MECHATRONICS SYSTEM INTEGRATION (MCTA 3203)

SEMESTER 2 2024/2025

WEEK 4: SERIAL AND USB INTERFACING WITH MICROCONTROLLER AND COMPUTER-BASED SYSTEM

SECTION 1

GROUP 5

LECTURER: ZULKIFLI BIN ZAINAL ABIDIN & WAHJU SEDIONO

Date of Experiment: Monday, 24 March 2025

Date of Submission: Monday, 12 April 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 develop and integrate two systems, a hand gesture recognition system using an MPU6050 accelerometer-gyroscope sensor, and an RFID-based access control system enhanced with LED indicators and servo motor control. For the gesture system, predefined hand movements were performed, and motion data was collected and visualized using Python to observe and classify gestures based on x-y coordinate paths. In the second part, the RFID system was programmed to detect authorized and unauthorized UIDs using structured JSON data. A green LED and servo movement indicated access approval, while a red LED signaled rejection. Users could also input desired servo angles for flexibility. The findings showed successful classification of simple gestures and reliable UID validation with accurate visual feedback. The project demonstrated effective sensor integration, real-time data handling, and user interaction, laying the groundwork for future improvements using machine learning and advanced access control features.

TABLE OF CONTENTS

1.0 INTRODUCTION

This experiment involves the development of two integrated embedded systems:

a) A hand gesture recognition system using accelerometer and gyroscope data from the MPU6050 sensor.
b) An RFID-based access control system that uses JSON data management for UID validation and provides visual feedback using LEDs and servo motor actuation.

This project aims to demonstrate real-time sensor data handling, classification, and actuator control, enabling gesture-based input recognition and secured RFID access with user feedback mechanisms.

2.0 MATERIALS AND EQUIPMENT

2.a. hand gesture system component;-

- Arduino board
- MPU6050 sensor
- Computer with Arduino IDE and Python installed
- Connecting wires: Jumper wires or breadboard wires to establish the connections between the
- Arduino, MPU6050, and the power source.
- USB cable: A USB cable connects the Arduino board to your computer. This will be used for uploading the Arduino code and serial communication.

2.a. RFID system component;-

- Arduino board
- RFID card reader with USB connectivity
- RFID tags or cards that can be used for authentication
- Servo Motor: A standard servo motor to control the angle
- Jumper wires
- Breadboard
- LED red and green colors.
- USB cables to connect the Arduino board and the RFID reader to your computer.
- RS232 to USB adapter cable
- Computer with Arduino IDE and Python installed

3.0 EXPERIMENTAL SETUP

3.a. Hand Gesture System 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.

3.b. RFID System Setup:

- TThe 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 according to the schematic diagram shown in *Figure 1*.
- The Arduino board was connected to the PC via USB for programming and serial communication.
- A UID database was created and stored in a structured JSON file format.
- The Arduino code was uploaded, and Python was used to run the serial interface for UID detection, LED activation, and servo angle control.

4.0 METHODOLOGY

4.a. Hand Gesture System:

<u>1 Circuit Assembly</u>

1. MPU6050
    - The VCC pin is connected to the 5V.
    - The GND pin is connected to the GND pin of the Arduino.
    - The SDA pin is connected to the A4 pin of the Arduino.

- The SCL pin is connected to the A5 pin of the Arduino.

2 Programming Logic

PYTHON

1. Import necessary libraries:
    - serial for reading data from Arduino.
    - matplotlib.pyplot for plotting.
    - deque for storing a limited number of recent data points.

2. Establish a serial connection with the Arduino on COM6 at 9600 baud rate.

3. Initialize real-time plotting using matplotlib:
    - Create a plot with X and Y axes labeled for accelerometer data.
    - Use two deque objects to store and update the last 100 X and Y values.
    - Create a dynamic plot line.

4. Enter an infinite loop to continuously:
    - Read each line of serial data from Arduino.
    - Decode and strip the line for clean processing.

5. If the line contains a gesture detection message:
    - Extract the gesture name.
    - Print the detected gesture.

6. If the line contains accelerometer data:
    - Split the values and convert them to integers.
    - Append the values to x_data and y_data.
    - Update the plot with the new values.
    - Refresh the plot using plt.draw() and plt.pause(0.01).

