REPORT 2 : PARALLEL,SERIAL AND USB INTERFACING WITH MICROCONTROLLER AND COMPUTER BASED SYSTEM(SENSORS AND ACTUATORS)

GROUP 4

MCTA 3203

SEMESTER 2 2024/2025

MECHATRONICS SYSTEM INTEGRATION

DATE OF SUBMISSION: 23 MARCH 2025

NO	NAME	MATRIC NUMBER
1.	IRDINA NABIHAH BINTI MOHD NAZRI	2214772
2.	IZZAH ZAHIRA BINTI NORAZLEE	2217696
3.	NOR MAISARAH BINTI ISMAIL	2213080

TABLE OF CONTENTS

INTRODUCTION

This project explores how Python and Arduino communicate with one another to manage electronic modules through serial communication. There are two aspects explored in this project:

First, we develop a system of communication where an Arduino transmits potentiometer values to a python program through a USB connection. The procedure demonstrates how analog input is read, processed, and utilized for monitoring and control applications.

The second part of the experiment is to control a servo motor using Python. By sending angle data from a Python program to the Arduino, we can make the servo motor turn to specific angles. This illustrates the practical use of servo motor control in automation and robotics.

In addition, this lab offers further understanding of serial communication protocols, the operation of analog sensors such as potentiometers, and the function of servo motors, familiarizing us with how these devices communicate with one another.

ABSTRACT

This experiment aims to show the interfacing of a potentiometer and servo motor with an Arduino Uno using serial communication through a Python script. The aim is to learn about reading analog input from a potentiometer and positioning a servo motor based on angle data sent from Python.

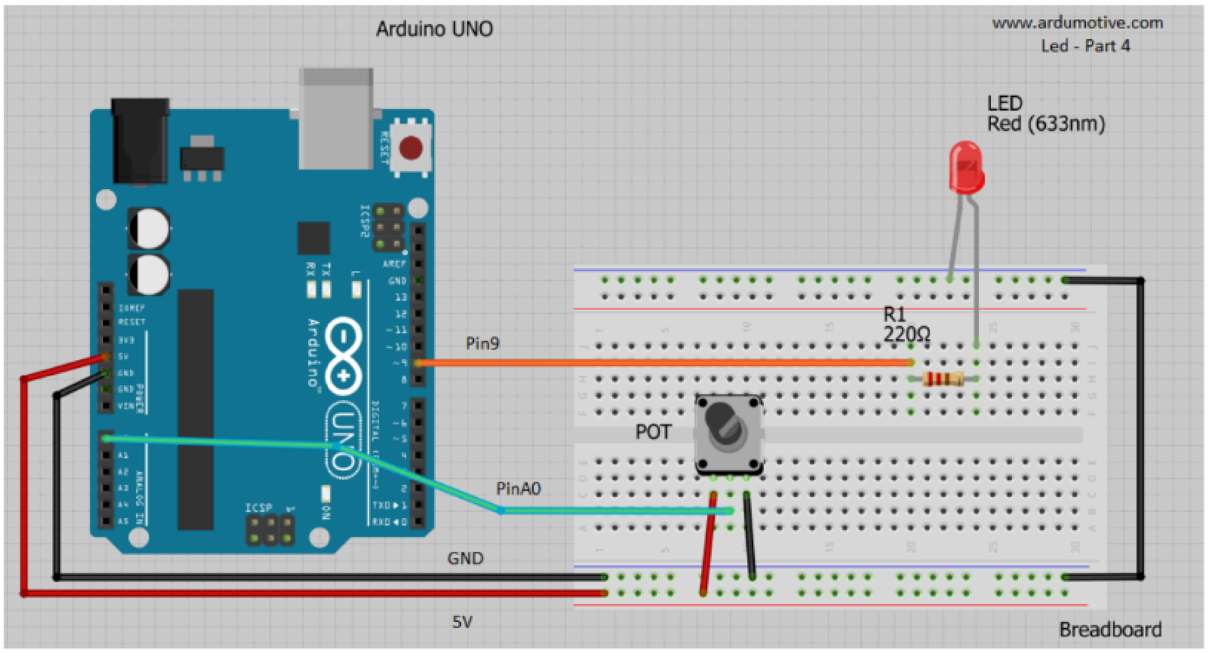
The process involves connecting the potentiometer to an analog input pin of the Arduino and configuring the servo motor on a PWM output pin. The Arduino takes a reading from the potentiometer and transmits it via serial communication, and the Python code translates this data to move the servo motor to specific angles. The experiment is able to effectively demonstrate that it is feasible to control servo motor movement based on readings from the potentiometer.

