



REPORT ON MECHATRONICS SYSTEM INTEGRATION

DIGITAL LOGIC DESIGN GROUP 7

SECTION 2, SEMESTER 1, 24/25

TEAM MEMBERS

| NAME | MATRIC NO. |
|---------------------------------------|------------|
| MUHAMMAD ZAMIR FIKRI BIN MOHD ZAMRI | 2212515 |
| MUHD AKMAL HAKIM BIN SAIFUDDIN | 2216093 |
| NUR SHADATUL BALQISH BINTI SAHRUNIZAM | 2212064 |
| NORHEZRY HAKIMIE BIN NOOR FAHMY | 2110061 |
| NUR AMIRA NAZIRA BINTI MOHD NASIR | 2110026 |

DATE OF EXPERIMENT: 16 OCTOBER 2024

DATE OF SUBMISSION:

ABSTRACT

The experiment aims to show how to integrate an Arduino microcontroller with a 7-segment display to count from 0 to 9 using a push button as an input trigger. By developing a fundamental counting mechanism, the goal is to look into fundamental digital logic systems and electrical circuit interfaces. Every display segment is connected to the digital pins of the Arduino, and the counting sequence is started by pressing a push button. A binary lookup table programs the display's segments. The capacity of the system to consistently count from 0 to 9 is the main learning; it showed accurate user input recognition as well as accurate counting output. The experiment displays the importance of simple digital circuits for real-world applications and underlines the need for bouncing when working with mechanical push buttons. In a nutshell, the experiment serves as a basic exercise for more complex digital systems and offers insightful information on digital logic user interfaces.

TABLE OF CONTENT

| | |
|------------------------------|----|
| TABLE OF CONTENT..... | 3 |
| INTRODUCTION..... | 4 |
| MATERIALS AND EQUIPMENT..... | 5 |
| EXPERIMENTAL SETUP..... | 5 |
| METHODOLOGY..... | 6 |
| DATA COLLECTION..... | 7 |
| DATA ANALYSIS..... | 8 |
| RESULT..... | 8 |
| DISCUSSION..... | 9 |
| CONCLUSION..... | 14 |
| RECOMMENDATION..... | 15 |
| REFERENCES..... | 16 |
| APPENDICES..... | 17 |
| STUDENT DECLARATION..... | 19 |

INTRODUCTION

The goal of this experiment is to investigate the basic ideas of digital logic systems through the combination of an Arduino microcontroller with a 7-segment display to develop a basic counting mechanism that displays numbers 0 through 9. The counting sequence is started by pressing a push button, which acts like an input trigger. This is a real-world example of how user inputs may work with microcontrollers to manage external displays. Typically found in digital clocks and calculators, the 7-segment display is constructed from up of seven LEDs that selectively activate segments to produce the numbers 0 through 9. To create the right digit, the Arduino microcontroller sends binary data to each display segment. A table for looking up information then converts the binary values into segment control signals. Bouncing methods are applied to remove the inconsistent behavior caused by signal noise when the button is pressed. The push button acts as a digital input, allowing users to increase the displayed number with each press. It is expected that the system would operate without problems counting from 0 to 9 with each button click and immediately returning to 0 when it reaches 9. It is projected that the 7-segment display would accurately depict the numbers and that the system will behave steadily and consistently, free from mistakes like missing counts or inaccurate displays brought on by inconsistent input. This experiment illustrates the core concepts of digital logic, electronic interface, and rudimentary programming, offering insight into the construction and operation of basic digital systems in real-world situations.

MATERIALS AND EQUIPMENT

- Arduino Uno board
- Common cathode 7-segment display
- 220-ohm resistors (7 of them)
- Pushbuttons (2 or more)
- Jumper wires
- Breadboard

EXPERIMENTAL SETUP

An Arduino Uno microcontroller was connected to a 7-segment display in this experiment, and each counting sequence was commandeered by a push button. Each of the seven display segments (designated A–G) was connected to the digital pins of the Arduino (pins 2–8) using a 220 Ω current-limiting resistor to prevent damage to the LEDs. The ground was connected to the display's common cathode. A 10k Ω pull-down resistor was placed between the button's input and ground to control the signal, and a push-button was connected to digital pin 9 of the Arduino to act as an input trigger. This ensures that the button will indicate LOW when it is not pressed and HIGH when it is. To turn on the corresponding segments and generate numerals 0-9. The Arduino was connected to the system through a power supply which is the laptop. The Arduino IDE was used to build and upload the software, and the circuit was put together on a breadboard. Using a table of information to connect each number to the appropriate segment control, the code was created to increase the number presented with each button click. Verification was done on the system to make sure the numbers, ranging from 0 to 9, were shown accurately and that the counting stopped when the number reached 9.