7. Handle exceptions:
    - Stop the loop and exit cleanly on KeyboardInterrupt (Ctrl+C).
    - Print error messages for other exceptions.

ARDUINO

1. Include necessary libraries:
    - Wire.h for I2C communication.
    - MPU6050.h to interact with the MPU6050 sensor.

2. Initialize components in setup():
    - Start serial communication at 9600 baud using Serial.begin().

    - Begin I2C communication with Wire.begin().
    - Initialize the MPU6050 sensor using mpu.initialize().
3. Inside the loop() function:
    - Call detectGesture() to check if a gesture has occurred.
    - If the detected gesture is different from the previous one:
        - Print the detected gesture over the serial port in the format:
          Detected Gesture: Gesture X
        - Update previousGesture to the current gesture.

4. Read real-time accelerometer and gyroscope values:
    - Use mpu.getMotion6() to get values for ax, ay, az, gx, gy, gz.
    - Continuously send ax and ay over the serial port, separated by a comma, for real-time plotting in Python.

5. Inside detectGesture() function:
    - Read MPU6050 sensor values again using getMotion6().
    - Apply conditional checks on ax and ay values to detect simple gestures:
        - If ax > threshold and ay < threshold → return gesture 1 (e.g., quick right movement).
        - If ax < -threshold and ay > threshold → return gesture 2 (e.g., quick left movement).
    - If no condition is met, return 0 (no gesture).

6. Add delay of 100ms between each loop iteration for smoother performance.

3 Code used

PYTHON

```
import serial
import matplotlib.pyplot as plt
from collections import deque

ser = serial.Serial('COM6', 9600, timeout=1)

plt.ion()
fig, ax = plt.subplots()
x_data = deque(maxlen=100)
y_data = deque(maxlen=100)
plot_line, = ax.plot([], [], 'ro-')

ax.set_xlim(-20000, 20000)
ax.set_ylim(-20000, 20000)
```

```python
ax.set_xlabel('X-axis (Accel)')
ax.set_ylabel('Y-axis (Accel)')
ax.set_title('Real-time Hand Gesture Plot')

while True:
    try:
        line = ser.readline().decode('utf-8').strip()

        if line.startswith("Detected Gesture: "):
            gesture = line.split(": ")[1]
            print(f"Gesture Detected: {gesture}")


        elif ',' in line:
            parts = line.split(',')
            if len(parts) == 2:
                ax_val = int(parts[0])
                ay_val = int(parts[1])
                x_data.append(ax_val)
                y_data.append(ay_val)
                plot_line.set_xdata(x_data)
                plot_line.set_ydata(y_data)
                ax.relim()
                ax.autoscale_view()
                plt.draw()
                plt.pause(0.01)

    except KeyboardInterrupt:
        print("Exiting...")
        break
    except Exception as e:
        print(f"Error: {e}")
```

<u>ARDUINO</u>

```cpp
#include <Wire.h>
#include <MPU6050.h>

MPU6050 mpu;

const int threshold = 10000;
```

```
int previousGesture = -1;

void setup() {
  Serial.begin(9600);
  Wire.begin();
  mpu.initialize();
}

void loop() {
  int gesture = detectGesture();

  if (gesture != previousGesture) {
    Serial.print("Detected Gesture: ");
    if (gesture == 1) {
      Serial.println("Gesture 1");
    } else if (gesture == 2) {
      Serial.println("Gesture 2");

    }
    previousGesture = gesture;
  }


  int ax, ay, az, gx, gy, gz;
  mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
  Serial.print(ax);
  Serial.print(",");
  Serial.println(ay);

  delay(100);
}

int detectGesture() {
  int ax, ay, az, gx, gy, gz;
  mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);


  if (ax > threshold && ay < threshold) {
    return 1;
  } else if (ax < -threshold && ay > threshold) {
    return 2;
  }

  return 0;
}
```

4.b. RFID System:

1 Circuit Assembly

1. Green LED:
    - Anode connected to digital pin 7.
    - Cathode connected to GND
2. Red LED:
    - Anode connected to digital pin 8.
    - Cathode connected to GND.
3. Servo Motor:
    - Orange wire connected to digital pin 9.
    - Red wire connected to 5V.
    - Brown wire connected to GND.
4. RFID Reader Connection:
    - The RFID reader uses a USB-to-RS232 adapter for communication with the laptop.
    - The RS232 female connector of the RFID reader was plugged into the male end of the USB-to-RS232 adapter.
    - The adapter was then connected to the laptop via USB.