EXPERIMENT 3A: <u>Sending potentiometer readings from an Arduino to a Python script through a USB connection.</u>

MATERIALS AND EQUIPMENT

1. Arduino board
2. Potentiometer
3. Jumper wires
4. LED
5. 220 resistor
6. Breadboard

EXPERIMENTAL SETUP



1. Connect one leg of the potentiometer to 5V on the Arduino
2. Connect the other leg of the potentiometer to GND on the Arduino
3. Connect the middle leg (wiper) of the potentiometer to an analog input pin on the Arduino such as A0.
4. Connect the Arduino to your computer via a USB cable.

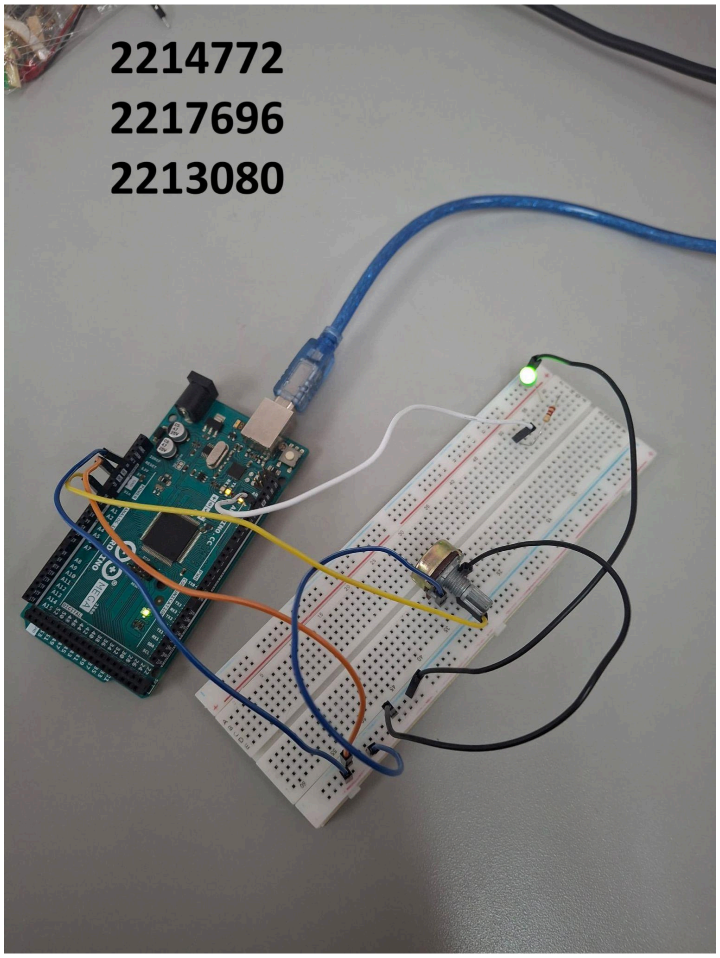
5. Power on the Arduino (upload the sketch to your Arduino using the Arduino IDE)
6. Run the Python script on your computer.
7. As you turn the potentiometer knob, you should see the potentiometer readings displayed in the Python terminal.

METHODOLOGY

1. Hardware Setup - Connecting the potentiometer to the Arduino
    a. Identify the three terminals of the potentiometer. The left terminal must be connected to the 5V pin of the Arduino to supply the necessary power.
    b. Connecting the right terminal to the GND pin on the Arduino closes the circuit.
    c. Middle terminal, or the wiper, gives varying voltage output. Plug it to an analog input pin on Arduino, such as A0, so the real-time voltage can be monitored while the potentiometer is turned.
2. Software Programming and Execution - Connecting the Arduino
    a. Connect Arduino Uno to the computer using USB cable. The connection delivers power and facilitates communication between the Arduino and computer.
    b. Uploading the Arduino Sketch
        i. Establish the pins connected to the potentiometer and any other hardware required in the experiment.
        ii. Utilize a loop that continually reads the value of the potentiometer at specified intervals.
        iii. Forward the potentiometer data acquired to the serial port to be processed.
    c. Running the Python Script
        i. Ensure all necessary libraries, such as pyserial, are installed and in good condition prior to executing the script.
        ii. Run the Python script in an effort to establish a connection with the Arduino.
        iii. When you rotate the potentiometer, the Python terminal will display the actual value of the potentiometer output.
3. Use of Readings

a. The collected potentiometer data can be utilized in various experiments, data logging, or control operations based on the needs of the project.

