MECHATRONICS SYSTEM INTEGRATION (MCTA 3203)

SEMESTER 2 2024/2025

WEEK 2: DIGITAL LOGIC DESIGN

SECTION 1

GROUP 5

LECTURER: ZULKIFLI BIN ZAINAL ABIDIN & WAHJU SEDIONO

Date of Experiment: Monday, 10 March 2025

Date of Submission: Monday, 17 March 2025

NO.	GROUP MEMBERS	MATRIC NO.
1.	BUSHRA BINTI A. RAHIM	2318514
2.	AUFA SIDQY BINIT MOHD SIDQY	2310542
3.	NUR HAMIZAH BINTI MUHAMAD HAZMI	2319132

ABSTRACT

This experiment aims to interface an Arduino Uno and a common cathode 7-segment display with two push buttons, one for incrementing the values and another one for resetting the display. The objective is to display numerical values from 0 to 9, increasing them sequentially with each button press, while the reset button returns the display to 0. This experiment involved hardware assembly and software programming, where the Arduino detect button presses and increase the numerical value or reset the system to 0 if another one button is pressed. The programming logic used digital input detection, and conditional statements to manage the display output. The results confirmed a successful interface between hardware and software, demonstrating how microcontrollers can be used for digital counting applications. This experiment highlights the importance of efficient coding, structured circuit design, and troubleshooting techniques in embedded systems, which are widely applicable in real-world devices like digital counters, timers, and automation systems.

TABLE OF CONTENTS

1.0 INTRODUCTION

This experiment focuses on interfacing an Arduino Uno with a common cathode 7-segment display, controlled by two push buttons—one for incrementing numbers and the other for resetting the display. The main objective is to develop a simple digital counter system that can display numbers from 0 to 9, increasing one step at a time with each button press while resetting to zero when the reset button is pressed. This project demonstrates microcontroller-based numerical display control, which is commonly used in digital counters, clocks, and other real-world embedded systems.

A 7-segment display is a visual indicator used to display numerical digits and some characters. It consists of seven LED segments arranged in a specific pattern, with each segment representing one of the digits from 0 to 9. In this experiment, a common cathode 7-segment display is used, meaning all cathodes are connected to ground (GND), and the individual segments are controlled by supplying HIGH (5V) signals from the Arduino. Push buttons act as input devices, allowing users to interact with the system. When pressed, the increment button increases the displayed number, while the reset button clears the display to zero.

This experiment involves key concepts in digital electronics and microcontroller programming, including:

1. Binary and Digital Logic: The Arduino processes binary signals (HIGH/LOW) to control the 7-segment display.
2. Microcontroller Programming: The Arduino uses digitalRead() to detect button inputs and digitalWrite() to control display segments.
3. State Machines & Counters: The system maintains a counter variable that updates based on user input, similar to real-world counters in electronic devices.

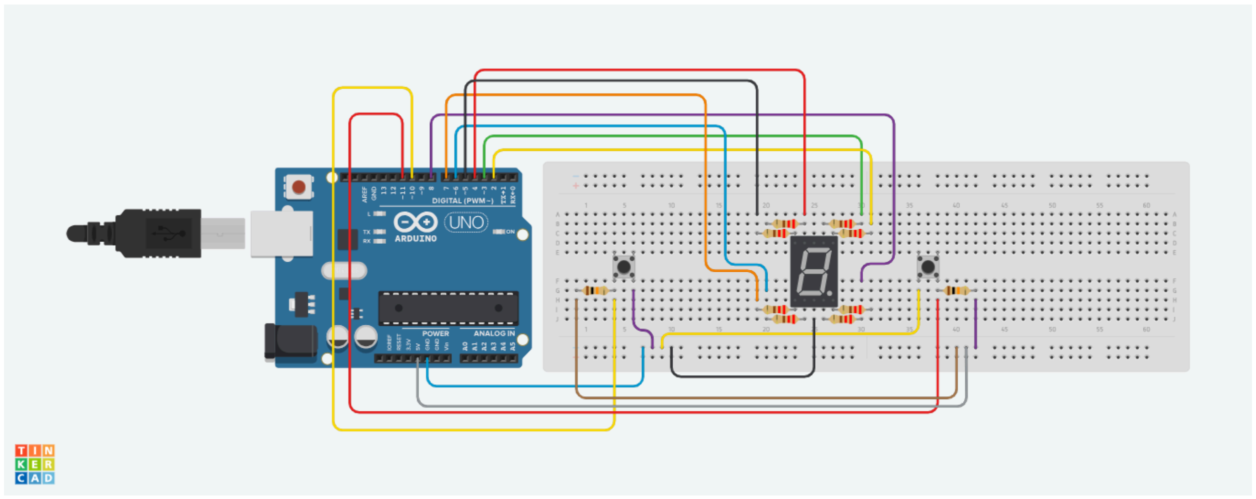
It is expected that the Arduino Uno will successfully control the 7-segment display, with numerical values increasing sequentially when the increment button is pressed and resetting to zero when the reset button is pressed. Besides, it should display correct digit representation on the display based on button presses, and proper debouncing, preventing multiple unintended increments.