2 Programming Logic

PYTHON

1. Import Modules
    - serial: for serial communication with hardware (RFID & Arduino).
    - time: for handling delays and debouncing.
    - json: to load UID data from a file.

2. Initialization
    - Serial Port Setup:
        - RFID_PORT = COM9 → RFID Reader.
        - ARDUINO_PORT = COM5 → Arduino.
        - Baud rate = 9600.
    - Open both serial connections using serial.Serial().
    - Wait 2 seconds to allow both devices to initialize.

3. Load UID Database

- Load a JSON file named uid_data.json.
- This file stores UID as keys and user info (like name, allowed) as values.

4. Main Loop: Listening for Tags
    - Continuously check if RFID has incoming data (rfid_ser.in_waiting).
    - Read one line from RFID reader.
    - Clean the UID string:
        - Strip control characters (\x01, \x02), spaces, and make uppercase.
    - Debounce the same UID (prevent multiple triggers in short time).
        - If it's a new UID or after 2 seconds from the last one:
            - Check if the UID exists in the JSON data.
            - If it exists and access is allowed:
            - Print success message with user's name.
            - Send "ACCESS_GRANTED\n" to Arduino.
        - Else:
            - Print access denied message.
            - Send "ACCESS_DENIED\n" to Arduino.

5. Exit Handling
    - On keyboard interrupt (Ctrl+C), stop the loop gracefully.
    - Close both serial connections to avoid port lock issues.

<u>ARDUINO</u>

1. Initialization
    - Set pin modes:
        - Pin 7 as output (green led).
        - Pin 8 as output (red led).
    - Attach servo motor to pin 9.
    - Move servo to 0° (locked position).
    - Start serial communication at 9600 bps.

2. Main Loop (loop)
    - Continuously check for incoming serial data.
    - If a string is received:
        - Read string until newline (\n).
        - Remove unwanted spaces using .trim().

3. If command is ACCESS_GRANTED:
    - Turn green led ON.
    - Move servo to 90°.
    - Wait for 3 seconds.
    - Move servo back to 0°.
    - Turn green led OFF.

4. If command is ACCESS_DENIED:
    - Turn red led ON.
    - Wait for 2 seconds.
    - Turn red led OFF.

3 Code used

1. Python

```
import serial
import time
import json



RFID_PORT = "COM9"
ARDUINO_PORT = "COM5"
BAUD_RATE = 9600

with open("uid_data.json", "r") as f:
    uid_data = json.load(f)

rfid_ser = serial.Serial(RFID_PORT, BAUD_RATE, timeout=1)
arduino_ser = serial.Serial(ARDUINO_PORT, BAUD_RATE, timeout=1)
time.sleep(2)

print("Listening for RFID tags...")

try:
    last_uid = ""
    last_time = 0

    while True:
        if rfid_ser.in_waiting:
            line = rfid_ser.readline().decode('utf-8', errors='ignore').strip()
            line = line.strip("\x01\x02").replace(" ", "").upper()
            print(f"Scanned UID: {repr(line)}")

            if line and (line != last_uid or time.time() - last_time > 2):
                last_uid = line
                last_time = time.time()
            user_info = uid_data.get(line)

            if isinstance(user_info, dict):
                if user_info.get("allowed", False):
                    print(f" Welcome, {user_info['name']}! Access granted.")
                    arduino_ser.write(b'ACCESS_GRANTED\n')
                else:
                    print(f" {user_info['name']} Access denied.")
                    arduino_ser.write(b'ACCESS_DENIED\n')
            else:
                print(" Unknown UID or Access denied.")
                arduino_ser.write(b'ACCESS_DENIED\n')

except KeyboardInterrupt:
    print("Stopping...")

finally:
    rfid_ser.close()
    arduino_ser.close()
```

2. JSON file

```
{
    "0008090590": {
      "name": "Hamizah",
      "allowed": true
    },
    "0008090485": {
      "name": "Aufa",
      "allowed": false
    },
    "0006493891": {
      "name": "Bushra",
      "allowed": true
    }
  }
```

