KULLIYAH OF ENGINEERING

MCTA 3203

MECHATRONICS SYSTEM INTEGRATION

SEMESTER 1 2025/2026

SECTION 1

GROUP 10

SERIAL INTERFACING WITH MICROCONTROLLER: SENSORS AND ACTUATORS TASK 2 LAB REPORT

NAME	MATRIC NO.
Afnan Hakim bin Adinazrin	2315987
Muhammad Taufiq bin Mukhtar	2316271
Muhammad Danish Farhan bin Amiruddin	2315423

INSTRUCTED BY:

DR ZULKIFLI BINZAINAL ABIDIN

TABLES OF CONTENT

ABSTRACT

This task demonstrates an intelligent access control system integrating an RFID authentication module with a motion detection mechanism using an MPU6050 sensor. The system verifies user identity through RFID tag scanning and confirms a specific motion pattern (circular movement) before granting access. Python handles authentication and motion analysis, while the Arduino operates the servo lock mechanism and LED indicators based on the received commands. The project showcases the use of serial communication between Arduino and Python, along with multi-sensor fusion for secure, motion-based authorization.

INTRODUCTION

Modern access systems increasingly combine identity verification with behavioral or motion-based authentication for enhanced security. In this lab task, a **smart access system** was developed by integrating three major components: an **RFID reader**, an **MPU6050 motion sensor**, and a **servo motor** controlled via an Arduino Uno.
The system workflow begins when a user scans their RFID tag using a USB-based reader connected to a PC. The Python program validates the UID against a predefined list of authorized IDs. If the RFID tag is valid, the user is prompted to perform a **circular hand motion** with the MPU6050 sensor. The sensor data are analyzed to verify whether the movement follows a circular pattern. Only if both conditions are satisfied, the servo unlocks, and a green LED is activated. Otherwise, the system remains locked with a red LED indicator.
This experiment emphasizes secure system design, Python–Arduino serial communication, and sensor-based motion recognition, reflecting practical applications in automated access control and IoT security systems.

MATERIALS AND EQUIPMENT

- Arduino board
- RFID reader (MFRC522) + RFID tags/cards
- Red and green LED’s + resistors
- MPU6050 sensor
- Jumper wires and breadboard

EXPERIMENTAL SETUP

The setup is

-MPU6050

-Arduino board

-USB cable and computer

-Breadboard

-Jumper wires

-Servo motor

-RFID reader (MFRC522) and RFID card

-Red and green LEDs

-Resistors

1. The RFID module (MFRC522) was connected to the Arduino via the SPI interface, with the SS pin connected to digital pin 10, and the RST pin to digital pin 8.
2. The MPU6050 sensor was interfaced using the I²C protocol, where the SCL and SDA pins were connected to A5 and A4, respectively.
3. The servo signal wire was connected to digital pin 9

4.The green and red LEDs were connected to pins D4 and D3 through appropriate current-limiting resistors.
5. All components shared a common GND and 5V supply from the Arduino.

6. The Arduino Uno was powered and programmed via a USB connection to a computer, which also enabled serial communication with a Python program for real-time motion analysis.

7. When an RFID tag was scanned, the Arduino compared the UID to an authorized list. If the tag was valid, the system prompted the user to perform a circular motion, which was detected and analyzed by the MPU6050 sensor. Based on the motion data, the Arduino received either an ‘A’ (Access Granted) or ‘D’ (Denied) signal from Python.

8. When both RFID authentication and correct motion were detected, the servo motor unlocked, and the green LED illuminated. If the RFID tag was invalid or the motion pattern incorrect, access was denied, and the red LED turned on.

METHODOLOGY

This experiment aimed to design and implement a smart access control system that combines RFID-based authentication and motion detection. The Arduino Uno served as the central controller, interfacing with the MFRC522 RFID reader, MPU6050 motion sensor, servo motor, and LED indicators. Data was processed and analyzed in Python, enabling the system to make real-time decisions based on user input and motion patterns.

