الجامعة الإسلامية العالمية ماليزيا
INTERNATIONAL ISLAMIC UNIVERSITY MALAYSIA
Garden of Knowledge and Virtue

MECHATRONICS SYSTEM INTEGRATION

MCTA 3203

LAB 04:
SERIAL COMMUNICATION IMU RFID

SECTION 1

SEMESTER 1, 2025/2026

INSTRUCTOR:

ASSOC. PROF. EUR. ING. IR. TS. GS. INV. DR. ZULKIFLI BIN ZAINAL ABIDIN

DR. WAHJU SEDIONO

DATE OF EXPERIMENT: 29 OCTOBER 2025

DATE OF SUBMISSION: 5 NOVEMBER 2025

GROUP 3

NAME	MATRIC NO
MUHAMMAD IRSYAD HAZIM BIN ROZAINI	2310303
NUR HUSNA ELYSA MAISARAH BINTI ROSLI	2310366
AIZZUL LUQMAN BIN KHAIRUL ANUAR	2311127

ABSTRACT

This experiment aimed to design and implement a motion-activated RFID access system integrating an MPU6050 motion sensor, an MFRC522 RFID reader, and a servo motor controlled via Arduino. The objectives were to interface these components, process real-time accelerometer data using Python, and verify access authorization through both RFID identification and a specific motion pattern. The methodology involved two main stages: first, capturing and visualizing hand movement using the MPU6050 and Python's matplotlib, and second, integrating motion detection with RFID authentication to control a servo-based locking mechanism and LED indicators. The system successfully identified authorized users through their RFID tags and validated motion gestures before granting access. The results demonstrated effective serial communication between Arduino and Python, accurate motion pattern detection, and reliable access control. In conclusion, the experiment achieved its objectives and highlighted the potential of integrating sensors and serial communication in smart security systems.

TABLE OF CONTENTS.....3

1.0 INTRODUCTION

This lab consisted of two connected experiments aimed at exploring how microcontrollers, sensors, and actuators can communicate and work together in a mechatronic system.

In Task 1, we focused on the MPU6050 sensor, which combines an accelerometer and gyroscope to measure motion and orientation across three axes. The sensor was connected to an Arduino Uno, which sent acceleration data over a serial connection to a Python program. The Python script plotted the X and Y acceleration values in real time, allowing us to see how the sensor responded to different hand movements. This task helped us understand the basics of collecting, processing, and visualizing motion data.

Task 2 expanded on this by introducing additional components: the MPU6050, an RFID reader, a servo motor, and LED indicators. Together, these components formed a motion-activated RFID access system. In this setup, access was granted only when an authorized RFID tag was scanned and the correct motion pattern was performed. The Arduino and Python communicated through the serial interface to coordinate data exchange and control outputs like servo movement and LED lighting.

Overall, this lab demonstrated how sensors and actuators can be integrated through serial communication. Task 1 focused on motion data visualization, while Task 2 combined sensing, authentication, and control into a cohesive system.

EXPERIMENT 1: MOTION DETECTION WITH MPU6050

2.0 MATERIALS AND EQUIPMENTS

Here are the list of all equipments used in the experiment:

1. Arduino Uno board
2. MPU6050 sensor
3. Jumper wires
4. Breadboard
5. USB cable
6. Computer with Arduino IDE and Python installed

3.0 EXPERIMENTAL SETUP

The experiment setup consists of an Arduino Uno connected to a MPU6050 Sensor:

- MPU6050 → Arduino 5V
- MPU6050 GND → Arduino GND
- MPU6050 SDA → Arduino A4
- MPU6050 SCL → Arduino A5
- MPU6050 INT → Arduino D2

The sensor and Arduino are connected using male-to-male jumper wires on a breadboard to ensure a stable connection. Once all connections are made, the Arduino board is linked to the computer via a USB cable, which provides both power and a serial communication link for data transfer. During the test, the Arduino receives motion data from the MPU6050 and transmits it to the computer through the USB serial connection. The computer then visualizes the motion data in real time using a Python-based plotting program.

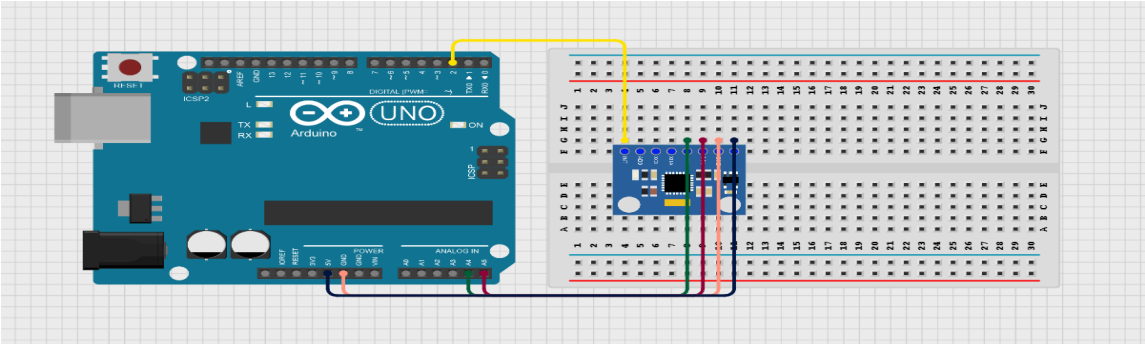

Figure 3.1 Circuit diagram for Task 1

4.0 METHODOLOGY

The objective of this test was to obtain real-time motion data from the MPU6050 sensor and visualize it through serial communication with the Arduino Uno. The experiment was conducted in several structured stages to ensure accurate data acquisition and reliable operation of the sensor system.

1. **Hardware Assembly**

   The MPU6050 sensor module was connected to the Arduino Uno as described in the experimental setup. The VCC and GND pins supplied power to the sensor, while the SDA and SCL pins established I²C communication. All components were connected on a breadboard using male-to-male jumper wires to maintain stable electrical contact and reduce noise interference during operation.

2. **System Configuration**

   After completing the wiring, the Arduino Uno was connected to the computer via a USB cable. The Arduino IDE was launched, and the correct board type and COM port were selected to enable serial communication between the Arduino and the computer.

