LAB REPORT 2: SERIAL AND USB INTERFACING (Part 2)

GROUP 1

MCTA 3203

SEMESTER 1 2024/2025

MECHATRONICS SYSTEM INTEGRATION

NO	NAME	MATRIC NO
1.	ABDUL A'LIM BIN MOHD RAJIB	2119687
2.	AHMAD HAZAMI BIN MOHD RAZIP	2211203
3.	ABDUL HADI BIN ZAWAWI	2210739
4.	AIN MAISARA BINTI ABDULLAH	2217856
5.	ADIBAH BINTI MOHD AZILAN	2212670

DATE OF SUBMISSION: 6/11/2024

ABSTRACT

This project integrates gesture recognition and RFID-based access control with an Arduino, MPU6050 sensor, servo motor, and RFID reader. The MPU6050 IMU collects accelerometer and gyroscope data to detect and classify hand gestures, which are then processed on a computer using Python over serial communication. The system identifies predefined gestures by analyzing motion data patterns and thresholds. The USB-connected RFID reader authenticates users by scanning RFID tags, with authorized tags causing the servo motor to adjust position, and LEDs indicating access status (green for authorized and red for unauthorized tags). JSON data handling increases flexibility by enabling configuration and customization, such as adjusting the servo angle. This setup successfully integrates real-time sensor data processing, gesture recognition, and RFID authentication in a mechatronic system.

TABLE OF CONTENTS

1. INTRODUCTION

The goal of this experiment is to create a dual-function mechatronic system that integrates gesture recognition with RFID-based access management. The system uses an MPU6050 Inertial Measurement Unit (IMU) to collect accelerometer and gyroscope data, allowing for basic hand gesture recognition, and an RFID reader to authenticate users using RFID tags. Integrating all functionalities into a single system demonstrates the ability of embedded systems to combine different interactive and security-related applications.

Gesture recognition is accomplished by analyzing motion data captured by the MPU6050 sensor and processed using an Arduino microcontroller and a linked PC running Python. This data enables the classification of predetermined motions using certain patterns and thresholds. An RFID reader scans tags to authorize users, causing servo motor movement and LEDs to indicate status: green for authorized access and red for denial. JSON data handling is incorporated to enable for more flexible configuration, such as threshold and servo positioning adjustments.

The system is expected to detect predefined gestures accurately and control access reliably using RFID authentication. The expected objectives include successful gesture classification and accurate, real-time RFID-based access control, proving the feasibility and versatility of integrating sensors, actuators, and security functions into embedded systems.

PART A : DISPLAYING ACCELEROMETER AND GYROSCOPE DATA

1. MATERIALS AND EQUIPMENT

- 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 to connect the Arduino board to your personal computer. This will be used for uploading the Arduino code and serial communication.
- Power supply: If your Arduino board and MPU6050 require an external power source, make sure to have the appropriate power supply.
- LEDs of different colours.

2. EXPERIMENTAL SETUP

a. Connect the MPU6050 sensor to the Arduino board using the appropriate pins. The MPU6050 typically uses I2C communication, so connect the SDA and SCL pins of the MPU6050 to the corresponding pins on the Arduino (usually A4 and A5 for most Arduino boards).

b. Connect the power supply and ground of the MPU6050 to the Arduino's 5V and GND pins.

c. Ensure that the Arduino board is connected to your PC via USB.

