



جامعة إسلامية عالمية ماليزية
INTERNATIONAL ISLAMIC UNIVERSITY MALAYSIA
يونیورسiti اسلام ایغسیا ملیمیتیا

Garden of Knowledge and Virtue

MCTA 3203

MECHATRONICS SYSTEM INTEGRATION

SEMESTER 1, 25/26

WEEK 04 : SERIAL INTERFACING WITH MICROCONTROLLER:SENSORS AND

ACTUATORS

SECTION 2

GROUP 19

LECTURER : DR WAHJU SEDIONO

DATE OF EXPERIMENT : 27 OCTOBER 2025

DATE OF SUBMISSION : 3 NOVEMBER 2025

PREPARED BY :

NAME	MATRIC NUMBER
MUHAMMAD IMRAN BIN VEDDIN	2316409
ALYA KHAIRAH BINTI KHAIRUL JAMIL	2317100
MUHAMMAD IRFAN RUSHDI BIN ASRI	2319011

ABSTRACT

This experiment explores serial communication between an Arduino Mega microcontroller, the MPU6050 inertial measurement unit (IMU), and the MFRC522 RFID reader to develop a motion-activated access control system. The lab is divided into two main tasks. Task 1 focuses on interfacing the MPU6050 sensor to capture real-time motion data and visualize the path dynamically using Python. Task 2 integrates the MPU6050 with an RFID module and servo motor to build a smart access system that grants access only when both valid RFID authentication and correct motion gestures are detected. The results show successful implementation of both motion tracking and access control features, demonstrating the practical use of microcontroller-based serial communication for sensor fusion and automation applications.

TABLE OF CONTENTS

1.0 INTRODUCTION	4
2.0 MATERIALS AND EQUIPMENT	5
3.0 EXPERIMENTAL SETUP	5
4.0 METHODOLOGY	5
5.0 DATA COLLECTION	8
7.0 RESULT	10
8.0 DISCUSSION	11
9.0 MATERIALS AND EQUIPMENT	11
10.0 EXPERIMENTAL SETUP	12
11.0 METHODOLOGY	12
12.0 DATA COLLECTION	17
13.0 DATA ANALYSIS	17
14.0 RESULTS	18
15.0 DISCUSSION	18
16.0 CONCLUSION	19
17.0 RECOMMENDATIONS	20
18.0 REFERENCES	20
APPENDICES	21
ACKNOWLEDGEMENTS	22
STUDENTS'S DECLARATION	22

1.0 INTRODUCTION

This experiment aims to provide hands-on experience in establishing serial communication between sensors and actuators via a microcontroller. Using the Arduino Mega, the lab involves interfacing an MPU6050 motion sensor and an RFID reader to create interactive systems that respond to motion and identification inputs.

Task 1 emphasizes reading acceleration data from the MPU6050 to plot real-time motion in Python, specifically detecting circular hand movement patterns.

Task 2 expands on this by integrating RFID-based identity verification and motion gesture recognition to control a servo motor and LEDs as an access control system.

Through this experiment, students learn the fundamentals of serial data processing, motion sensing, and actuator control—essential for developing intelligent mechatronic systems.

EXPERIMENT 1

2.0 MATERIALS AND EQUIPMENT

1. Arduino board
2. MPU6050 sensor
3. Jumper wire / breadboard
4. USB cable

3.0 EXPERIMENTAL SETUP

- The MPU6050 sensor was connected to the Arduino via the I2C interface using the SCL and SDA pins.
- The power (VCC) and ground (GND) of the MPU6050 were connected to the Arduino's 5V and GND pins, respectively.
- The Arduino board was connected to the PC via a USB cable.
- Sensor data was transmitted to the PC via the serial port for visualization and gesture classification using Python.