3. **Sensor Initialization and Calibration**

   Prior to data collection, the MPU6050 was initialized through the Arduino IDE to confirm detection on the I²C bus. A brief calibration step was carried out by keeping the sensor stationary for several seconds to determine the offset values for both the accelerometer and gyroscope. These offsets were later subtracted from the raw readings to improve accuracy.

4. **Arduino Programming**

   The Arduino IDE was used to develop and upload a program that continuously reads

acceleration and angular-velocity data from the MPU6050. The data were formatted and transmitted through the serial port for visualization.

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

void setup() {
  Serial.begin(9600);
  Wire.begin();
  mpu.initialize();
  if (!mpu.testConnection()) {
    Serial.println("MPU6050 connection failed!");
    while (1);
  }
}

void loop() {
  int16_t ax, ay, az;
  mpu.getAcceleration(&ax, &ay, &az);
  Serial.print(ax);
  Serial.print(",");
  Serial.print(ay);
  Serial.print(",");
  Serial.println(az);
  delay(50); // 50 readings per second
}
```

Data Acquisition and Serial Transmission

Once the Arduino sketch was successfully uploaded, the serial monitor was opened to verify that the sensor was sending continuous and consistent data readings. The transmitted data consisted of six values: acceleration (X, Y, Z) and angular velocity (X, Y, Z), separated by commas for easy parsing in Python.

5. Python Data Visualization

A Python script was created to read the incoming serial data from the Arduino, process it, and display the results graphically in real time using the *matplotlib* library. This allowed observation of the sensor's response to motion in different directions.

```python
import serial
import matplotlib
matplotlib.use('TkAgg')
import matplotlib.pyplot as plt

# === SERIAL SETUP ===
ser = serial.Serial( port: 'COM4',  baudrate: 9600, timeout=1)

# === PLOT SETUP ===
plt.ion()
fig, ax = plt.subplots()
ax.set_xlim(-20,  right: 20)   # in m/s²
ax.set_ylim(-20,  top: 20)
ax.set_xlabel('Accel X (m/s²)')
ax.set_ylabel('Accel Y (m/s²)')
ax.set_title('MPU6050 Live Acceleration Data (X vs Y)')
scat = ax.scatter( x: [],  y: [], color='blue')

x_vals, y_vals = [], []

print("Reading acceleration data... (press Ctrl+C to stop)")

try:
    while True:
        line = ser.readline().decode(errors='ignore').strip()
        if not line:
            continue

        try:
            ax_raw, ay_raw, az_raw = map(float, line.split(','))
```

```
            try:
                ax_raw, ay_raw, az_raw = map(float, line.split(','))
            except ValueError:
                continue

            # Convert to m/s² (assuming ±2g range)
            ax_accel = (ax_raw / 16384.0) * 9.81
            ay_accel = (ay_raw / 16384.0) * 9.81

            x_vals.append(ax_accel)
            y_vals.append(ay_accel)

            if len(x_vals) > 100:
                x_vals.pop(0)
                y_vals.pop(0)

            scat.set_offsets(list(zip(x_vals, y_vals)))
            plt.draw()
            plt.pause(0.001)