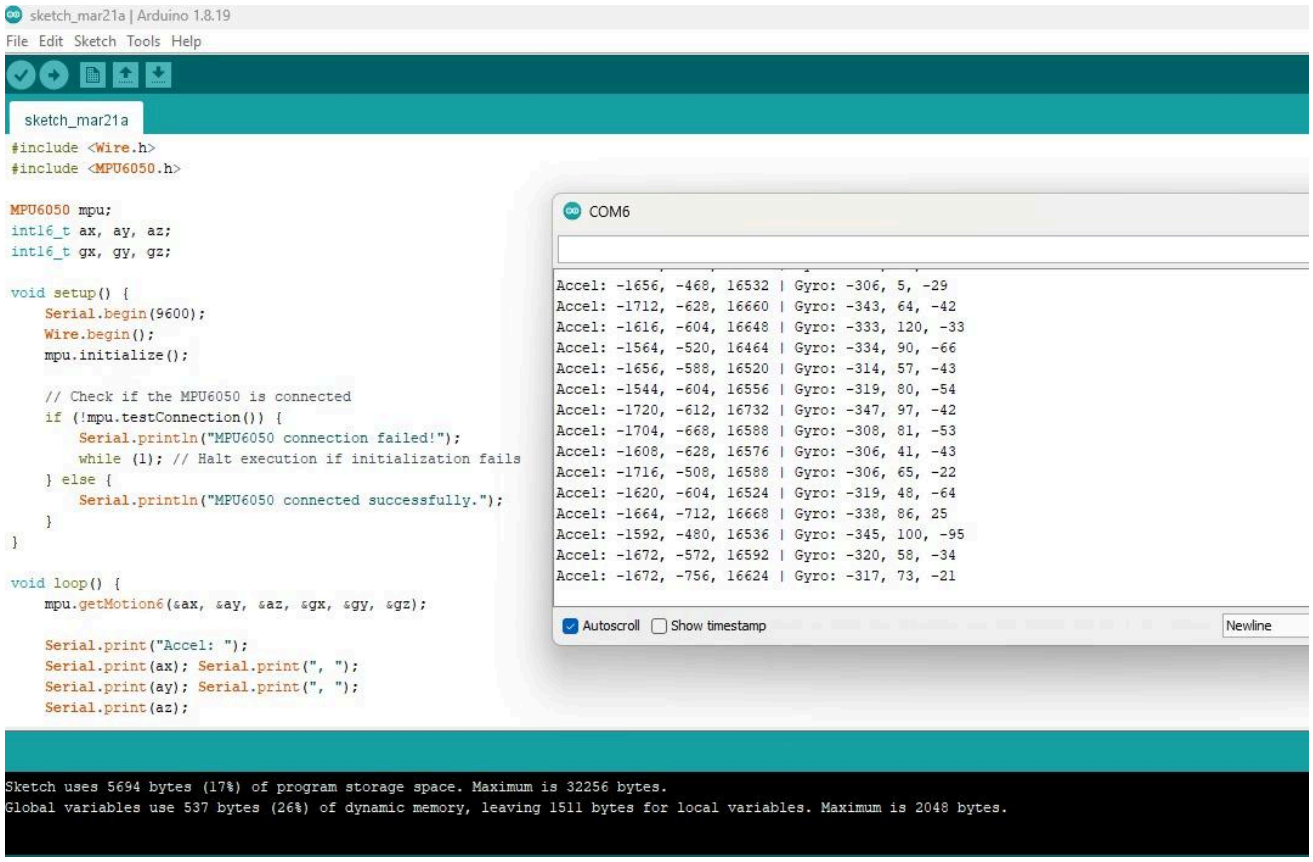
3. Arduino

```
#include <Servo.h>

Servo gateServo;
```

```
const int greenLED = 7;
const int redLED = 8;
const int servoPin = 9;

void setup() {
  pinMode(greenLED, OUTPUT);
  pinMode(redLED, OUTPUT);
  gateServo.attach(servoPin);

  gateServo.write(0);
  Serial.begin(9600);
}

void loop() {
  if (Serial.available()) {
    String cmd = Serial.readStringUntil('\n');
    cmd.trim();

    if (cmd == "ACCESS_GRANTED") {
      digitalWrite(greenLED, HIGH);
      digitalWrite(redLED, LOW);

      gateServo.write(90);
      delay(3000);

      gateServo.write(0);
      digitalWrite(greenLED, LOW);
    }
    else if (cmd == "ACCESS_DENIED") {
      digitalWrite(redLED, HIGH);
      digitalWrite(greenLED, LOW);
      delay(2000);
      digitalWrite(redLED, LOW);
    }
  }
}
```