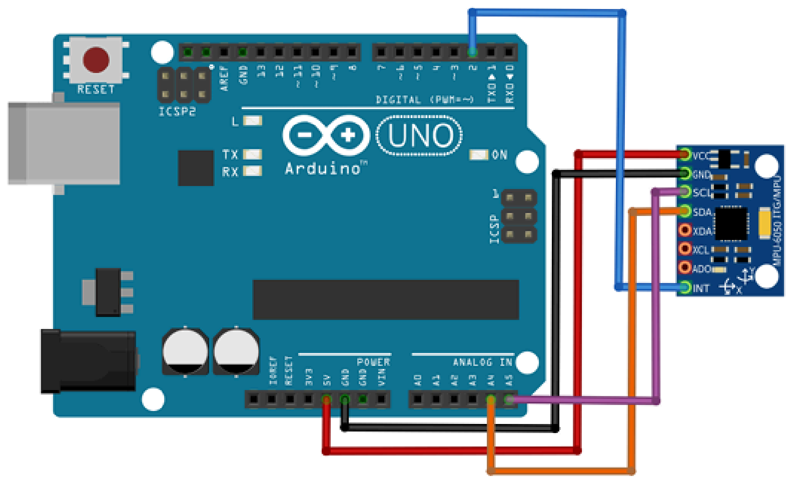


Fig. 1: Arduino-MPU6050 Connections

3. METHODOLOGY

PROCEDURES

a. Connect the Arduino to the computer via a USB cable.

b. Open the Arduino IDE and write code to read data from the MPU6050 and send it to the PC via the serial port:

c. Upload the code to the Arduino. This will send accelerometer and gyroscope data from the MPU6050 sensor to the PC via serial communication.

d. Create a Python script to read data from the Arduino and print it on your PC's console. Save this code as a .py file:

e. With the Arduino code uploaded and the Python script running, you should see accelerometer and gyroscope data from the MPU6050 sensor printed to your PC's console.

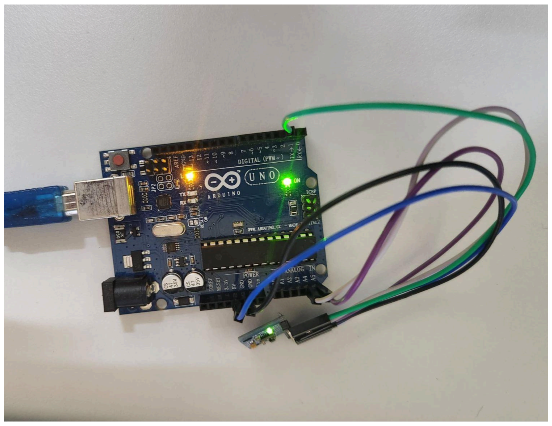
Arduino code:

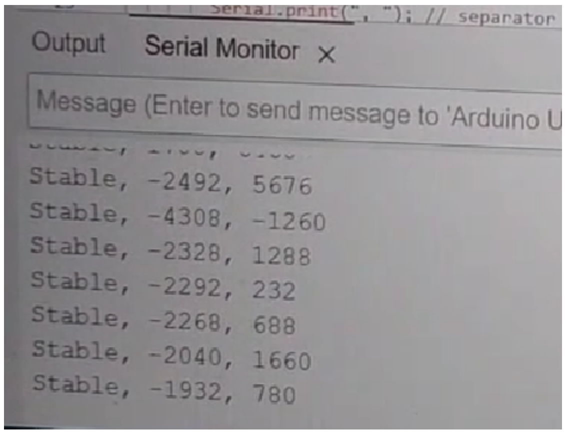
```
#include <Wire.h>
#include <MPU6050.h>

MPU6050 mpu;

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

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);
  Serial.print("  Gyro: ");
  Serial.print(gx);
  Serial.print(", ");
  Serial.print(gy);
  Serial.print(", ");
  Serial.println(gz);

  delay(100); // Adjust the delay as needed
}
```

Python code:

```
import serial

ser = serial.Serial('COM4', 9600)

while True:
    data = ser.readline().decode('utf-8').strip()
    print(data)
```