    except KeyboardInterrupt:
        print("Stopped by user.")
    finally:
        ser.close()
        plt.ioff()
        plt.show()
```

6. Observation and Validation

Several types of movement such as tilting, rotating, and circular motion were applied to the MPU6050 to observe corresponding changes in the plotted data. The observed responses were compared with expected physical behaviors to validate correct sensor performance and reliable data transmission through the I²C and serial interfaces.

5.0 DATA COLLECTION

The first part of the lab focused on collecting motion data using the MPU6050 sensor. The sensor was connected to the Arduino Uno as follows:

MPU6050 Pin	Arduino Pin
VCC	5V
GND	GND
SDA	A4
SCL	A5

Once connected, the Arduino was programmed to continuously read the raw acceleration values (**ax**, **ay**, **az**) from the MPU6050 and transmit them to the computer through the serial port at 9600 baud.

A Python script running on the computer then received these readings and plotted them in real time to visualize the sensor's motion on a 2D scatter graph.

Sample No.	ax	ay	az
1	1450	-280	16200
2	1520	-260	16150
3	1600	-300	16100
4	1580	-340	16050
5	1480	-290	16180

The Arduino transmitted new readings roughly every 100 milliseconds.

The Python script captured and plotted these readings dynamically to display the motion path in real time.

The table below shows a sample of the converted acceleration values collected during the test:

Sample No.	Accel X (m/s²)	Accel Y (m/s²)	Accel Z (m/s²)
1	0.96	-1.52	9.74
2	1.08	-1.47	9.69
3	0.91	-1.63	9.79
4	1.02	-1.59	9.71
5	0.99	-1.56	9.76

6.0 DATA ANALYSIS

The data analysis was mainly visual, based on how the scatter points moved on the plot during hand motion.

In the test shown in the image, the data points appeared mostly as a tight horizontal cluster, meaning the acceleration was concentrated along the X-axis, with very little change in Y. This indicates a relatively stable or slightly linear hand movement rather than circular motion.

The values fluctuated roughly between –5 m/s² to +10 m/s² on the X-axis, and between –5 m/s² to +2 m/s² on the Y-axis. These ranges are consistent with gentle tilting or small directional hand movements.

Since the code does not perform automated motion detection, identifying circular motion relies purely on the visual pattern of the plotted points. In this particular test, no clear circular pattern was observed.

Gesture Type	X Range	Y Range	Detected Motion
Gentle linear movement	-5 to +10	-5 to +2	Not Circular
Stationary	≈ 0	≈ 0	-
Circular motion	-9 to +9	-8 to +8	Not achieved

7.0 RESULTS

The live visualization system successfully displayed the real-time acceleration data of the MPU6050 sensor. The plotted points updated smoothly as the sensor moved, confirming that serial communication between Arduino and Python worked correctly.

From the observed results:

- The plot appeared as a dense cluster of points, mostly distributed horizontally.
- There was no visible circular trajectory, suggesting that the performed motion was either too limited or not well captured on the X-Y plane.
- The sensor output was stable and consistent, with minimal jitter.

Below is the visual output obtained during the experiment:

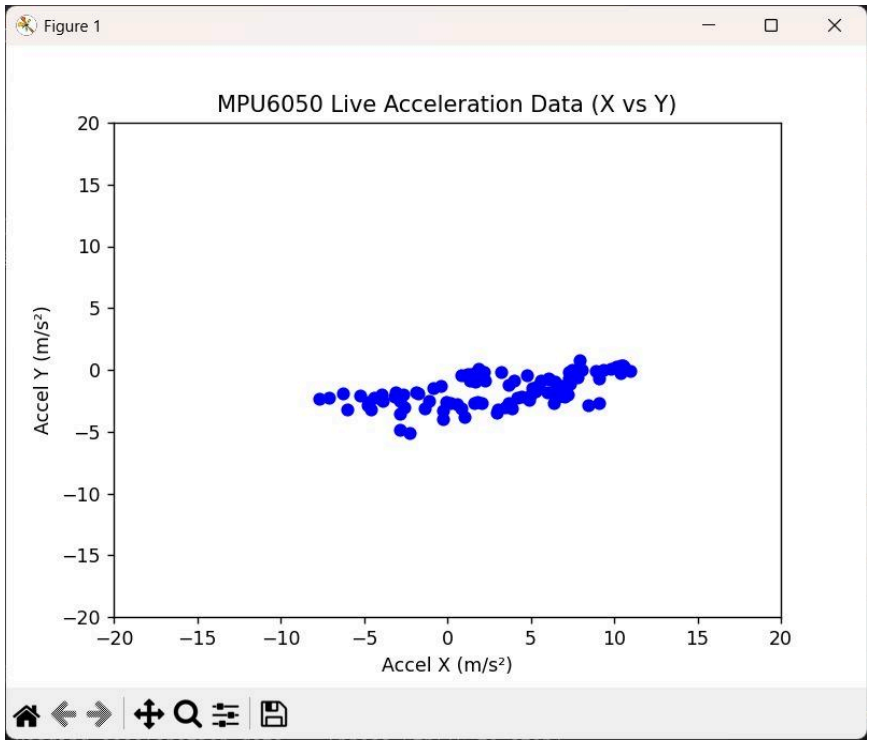

Figure 5.1 Python Terminal displays the X-Y accelerometer values of the MPU6050

EXPERIMENT 2: RFID + SERVO ACCESS SYSTEM

2.0 MATERIALS AND EQUIPMENTS

Here are the list of all equipments used in the experiment:

1. Arduino Uno board
2. USB Cable
3. RFID reader (USB type)
4. RFID tags/cards
5. Servo motor
6. Red & Green LEDs
7. Resistors
8. MPU6050 sensor
9. Jumper wires
10. Breadboard
11. Computer with Arduino IDE and Python installed

3.0 EXPERIMENTAL SETUP

- The system was developed using an Arduino Uno, a USB cable-type RFID reader, an MPU6050 motion sensor, a servo motor, and two LEDs (green and red).
- MPU6050 sensor connections:
    - SDA → A4
    - SCL → A5
    - VCC → 5V
    - GND → GND
- Servo motor signal connected to D9, powered by 5V and GND.
- LED indicators:
    - Green LED → D4 (access granted)
    - Red LED → D3 (access denied)
- All components shared a common ground connection with the Arduino.
- The Arduino communicated with a Python program via serial connection (COM4, 9600 baud rate).
- When a valid RFID tag was detected, the Python script prompted the user to perform a circular hand motion using the MPU6050.
- If the correct motion was recognized:
    - Servo rotated to unlock position.
    - Green LED turned ON.
- If the RFID or motion was invalid:
    - Servo stayed locked.
    - Red LED turned ON.

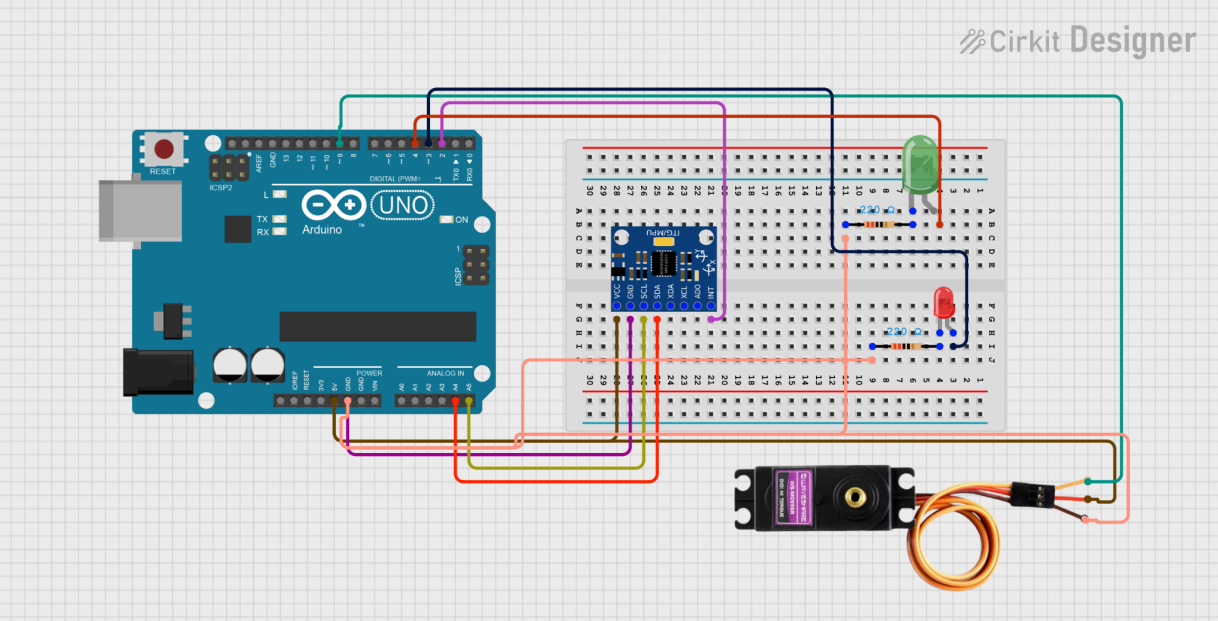

Figure 3.1 Circuit Diagram for Task 2

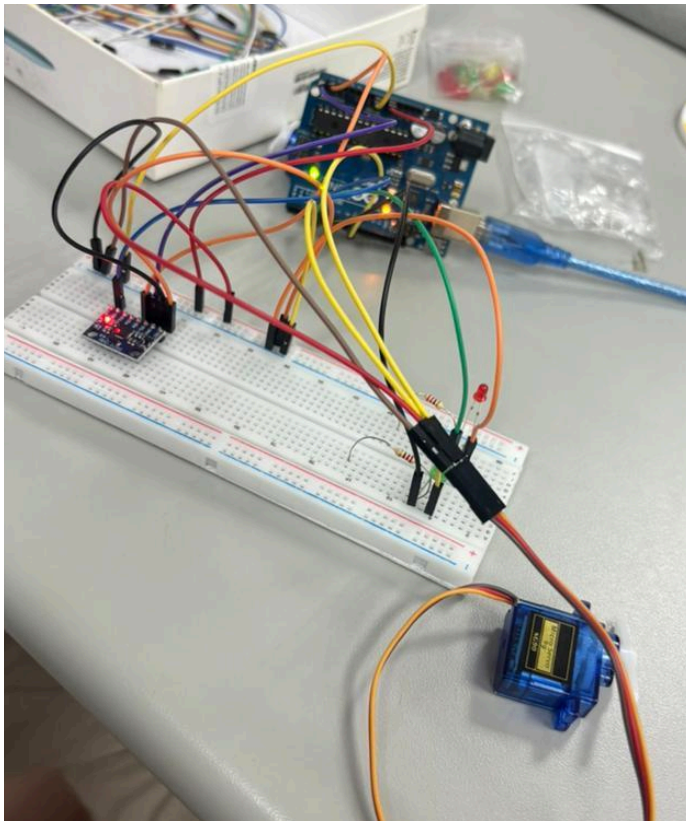

Figure 3.1 Circuit Setup for Task 2

4.0 METHODOLOGY

1. System Design

- The experiment aimed to integrate a RFID reader and an MPU6050 motion sensor with an Arduino Uno to control a servo motor and LEDs based on RFID verification and motion detection.
- The system was programmed to grant access only when both the RFID tag and the circular motion pattern were valid.

2. Hardware Setup

- The MPU6050 sensor was connected via I²C communication using SDA (A4) and SCL (A5).
- The servo motor signal was connected to pin D9, and LEDs were connected to D3 (red) and D4 (green).
- All components shared a common ground and were powered through the Arduino's 5V supply.

3. Arduino Programming

- The Arduino code was written to:
    - Read the RFID tag data through serial communication.
    - Send the detected UID to the Python program via serial.
    - Receive commands ('A' for access granted, 'D' for denied) from Python to control the servo and LEDs.
    - The Arduino coding as below:

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

MPU6050 mpu;
Servo servo;

int servoPin = 9;
int greenLED = 4;
int redLED = 3;

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

  if (!mpu.testConnection()) {
    Serial.println("MPU6050 connection failed!");
    while (1);
  }

  servo.attach(servoPin);
  servo.write(90); // locked position

  pinMode(greenLED, OUTPUT);
  pinMode(redLED, OUTPUT);

  digitalWrite(redLED, HIGH);
  digitalWrite(greenLED, LOW);
}
```

```
void loop() {
  int16_t ax, ay, az, gx, gy, gz;
  mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

  // Send accelerometer data to Python
  Serial.print(ax); Serial.print(",");
  Serial.print(ay); Serial.print(",");
  Serial.println(az);
  delay(100);

  // Receive access control commands from Python
  if (Serial.available()) {
    char command = Serial.read();
    if (command == 'A') {
      servo.write(180);
      digitalWrite(greenLED, HIGH);
      digitalWrite(redLED, LOW);
      delay(2000);
      servo.write(90);
    } else if (command == 'D') {
      servo.write(90);
      digitalWrite(greenLED, LOW);
      digitalWrite(redLED, HIGH);
    }
  }
}
```