METHODOLOGY

Connect the common cathode 7-segment display to the Arduino Uno as follows:

- Connect each of the 7 segments (a, b, c, d, e, f, g) of the display to separate digital pins on the Arduino (e.g., D0 to D6).
 - Connect the common cathode pin of the display to one of the GND (ground) pins on the Arduino.
 - Use 220-ohm resistors to connect each of the segment pins to the Arduino pins to limit the current.
1. Connect the pushbuttons to the Arduino:
 - Connect one leg of each pushbutton to a separate digital pin (e.g., D9 and D10) and connect the other leg of each pushbutton to GND.
 - Use 10K-ohm pull-up resistors for each pushbutton by connecting one end of each resistor to the digital pin and the other end to the 5V output of the Arduino.

Experiment Steps:

1. Build the circuit according to the circuit setup instructions.
2. Upload the provided Arduino code to your Arduino Uno.
3. Open the Serial Monitor in the Arduino IDE.
4. Press the increment button to increase the count. The 7-segment display should show the numbers from 0 to 9 sequentially.
5. Press the reset button to reset the count to 0.

This simple experiment helps you understand how to interface a 7-segment display with an Arduino Uno and how to control it manually using buttons. You can extend this experiment by adding more functionality or additional displays for more complex projects.

DATA COLLECTION

| LIGHTEN SEGMENTS | | | | | | | NUMBER DISPLAY |
|------------------|---|---|---|---|---|---|-------------------|
| A | B | C | D | E | F | G | |
| 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 |

Table 1: Truth Table

Attach the Arduino Uno to the common cathode 7-segment display in the manner described below:

- 1) Attach each of the display's seven segments (a, b, c, d, e, f, and g) to a different digital pin on the Arduino (for example, D0 to D6).
- 2) Attach one of the Arduino's GND (ground) pins to the display's common cathode pin.
- 3) To limit the current, connect each segment pin to the Arduino pin using a 220-ohm resistor.

Attach the Arduino to the pushbuttons:

- 1) Each pushbutton should have one leg connected to a different digital pin (such as D9 or D10) and the other leg connected to GND.

- 2) Each pushbutton should have a 10K-ohm pull-up resistor connected to the digital pin at one end.

DATA ANALYSIS

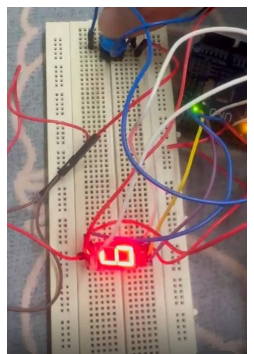
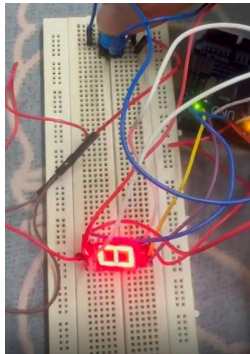
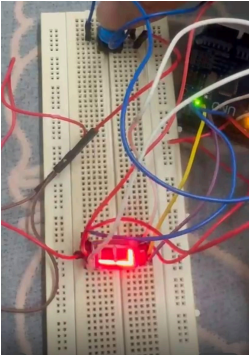
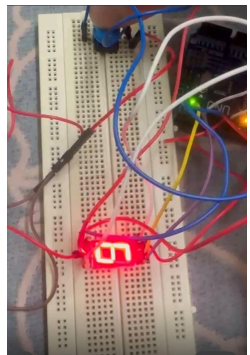
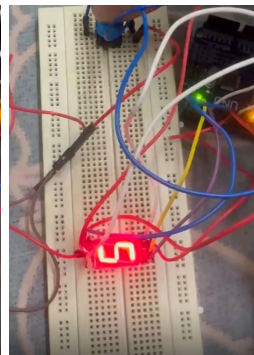
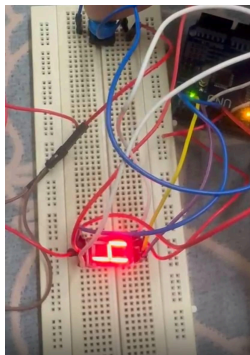
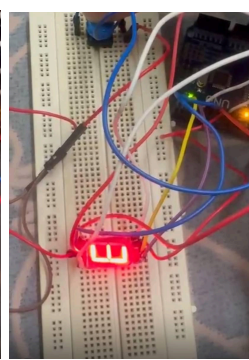
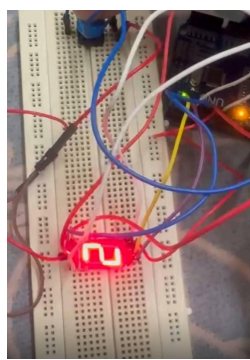
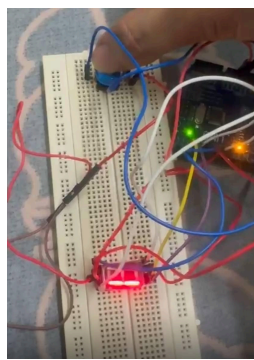
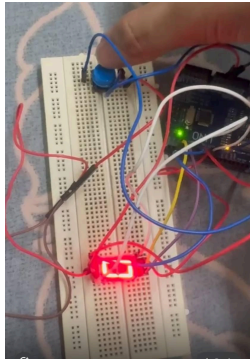
| BINARY INPUT | LIGHTEN SEGMENTS | | | | | | | NUMBER DISPLAY |
|--------------|------------------|---|---|---|---|---|---|----------------|
| | A | B | C | D | E | F | G | |
| 0000 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| 0001 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0010 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 2 |
| 0011 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 3 |
| 0100 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 4 |
| 0101 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 5 |
| 0110 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 6 |
| 0111 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 7 |
| 1000 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 8 |
| 1001 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 9 |