4. RESULTS





```
Serial.print(", "); // separator
```

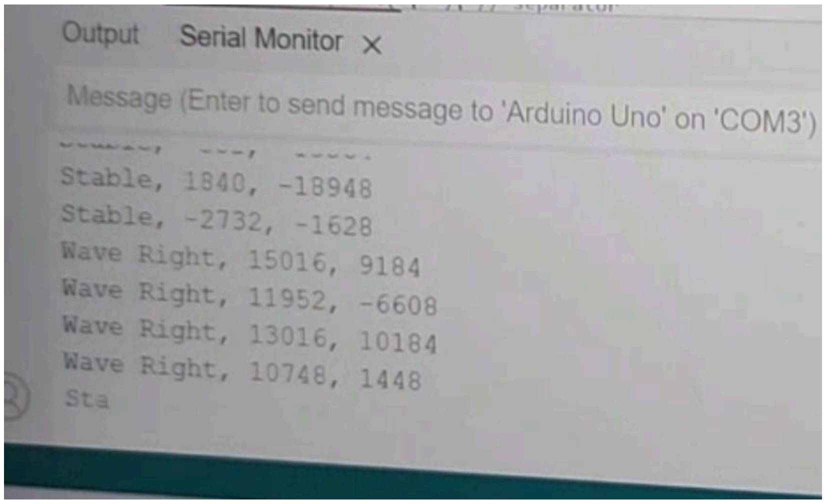
```
Stable, -2492, 5676
Stable, -4308, -1260
Stable, -2328, 1288
Stable, -2292, 232
Stable, -2268, 688
Stable, -2040, 1660
Stable, -1932, 780
```

5. DISCUSSION

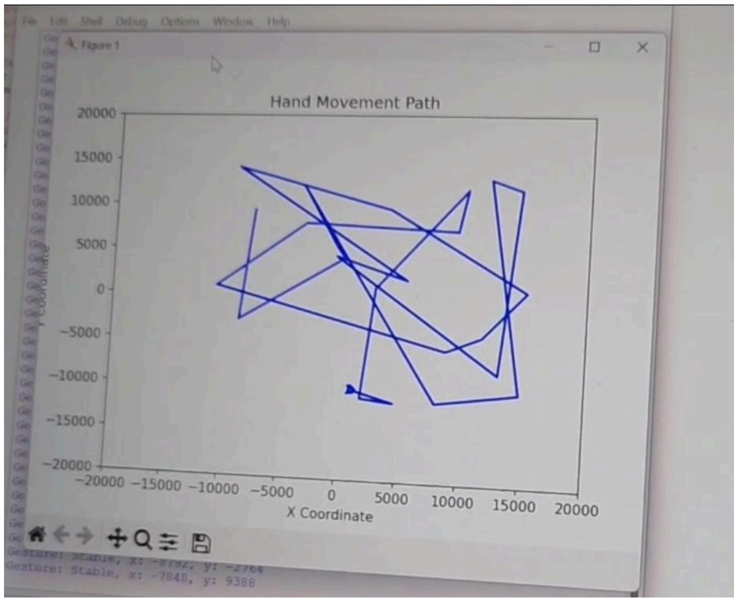
This experiment successfully established a connection between the MPU6050 sensor with an Arduino board, and a personal computer by enabling the collection of accelerometer and gyroscope data for hand gesture recognition. The sensor is capable of readings during predefined hand movements, the system was able to categorize these gestures using an algorithm. Visualizing the hand movement paths in an x-y coordinate system provided valuable insights into the accuracy and responsiveness of the gesture recognition process. However, a challenge faced during the experiment was dealing with sensor noise and drift, which affected the precision of the gesture recognition and path visualization. To mitigate this, further calibration and tuning are required for more accurate results. Despite this issue, the integration of the MPU6050 with Arduino demonstrated a practical approach to motion tracking and gesture-based control, with potential applications in interactive systems and human-computer interfaces.

Question

Create a straightforward hand gesture recognition system by capturing accelerometer and gyroscope data during the execution of predefined hand movements. Employ an algorithm to identify and categorize these gestures using the collected sensor data. Additionally, visualize the paths of hand movement in an x-y coordinate system.



```
Stable, 1840, -18948
Stable, -2732, -1628
Wave Right, 15016, 9184
Wave Right, 11952, -6608
Wave Right, 13016, 10184
Wave Right, 10748, 1448
Sta
```



```
Gesture: Stable, x: -8792, y: -2764
Gesture: Stable, x: -7848, y: 9388
```

6. CONCLUSION

The integration of the Arduino and sensor successfully facilitated the display of accelerometer and gyroscope data. By using serial communication, the Arduino transmitted real-time sensor data to a Python interface, which allowed for visualizing motion data and enhancing gesture recognition capabilities. This setup demonstrated a reliable method for capturing and interpreting hand gestures, proving the practicality of combining sensors and software for motion detection in embedded systems.