5.0 DATA COLLECTION

5.a. Hand Gesture System:



```
#include <Wire.h>
#include <MPU6050.h>

MPU6050 mpu;
int16_t ax, ay, az;
int16_t gx, gy, gz;

void setup() {
    Serial.begin(9600);
    Wire.begin();
    mpu.initialize();

    // Check if the MPU6050 is connected
    if (!mpu.testConnection()) {
        Serial.println("MPU6050 connection failed!");
        while (1); // Halt execution if initialization fails
    } else {
        Serial.println("MPU6050 connected successfully.");
    }
}

void loop() {
    mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

    Serial.print("Accel: ");
    Serial.print(ax); Serial.print(", ");
    Serial.print(ay); Serial.print(", ");
    Serial.print(az);
```

```
Accel: -1656, -468, 16532 | Gyro: -306, 5, -29
Accel: -1712, -628, 16660 | Gyro: -343, 64, -42
Accel: -1616, -604, 16648 | Gyro: -333, 120, -33
Accel: -1564, -520, 16464 | Gyro: -334, 90, -66
Accel: -1656, -588, 16520 | Gyro: -314, 57, -43
Accel: -1544, -604, 16556 | Gyro: -319, 80, -54
Accel: -1720, -612, 16732 | Gyro: -347, 97, -42
Accel: -1704, -668, 16588 | Gyro: -308, 81, -53
Accel: -1608, -628, 16576 | Gyro: -306, 41, -43
Accel: -1716, -508, 16588 | Gyro: -306, 65, -22
Accel: -1620, -604, 16524 | Gyro: -319, 48, -64
Accel: -1664, -712, 16668 | Gyro: -338, 86, 25
Accel: -1592, -480, 16536 | Gyro: -345, 100, -95
Accel: -1672, -572, 16592 | Gyro: -320, 58, -34
Accel: -1672, -756, 16624 | Gyro: -317, 73, -21
```

```
Sketch uses 5694 bytes (17%) of program storage space. Maximum is 32256 bytes.
Global variables use 537 bytes (26%) of dynamic memory, leaving 1511 bytes for local variables. Maximum is 2048 bytes.
```

5.b. RFID System:

UID	Name	Accessible	Green LED	Red LED	Servo
0008090590	Hamizah	Granted	Turn on	Remains off	Move 90° and turn back to 0°
0008090485	Aufa	Denied	Remains off	Turn on	Remains 0°
0006493891	Bushra	Granted	Turn on	Remains off	Move 90° and turn back to 0°

6.0 DATA ANALYSIS

6.a. Hand Gesture System:

In this experiment, raw data from the MPU6050 sensor—which combines a 3-axis accelerometer and 3-axis gyroscope—was analyzed to understand hand movement gestures. The data includes linear acceleration (Accel) and angular velocity (Gyro) along the X, Y, and Z axes.

From the data:
Accel Z-axis : Indicates that the sensor remained relatively upright throughout the test, as the Z-axis points against gravity. Values close to 16384 typically represent 1g in raw 16-bit output for ±2g sensitivity.

Accel X and Y axes: Fluctuate in the range of -1500 to -700, indicating minor tilting or hand rotation in the X and Y directions.

Gyro Y-axis: Reaches up to 120°/s, suggesting a significant angular rotation, which could correspond to a wrist-twisting motion.

6.b. RFID System:

The data collected from the RFID experiment consists of the following columns: UID, Name, Accessible, Green LED, Red LED, and Servo. The analysis of this data reveals several important insights:

1. Access Granted vs. Denied
    - A total of 2 access requests were granted, while 1 access request was denied. The individuals granted access were Hamizah and Bushra, while Aufa was denied access.

2. LED Behavior
    - The Green LED turned on in the cases where access was granted (Hamizah and Bushra), which is the expected behavior.
    - The Red LED turned on in the case where access was denied (Aufa), as expected. In the other two cases (Hamizah and Bushra), the Red LED remained off.