RESULTS



Link :
https://github.com/irdinazri/MSI_G4/blob/main/Week%203/3A/VIDEO_WEEK%203A.mp4

QUESTION

To present potentiometer readings graphically in your Python script, you may enhance your code by introducing the capability to generate and showcase a graph. This graphical visualization can deliver a more intuitive and informative perspective for data interpretation. Be sure to showcase the steps involved in your work (Hint: use matplotlib in your Python script).

To enhance the presentation of potentiometer readings graphically, the Python script was modified to include real-time data visualization by using the matplotlib library. The Arduino was configured to use the serial communication to transmit analogue readings from the potentiometer to the PC. This data was then received and plotted dynamically using a Python script.

The pyserial package was then used by the Python script to connect to the Arduino and continuously read incoming data. A real-time graph with the X-axis representing time (in seconds) and the Y-axis representing the potentiometer's analogue reading (ranging from 0 to 1023) was created using the matplotlib package. The script ensures a continuous and seamless visualisation by dynamically gathering and updating data points. To achieve it, an interactive plotting mode (plt.ion()) was enabled, allowing the graph to refresh at regular intervals. With each new value obtained, the plot was updated, and the readings were kept in lists. This method offered a simple and straightforward technique to track real-time potentiometer position changes. When the serial connection was cut off and the final plot was shown, the program was intended to run until it was manually stopped (Ctrl + C).

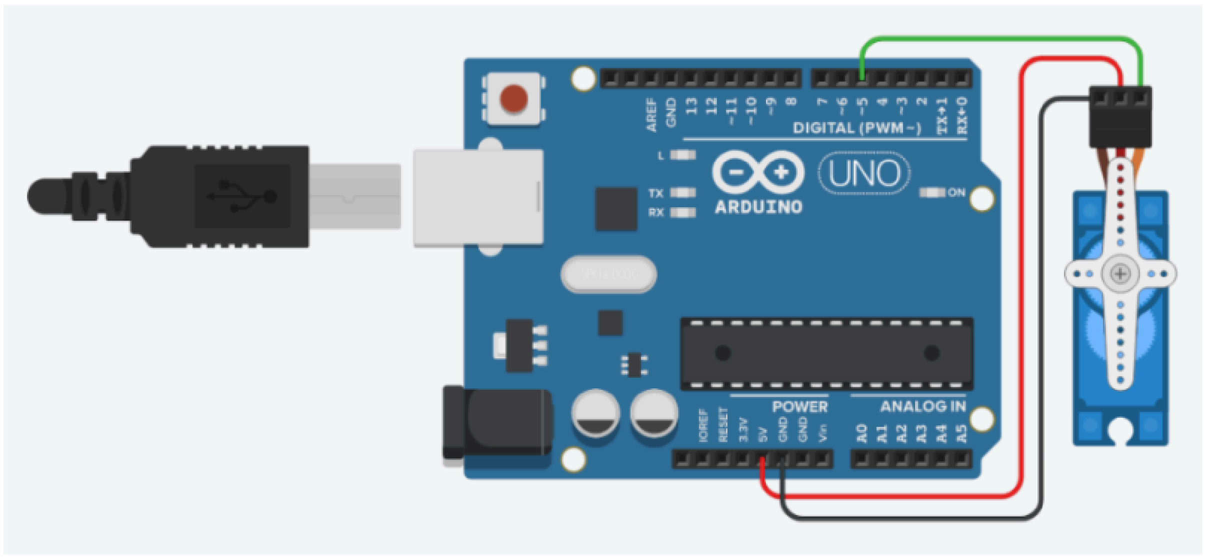
For real-time monitoring, analysis, and further uses like data logging or control systems, this graphical depiction of sensor data is quite handy. Moreover, the implementation showed how Arduino and Python may be successfully integrated for serial communication and visualisation.