PART B : RFID-BASED AUTHENTICATION AND SERVO MOTOR CONTROL USING ARDUINO AND PYTHON

1. MATERIALS AND EQUIPMENT

- 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
- LEDs of various colors
- USB cables to connect the Arduino board and the RFID reader to your computer.
- Computer with Arduino IDE and Python installed
- Datasheets and Manuals: Make sure you have the datasheets or manuals for the RFID reader, servo motor, and any other components you are using. Most of them can be downloaded from the internet. Before starting the experiment, carefully read the documentation for each component and understand the electrical and mechanical requirements. Also, consider safety protocols and guidelines to ensure a safe working environment in the lab.
- Power Supply (optional): If the servo motor requires a power supply other than what the Arduino can provide, you'll need the appropriate power supply.
- Mounting Hardware (for the servo): If you want to mount the servo in a specific orientation or location, you might need screws, brackets, or other mounting hardware.

2. EXPERIMENTAL SETUP

a. Servo Motor Wiring:

b. Connect the servo's power wire (usually red) to the 5V output on the Arduino.

c. Connect the servo's ground wire (usually brown or black) to one of the ground (GND) pins on the Arduino.

d. Connect the servo's signal wire (usually orange or yellow) to one of the PWM pins on the Arduino (e.g., pin 9).

e. Ensure that you have a common ground connection between the Arduino and the servo motor to complete the circuit.

f. As for the USB RFID reader, it's usually powered via the USB connection, and you don't need any additional wiring for power. The communication with the RFID reader is handled through the USB cable.

3. METHODOLOGY

PROCEDURES

a. Connect the servo motor to the Arduino.

b. Connect the RFID card reader to the computer via USB.

c. Ensure a common ground connection between the Arduino and the servo motor.

d. Open the Arduino IDE and upload the code

e. Install the pyusb library

f. Write a Python script for RFID authentication and to control the servo. Save it as a .py file.

g. Run the Python script, and when you present an RFID card, the card's UID will be checked

h. If authorized, the Python script sends a signal to the Arduino to move the servo; if not, the servo returns to its default position.

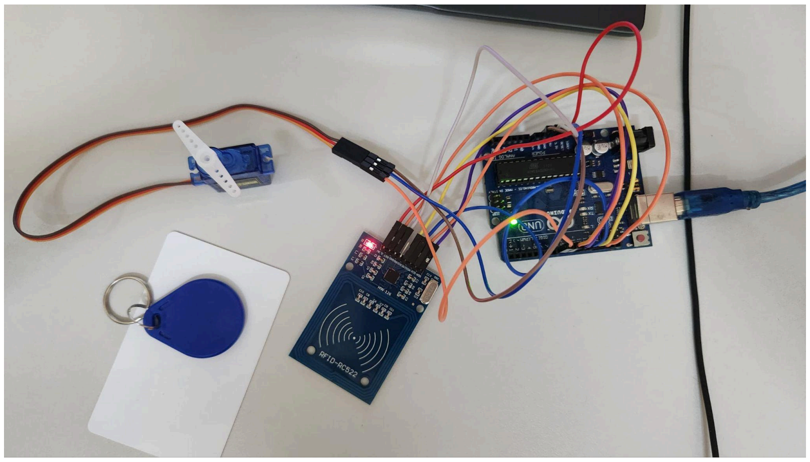
Arduino code:

```
#include <Servo.h>

Servo servo;
int servoPosition = 90;  // Initial servo position

void setup() {
  servo.attach(9);  // Attach the servo to pin 9
  servo.write(servoPosition);
  Serial.begin(9600);
}

void loop() {
  // Listen for signals from Python
  if (Serial.available() > 0) {
    char command = Serial.read();
    if (command == 'A') {
      // Allow servo control
      servoPosition = 180;  // Set the servo to a specific angle
      servo.write(servoPosition);
    } else if (command == 'D') {
      // Disallow servo control
      servoPosition = 90;  // Set the servo back to the default angle
      servo.write(servoPosition);
    }
  }
}
```