Table 2: Truth Table for Binary Input

Mathematical models that we have used to run this common cathode-7 segments is by using truth table construction. 1 and 0 that represent Table 1 is the segment is ON and OFF state. Which means the segment will light if the segment is 1 (ON) and otherwise the segments 0(OFF).Based on this experiment as well, we can conclude that this logic system has its efficiency, accuracy, and reliability in representing digits (0-9).By employing binary input and combinational logic, the truth table precisely regulates the 7-segment display, enabling it to depict the numbers 0–9. To guarantee that the proper digit is shown, each binary input sets off the appropriate combination of segment activations.Because not every segment is illuminated at once unless the number eight is displayed, the design uses less power.The truth table can be used to construct boolean logic equations that can be used to implement this system in hardware, such as logic

gates or decoders. The logic design is accurate and dependable for numerical displays in a range of digital applications, according to the examination.

RESULT



DISCUSSION

QUESTION

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

When compared with a substitute displays such as matrix LEDs or 7-segment displays, connecting an Arduino to an I2C LCD is a simple yet efficient procedure. By using the I2C communication standard, the I2C LCD drastically reduces the amount of cables needed. Usually, just four connections are required: SDA (data line), SCL (clock line), GND, and VCC. Advanced displays with fewer pins are made possible by these connections, which enable communication between the Arduino and the LCD over a common serial data bus. The LiquidCrystal_I2C library is frequently used to operate the display, providing a simple method for controlling the brightness, printing text, and setting cursor areas. The I2C LCD combines this control into two data lines, as opposed to seven-segment displays that need plenty pins for each segment (one for each of the seven segments to produce numbers). In addition to this, I2C displays are better suited to displaying text or many lines of data, which are examples of comprehensive information.

Coding-wise, the I2C LCD works on the basis of transmitting commands to the display using the I2C protocol. The process of communicating is made simpler by the use of high-level functions like `lcd.print()`. A seven-segment display, on the other hand, requires manual segment control by modifying each pin's level. Since the display can only show one digit at a time, it is more laborious and limiting to demonstrate the number "1" by turning on only the B and C segments. Although matrix LEDs offer greater graphical output versatility, they necessitate a more intricate control system. This involves addressing each LED in a grid by turning on certain rows and columns, which frequently calls for shift registers or additional drivers. While matrix LEDs are excellent for animations and scrolling text, they involve more complicated logic compared to the straightforward text display on an I2C LCD.

Software Discussion

In this project, a 7-segment display interfaced with an Arduino was used to develop a digital counting system. The display was incremented from 0 to 9 using a push button. Connecting each segment of the seven-segment display to the digital pins of the Arduino was the first step in bringing the circuit together. In particular, segments A through G were controlled by pins 2 through 8, correspondingly, with each connection going via a 220Ω resistor to guard against excess current flowing to the display's LEDs. With the goal to use the button as an input to control the counting sequence, a push button was then attached to pin 9 of the Arduino and set up with an internal pull-up resistor.