EXPERIMENT 3B : <u>Transmitting angle data from a Python script to an Arduino, which then actuates a servo to move to the specified angle</u>

MATERIALS AND EQUIPMENT

1. Arduino Uno board
2. Servo motor
3. Jumper wires
4. Potentiometer (for manual angle input)
5. USB cable for Arduino
6. Computer with Arduino IDE and Python installed

EXPERIMENTAL SETUP

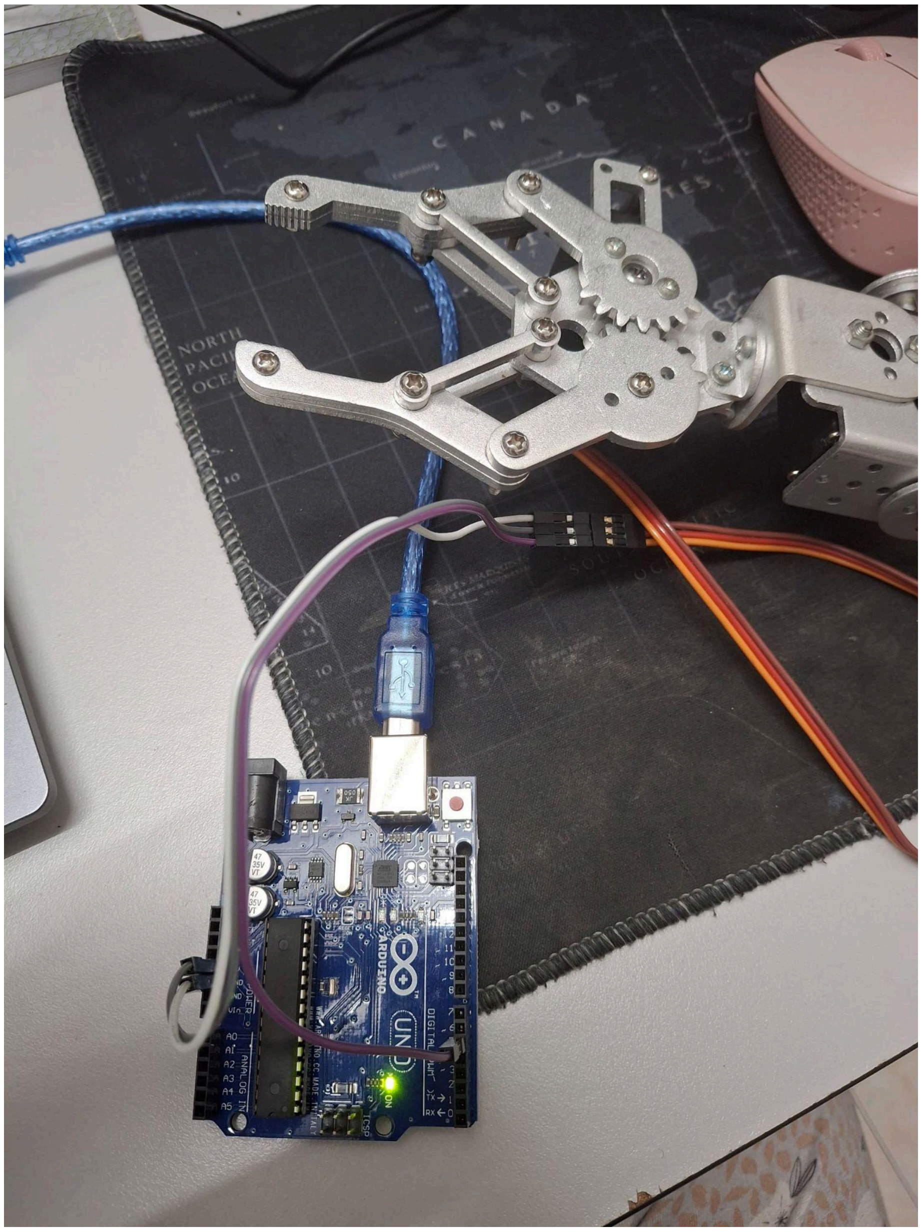


1. Connect the servo's signal wire to a PWM-capable pin on the Arduino (digital pin 5)
2. Power the servo using the Arduino's 5V and GND pins. Servos typically require a supply voltage of +5V. You can connect the servo's power wire to the 5V output on the Arduino board..
3. Connect the servo's ground wire to one of the ground (GND) pins on the Arduino