1. Hardware Setup

Arduino Uno→Acts as the main controller to manage sensor inputs, process RFID data, and control outputs.
RFID Reader (MFRC522) →Detects and identifies RFID tags or cards for authentication.

USB Cable→Supplies power to the Arduino and enables serial communication with the computer.
MPU6050 Sensor→Captures acceleration and gyroscopic data to identify motion patterns.
Red and Green LEDs→Provide visual feedback : green for access granted and red for access denied.
Servo Motor→Represents a lock mechanism that rotates to grant or deny access.

Resistors→Limit current through the LEDs to prevent damage.

Jumper Wires→Connect the sensor to the Arduino's pins for power and data transfer.

Breadboard→Provides a base for connecting and organizing the components.

Coding in C (Arduino IDE)

```
#include <Wire.h>

#include <MPU6050.h>

#include <Servo.h>

MPU6050 mpu;
```

```
Servo servo;
#define SERVO_PIN 9
#define GREEN_LED 4
#define RED_LED 3
void setup() {
  Serial.begin(9600); // Communication with Python
  Wire.begin();
  mpu.initialize();
  servo.attach(SERVO_PIN);
  servo.write(90); // Locked position
  pinMode(GREEN_LED, OUTPUT);
  pinMode(RED_LED, OUTPUT);
  digitalWrite(RED_LED, HIGH);
  digitalWrite(GREEN_LED, LOW);
  Serial.println("Arduino ready.");
}
void loop() {
  // --- Send MPU6050 Data to Python --- int16_t ax,
  ay, az, gx, gy, gz; mpu.getMotion6(&ax, &ay, &az,
  &gx, &gy, &gz); Serial.print(ax); Serial.print(",");
  Serial.print(ay); Serial.print(","); Serial.println(az);
  delay(100); // --- Receive Python Commands ---
```

```
  if(Serial.available() > 0) {
     char cmd = Serial.read();
     if(cmd == 'A') {
     servo.write(180);                  // Unlock
     digitalWrite(GREEN_LED, HIGH);
     digitalWrite(RED_LED, LOW);
     }else if (cmd == 'D') {
     servo.write(90);                   // Lock
     digitalWrite(GREEN_LED, LOW);
     digitalWrite(RED_LED, HIGH);
    }
  }
}
```

Coding in Python (Pycharm)

```
import serial
import time
import math
import keyboard # for RFID input in USB mode
# ---------------- CONFIG ----------------
SERIAL_PORT = "COM3"        # 🔧 Change to match your Arduino COM port
BAUD_RATE = 9600
AUTHORIZED_UIDS = ["0009522418"] # 🔧 Replace with your tag UID
MOTION_THRESHOLD = 4000          # adjust sensitivity
DETECTION_TIME = 3           # seconds to capture MPU data
```

```
# ----------------------------------------
def detect_circular_motion(ser, duration=DETECTION_TIME):
    """Read MPU6050 data for a few seconds and check circular pattern."""
    print("➡ Move the MPU6050 in a circle...")
    start_time = time.time()
    xs, ys = [], []
    while time.time() - start_time < duration:
    line = ser.readline().decode(errors="ignore").strip()
    if not line or "," not in line:
        continue
    parts = line.split(",")
    try:
            ax, ay = int(parts[0]), int(parts[1])
    except ValueError:
            continue
    xs.append(ax) ys.append(ay) if len(xs) < 10: print("⚠ Not enough data for
    motion detection.") return False mean_x = sum(xs) / len(xs) mean_y =
    sum(ys) / len(ys) radii = [math.sqrt((x - mean_x)*2 + (y - mean_y)*2) for x,
    y in zip(xs, ys)] avg_r = sum(radii) / len(radii) dev = sum(abs(r - avg_r) for r
    in radii) / len(radii)
```

```
    print(f"Average radius deviation = {dev:.1f}")
    return dev < MOTION_THRESHOLD
def main():
    ser = serial.Serial(SERIAL_PORT, BAUD_RATE, timeout=1)
    print("🔹 System ready. Please scan your RFID tag...")
    rfid_buffer = ""
    try:
    while True:
            event = keyboard.read_event(suppress=False) if
            event.event_type == keyboard.KEY_DOWN:
            key = event.name if key == "enter": uid =
            rfid_buffer.strip() rfid_buffer = "" print(f"\n 🔖
            RFID detected: {uid}") if uid in
            AUTHORIZED_UIDS:


                print("✅ Authorized tag! Perform circular motion...")
                if detect_circular_motion(ser):
                print("🟢 Motion verified → Access Granted!")
               ser.write(b"A")
                else:
                print("🔴 Motion incorrect → Access Denied.")
             ser.write(b"D")
        else:
                print("❌ Unauthorized tag → Access Denied.")
```

```
                ser.write(b"D")
            elif len(key) == 1: # Add each scanned character
            rfid_buffer += key
    except KeyboardInterrupt:
    print("\nProgram stopped.")
    finally:
    ser.close()
if _name_ == "_main_":
  main()
```