A control algorithm was programmed into the Arduino that can identify button pushes and show the associated number on the 7-segment display. In order to manage button debouncing, the code was developed with a 200 ms delay. This was put in place to prevent mistakes brought on by the button's mechanical design, which might allow it to record several pushes quickly. When an actual push was detected, the system increased the number that was shown and reset it to zero when it reached nine. A custom function called `displayNumber()` was then given the number to be shown. It implemented a switch statement to decide which segments to activate, switching on the relevant LEDs to create the right digit.

A function for cleaning the display before updating it with a new number was also incorporated into the code's structure. This gave an accurate and clear representation of the current number by guaranteeing that none of the parts from the previous number stayed lighted. The `displayNumber()` function defines each number by activating the appropriate segment combination. For instance, although the number "1" only required segments B and C to be active, the number "0" required segments A, B, C, D, E, and F to be active.

The push button was pressed to test the configuration once the circuit and code were put together. The displayed number increased with each press, and when it reached "9," it reset to "0" as planned. Based on the debounce logic, the system processed input without any problems and displayed the right sequence without any button bouncing issues. This method highlighted how straightforward input control techniques may be used to useful digital logic applications by successfully implementing a simple counting system using an Arduino and a 7-segment display.

CODING

```
// Define the pins for
each segment (D0 to
D6)
const int segmentA = 2;
// D0
const int segmentB = 3;
// D1
const int segmentC = 4;
// D2
const int segmentD = 5;
// D3
const int segmentE = 6;
// D4
const int segmentF = 7;
// D5
const int segmentG = 8;
// D6

const int buttonPin = 9;
// Button connected to
pin 9
int currentNumber = 0;
// Track the current
number to display

unsigned long
lastDebounceTime = 0;
// Last time the button
was pressed
const unsigned long
debounceDelay = 200; //
Debounce delay in
milliseconds

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);

    // Initialize the button
    pin as INPUT with
    internal pull-up resistor
    pinMode(buttonPin,
INPUT_PULLUP);

    // Display the initial
    number (0)

    displayNumber(currentN
umber);
}

void loop() {
    // Check if the button is
    pressed
    if
    (digitalRead(buttonPin)
    == LOW) {
        // Check if enough
        time has passed to
        account for button
        debounce
        if (millis() -
lastDebounceTime >
debounceDelay) {
            // Increment the
            number and wrap
            around to 0 after 9
            currentNumber =
(currentNumber + 1) %
10;

            displayNumber(currentN
umber); // Update the
            display

            // Update the last
            debounce time
            lastDebounceTime =
            millis();
        }
    }
}

// Function to display a
specific number on the
7-segment display
void displayNumber(int
number) {
    // Turn off all segments
    initially
    clearDisplay();

    // Set segments based
    on the current number
    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(segmentG,
HIGH);

digitalWrite(segmentE,
HIGH);
```

```
digitalWrite(segmentD,
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;
```

```
    }  
}  
  
// Helper function to turn  
off all segments  
void clearDisplay() {  
    digitalWrite(segmentA,  
LOW);  
    digitalWrite(segmentB,  
LOW);  
    digitalWrite(segmentC,  
LOW);  
    digitalWrite(segmentD,  
LOW);  
    digitalWrite(segmentE,  
LOW);  
    digitalWrite(segmentF,  
LOW);  
    digitalWrite(segmentG,  
LOW);  
}
```

CONCLUSION

Using an Arduino Uno board for communicating with a common cathode 7-segment display turned out to be a useful and interesting project in the fields of electrical circuit interfacing and digital logic systems. by uploading the supplied file and according to the circuit setup instructions. Using Arduino code, participants were able to program the display to display numbers 0 through 9 in a sequential fashion and use push buttons to reset the count to zero.

Through the use of push buttons for input, linking segment displays, and writing Arduino code to regulate display actions, this experiment enabled an understanding of Arduino-based interface approaches. We learnt a lot about electronic circuit interface, fundamental logic gates, and using Arduino for basic arithmetic functions.

In the future, more complex tasks including intricate logic systems, extra displays, or integration with other electrical components can be performed using this core knowledge. All things considered, this project provides a fundamental basis for investigating the enormous potential of digital systems and interface applications based on Arduino.

RECOMMENDATION

A variety of improvements and changes might be taken into consideration for further versions of the experiment in order to improve the system's functioning and the learning process. First, there may be more area for displaying more data, such as error messages or real-time sensor data, if you choose a more sophisticated LCD with more character space, such a 20x4 I2C display. This would enable students to experiment with a greater variety of uses, such connecting sensors or other external parts to the Arduino. To improve the input system's responsiveness, a more reliable button debouncing technique utilising hardware-based debounce circuits or software libraries is additionally suggested. Students might further expand their knowledge of digital interface by experimenting with different communication protocols.