METHODOLOGY

1. Configuration of hardware
    a. Attach the servo's signal wire to an Arduino digital pin B.
    b. Next, attach the servo power to the 5V pin and the ground wire to the Arduino's GND pin.
    c. Connect the middle terminal of the potentiometer to pin A0, the analogue input pin, then connect the other terminal to the GND pin and connect one terminal to the 5V.
2. Software setup and execution
    a. Connection to the Arduino
        i. Connect the Arduino to the computer by using a USB cable. This connection not only powers the Arduino but also allows for data transmission between the Arduino and the computer.
        ii. Install the Servo library it may be easier to code and also control the servo.
    b. Upload the Arduino Sketch
        i. Initializing the pins connected to the potentiometer and servo that were used in the experiment
        ii. Read the angle data from the serial port to adjust the servo position as directed by the Python script.
        iii. Reads the potentiometer value and maps it to an angle (0–180 degrees) to control the servo position in real-time.
    c. Run the Python script
        i. Create a Python script that prompts the user to enter the desired angle, sends it to the Arduino through serial communication, then allow quitting the program.
    d. Output of the execution of code
        i. Run the Python script, enter an angle (0–180 degrees) to control the servo, and terminate the program by entering 'Q'.
        ii. If no angle is entered through Python, the potentiometer's position determines the servo's angle, enabling dynamic movement based on potentiometer adjustments.

RESULTS



Video Link :

1. https://github.com/irdinazri/MSI_G4/blob/main/Week%203/3B/Week3_3B_loop.mp4
2. https://github.com/irdinazri/MSI_G4/blob/main/Week%203/3B/Week3_3B_python.mp4

QUESTION

Enhance your Arduino and Python code to incorporate a potentiometer for real-time adjustments of the servo motor's angle. Ensure that, in the updated Arduino code, you have the ability to halt its execution by pressing a designated key on your computer's keyboard. Following the modification, restart the Python script to receive and display servo position data from the Arduino over the serial connection. While experimenting with the potentiometer, observe the corresponding changes in the servo motor's position.

To add an improvement to the Arduino and Python code by the addition of a potentiometer for dynamic angle adjustments of the servo motor in real time, we adapt the system to be under dynamic control while also being able to stop execution through keyboard input. The Arduino takes the analog input of the potentiometer, maps the values from the range it reads to the range of 0 to 180 degrees, and adjusts the servo position correspondingly. To make it easier to communicate with Python, the Arduino sends the current servo angle back over the serial connection repeatedly. Also, it listens for data from Python and, when it reads the character `'q'`, halts operation by entering into an infinite loop.

On the Python side, the program establishes a serial connection to the Arduino and reads the servo angle coming across repeatedly, displaying it in real time. The user can interrupt the program manually by typing `'q'`, which sends the same character to the Arduino, making it interrupt further execution. The setup enables easy and interactive control of the servo motor via the potentiometer with real-time feedback between Python and the Arduino. The outcome is an interactive system in which the user is able to both monitor and control servo motion dynamically while still retaining the capability of safely halting execution when necessary.

DISCUSSION

In this lab, we learned how to control a servo motor via serial communication between an Arduino and computer. This experiment had two part which the first part showed how the Arduino, using angle values sent from a Python script, we used it to instruct the servo motor to rotate at the desired angle. Second part we introduced a potentiometer to adjust the servo position in real time. Both methods highlighted different ways of interacting with hardware which through programmed commands and the other through direct, sensor based adjustments.

In the first phase, we set up serial communication to allow the user to control the servo's position within a range of 0 to 180 degrees by sending the angle values from a Python script to the Arduino. This technique demonstrated how external data inputs from a computer can control hardware components. Stable communication was maintained by ensuring the baud rates of Python script to the Arduino were balanced. While this method provided precise and programmable control, it required constant user input and was not ideal for applications needing real time responsiveness.

The second part is to enable real-time servo control without additional computer inputs, by adding potentiometer as an analog input. By mapping the potentiometer's reading to angle values, the servo's position could be adjusted continuously based on changes in the potentiometer's position. This method provided responsive, manual control and made it useful for applications like user operated robotic systems. However, while it allowed for real time adjustments, the potentiometer lacked the programmability of Python which made it better suited for manual control rather than automated tasks requiring external commands.

Overall, this experiment highlighted the trade off between automated and real time manual control showing how different input methods can be used depending on the needs of a given application.