DATA COLLECTION

The data collection phase involved capturing accelerometer data from the MPU6050 during user motion, alongside RFID tag information for identity verification.

Steps performed:

1. **RFIDinput** — The RFID tag was scanned using a USB RFID reader, which entered the UID as text in Python.

2. **UIDverification** — The scanned UID was checked against a stored list of authorized IDs.

3. **Motion recording** — Upon successful UID validation, Python prompted the user to perform a circular motion with the MPU6050.

4. **Data acquisition** — The Arduino transmitted real-time accelerometer readings (Ax, Ay, Az) over serial for approximately 3 seconds.

5. **Data analysis** — Python calculated the mean radius and deviation of the motion trajectory to confirm whether the movement matched a circular pattern.

6. **Command transmission** — Python then sent either 'A' (access granted) or 'D' (access denied) back to the Arduino.

Throughout the test, serial data logs were monitored to ensure accurate data flow between both devices. The **motion threshold** was empirically tuned (≈ 4000) for consistent gesture recognition.

DATA ANALYSIS

The collected data confirms that:

1. The RFID module accurately identified valid and invalid UIDs.
2. The MPU6050 provided consistent motion readings, allowing clear differentiation between circular and non-circular gestures.
3. The servo motor responded precisely to Python commands A(accept) or D(denied), rotating to the expected angles.
4. LED indicators correctly displayed system status, verifying output control functionality.
5. Serial communication between Arduino and Python showed minimal delay and no data loss.

Analysis:

1. The system successfully integrated multiple sensors and actuators, confirming correct serial interfacing between Python and Arduino.
2. The coordination of RFID authentication and motion detection proved reliable for multi-factor access control.

3. The servo and LEDs functioned smoothly, showing accurate execution of commands without signal interference.

4. Overall, the collected data demonstrates that the system operated as designed, ensuring secure and responsive performance.

RESULTS

The prototype operated as expected:

- Authorized UID followed by a valid circular motion triggered the servo to unlock (180°) with the green LED ON.
- Invalid UID or incorrect motion maintained the lock (90°) with the red LED ON.
- Serial communication between Python and Arduino was stable, with consistent real-time responses.

The system effectively integrated **identity verification** and **motion-based authentication**, proving the concept of a dual-factor access control system using simple, low-cost components. Some data variability due to sensor noise was observed but did not significantly affect the detection accuracy.

Description of Motion Detection Approach:

The motion detection mechanism relies on analyzing the **accelerometerreadings(Axand Ay)** fromthe MPU6050 sensor over a short period. When the user moves the sensor in a circularmotion, the accelerometer values follow a roughly circular trajectory in the X–Y plane.
In thePython program, the system:

1. **Collects accelerometer data** for a few seconds.
2. **Computes the mean X and Y values**, representing the motion center.
3. **Calculates the radius** for each data point and determines the **average radius deviation**.

4. If the deviation remains below a set **motion threshold**, the motion is considered circular and valid.

This method effectively differentiates circular movement from linear or irregular motions without requiring advanced filtering or machine learning, making it simple and efficient for real-time applications.