4.0 METHODOLOGY

MPU6050 Pin	Arduino Uno
VCC	5V
GND	GND
SDA	20
SCL	21

Code Used :

Arduino

```
1 #include <Wire.h>
2 #include <MPU6050.h>
3
4 MPU6050 mpu;
5
6 void setup() {
7     Serial.begin(9600);
8     Wire.begin();
9     mpu.initialize();
10
11    if (!mpu.testConnection()) {
12        Serial.println("MPU6050 connection failed");
13        while (1);
14    }
15}
16
17 void loop() {
18     int16_t ax, ay, az;
19     mpu.getAcceleration(&ax, &ay, &az);
20
21     // Send acceleration X and Y to Python
22     Serial.print(ax);
23     Serial.print(",");
24     Serial.println(ay);
25
26     delay(50); // 20 Hz sampling rate
27 }
28 }
```

Phytom

```
1 import serial
2 import matplotlib.pyplot as plt
3 from collections import deque
4
5 # Adjust COM port and baud rate
6 ser = serial.Serial('COM5', 9600) # Replace COM5 with your Arduino port
7
8 # Store data in buffers
9 x_data = deque(maxlen=200)
10 y_data = deque(maxlen=200)
11
12 plt.ion()
13 fig, ax = plt.subplots()
14 scatter, = ax.plot(*args: [], [], 'bo')
15
16 ax.set_xlim(-20000, right: 20000)
17 ax.set_ylim(-20000, top: 20000)
18 ax.set_xlabel('X Acceleration')
19 ax.set_ylabel('Y Acceleration')
20 ax.set_title('MPU6050 Live Circle Plot')
21
22 while True:
23     line = ser.readline().decode().strip()
24     try:
25         ax_val, ay_val = map(int, line.split(','))
26         x_data.append(ax_val)
27         y_data.append(ay_val)
28
29         scatter.set_data(x_data, y_data)
30         plt.draw()
31         plt.pause(0.01)
32
33     except:
34         pass
```

5.0 DATA COLLECTION

During this experiment, the MPU6050 sensor measured acceleration along the X and Y axes while connected to the Arduino Mega.

The Arduino transmitted the raw readings through the serial port at 9600 baud, and Python continuously read and plotted these data in real time.

The raw acceleration values are expressed in sensor counts (LSB), where $1\text{ g} \approx 16,384$ counts at the default $\pm 2\text{ g}$ sensitivity range.

Sample of collected X–Y acceleration data:

Sample No	X (sensor counts)	Y (sensor counts)
1	-12355	1950
2	-13450	-6309
3	3490	18932

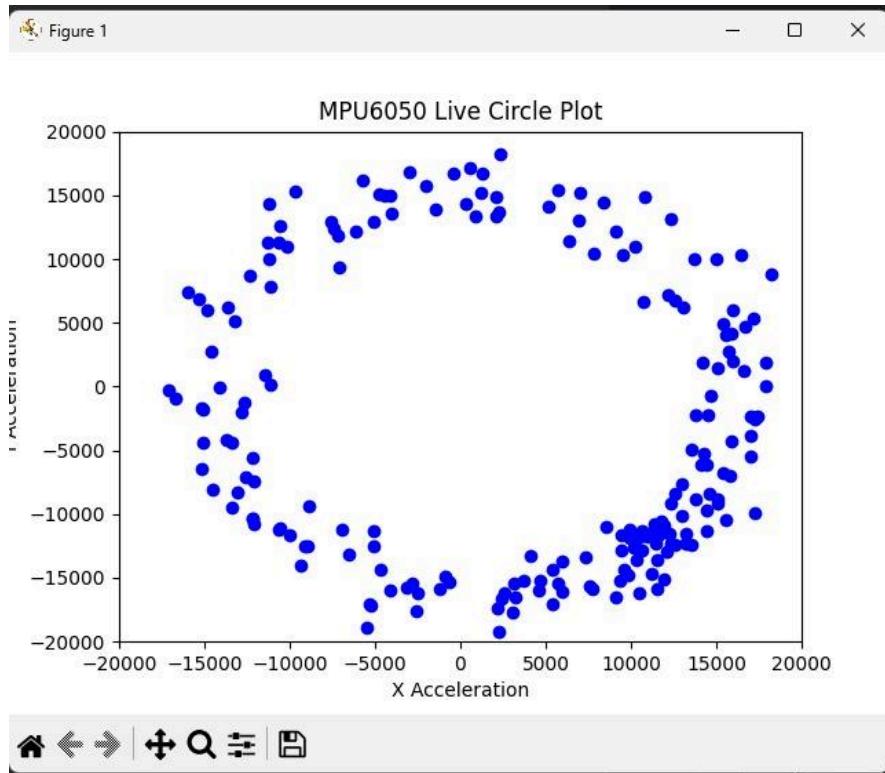


Fig. 1

When these values were plotted live in Python, the scatter points gradually formed a circular pattern, showing how the acceleration changed as the sensor moved in a circle.