It is evident from the lessons learnt that handling various display types—such as LCD, matrix, and 7-segment—offers a greater comprehension of the many visual styles in which data may be displayed. It strongly encourages students to experiment with different display drivers and explore libraries such as `LiquidCrystal_I2C`. Investigating real-world uses for these displays, such interactive dashboards or digital counters, might be helpful to future students. By offering a more comprehensive understanding of embedded systems, providing projects that involve various inputs (such as buttons, sensors) and outputs (such as screens, buzzers) will further improve the experiment. Future students may find the experiment more interesting and relevant as a result of these advancements.

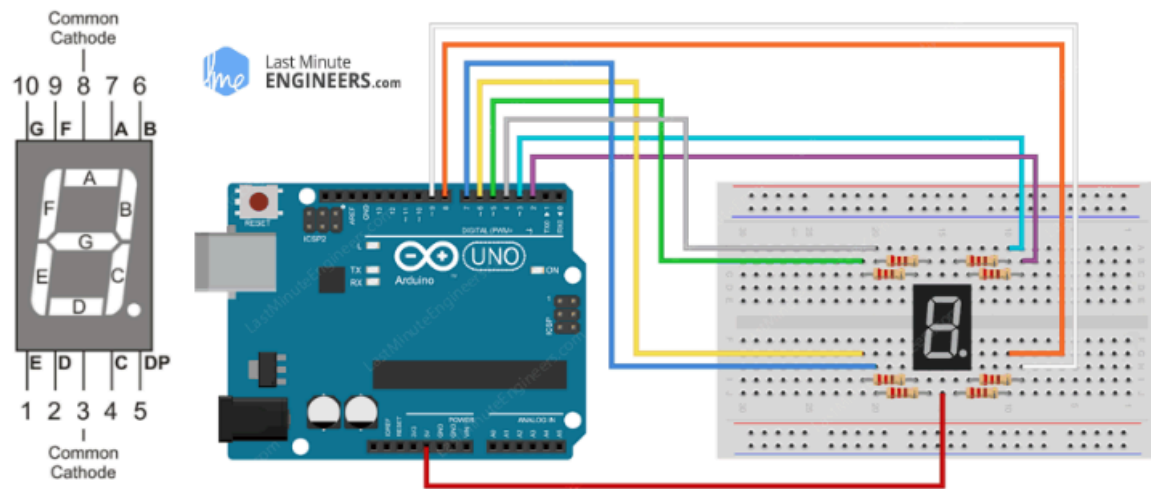
REFERENCES

Arduino. (n.d.). *Arduino reference: LiquidCrystal I2C library*. Arduino.
<https://www.arduino.cc/en/Reference/LiquidCrystalI2C>

Tech Explorations. (2020). *Arduino step-by-step: Getting started*. Tech Explorations.
<https://techexplorations.com/guides/arduino/arduino-i2c-lcd>

SparkFun Electronics. (n.d.). *Using a 7-segment display*. SparkFun.
<https://learn.sparkfun.com/tutorials/using-a-7-segment-display>

APPENDICES



Circuit diagram

STUDENT 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 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.

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

| | | |
|---|------------|---|
| Name : Muhammad Zamir Fikri Bin Mohd Zamri | Read | / |
| Matric No. : 2212515 | Understand | / |
|  Signatures : | Agree | / |

| | | |
|---------------------------------------|------------|---|
| Name : Muhd Akmal Hakim Bin Saifuddin | Read | / |
| Matric No. : 2216093 | Understand | / |

| | | |
|---------------------------|-------|---|
| Signatures : <i>akmal</i> | Agree | / |
|---------------------------|-------|---|

| | | |
|---|------------|---|
| Name : Nur Shadatul Balqish Binti Sahrnizam | Read | / |
| Matric No. : 2212064 | Understand | / |
| Signatures : <i>shadatul</i> | Agree | / |

| | | |
|---------------------------------------|------------|---|
| Name :NORHEZRY HAKIMIE BIN NOOR FAHMY | Read | / |
| Matric No. :2110061 | Understand | / |
| Signatures : <i>hezry</i> | Agree | / |

| | | |
|--|------------|---|
| Name : Nur Amira Nazira Binti Mohd Nasir | Read | / |
| Matric No. :2110026 | Understand | / |
| Signatures : <i>amira</i> | Agree | / |