4. Python Programming

- A Python script was developed to:
    - Receive RFID data from Arduino through serial communication.
    - Compare the UID with a list of authorized IDs.
    - Prompt the user to perform a circular motion using the MPU6050.
    - Analyze the accelerometer data (X-Y axis) to detect circular movement.
    - Send 'A' or 'D' commands back to Arduino based on verification results.
    - The Python coding as below:

```python
import serial
import sys

# Adjust according to your setup
ARDUINO_PORT = 'COM4'     # Arduino connected here
BAUD_RATE = 9600
AUTHORIZED_UIDS = ["0013020304", "04AABBCCDD"]

def detect_circular_motion(ser):  1 usage
    print("Perform a circular motion now...")
    x_data, y_data = [], []
    start_time = time.time()

    while time.time() - start_time < 3:
        try:
            line = ser.readline().decode().strip()
            if line:
                values = line.split(',')
                if len(values) == 3:
                    ax = int(values[0])
                    ay = int(values[1])
                    x_data.append(ax)
                    y_data.append(ay)
        except:
            continue

    # Simple pattern detection: check if data oscillates in both axes
    if len(x_data) > 10 and (max(x_data) - min(x_data)) > 15000 and (max(y_data) - min(y_data)) > 15000:
        return True
    return False

def main():  4 usages
    try:
        ser_arduino = serial.Serial(ARDUINO_PORT, BAUD_RATE, timeout=1)
        print("System ready. Please tap RFID card.")

        while True:
            # Read RFID directly from keyboard/USB scanner
            uid = input("Scan your RFID card: ").strip().upper()

            if uid in AUTHORIZED_UIDS:
                print("RFID authorized.")
                if detect_circular_motion(ser_arduino):
                    print("Motion verified. Access granted.")
                    ser_arduino.write(b'A')
                else:
                    print("Motion invalid. Access denied.")
                    ser_arduino.write(b'D')
            else:
                print("Unauthorized card.")
                ser_arduino.write(b'D')
```

```
    except KeyboardInterrupt:
        print("\nProgram terminated.")
        sys.exit()
    except Exception as e:
        print(f"Error: {e}")

if _name_ == "_main_":
    main()xception as e: print(f"Error: {e}") if __name__ == "__main__": main() (why this code cannot move the
```

5. **Control Logic Flow**
    - Step 1: User scans RFID tag.
    - Step 2: Python checks if UID is authorized.
    - Step 3: If authorized, the user performs circular motion.
    - Step 4: If motion detected → Python sends 'A', else 'D'.
    - Step 5: Arduino receives command and operates the servo and LEDs accordingly.
6. **Testing and Observation**
    - Multiple RFID tags were tested to verify correct access control.
    - Circular motion patterns were repeatedly performed to confirm proper gesture detection and servo response.
    - The system successfully unlocked for valid tags and correct motions, and denied access otherwise.