Python code:

```
import usb.core
import usb.util

# Define your RFID reader's vendor and product IDs
vendor_id = 0x1234  # Replace with your RFID reader's vendor ID
product_id = 0x5678  # Replace with your RFID reader's product ID

# Authorized card IDs
authorized_cards = ["YourCardID1", "YourCardID2"]

# Initialize the USB RFID reader
dev = usb.core.find(idVendor=vendor_id, idProduct=product_id)
if dev is None:
    raise ValueError("RFID reader not found")

# Detach kernel driver if it's attached
if dev.is_kernel_driver_active(0):
    dev.detach_kernel_driver(0)

# Set the configuration of the device
dev.set_configuration()

# Define the endpoint
endpoint = dev[0][(0, 0)][0]

while True:
    try:
        data = dev.read(endpoint.bEndpointAddress,
endpoint.wMaxPacketSize)
        card_id = ''.join([chr(byte) for byte in data])

        if card_id in authorized_cards:
            print("Access granted. You can now control the servo.")
            # Send the 'A' signal to the Arduino
            ser.write(b'A')
        else:
            print("Access denied. Unauthorized card.")
            ser.write(b'D')
    except usb.core.USBError:
        pass
```

4. RESULTS

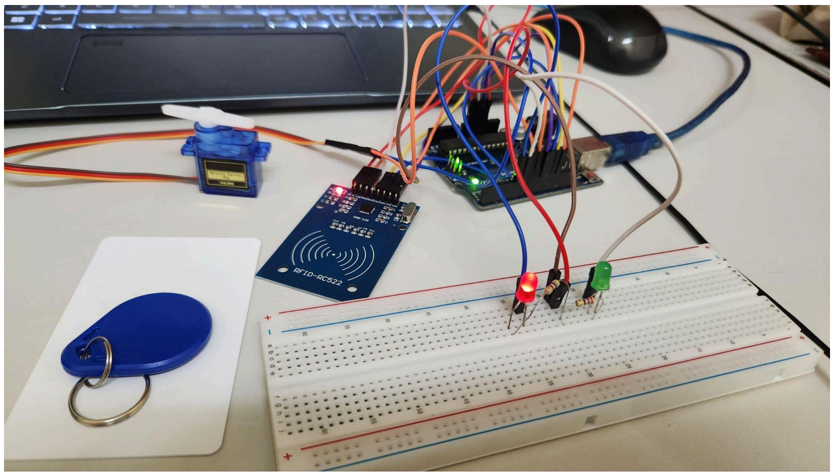