2.0 MATERIALS AND EQUIPMENTS

- Arduino Uno board
- Common cathode 7-segment display
- 10k-ohm resistor (2)
- 220-ohm resistor (7)
- push buttons (2)
- breadboard
- jumpers

3.0 EXPERIMENTAL SETUP

- Each pin on common cathode 7-segment display are connected separately to digital pins between D0 until D6 on Arduino Uno board.
- 220-ohm resistors are connected between Arduino Uno board and common cathode 7-segment display to limit the current.
- One leg of each Push buttons are connected to different digital pins on Arduino Uno board, one connected to D11 as an increment button and another one is connected to D10 as a reset button.
- Another leg of the push buttons are connected to GND.
- Use a 10k-ohm pull up resistor for each push button. Connect one leg to digital pin and another one leg to 5V output on the Arduino Uno.



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 count button displayed the following number on the 7-segment display.
4. Press again and record the outcome until 9.
5. To see the outcome of rest button was pressed and it turn to 0;

4.1 Circuit Assembly

- The 7-segment display is connected to the Arduino Uno with individual segment pins (D0 - D6) assigned to specific digital output pins.
- Two push buttons are used, one for incrementing the displayed number and another for resetting it.
- Internal pull-up resistors are enabled for the buttons to ensure stable readings.

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

4.3 Code used:

```
// Define the pins for each segment
const int segmentA = 3; // D3
const int segmentB = 2; // D2
const int segmentC = 8; // D8
const int segmentD = 6; // D6
const int segmentE = 7; // D7
const int segmentF = 4; // D4
const int segmentG = 5; // D5

// Define button pins
const int buttonIncrement = 11; // Button to increment the count
const int buttonReset = 10;     // Button to reset the count

int count = 0; // Current count

// Button states
bool lastButtonStateIncrement = HIGH;
bool lastButtonStateReset = HIGH;
```

```
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 pins as INPUTs with pull-up resistors
  pinMode(buttonIncrement, INPUT_PULLUP);
  pinMode(buttonReset, INPUT_PULLUP);
}

void loop() {
  bool currentButtonStateIncrement = digitalRead(buttonIncrement);
  bool currentButtonStateReset = digitalRead(buttonReset);

  // Detect button press (LOW means pressed due to pull-up)
  if (currentButtonStateIncrement == LOW && lastButtonStateIncrement
== HIGH) {
    count++; // Increment count
    if (count > 9) {
      count = 0; // Wrap around if count exceeds 9
    }
    delay(200); // Debounce delay
  }

  if (currentButtonStateReset == LOW && lastButtonStateReset ==
HIGH) {
    count = 0; // Reset count to 0
    delay(200); // Debounce delay
  }

  // Update last button state
  lastButtonStateIncrement = currentButtonStateIncrement;
  lastButtonStateReset = currentButtonStateReset;

  // Display the current count on the 7-segment display
  displayNumber(count);
}

// Function to display a number on the 7-segment display
void displayNumber(int number) {
  // Reset all segments first
  digitalWrite(segmentA, LOW);
  digitalWrite(segmentB, LOW);
  digitalWrite(segmentC, LOW);
  digitalWrite(segmentD, LOW);
  digitalWrite(segmentE, LOW);
  digitalWrite(segmentF, LOW);
  digitalWrite(segmentG, LOW);

  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(segmentD, HIGH);
      digitalWrite(segmentE, HIGH);
      digitalWrite(segmentG, 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;
        }
    }
```