CONCLUSION

Two experiments were carried out in this lab to investigate the fundamentals of serial communication between an Arduino and a Python computer. In the first experiment, the potentiometer readings were obtained from an Arduino and sent to a Python script using a USB serial connection. After processing the incoming data, the Python script created a real-time graph of the potentiometer values using the matplotlib module. This visualisation showed how sensor data can be effectively communicated, processed, and shown via serial connection, and it gave users an easy method to see how the potentiometer's position changed over time. The principles of interpreting sensor input and connecting microcontrollers to external software were strengthened by this experiment.The second experiment extended the application of serial communication by integrating a servo motor that was controlled using a potentiometer. In this setup, the potentiometer readings were mapped to an angle range of 0 to 180 degrees, and the Arduino adjusted the servo's position accordingly. Then, this experiment showcased on how Python can be used not only for sensor data acquisition but also for real-time actuator control, an essential concept in robotics and automation.

In mechatronic systems, where computers and microcontrollers collaborate to carry out intricate tasks, serial communication is crucial. These experiments demonstrated the point. While matplotlib made data visualisation easier, the pyserial package was essential in establishing communication. In many technical applications, such as robotics, industrial automation, and Internet of Things (IoT) systems, the capacity to read sensor values, transmit control signals, and process data in real time is essential. The experiments reaffirmed the usefulness of serial communication in contemporary embedded systems by effectively establishing real-time hardware-software interaction. All things considered, these activities gave participants a practical grasp of how microcontrollers and computers may work together to process sensor inputs and dynamically control actuators. More complex projects requiring data logging, automated control systems, and remote monitoring apps can benefit from the abilities acquired in this lab. More complex mechatronic and embedded system designs can be created by engineers using serial connection to create interactive, effective systems that smoothly combine hardware and software.

RECOMMENDATION

To increase functionality and efficiency, a number of enhancements and alterations can be made for further iterations of similar studies. To stabilise the potentiometer readings, one suggestion is to use real-time data smoothing methods, including using a moving average filter. As a result, sensor data would be represented more accurately and with fewer variations. Additionally, rather than depending just on terminal input, users could control the servo motor and visualise data in a more dynamic way by integrating a graphical user interface (GUI) using Tkinter or PyQt. The use of wireless communication protocols, like Bluetooth (HC-05 module) or Wi-Fi (ESP8266/ESP32), to do away with the necessity for a USB connection and enable remote system control and monitoring is another possible enhancement. Implementing multi-threading in Python could also enhance performance by enabling simultaneous data acquisition, processing, and visualization without delays or interference.

Besides that, the practical uses of serial communication in embedded systems and mechatronics were made clear to the students by these demonstrations. The significance of synchronising the baud rate between the Python script and Arduino for seamless data transfer was one of the most important lessons discovered. The necessity of appropriate port management and debugging approaches was further highlighted by the students' observations that incorrect handling of serial ports, such as trying to access the port while it is already in use, might result in errors or communication failures. A key idea in automation and control systems, mapping sensor inputs to actuator outputs, was also reaffirmed by the trials. Furthermore, understanding how to interpret, process, and visualize real-time data is a crucial skill for future engineers working in fields such as robotics, IoT, and also industrial automation.

In a nutshell, through the integration of these advancements and the application of the acquired knowledge, students can create increasingly sophisticated and effective embedded systems and acquire a more profound comprehension of hardware technology. Additionally, these trials established the groundwork for more intricate initiatives in which the development of intelligent and autonomous systems depends heavily on real-time data processing and accurate actuator control.

ACKNOWLEDGEMENTS

A special thanks goes out to Dr. Wahju Sediono and Dr. Zulkifli Bin Zainal Abidin, my teaching assistant, and my peers for their invaluable help and support in finishing this report. Their advice, feedback, and experience have greatly influenced the level of quality and understanding of this work. Their time, patience, and commitment to supporting my academic success are greatly appreciated.

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 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 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: [illegible]	Read ✓
Name: IRDINA NABIHAH BINTI MOHD NAZRI	Understand ✓
Matric Number: 2214772	Agree ✓

Signature: [illegible]	Read ✓
Name: IZZAH ZAHIRA BINTI NORAZLEE	Understand ✓
Matric Number: 2217696	Agree ✓

Signature: [illegible]	Read ☑
Name: NOR MAISARAH BINTI ISMAIL	Understand ☑
Matric Number: 2213080	Agree ☑