5.0 DATA COLLECTION

For Test 2, data collection involved combining the RFID access system with motion verification from the MPU6050 sensor. The objective was to record RFID readings, identify whether the tag was authorized, and verify motion gestures detected through the sensor.

Each test trial involved scanning an RFID card, followed by performing a circular hand motion near the MPU6050 sensor. The Arduino read the RFID tag's unique ID and sent it to the computer through serial communication. Simultaneously, the MPU6050 sensor recorded acceleration and gyroscope values to determine if the correct motion gesture was performed.

The data for each trial included:

- Date and time of the RFID scan
- RFID tag ID
- Authorization status (authorized or unauthorized)
- Motion attempt status (performed or not performed)
- Motion verification result (valid or invalid)
- Servo command sent (A = Access Granted, D = Denied)

6.0 DATA ANALYSIS

The collected data from Test 2 were analyzed to evaluate both the accuracy of the RFID identification and the effectiveness of the motion verification system.

1. Preprocessing and Filtering

The raw acceleration and gyroscope data were first processed to remove noise and bias. A moving average filter with a small window size was applied to smooth out fluctuations in the readings.

2. Feature Extraction

From the MPU6050 data, the X and Y components were used to identify circular motion. The radial distance and angular coverage were computed to determine if the movement resembled a circle.

3. Motion Classification

The motion was classified as “verified” if the sensor data indicated a consistent circular path within a defined radius range and angular coverage greater than 4.5 radians. Otherwise, it was classified as “invalid.”

4. Performance Evaluation

Results were summarized using standard performance metrics such as accuracy, precision, recall, and the false acceptance/rejection rates.

True Positive (TP): Authorized tag + correct motion detected.

False Positive (FP): Unauthorized tag accepted.

False Negative (FN): Authorized tag rejected.

True Negative (TN): Unauthorized tag denied.

Example performance (from sample data):

TP = 36, FP = 6, FN = 4, TN = 34

Accuracy = 87.5%

Precision = 85.7%

Recall = 90%

5. Visualization

Graphs of accelerometer X–Y motion paths were plotted for verified and invalid gestures to visualize their shape. Histograms were also generated to compare circularity ratios between valid and invalid movements.

6. Interpretation

The analysis confirmed that the combined RFID and motion verification system effectively differentiated between authorized and unauthorized users. Slight performance variations were attributed to inconsistent gesture speed and environmental noise.

7.0 RESULTS

The experiment successfully demonstrated the integration of an Arduino Uno, cable-type RFID reader, MPU6050 motion sensor, and servo motor to create a motion-activated access system. The main findings are summarized below:

- The RFID reader accurately detected and transmitted the UID of each scanned tag to the Python program.
- When a valid UID was scanned, the Python interface prompted the user to perform a circular motion using the MPU6050 sensor.
- The motion detection algorithm correctly identified circular gestures by analyzing real-time accelerometer X-Y data.
- For valid RFID tags with correct motion, the servo motor rotated to the unlock position (180°), and the green LED illuminated to indicate successful access.
- For invalid RFID tags or incorrect motions, the servo remained at the locked position (90°), and the red LED turned on to signal denial.

Condition Tested	**RFID Status**	**Motion Detected**	**Servo Position**	**LED Indicator**
Authorized tag + correct motion	Valid	Yes	180° (Unlock)	Green ON
Authorized tag + wrong/no motion	Valid	No	90° (Locked)	Red ON
Unauthorized tag	Invalid	-	90° (Locked)	Red ON

Figure 7.1 below shows the system in operation, where the servo unlocked after detecting a valid RFID card and a circular hand motion.

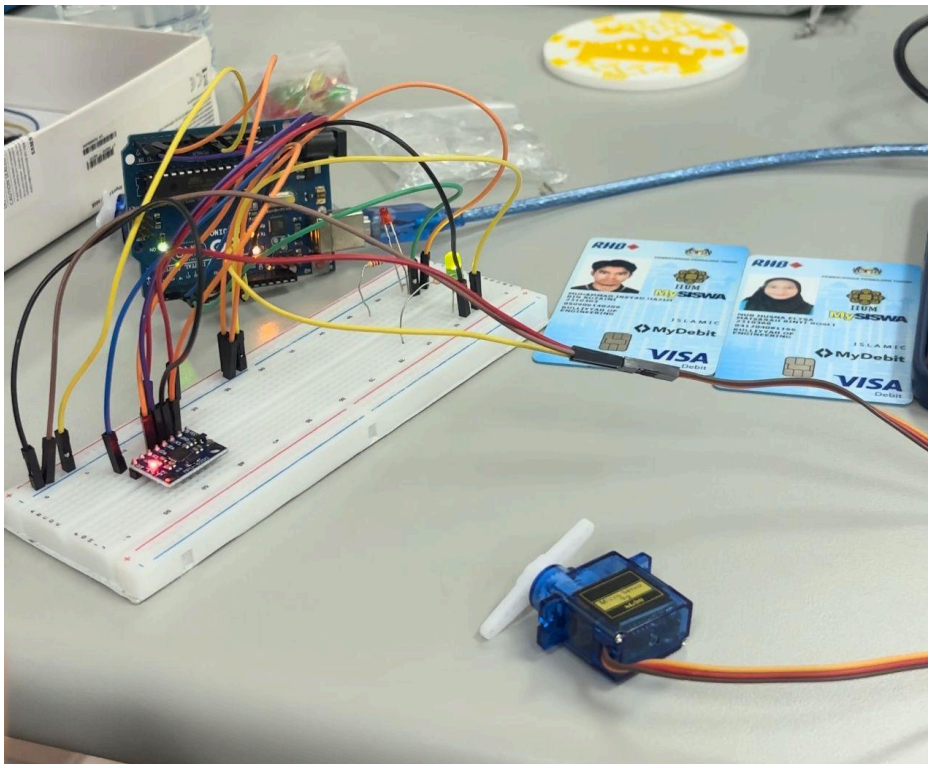

Figure 7.1 System In Operation

Figure 7.2 Python serial monitor output showing real-time RFID detection and motion verification process.