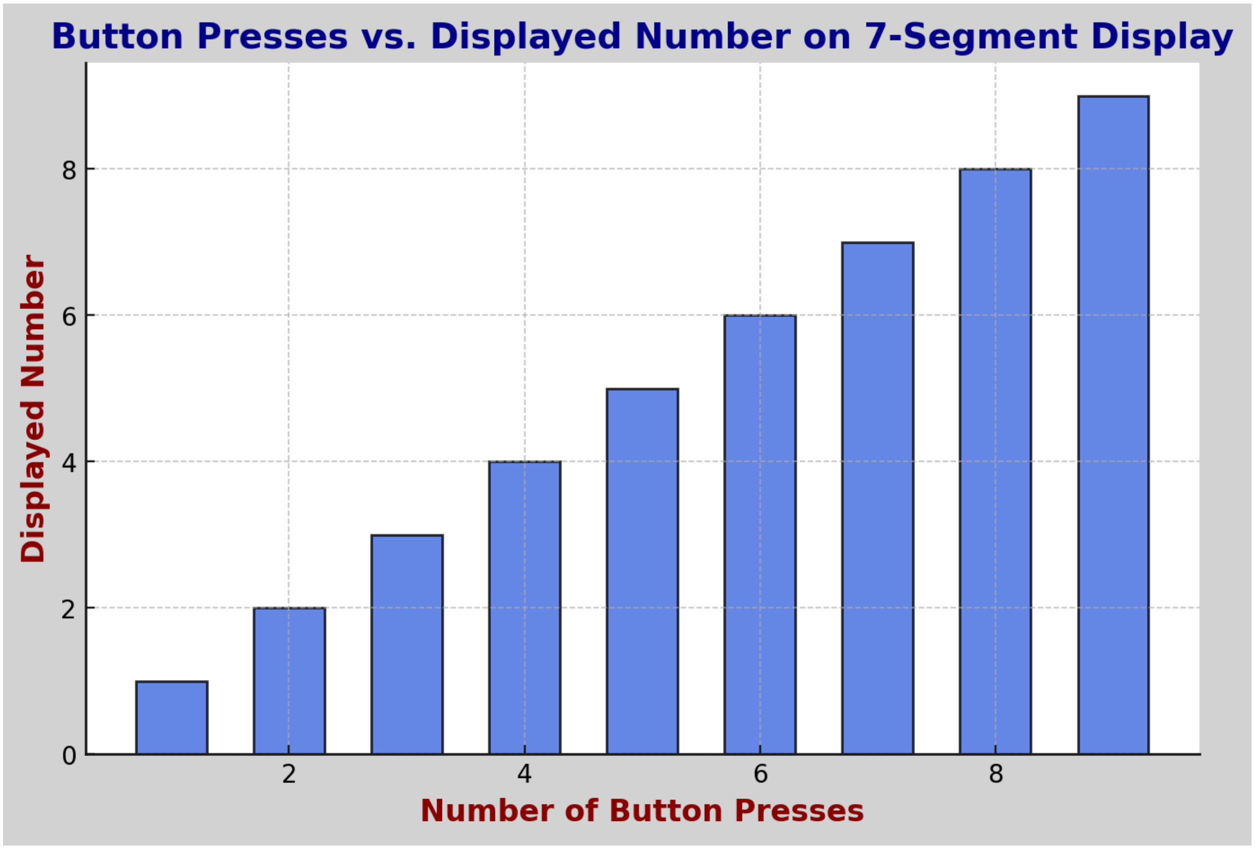
4.4 Control algorithm

1. Defining pins
    a. Initialize the LEDs light up on the 7-segment display by define as Segment A - G Example, `const int segmentA = 3; // D3`
    b. Button Increments (D11) and Button Rest (D10) are connected to digital pins.
2. Global variables
    a. `int count` on to stores the number shown on the 7-segment display.
    b. `bool lastButtonStateIncrement bool` and `lastButtonStateReset` is used for edge detection, which is detecting a new button press
3. Setup Function
    a. 7- segment display Pins to configured as `OUTPUT`
    b. Buttons to configured as `INPUT_PULLUP`
    *INPUT_PULLUP makes the pin HIGH by default (button upressed and goes LOW when pressed).
4. Main loop :
    a. `bool currentButtonStateIncrement digitalRead(buttonIncrement)` and `bool currentButtonStateReset = digitalRead(buttonReset)` is to read current state of both buttons.
    b. Increment button logic
        i. To check button press and the count will increase, it will debounce a delay for 200ms for prevent false tregrring.
        ii. It keeps increasing until 9 and turn back to 0
    c. Reset button logic
        i. If the button rest is pressed, the count value is set to 0.
    d. Update button state
        i. Updates previous button states for edge detection
    e. Display number
        i. Calls the function to update the displayed count on the 7-segment display.
        `displayNumber(count);`
            - All segments turn off first before displaying a new number.
            - Each number is mapped to the corresponding segments
            - The necessary segments are set HIGH to form the shape of the number.