3. Servo Movement
    - The Servo moved to 90° and then returned to 0° in the cases where access was granted (Hamizah and Bushra), indicating that the servo performed the expected motion.

    - In the case where access was denied (Aufa), the servo remained at 0°, as expected.

4. Consistency and System Behavior
    - The system correctly responded to access requests by turning on the appropriate LED and moving the servo as per the access status.
    - No discrepancies were observed between the expected behavior and the actual actions of the LEDs and servo based on the access status

7.0 RESULT

7.a. Hand Gesture System:

1. MPU6050 Sensor Performance: The system successfully acquired accelerometer and gyroscope data through the I2C interface, enabling real-time data processing.

2. Gesture Detection: The Arduino successfully recognized basic gestures by comparing the acceleration values (ax, ay) to predefined thresholds. When a gesture was performed, the system accurately classified the gesture and transmitted the gesture data to the PC via the serial port.

Video Link:

7.b. RFID System:

1. UID Detection: The RFID reader successfully detected and read the UID of the RFID tags. The system correctly transmitted the UID to a Python program for validation.

2. Access Validation: The Python program correctly compared the read UID against a pre-stored list in JSON format and successfully determined whether the UID was valid or invalid. Valid UIDs triggered visual feedback via LEDs and servo motor actuation, simulating a real-world access control mechanism.

3. System Response: When a valid RFID tag was detected, the system provided immediate visual feedback, such as rotating the servo motor, confirming that the access control mechanism was functioning as intended.

Video Link:

8.0 DISCUSSION

This experiment explored two standalone embedded systems: a gesture recognition system using the MPU6050 sensor and a RFID-based access control system. Both systems were designed to demonstrate real-time sensor data handling, classification, and actuator control, aligning with modern applications in human-computer interaction and secure access management.

LIMITATIONS & SOURCES OF ERROR:

8.a. Hand Gesture System:

1. Fixed thresholds may not generalize well across different users or movement intensities.
2. Sensor readings were susceptible to noise, drift, and external vibrations.
3. Gesture overlap or slight movements could lead to misclassification.
4. The 100ms loop delay, while smoothing data, introduced minor latency.
5. No filtering or sensor fusion was applied, which could have improved gesture stability and accuracy.

8.b. RFID System:

1. RFID readers can sometimes fail to read a tag if the orientation or distance isn't optimal.
2. Manual UID registration in JSON can become inefficient as the user database scales.
3. No encryption was implemented for data transmission, making it less secure in a real-world application.
4. Serial communication speed must be managed carefully to avoid delays or data corruption, especially during rapid scanning.

9.0 RECOMMENDATIONS

To enhance the performance and scalability of both systems, it is recommended to improve data handling and user feedback mechanisms. Incorporating real-time data logging in Python would allow users to analyze recorded sensor and RFID data for debugging and optimization purposes. Additionally, adopting structured JSON formatting for data communication will improve organization and facilitate future system expansion.

For the hand gesture recognition system using the MPU6050, integrating a more advanced algorithm such as a machine learning model (e.g., k-nearest neighbors or decision trees) could significantly increase accuracy in gesture detection compared to the current threshold-based method. This would be particularly beneficial in distinguishing between subtle or similar hand movements. It is also advisable to adopt a modular design approach in both

Arduino and Python code. By separating hardware interaction, logical processing, and user interface components, the systems will become more maintainable and easier to debug or upgrade. In the context of the RFID system, implementing secure authentication features—such as encrypted UID verification—will enhance security, especially if the system is to be deployed in real-world access control scenarios.

Lastly, to improve usability, integrating a graphical user interface (GUI) in the Python application using libraries like Tkinter or PyQt could provide a more intuitive user experience. This would allow users to interact with the system visually, such as setting custom servo angles or monitoring recognized gestures and card scans more effectively.

10.0 REFERENCES

1. https://learn.sparkfun.com/tutorials/mpu-6050-gyroscope-and-accelerometer-hookup-guide
2. https://randomnerdtutorials.com/security-access-using-rfid-reader-arduino/

11.0 APPENDICES

12.0 ACKNOWLEDGEMENT

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 [/]