(segmentF, OUTPUT);
40     pinMode(segmentG, OUTPUT);
41     pinMode(segmentDP, OUTPUT); // Optional, if using the decimal point
42
43     // Set button pin as input with an internal pull-up resistor
44     pinMode(buttonPin, INPUT_PULLUP);
45
46     // Initially display 0
47     displayDigit(currentDigit);
48 }
49

```

```

50 void loop() {
51     // Read the button state (LOW when pressed, HIGH when not pressed)
52     buttonState = digitalRead(buttonPin);
53
54     // Check if the button has been pressed (change from HIGH to LOW)
55     if (buttonState == LOW && previousButtonState == HIGH) {
56         delay(50); // Debounce delay to avoid multiple triggers
57         // Increment the digit and wrap around if it exceeds 9
58         currentDigit++;
59         if (currentDigit > 9) {
60             currentDigit = 0;
61         }
62         // Display the new digit
63         displayDigit(currentDigit);
64     }
65
66     // Store the current button state for the next loop iteration
67     previousButtonState = buttonState;
68 }
69
70 // Function to display a digit on the 7-segment display
71 void displayDigit(int num) {
72     digitalWrite(segmentA, digitArray[num][0]);
73     digitalWrite(segmentB, digitArray[num][1]);
74     digitalWrite(segmentC, digitArray[num][2]);
75     digitalWrite(segmentD, digitArray[num][3]);
76     digitalWrite(segmentE, digitArray[num][4]);
77     digitalWrite(segmentF, digitArray[num][5]);
78     digitalWrite(segmentG, digitArray[num][6]);
79     digitalWrite(segmentDP, HIGH); // Turn off decimal point (optional)
80 }
81

```

- **Electrical**

The key electrical components utilised in this project are the 7-segment display and the push button, which are connected to the Arduino Mega 2560. Each segment of the display labelled A, B, C, D, E, F, G, and DP is connected to a specific PWM (Pulse Width Modulation) pin on the Arduino Mega. This pin allocation enables individual control of each segment. The connections are as follows :

Segment A = PWM 2

Segment B = PWM 3

Segment C = PWM 4

Segment D = PWM 5

Segment E = PWM 6

Segment F = PWM 7

Segment G = PWM 8

Segment DP = PWM 9

Moreover, the push button is connected to PWM pin 10 and GND (ground) on the Arduino Mega. This button acts as an input mechanism as it allows the user to interact with the project. By pressing the button, users can trigger specific actions programmed into the microcontroller. In this project, pressing the push button will trigger each segment in the 7-segment display accordingly.

However, in our project, segment E on the 7-segment display seems to be not working although the wiring and coding are correct. It is suspected that the LED in segment E was burnt out. Therefore, the number displayed on the 7-segment display on each press of the push button is not exact.

- **Hardware**

The digital counter system is created by employing an Arduino Mega 2560 microcontroller board, a 7-segment display, and a push button. The Arduino oversees the counting operation by increasing the displayed number with every touch of the push button. The circuit interconnects the components using male-to-male jumper wires on a breadboard to guarantee accurate wiring for optimal functioning. The operating logic depends on the Arduino's digital input/output pins to connect with the push button for detecting input and the 7-segment display for displaying numbers. Throughout the experiment, the circuit exhibited stability with reliable operation of the push button and jumper wires except for the 7-segment display. There were no observed instances of erratic behaviour or unexpected resets except for the suspected burnt-out LED in segment E of the 7-segment display.

CONCLUSION

To conclude, the interfacing of a 7-segment display with an Arduino board to display values from 0 to 9 was successful. Each individual segment displayed was controlled by the Arduino which validates the electrical and programming parts of the experiment. However, a minor inconvenience came in the form of hardware as one of the segments was not able to be displayed due to a possible burnt-out. Apart from that, the setup works perfectly well avoiding any flickering and wiring problems. This experiment accurately showcases the principles of the digital logic system that was stated at the start of this report. It provides practical insight into developing a user-friendly electronic display as well as reinforcing our understanding of hardware interfacing. Further enhancements could be explored to help make the system more effective.

RECOMMENDATIONS

Several modifications can be made to improve the performance of this 7-segment display system. As mentioned above, it is suspected that one of the segments in the LED has been burnt out. Using a different LED or upgrading to a multi-digit display would significantly improve the functionality of this digital logic system. This would require changes in the programming code as well as the connections between the electrical components and the Arduino board. Adding other input devices such as a potentiometer would add a different dynamic to the already functioning system. It is a variable resistor that provides a varying amount of resistance when its shaft is turned, thus the brightness of the LED on the segment display can be altered. This could also be done by implementing pulse-width modulation (PWM) to adjust the duty cycle and thus the brightness of the LED. Lastly, incorporating error detection in the code could improve system reliability for real-world applications.

REFERENCES

“Arduino Integrated Development Environment (IDE) v1 | Arduino Documentation | Arduino Documentation,” *Arduino.cc*, 2022.

<https://docs.arduino.cc/software/ide-v1/tutorials/arduino-ide-v1-basics/>

ACKNOWLEDGEMENT

We would like to express our gratitude to Dr. Wahju Sediono, Dr. Ali Sophian, Dr. Zulkifli bin Zainal Abidin, my teaching assistant, and my peers for their invaluable help and support in finishing this project. Their advice, feedback, and experience have greatly influenced the level of quality and understanding of this work.

STUDENT’S DECLARATION


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 has 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.


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

Signature: 	Read	/
Name: AKMAL ZARIF BIN NAJIB	Understand	/
Matric No: 2211971	Agree	/

Signature: 	Read	/
Name: AMIR FARHAN BIN GHAFAR	Understand	/
Matric No: 2115617	Agree	/

Signature: 	Read	/
Name: AMIRAH HUDA BINTI JAMALULAIL ASRI	Understand	/
Matric No: 2210776	Agree	/

Signature: 	Read	/
Name: AMIRUL AIZAD BIN AMIR	Understand	/
Matric No: 2211263	Agree	/

Signature: 	Read	/
Name: ANAS BIN BADRULZAMAN	Understand	/
Matric No: 2219945	Agree	/