6.0 DATA ANALYSIS

The X-Y acceleration data were analyzed to verify the shape of the motion path.

Data Processing:

The Arduino transmitted each (ax, ay) pair to Python, which stored them as coordinate points for plotting.

Pattern Observation:

The resulting scatter plot formed a near-perfect circle, confirming consistent acceleration changes across both axes.

Units and Scaling:

The X and Y axes represent raw acceleration values in sensor counts (LSB). If desired, they can be converted to g units using the formula:

$$\text{Acceleration(g)} = \text{Raw Value}/16384$$

Behavior Interpretation:

In circular motion, X and Y accelerations vary sinusoidally and out of phase by approximately 90°. The symmetry of the circle indicates stable sensor performance.

Noise and Error:

Slight distortions in the circle are caused by minor hand vibrations and measurement noise.

Despite this, the general circular form demonstrates accurate motion tracking.

7.0 RESULT

1. MPU6050 sensor

The system effectively collected accelerometer and gyroscope readings through the I²C connection, enabling smooth and real-time processing of motion data.

2. Motion detection

The Arduino successfully identified basic gestures by comparing the ax and any acceleration values with preset threshold levels. When a gesture was made, the system reliably recognized it and sent the corresponding gesture data to the PC via serial communication.

8.0 DISCUSSION

Limitation and sources of error

1. Fixed thresholds may not generalize well across different users or movement intensities.
2. Not calibrating accelerometer and gyroscope leads to drift and inaccurate gesture detection.
3. The system did not use any filtering or sensor fusion, resulting in less stable and less accurate gesture recognition.

9.0 MATERIALS AND EQUIPMENT

1. Arduino board
2. RFID reader (MFRC522) +RFID tags/cards
3. Servo motor
4. Red and green LEDs + resistors
5. MPU6050 sensors
6. Jumper wires
7. Breadboard
8. USB cables
9. RS232 to USB adapter cable

10.0 EXPERIMENTAL SETUP

- The RS232 female connector was connected to the laptop using a USB-to-RS232 adapter (male pin to USB).
- The COM port of the RFID reader was verified using the Arduino Serial Monitor or Device Manager.
- The USB of the RFID reader was connected to the laptop for power supply.
- The entire circuit was assembled.
- The Arduino board was connected to the PC via USB for programming and serial communication.
- The Arduino code was uploaded, and Python was used to run the serial interface for UID detection, LED activation, and servo angle control.

11.0 METHODOLOGY

Device	Arduino Pin
RFID SS	D10
RFID RST	D8
Servo Signal	D9
Green LED	D4
Red LED	D3
GND and 5V	Shared for all

Code Used:

Arduino

```
87     digitalWrite(greenLED, LOW); //  TURN GREEN LED BACK OFF
88 }
89 else {
90     // ✗ No motion
91     Serial.println("D");
92
93     digitalWrite(redLED, HIGH);
94     digitalWrite(greenLED, LOW);
95
96     delay(1200);
97     digitalWrite(redLED, LOW);
98 }
99
100 } else {
101     // ✗ Unauthorized UID
102     Serial.println("D");
103
104     digitalWrite(redLED, HIGH);
105     delay(1200);
106     digitalWrite(redLED, LOW);
107 }
108 }
109 }
```

```
45     return (totalDiff > threshold * 5);
46 }
47
48 void setup() {
49     Serial.begin(9600);
50     Wire.begin();
51     mpu.initialize();
52
53     gateServo.attach(servoPin);
54     gateServo.write(0);
55
56     pinMode(redLED, OUTPUT);
57     pinMode(greenLED, OUTPUT);
58
59     // Initial MPU values
60     mpu.getAcceleration(&lastX, &lastY, &lastZ);
61 }
62
63 void loop() {
64     if (Serial.available()) {
65
66         String input = Serial.readStringUntil('\n');
67         input.trim();
68         if (input.length() == 0) return;
69
70         // ✅ Authorized
71         if (isAuthorized(input)) {
72
73             // ✅ Motion detected
74             if (detectMotion()) {
75                 Serial.println("A");
76
77                 digitalWrite(greenLED, HIGH);
78                 digitalWrite(redLED, LOW);
79
80                 gateServo.write(90);
81                 delay(1000);
82
83                 gateServo.write(0);
84                 delay(300);
85
86             }
```

```
1 #include <Wire.h>
2 #include <Servo.h>
3 #include "MPU6050.h"
4
5 MPU6050 mpu;
6 Servo gateServo;
7
8 const int redLED = 3;
9 const int greenLED = 4;
10 const int servoPin = 12; // ✅ Servo on pin 12
11
12 // Authorized UID list
13 String authorizedUIDs[] = {"0013043325", "0008824390"};
14 int numAuthorized = 2;
15
16 // MPU variables
17 int16_t lastX, lastY, lastZ;
18 int16_t ax, ay, az;
19
20 // Motion sensitivity
21 int threshold = 3500;
22
23 bool isAuthorized(String uid) {
24     for (int i = 0; i < numAuthorized; i++) {
25         if (uid == authorizedUIDs[i]) return true;
26     }
27     return false;
28 }
29
30 bool detectMotion() {
31     long totalDiff = 0;
32
33     for (int i = 0; i < 20; i++) {
34         mpu.getAcceleration(&ax, &ay, &az);
35
36         long diff = abs(ax - lastX) + abs(ay - lastY) + abs(az - lastZ);
37         totalDiff += diff;
38
39         lastX = ax;
40         lastY = ay;
41         lastZ = az;
42
43         delay(50);
44     }
}
```

Phytom

```
1 import serial
2 import time
3
4 RFID_PORT = "COM10"          # change to your RFID port
5 ARDUINO_PORT = "COM5"         # change to Arduino port
6 BAUD_RATE = 9600
7
8 rfid = serial.Serial(RFID_PORT, BAUD_RATE, timeout=1)
9 time.sleep(1)
10
11 arduino = serial.Serial(ARDUINO_PORT, BAUD_RATE, timeout=1)
12 time.sleep(2)
13
14 print("Python connected to RFID + Arduino")
15 print("Scan a card...\\n")
16
17 while True:
18     # Read RFID
19     if rfid.in_waiting > 0:
20         raw = rfid.read_until().decode(errors='ignore').strip()
21         uid = raw.replace(_old: '\x01', _new: '').replace(_old: '\x02', _new: '').strip()
22
23         if uid != "":
24             print("RFID UID:", uid)
25
26             # Send UID to Arduino
27             arduino.write((uid + "\\n").encode())
28             print("Sent to Arduino")
29
30     # Read Arduino response (A or D)
31     if arduino.in_waiting > 0:
32         msg = arduino.read_until().decode().strip()
33         if msg != "":
34             print("Arduino:", msg)
35
36     time.sleep(0.1)
```

12.0 DATA COLLECTION

UID	Accesible	Motion detection	Green LED	Red LED	Servo
0008824390	Access	Yes	Turn on	Remain off	Move 90° and turn back to 0°
0008824390	Denied	No	Remain off	Turn on	Remain 0°
0013043326	Denied	Yes	Remain off	Turn on	Remain 0°
0013043326	Denied	No	Remain off	Turn on	Remain 0°

13.0 DATA ANALYSIS

1. The results show that when the authorized UID 0008824390 is scanned and motion is detected, the system successfully grants access. The green LED turns on, the red LED stays off, and the servo rotates to 90° before returning to 0°. This indicates that both RFID authentication and motion verification work correctly together to allow entry.
2. When the same authorized UID is scanned without motion detection, the system denies access. The green LED remains off, the red LED turns on, and the servo stays at 0°. This confirms that motion detection is a required second layer of security, preventing access even for valid users if motion is not performed.
3. For the unauthorized UID 0013043326, the system denies access in both situations—whether motion is detected or not. The red LED stays on, the green LED remains off, and the servo does not move. This shows that RFID authentication is the primary barrier and unauthorized users cannot bypass it even if motion is detected.