5.0 DATA COLLECTION

Observations:

Button press	**Displayed Number**
Increment	1
Increment	2
Increment	3
Increment	4
Increment	5
Increment	6
Increment	7
Increment	8
Increment	9
Increment	0 (Wraps around)
Reset	0



6.0 DATA ANALYSIS

- The number increases sequentially as expected when the button is pressed.
- Its showing number of time pressed will show the same number of display on the LCD 7-segment..
- A debounce delay of 200 ms prevents unintended multiple triggers.
- The reset button sets the display back to 0.

7.0 RESULT

The system successfully displayed numbers from 0 to 9 with button control. Incrementing and resetting functions operated correctly. The debounced method effectively eliminated false triggers.

How to interface an I2C LCD with Arduino?

Interfacing an 12C LCD display with Arduino are more easier than 7 segment display. It only require two I/O pins. Which is SDA (Serial Data) and SCL (Serial Clock).

Steps to interface an 12C LCD with arduino

1. Connect VCC into pin 5V.
2. Connect SDA and SCL into pins Analog (eg. A5 and A4)
3. Connect GND into GND.
4. On the Arduino IDE, install the library LiquidCrystal_12C. This library helps to control the LCD easily.
5. Construct the Arduino code.

Code to sketch Hellp Word on the 12C LCD

- Call the library (`#include <LiquidCrystal_I2C.h>`)
- Pass the the 12C address into LCD as well as the display dimension to the LiquidCrystakl_12C. (eg. `LiquidCrystal_I2C lcd(0x3F, 16, 2)`)
- Call function `lcd.init(),lcd.clear(),lcd.backlight()`

  init() : to initialize the interface to the LCD

  clear() : to clear the LCD screen and position the cursor in the upper-left corner

  blacklight() : to turn on LCD backlight
- Call the function setCursor and state the address position for specifies where the text will appear on the LCD. (eg, setCursor (2,0) which is column = 2 and row = 0)
- Call print function. (eg, lcd.print("Hello World"))

Explain the coding principle behind it compared with 7 segments display and matrix LED.

Feature	**12C LCD**	**7-Segment Display**	**LED Matrix**
Communication	12C (2 wires : SDA and SDL)	Parallel (multiple pins)	Multiplexed (row/col scanning)
Control Library	LiquidCrystal_12C.h	Manual control segments	Can put manual or upload the library depend on type of brand LED MAtrix board. Example library 1. MD_Parola.h 2. Arduino_LED_Matrix.h
Pins required	2	7+ (for single digit)	8+ (for 8x8 matrix)
Text Capability	Yes (custom text output)	No (only numbers 0-9)	High (LED multiplexing)

The easiest and most advisable is using 12C LCD because it allows easy modification of numbers and text without the need for manually setting HIGH or LOW states in the code. The built-in LiquidCrytsal_12C library simplifies programming by providing predefined functions, making it more user-friendly compared to the 7-segment display and LED matrix. Additionally, the circuit connection is much simpler, requiring only two communication wires (SDA and SCL), whereas the other displays require multiple connections and complex wiring setups.

8.0 DISCUSSION

Expected vs observed outcomes :

Function	**Expected Outcomes**	**Observed Outcomes**
Increment button	Number increases by 1	Works as expected
Reset button	Display resets to 0	Works as expected
Wrap around	9 into 0	Works as expected

Source of error:

- Mechanical button bounce

  Initially, it caused multiple unintended counts, which were resolved by debounce logic.
- Wiring issues

  Incorrect wiring of segment pins led to wrong digit display initially. Additionally, the short length of wire caused it to keep disconnecting which effecting the circuit.

Improvement

- Implement an interrupt-based button reading system for improved responsiveness.
- Replace with the long wire to make an effective structure of wiring.

9.0 CONCLUSION

From this experiment, we have successfully demonstrated the practical implementation of digital logic systems through the interfacing of a 7-segment display with an Arduino. The push buttons allowed us to control the number shown on the LCD display, which then helps us in understanding how digital signals interact with electronic components. This project simultaneously showed several important aspects of binary counting, circuit connections, and the microcontroller programming. One of the key lessons that we have discovered was the need for the pull-up resistors, which helped the push buttons work correctly without errors. This experiment also helps us in hands-on real life experience in identifying and fixing the wiring and the troubleshooting skills. By the end of this experiment, we successfully built a strong base for better advanced digital applications, such as multi-digit displays and automatic counters. We can conclude, this experiment strengthened both theoretical concepts that we learnt in class and practical skills, making it a very valuable learning experience.

