REPORT 3: SERIAL COMMUNICATION

GROUP 2

MCTA 3203

SEMESTER 1 2024/2025

MECHATRONICS SYSTEM INTEGRATION

DATE OF SUBMISSION: 6/11/2024

	NAME	MATRIC NUMBER
1.	AKMAL ZARIF BIN NAJIB	2211971
2.	AMIR FARHAN BIN GHAFFAR	2115617
3.	AMIRAH HUDA BINTI JAMALULAIL ASRI	2210776
4.	AMIRUL AIZAD BIN AMIR	2211263
5.	ANAS BIN BADRULZAMAN	2219945

TABLE OF CONTENTS

ABSTRACT

This experiment demonstrates the use of serial and USB interfacing with an Arduino microcontroller and computer-based system. We established a connection between a computer with an IMU (Inertial Measurement Unit) MPU6050 and an RFID card reader through an Arduino microcontroller. To establish this serial communication, we wrote codes for both Python and Arduino. The Python script processes the measurement data from the MPU6050 and RFID card which was received through Arduino. It enables us to create a straightforward hand gesture recognition system by capturing accelerometer and gyroscope data during the execution of predefined hand movements. Linking an RFID card reader to the computer through a USB cable allows us to engage with it using the Python code. An experiment is conducted, where when a recognized UID is detected by the RFID reader, an LED will light up as a visual indicator and we can control the servo motor. The objective of both experiments is to establish a reliable communication channel, enabling data exchange between the two platforms.

INTRODUCTION

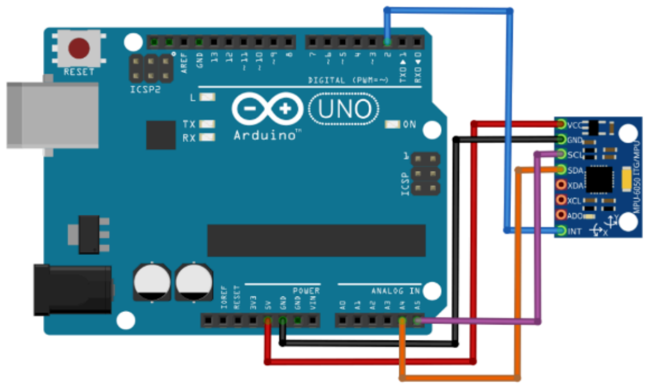
Serial communication between Python and an Arduino is one of the ways to exchange data between a computer and an Arduino microcontroller. In this experiment, we read data received from the sensors using Python. For this experiment, we divide it into two parts. For experiment 4A, we are establishing a connection between a personal computer and MPU6050. MPU6050 is a complete 6-axis Motion Tracking Device, which combines a 3-axis Gyroscope, a 3-axis Accelerometer and a Digital Motion Processor. MPU6050 has the ability to combine accelerometer and gyroscope measurements, providing a valuable source of information for a wide range of projects and devices. It is a versatile sensor for a variety of applications that require motion and orientation data. By employing an algorithm, we could identify and categorize hand gestures using the collected data from the sensors in the MPU6050. For experiment 4B, we are linking the RFID card reader to the computer through a USB cable. RFID cards contain a chip and an antenna for wireless identification of the objects they are attached to with the help of an RFID reader. The reader is a device that has one or more antennas that emit radio waves and receive signals back from the RFID card. The RFID card is placed near the reader so the card's UID will be sent from the Arduino to the Python script. From the Python script, we perform actions and decisions based on the RFID card data that we received.

EXPERIMENT 4A - <u>Real-time hand gesture recognition using MPU6050 sensor.</u>

MATERIALS AND EQUIPMENT

- Arduino Mega 2560
- MPU6050 sensor
- Jumper wires
- Breadboard
- LED

EXPERIMENTAL SETUP



METHODOLOGY

1. Purpose: To read accelerometer and gyroscope data from the MPU6050 sensor via the Arduino and send it to a computer for processing.

2. Software Setup: Write Arduino code to interface with the MPU6050 sensor using I2C communication protocol, and a Python script to receive and display sensor data on a computer.

3. Data Processing: Data is processed in real-time to recognize specific hand gestures.

PROCEDURE

1. Hardware Setup:

- Connect the MPU6050 sensor to the Arduino using I2C pins.
- Link an LED as a visual indicator.

2. Arduino Coding:

- Program the Arduino to capture motion data and transmit it serially.

3. Python Script:

- Develop a script to receive serial data, decode it, and display it for further analysis.

4. Testing:

- Observe LED indicators or other feedback mechanisms to confirm data transmission.