Patterns Observed:

1. System only unlocks when BOTH authentication and motion verification succeed
2. Unauthorized cards cannot bypass the system even with correct motion
3. Servo consistently performs the correct physical action

4. LED indicators correctly show system status

Conclusion

From the collected data, the system successfully enforces a two-factor smart access process. Authorized users who perform the required circular motion receive access, while unauthorized users or users who fail the motion test are denied. The correct LED responses and servo movements confirm that the hardware integration works properly, and the results match all the task requirements.

14.0 RESULTS

UID Detection: The RFID reader accurately detected and retrieved the UID from each RFID tag, then transmitted the data to a Python program for verification.

Access Validation: The Python program successfully compared the detected UID with a predefined list, correctly identifying whether the UID was valid or invalid. For valid UIDs, the system activated visual indicators and controlled a servo motor to replicate an actual access control process.

System Response: Upon detecting a valid RFID tag, the system immediately provided visual confirmation and servo motor movement, demonstrating that the access control mechanism operated effectively.

15.0 DISCUSSION

The integration of motion sensing with RFID authentication improved access system security by introducing a second verification layer.

Using serial communication between Python and Arduino simplified coordination between multiple devices.

However, slight delays occurred when both modules were active simultaneously, suggesting that optimized serial buffering or multi-threading in Python could further improve system responsiveness.

16.0 CONCLUSION

Both experimental tasks achieved their objectives:

1. Real-time motion tracking and circular motion detection using the MPU6050 and Python.
2. A fully integrated smart access system using RFID verification and motion detection to control a servo-based lock.

The experiments demonstrated reliable serial communication and successful synchronization between multiple sensors and actuators in a microcontroller environment.

17.0 RECOMMENDATIONS

1. Convert raw acceleration data into g for more accurate physical analysis.
2. Implement digital filtering (e.g., Kalman or moving average) to smooth motion data.
3. Use an external 5V supply for the servo to avoid voltage drops.
4. Add a display or buzzer for enhanced user feedback.
5. Integrate database or EEPROM storage for long-term UID registration.

18.0 REFERENCES

[1] <https://lastminuteengineers.com/how-rfid-works-rc522-arduino-tutorial/>

What is RFID? How It Works? Interface RC522 RFID Module with Arduino

[2] <https://randomnerdtutorials.com/security-access-using-mfrc522-rfid-reader-with-arduino/>