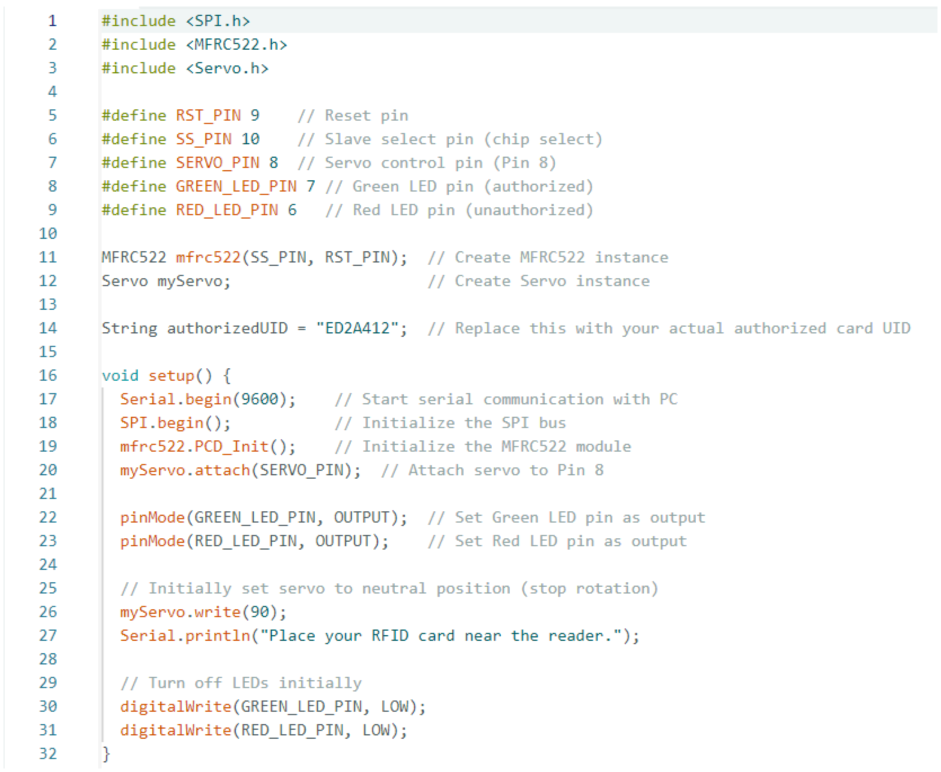
5. DISCUSSION

The experiment was successful in integrating the RFID card reader with Python and Arduino to create a functional authentication system that controls a servo motor. The RFID reader reliably transmitted the card's UID to Python, where it was authenticated against a stored list of authorized cards. Upon successful authentication, Python communicated with the Arduino via serial connection to control the servo motor's movement, as intended. Despite some initial challenges with data parsing and communication synchronization, these were successfully addressed, leading to a seamless interaction between the components. Overall, the system demonstrated effective RFID-based access control and motor automation, achieving the experiment's goals.

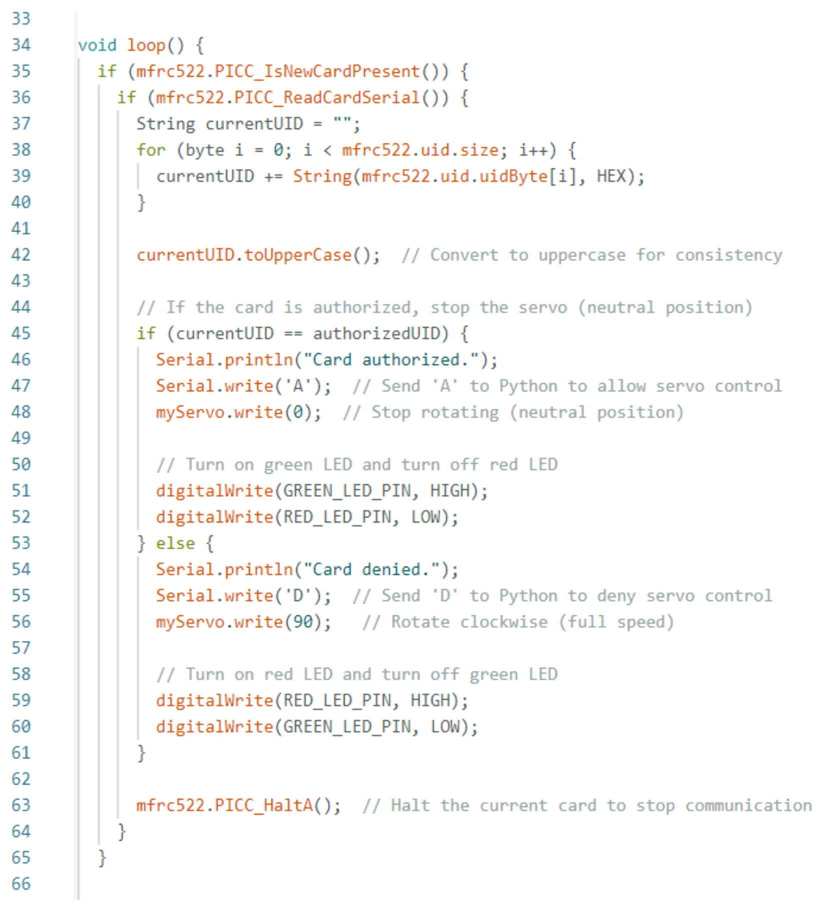
Question

Enhance the existing code to introduce a visual indicator, such as illuminating a green LED, when a recognized UID is detected by the RFID reader, and conversely, activate a *red LED when an unrecognized card is read. Incorporate structured JSON data handling within your code for better organization and flexibility. Add some options for the user to freely set the angle position of the servo.*