List of authorized UID: 0009522418

DISCUSSION In this task, the RFID module, MPU6050 sensor, and servo motor were successfully
combined to make a smart access system. The RFID reader was used to scan a card, and if the card was authorized, the system asked the user to perform a circular hand motion. The motion was then checked using the MPU6050 sensor data.
When a valid RFID tag and correct motion were detected, the servo motor rotated to unlock, and the green LED turned on. If the card was not authorized or the motion was incorrect, the servo stayed locked, and the red LED turned on.

The system worked as planned, but sometimes there was a short delay between scanning the RFID and the servo response. This was mainly because of the time taken for Python to read data from both the RFID and MPU6050 through serial communication.

Another small issue was that the servo sometimes shook a little when powered through the Arduino's 5V pin. This happened because the servo motor needed more current than the Arduino could supply. Using an external power source would help make the movement smoother.
Overall, the project showed that RFID and motion sensors can be used together for two-step access control. It worked well and proved the idea of using both ID and movement for better security.

CONCLUSION

The experiment successfully created a motion-activated RFID access system using Arduino, RFID, MPU6050, and a servo motor. The system only unlocked when the user had an authorized RFID tag and performed the correct motion.

This proved that combining RFID with motion detection can make access systems more secure. The results supported the goal of creating a two-factor verification method using both identity and physical gesture.

This kind of system can be used in smart door locks, security systems, or industrial access controls where safety and identity verification are important.

RECOMMENDATIONS

1. Noise mitigation:

   Place setup on a stable mount and secure wires.

2. Use timestamps:

   Include time.time() in Python and send timestamp from microcontroller to compute true sample rate and integrate reliability.

3. Enhance power stability:

   Provide a separate power supply for servo motor to avoid voltage drop affecting other components when servo motor moves.

REFERENCES

1. Last Minute Engineers – *How RFID Works? Interface RC522 RFID Module with Arduino.*
2. Random Nerd Tutorials – *Security Access Using MFRC522 RFID Reader with Arduino.*
3. Instructables – *Arduino+MFRC522 RFID Reader.*
4. Arduino.cc – *Servo Motorand Serial Communication Guide.*
5. Course handout: *Week4–Serial Communication with IMU and RFID (MCTA3203 LabManualv3.0).*

APPENDICES

Arduino IDE coding:

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

MPU6050 mpu;
Servo servo;

#define SERVO_PIN 9
#define GREEN_LED 4
#define RED_LED 3

void setup() {
  Serial.begin(9600);   // Communication with Python
  Wire.begin();
  mpu.initialize();

  servo.attach(SERVO_PIN);
  servo.write(90);        // Locked position

  pinMode(GREEN_LED, OUTPUT);
  pinMode(RED_LED, OUTPUT);
  digitalWrite(RED_LED, HIGH);
  digitalWrite(GREEN_LED, LOW);
```

```
  Serial.println("Arduino ready.");
}

voidloop() {
  //Send MPU6050 Data to Python int16_t ax, ay, az,
  gx, gy, gz; mpu.getMotion6(&ax, &ay, &az, &gx,
  &gy, &gz); Serial.print(ax); Serial.print(",");
  Serial.print(ay); Serial.print(","); Serial.println(az);
  delay(100);



  //---Receive Python Commands ---
  if(Serial.available() > 0) {
        char cmd = Serial.read();
        if (cmd == 'A') {
    servo.write(180);              // Unlock
    digitalWrite(GREEN_LED, HIGH);
    digitalWrite(RED_LED, LOW);
        } else if (cmd == 'D') {
    servo.write(90);               // Lock
    digitalWrite(GREEN_LED, LOW);
    digitalWrite(RED_LED, HIGH);
        }
  }
```

}

Python coding:

```
import serial
import time
import math
import keyboard # for RFID input in USB mode