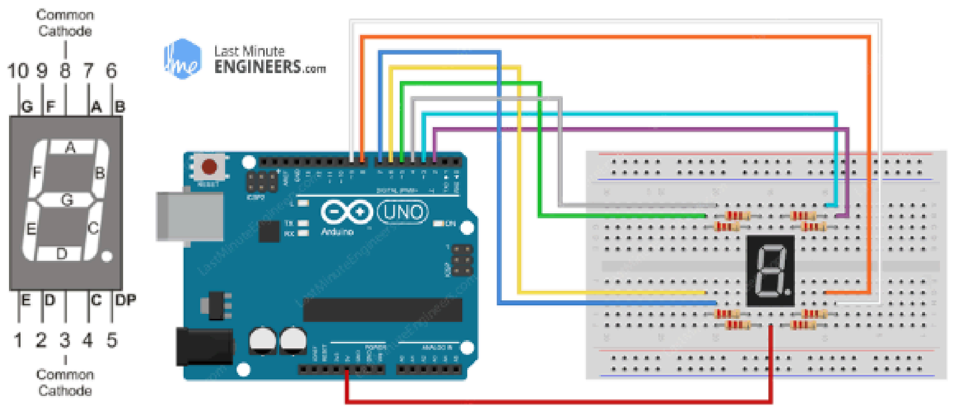
10.0 RECOMMENDATIONS

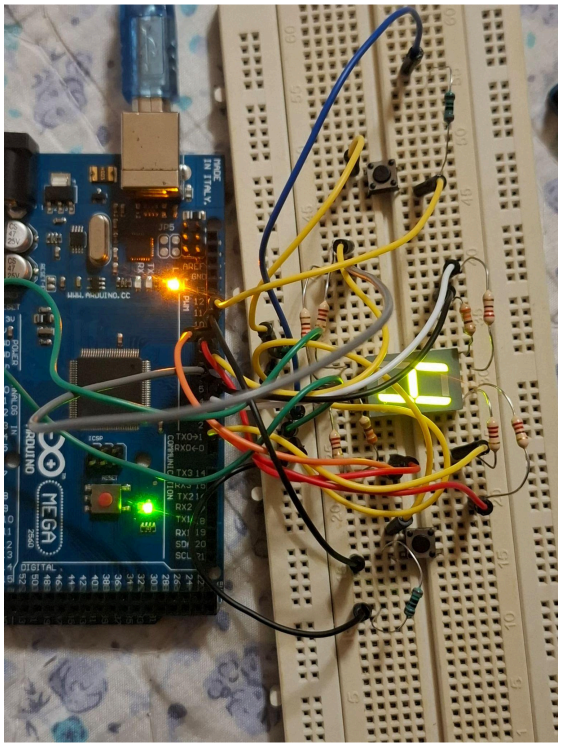
The improvement that we can do to improvise this experiment is to use a 12C LCD instead of a 7- segment display, which can help in displaying the number more efficiently. Other than that, by adding a debounce system system for push buttons will prevent accidental multiple presses , this will make the counting more accurate. We can also use a multiplexing method, this can allow controlling multiple 7-segment displays with fewer pins, which then will make it easier for us. Lastly, the simplest but one of the most important to improve is the length of wire we use. The shorter the length of the wire, the easier it is to make mistakes. We believe these improvements will make the system easier and more reliable to use while learning to understand the advanced digital logic concepts.

11.0 REFERENCES

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

12.0 APPENDICES





ACKNOWLEDGEMENTS

Special thanks to ZULKIFLI BIN ZAINAL ABIDIN & WAHJU SEDIONO for their guidance and support during this experiment.

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 qno 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: [illegible] Read [/]
Name: BUSHRA BINTI A. RAHIM Understand [/]
Matric Number: 2318514 Agree [/]

Signature: aufa Read [/]
Name: AUFA SIDQY BINTI MOHD SIDQY Understand [/]
Matric Number: 2310542 Agree [/]

Signature: [illegible] Read [/]
Name: NUR HAMIZAH BINTI MUHAMAD HAZMI Understand [/]
Matric Number:2319132 Agree [/]