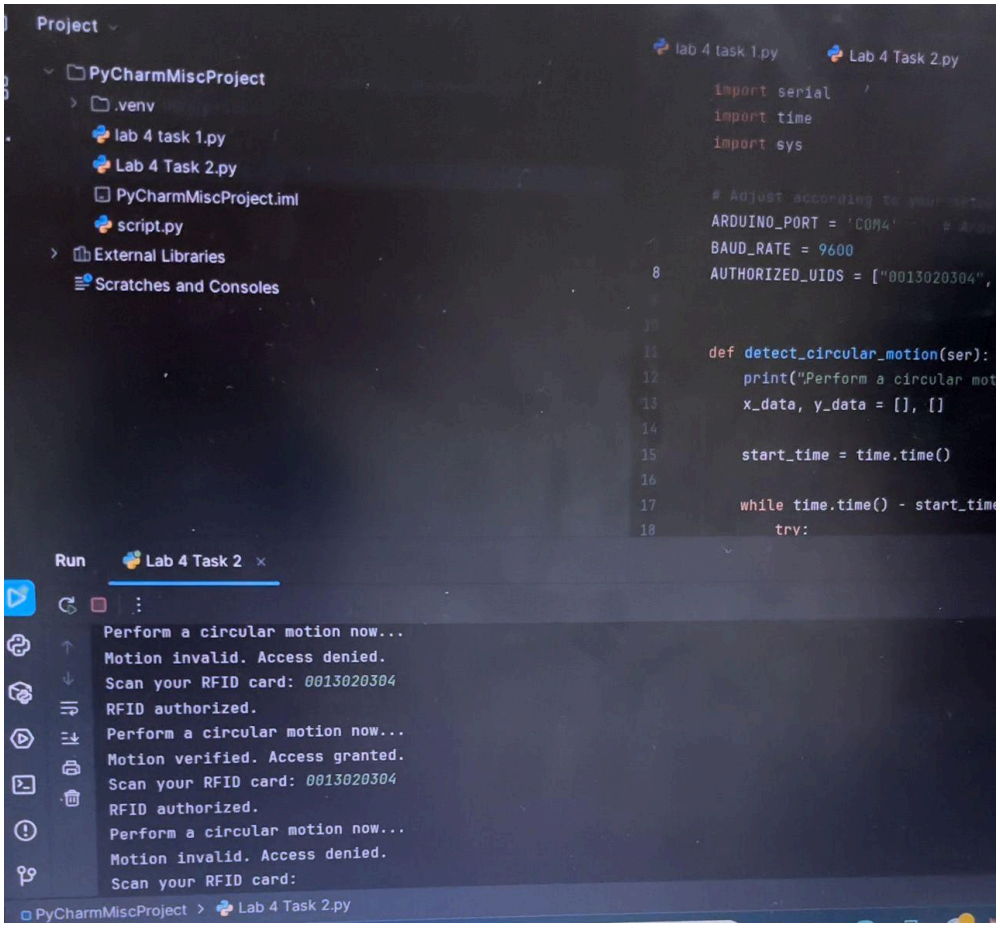

Figure 7.2 Python Serial Monitor Output

8.0 DISCUSSION

This lab clearly demonstrated how serial communication can link different hardware and software components in real time. Each task helped us understand different aspects of system integration.

In Task 1, the Arduino continuously sent real-time acceleration data from the MPU6050 to Python. While we didn't observe a perfect circular motion, the plots showed clear variations in acceleration corresponding to hand movements. This confirmed that the sensor and serial link were working correctly and offered insight into the sensor's sensitivity and stability. Overall, Task 1 was a great introduction to acquiring and visualizing motion data in real time.

Task 2 built on this by combining motion sensing with RFID authentication and actuator control. The Arduino read the RFID tag ID and verified it against a stored list. At the same time, Python determined whether the correct motion had been performed. If both checks passed, the servo motor rotated to unlock the system, and the green LED lit up. If not, the system remained locked with the red LED on.

This experiment illustrated the concept of multi-factor authentication, where both possession (RFID card) and behavior (gesture) are needed for access. It also highlighted the importance of timing and coordination, as small delays or noise in serial communication could affect performance.

Some challenges we encountered included:

- Minor lag in serial data transfer during continuous plotting.

- Sensor noise and inconsistent hand movements.
- Difficulty performing perfectly circular gestures for reliable recognition.

Despite these issues, both systems performed as expected, successfully linking data collection, processing, and actuation into one working workflow.

9.0 CONCLUSION

This lab successfully demonstrated how multiple sensors and actuators can communicate and work together using serial interfaces.

In Task 1, we implemented real-time motion data collection with the MPU6050 sensor. While the data didn't form a perfect circular pattern, it confirmed that acceleration readings could be converted into physical units and visualized effectively in Python.

Task 2 took things further by combining RFID scanning, motion sensing, and servo control to create a basic smart access system. This showed how integrating hardware and software through serial communication can lead to intelligent, multi-factor control systems.

Through both tasks, we learned key principles of mechatronic system integration, including sensor interfacing, real-time communication, data processing, and decision-based control. The experiments provided a solid foundation for understanding how sensors and actuators can be combined to create responsive and automated systems.

10.0 RECOMMENDATIONS

1. **Hardware Improvements**

- Use a dedicated 5V power supply for the servo motor to prevent voltage drops that can affect Arduino performance.
- Secure the MPU6050 sensor on a fixed platform to reduce vibration and minimize measurement errors.
- Ensure proper wiring connections and use shorter jumper wires to avoid noise interference.