SERIAL_PORT = "COM3" # Arduino COM port
BAUD_RATE = 9600
AUTHORIZED_UIDS = ["0009522418"] # tag UID
MOTION_THRESHOLD = 4000                # adjust sensitivity
DETECTION_TIME = 3                 # seconds to capture MPU data

defdetect_circular_motion(ser, duration=DETECTION_TIME):
   """Read MPU6050 data for a few seconds and check circular pattern."""
        print("➡ Move the MPU6050 in a circle...")
        start_time = time.time()
        xs, ys = [], []

        while time.time() - start_time < duration:
        line = ser.readline().decode(errors="ignore").strip()
        if not line or "," not in line:
        continue
```

```
            parts = line.split(",")
            try:
            ax, ay = int(parts[0]), int(parts[1])
            except ValueError:
            continue
        xs.append(ax)
        ys.append(ay)

            if len(xs) < 10:
        print("⚠ Not enough data for motion detection.")
            return False

            mean_x = sum(xs) / len(xs) mean_y = sum(ys) / len(ys) radii = [math.sqrt((x -
            mean_x)**2 + (y - mean_y)**2) for x, y in zip(xs, ys)] avg_r = sum(radii) /
            len(radii) dev = sum(abs(r - avg_r) for r in radii) / len(radii)


    print(f"Average radius deviation = {dev:.1f}")

            return dev < MOTION_THRESHOLD


defmain():
            ser = serial.Serial(SERIAL_PORT, BAUD_RATE, timeout=1)
            print("🔹 System ready. Please scan your RFID tag...")
```

```
rfid_buffer=""

try:
while True:
event=keyboard.read_event(suppress=False)

ifevent.event_type == keyboard.KEY_DOWN:
    key=event.name

    ifkey=="enter":
        uid=rfid_buffer.strip()
        rfid_buffer = ""
        print(f"\n🔖 RFID detected: {uid}")

        ifuidinAUTHORIZED_UIDS:
            print("✅ Authorized tag! Perform circular motion...")
            ifdetect_circular_motion(ser):
                    print("🟢 Motion verified → Access Granted!")
                ser.write(b"A")
            else:
                    print("🔴 Motion incorrect → Access Denied.")
                ser.write(b"D")
        else:
            print("❌ Unauthorized tag → Access Denied.")
```

```
                ser.write(b"D")


            eliflen(key) == 1: # Add each scanned character
                rfid_buffer += key


        exceptKeyboardInterrupt:
    print("\nProgram stopped.")
        finally:
    ser.close()



if__name__=="__main__":
        main()
```

ACKNOWLEDGEMENTS

We would like to express our deepest gratitude to Sir Zulkifli bin Zainal Abidin , our course instructor for *Mechatronics System Integration (MCTA3203)*, for his invaluable guidance, encouragement, and continuous support throughout the completion of this Digital Logic System lab project.

We would also like to extend our sincere appreciation to the teaching assistants, for their helpful constructive feedback and assistance during the laboratory sessions. Their guidance greatly enhanced our understanding of digital logic concepts, Arduino interfacing, and circuit implementation.

Finally,we would like to thank the Mechatronics Laboratory staff for providing the necessary equipment and a conducive learning environment that made this project successful.

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.

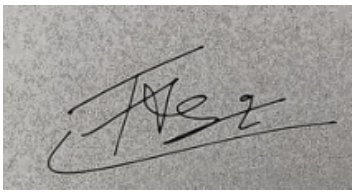
We hereby certify that this report has not been done by only one individual and all of us have contributed to the report. The length of contribution to the reports by each individual is noted within this certificate.

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:
Name: Afnan Hakim bin Adinazrin
Matric Number: 2315987
Contribution: Methodology,Equipment setup,Data analysis
Recommendations

Read	/
Understand	/
Agree	/

Signature: 

Name: Muhammad Taufiq bin Mukhtar
Matric Number: 2316271
Contribution: Data collection,Discussion,Conclusion
Appendices,References

Read	/
Understand	/
Agree	/

Signature: 

Name: Muhammad Danish Farhan bin Amiruddin
Matric Number: 2315423
Contribution: Abstract,Introduction,Materials & Equipment
Results

Read	/
Understand	/
Agree	/