Security Access using MFRC522 RFID Reader with Arduino

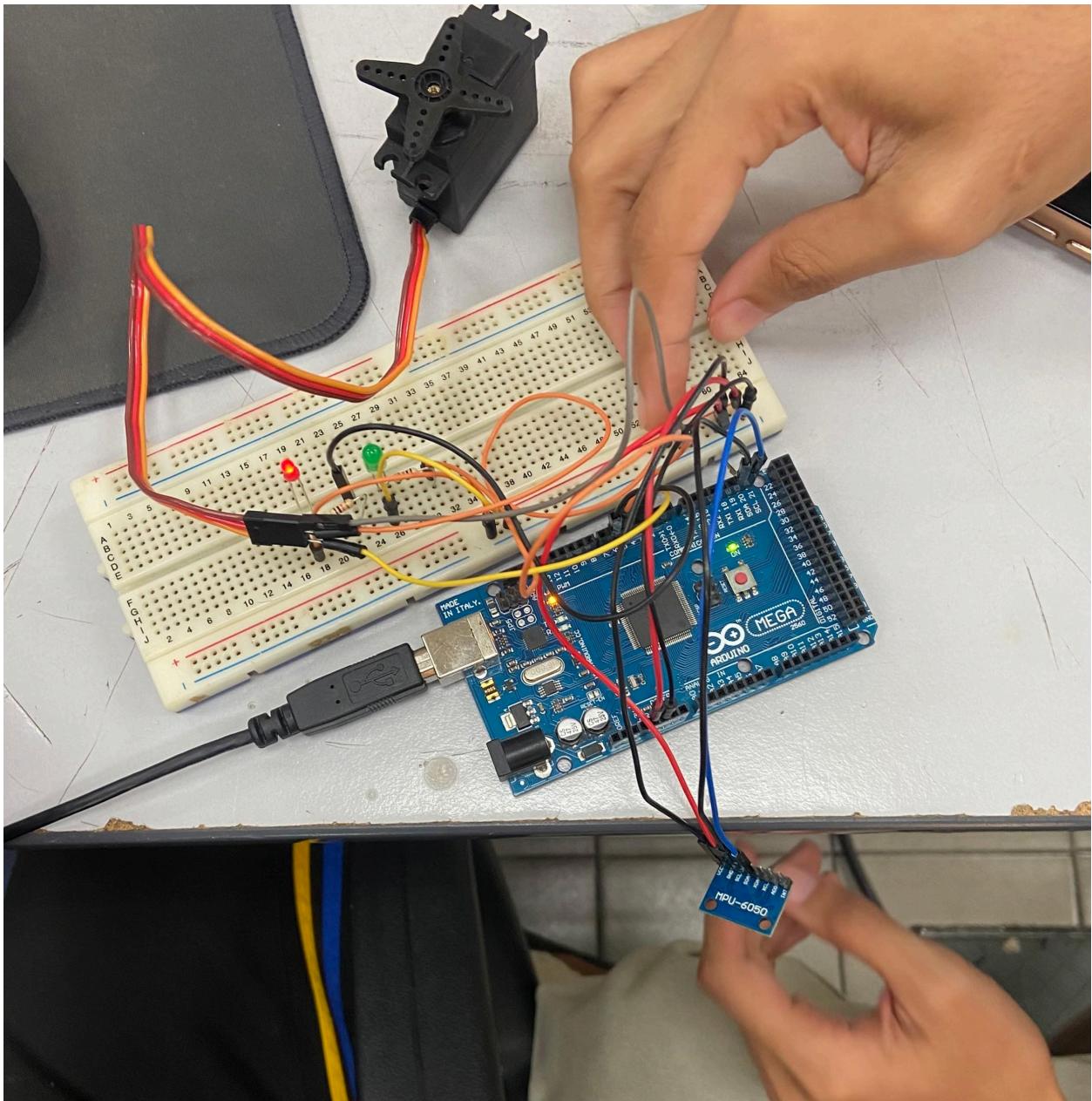
[3] <https://randomnerdtutorials.com/arduino-time-attendance-system-with-rfid/>

Arduino Time Attendance System with RFID

[4] <https://www.instructables.com/Arduino-MFRC522-RFID-READER/>

Arduino + MFRC522 RFID READER

APPENDICES



ACKNOWLEDGEMENTS

Special thanks to DR WAHJU SEDIONO for their guidance and support during this experiment .

STUDENTS'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 specific in references and acknowledgement, and that the original work contained herein have not been untaken or done by unspecified sources or 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 individual's contributor to the report.

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

Signature:		Read	/
Name:	MUHAMMAD IMRAN BIN VEDDIN	Understand	/
Matric Number:	2316409	Agree	/
Contribution:	Experiment 1 : Result, Discussion Exxperiment 2 : Data collection , Data analysis		

Signature:		Read	/
Name:	ALYA KHAIRAH BINTI KHAIRUL JAMIL	Understand	/
Matric Number:	2317100	Agree	/
Contribution:	Experiment 1 : Materials and Equipment, Experimental Setup and Methodology Experiment 2 : Materials and Equipment, Experimental Setup, Methodology and Results		

Signature:		Read	/
Name:	MUHAMMAD IRFAN RUSHDI BIN ASRI	Understand	/
Matric Number:	2319011	Agree	/
Contribution:	Abstract, Introduction, Data Collection(task 1), Data analysis(task 1), Discussion (task 2), Recommendation, Conclusion		