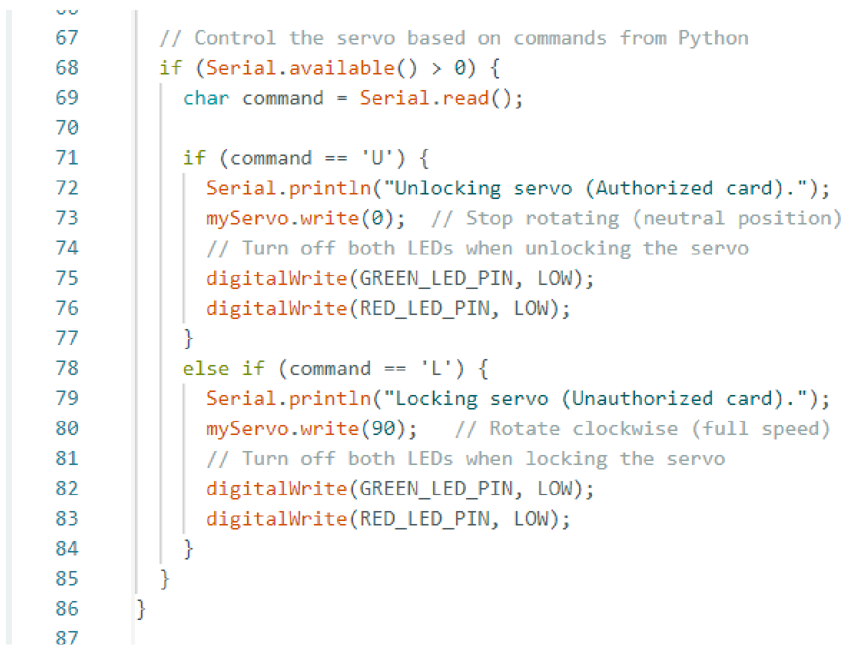




```
#include <SPI.h>
#include <MFRC522.h>
#include <Servo.h>

#define RST_PIN 9     // Reset pin
#define SS_PIN 10     // Slave select pin (chip select)
#define SERVO_PIN 8   // Servo control pin (Pin 8)
#define GREEN_LED_PIN 7 // Green LED pin (authorized)
#define RED_LED_PIN 6   // Red LED pin (unauthorized)

MFRC522 mfrc522(SS_PIN, RST_PIN);  // Create MFRC522 instance
Servo myServo;                     // Create Servo instance

String authorizedUID = "ED2A412";  // Replace this with your actual authorized card UID

void setup() {
  Serial.begin(9600);     // Start serial communication with PC
  SPI.begin();            // Initialize the SPI bus
  mfrc522.PCD_Init();     // Initialize the MFRC522 module
  myServo.attach(SERVO_PIN);  // Attach servo to Pin 8

  pinMode(GREEN_LED_PIN, OUTPUT);  // Set Green LED pin as output
  pinMode(RED_LED_PIN, OUTPUT);    // Set Red LED pin as output

  // Initially set servo to neutral position (stop rotation)
  myServo.write(90);
  Serial.println("Place your RFID card near the reader.");

  // Turn off LEDs initially
  digitalWrite(GREEN_LED_PIN, LOW);
  digitalWrite(RED_LED_PIN, LOW);
}
```



```
void loop() {
  if (mfrc522.PICC_IsNewCardPresent()) {
    if (mfrc522.PICC_ReadCardSerial()) {
      String currentUID = "";
      for (byte i = 0; i < mfrc522.uid.size; i++) {
        currentUID += String(mfrc522.uid.uidByte[i], HEX);
      }

      currentUID.toUpperCase();  // Convert to uppercase for consistency

      // If the card is authorized, stop the servo (neutral position)
      if (currentUID == authorizedUID) {
        Serial.println("Card authorized.");
        Serial.write('A');  // Send 'A' to Python to allow servo control
        mySerVo.write(0);  // Stop rotating (neutral position)

        // Turn on green LED and turn off red LED
        digitalWrite(GREEN_LED_PIN, HIGH);
        digitalWrite(RED_LED_PIN, LOW);
      } else {
        Serial.println("Card denied.");
        Serial.write('D');  // Send 'D' to Python to deny servo control
        myServo.write(90);   // Rotate clockwise (full speed)

        // Turn on red LED and turn off green LED
        digitalWrite(RED_LED_PIN, HIGH);
        digitalWrite(GREEN_LED_PIN, LOW);
      }

      mfrc522.PICC_HaltA();  // Halt the current card to stop communication
    }
  }