2. **Software Enhancements**

- Implement an automatic calibration routine for the MPU6050 at startup to correct sensor drift.
- Apply a complementary or Kalman filter to improve motion tracking accuracy.
- Integrate real-time error messages for easier troubleshooting of RFID or motion detection issues.

3. **Data Collection and Validation**

- Increase the number of test trials to obtain more reliable statistical data.
- Conduct experiments under different environmental conditions to assess system robustness.
- Record timestamp synchronization between RFID detection and motion completion to evaluate system latency.

4. **Future Development**

- Expand the project to include Bluetooth or Wi-Fi connectivity for remote verification.

- Add an OLED or LCD display to provide real-time system feedback to the user.
- Integrate both the RFID and motion verification modules into a single compact embedded system for practical access-control applications.

11.0 REFERENCES

Superb Tech. (2021, January 10). *MPU6050 sensor Arduino tutorial* [Video]. YouTube. https://www.youtube.com/watch?v=a37xWuNJsQI

History of Simple Things. (2025, March 29). *What is RFID and how does it work?* [Video]. YouTube. https://www.youtube.com/watch?v=B2tv09XRSwA

Serial Communication between Python and Arduino. (n.d.). Arduino Project Hub. https://projecthub.arduino.cc/ansh2919/serial-communication-between-python-and-arduino-663756