```



```
  // Control the servo based on commands from Python
  if (Serial.available() > 0) {
    char command = Serial.read();

    if (command == 'U') {
      Serial.println("Unlocking servo (Authorized card).");
      myServo.write(0);  // Stop rotating (neutral position)
      // Turn off both LEDs when unlocking the servo
      digitalWrite(GREEN_LED_PIN, LOW);
      digitalWrite(RED_LED_PIN, LOW);
    }
    else if (command == 'L') {
      Serial.println("Locking servo (Unauthorized card).");
      myServo.write(90);   // Rotate clockwise (full speed)
      // Turn off both LEDs when locking the servo
      digitalWrite(GREEN_LED_PIN, LOW);
      digitalWrite(RED_LED_PIN, LOW);
    }
  }
}

```

6. CONCLUSION

The implementation of an RFID-based authentication system integrated with servo motor control provided a functional and secure access control mechanism. The Arduino and Python scripts enabled RFID tag verification, with successful authentication activating the servo motor to grant access. LED indicators, controlled through structured JSON data, further enhanced the system's usability by providing visual feedback on access status. This part of the project demonstrated effective user identification and secure access management through mechatronic integration.

RECOMENDATIONS

To improve this experiment, consider using machine learning algorithms such as k-nearest neighbors (KNN) for gesture recognition, which may improve accuracy beyond simple threshold-based detection. Adding a magnetometer would also help with orientation tracking and gesture classification. Improving the Python user interface with real-time visual feedback, such as an x-y plot of gesture paths, may make data interpretation more intuitive. The application of secure, encrypted tags or NFC technology in RFID access control would improve security. Additionally, allowing custom configuration of servo angles and access settings via a JSON file would increase flexibility. Lessons learned include the importance of secure sensor connections for accurate data and careful calibration to reduce noise. Future students may benefit from reorganizing code in order to simplify debugging and modifications.

REFERENCES

1. Components101, (2021, March 17), MPU6050 Accelerometer and Gyroscope Module. Retrieved from https://components101.com/sensors/mpu6050-module

ACKNOWLEDGEMENTS

We would like to express our sincere gratitude to our lecturers, Dr. Wahju Sediono and Dr. Zulkifli bin Zainal Abidin for their invaluable guidance in this experiment. Their expertise in the mechatronics field and encouragement have been a big support in the successful integration of our system. Not forgetting our teaching assistants in the lab for helping us indirectly.

To add, we also wish to extend a special thank you to our fellow team members for their collaboration and hard work during the course of this project. Their contributions were pivotal in driving our efforts forward and enhancing the overall quality of our work.

STUDENTS' DECLARATION

<u>Certificate of Originality and Authenticity</u>

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 that no further improvement on the reports is needed from any of the individual 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: A'LIM	**Read**	/
Name: ABDUL A'LIM BIN MOHD RAJIB	**Understand**	/

Matric No: 2119687	**Agree**	/

Signature: hazami	**Read**	/
Name: AHMAD HAZAMI BIN MOHD RAZIP	**Understand**	/
Matric No: 2211203	**Agree**	/

Signature: [illegible]	**Read**	/
Name: ABDUL HADI BIN ZAWAWI	**Understand**	/
Matric No: 2210739	**Agree**	/

Signature: [illegible]	**Read**	/
Name: AIN MAISARA BT. ABDULLAH	**Understand**	/
Matric No: 2217856	**Agree**	/

Signature: [illegible]	**Read**	/
Name: ADIBAH BINTI MOHD AZILAN	**Understand**	/
Matric No: 2212670	**Agree**	/