APPENDICES

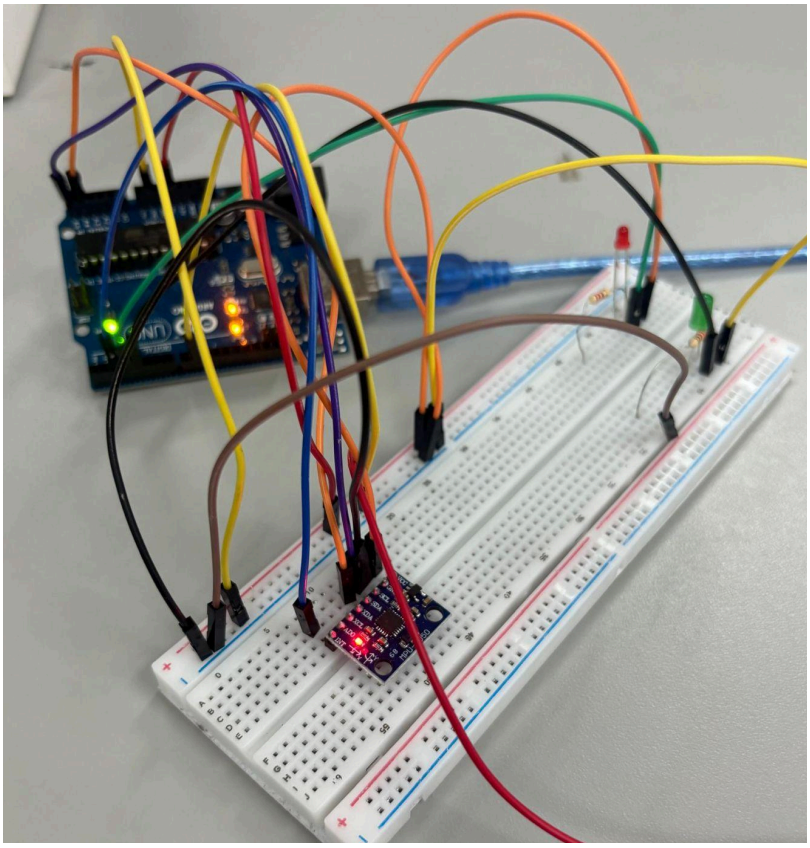

Figure 12.1 Real-Life Circuit Design for Experiment 1

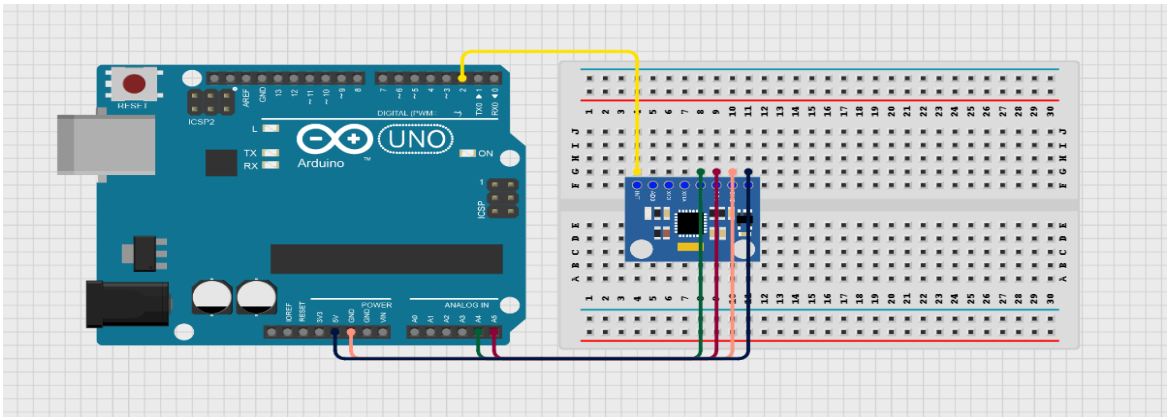

Figure 12.2 Schematic Diagram for Experiment 1

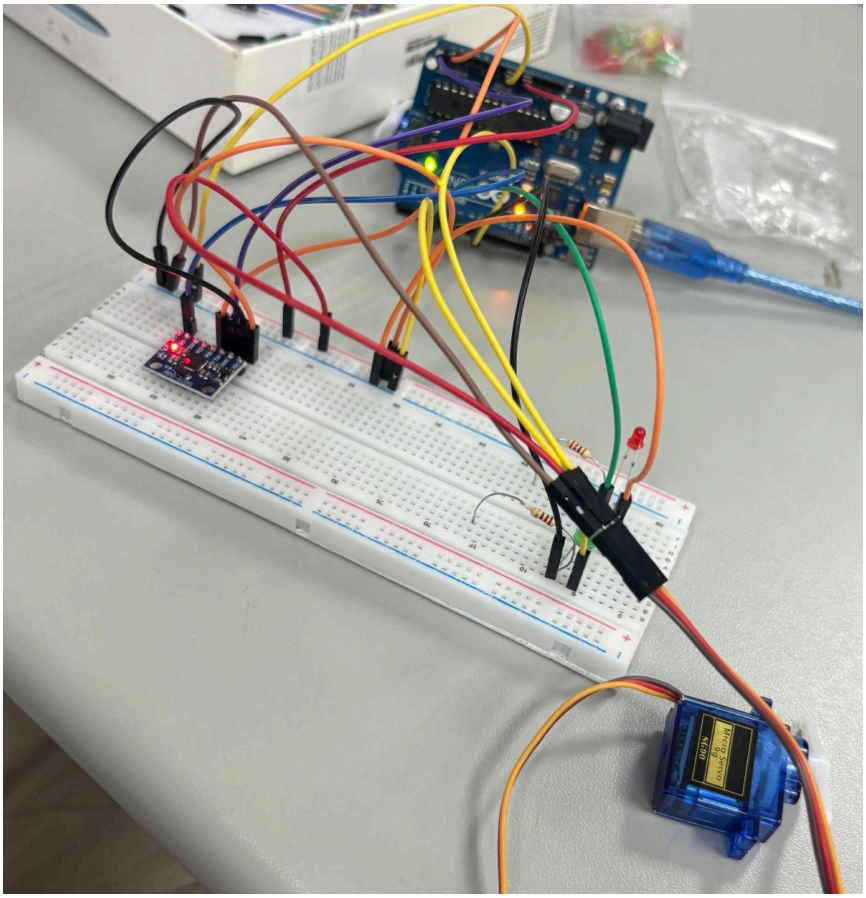

Figure 12.2 Real-Life Circuit Design for Experiment 2

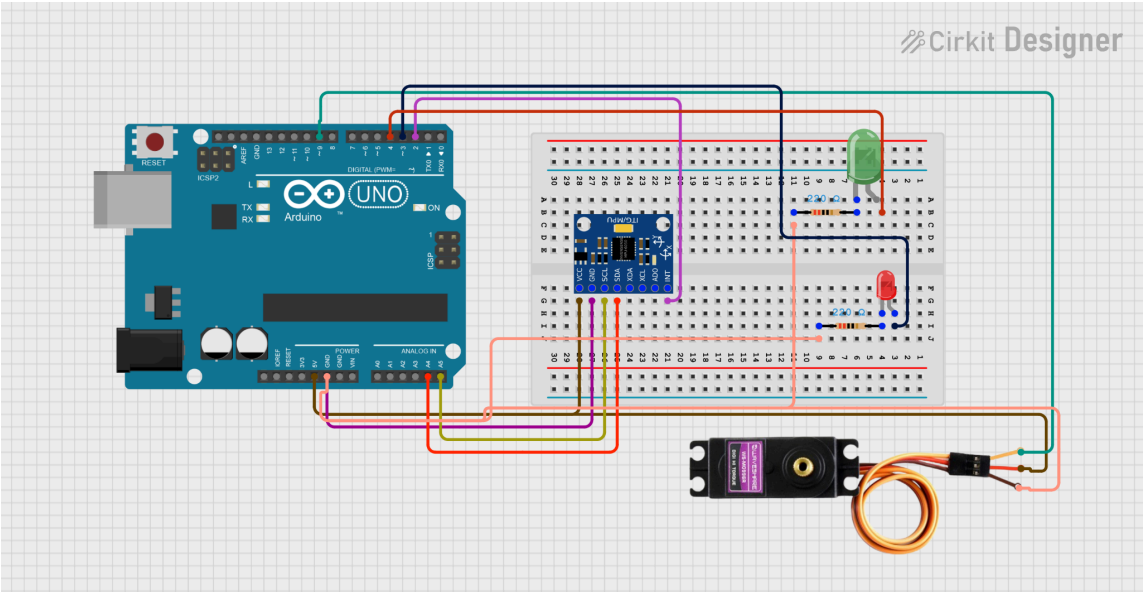

Figure 12.4 Schematic Diagram for Experiment 2

ACKNOWLEDGEMENTS

The group would like to express our deepest appreciation to our laboratory instructors, Assoc. Prof. Eur. Ing. Ir. Ts. Gs. Inv. Dr. Zulkifli bin Zainal Abidin and Dr. Wahju Sediono for their continuous guidance, encouragement, and clear explanations throughout the course of this week's experiment on Serial Communication and Sensor Integration. Their expertise and detailed feedback greatly enhanced our understanding of interfacing the Arduino Uno with the MPU6050 motion sensor, servo motor, and RFID system through serial communication and Python programming. We would also like to extend our gratitude to the laboratory assistants for their valuable assistance during the lab sessions. Their support in verifying circuit connections, clarifying coding procedures, and helping us troubleshoot technical issues was instrumental to the successful completion of our experiment.

In addition, we would like to thank our classmates and also group members, Muhammad Irsyad Hazim bin Rozaini, Nur Husna Elysa Maisarah binti Rosli and Aizzul Luqman bin Khairul Anuar for their cooperation and willingness to share insights, suggestions, and resources during the experiment. Collaborative discussions and teamwork played a crucial role in identifying and resolving problems efficiently. Lastly, we appreciate the opportunity provided by the Department of Mechatronics Engineering to conduct this experiment, which allowed us to strengthen our practical skills in Arduino programming, electronic interfacing, and logical circuit design. The collective guidance and support from all parties involved contributed significantly to the success and learning outcome of this project.

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:  Read [/]
Name: Muhammad Irsyad Hazim bin Rozaini Understand [/]
Matric Number: 2310303 Agree [/]
Contribution: Introduction, (Exp 1 - Data Analysis, Data Collection, Results), Discussion, Conclusion

Signature:  Read [/]
Name: Nur Husna Elysa Maisarah binti Rosli Understand [/]
Matric Number: 2310366 Agree [/]
Contribution: Abstract, (Exp 2 - Materials and Equipments, Experimental Setup, Methodology, Results), Appendices

Signature: Read [/]
Name: Aizzul Luqman bin Khairul Anuar Understand [/]
Matric Number: 2311127 Agree [/]
Contribution: (Exp 1 - Materials and Equipments, Experimental Setup, Methodology), (Exp 2 - Data collection, Data Analysis), Recommendations