RESULTS

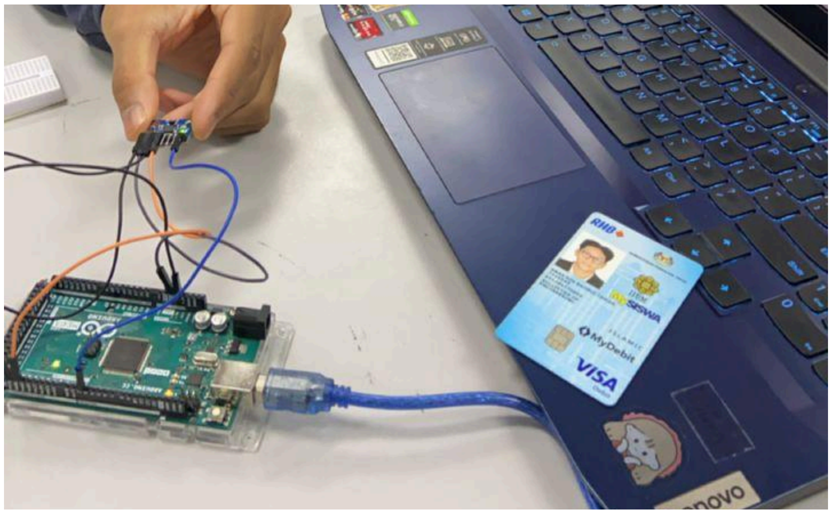
After establishing the serial connection between the MPU6050 sensor and the Arduino, followed by data transmission to a computer, we observed the following outcomes:

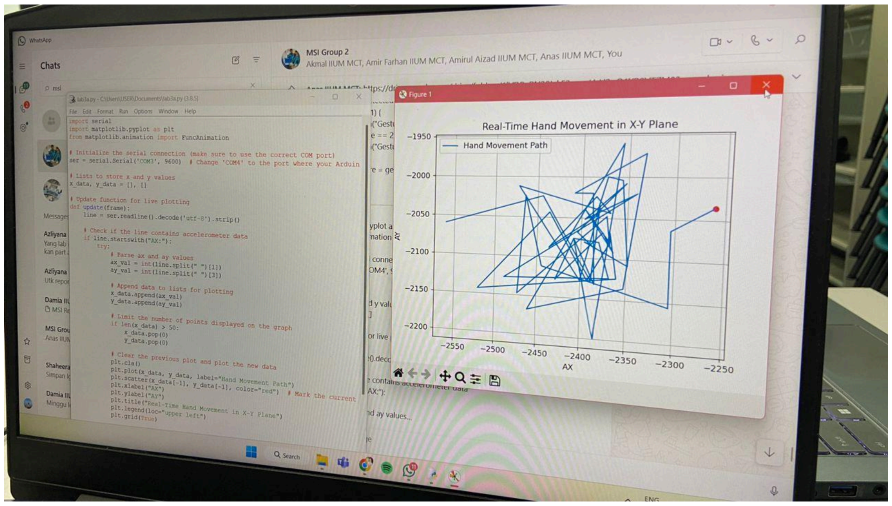
1. **Data Collection**: The MPU6050 successfully transmitted accelerometer and gyroscope data to the Arduino. This data included values for motion along the X, Y, and Z axes.

2. **Real-time Display**: The Python script on the computer received and displayed real-time motion data from the Arduino. Data updates were consistent and displayed without noticeable lag.

3. **Gesture Recognition**: Specific gestures, such as tilting the sensor to the right and left, were successfully detected and distinguished based on pre-set thresholds in the Python script.

4. **Indicator Feedback**: The LED connected to the Arduino lit up in response to recognized gestures, confirming that the serial communication was reliable and **responsive.**




lab3a.py - C:\Users\USER\Documents\lab3a.py (3.8.5)
File Edit Format Run Options Window Help
Figure 1
Real-Time Hand Movement in X-Y Plane
Hand Movement Path
AX
AY
−1950
−2000
−2050
−2100
−2150
−2200
−2550
−2500
−2450
−2400
−2350
−2300
−2250

```
import serial
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

# Initialize the serial connection (make sure to use the correct COM port)
ser = serial.Serial('COM3', 9600)  # Change 'COM4' to the port where your Arduin

# Lists to store x and y values
x_data, y_data = [], []

# Update function for live plotting
def update(frame):
    line = ser.readline().decode('utf-8').strip()

    # Check if the line contains accelerometer data
    if line.startswith("AX:"):
        try:
            # Parse ax and ay values
            ax_val = int(line.split(" ")[1])
            ay_val = int(line.split(" ")[3])

            # Append data to lists for plotting
            x_data.append(ax_val)
            y_data.append(ay_val)

            # Limit the number of points displayed on the graph
            if len(x_data) > 50:
                x_data.pop(0)
                y_data.pop(0)

            # Clear the previous plot and plot the new data
            plt.cla()
            plt.plot(x_data, y_data, label="Hand Movement Path")
            plt.scatter(x_data[-1], y_data[-1], color="red")  # Mark the current
            plt.xlabel("AX")
            plt.ylabel("AY")
            plt.title("Real-Time Hand Movement in X-Y Plane")
            plt.legend(loc="upper left")
            plt.grid(True)
```

DISCUSSION

This experiment highlights the capabilities and limitations of the MPU6050 sensor and the effectiveness of Arduino-Python serial communication:

1. Effectiveness of Data Transfer:

The serial connection allowed smooth, real-time data transfer between the Arduino and the computer, demonstrating the effectiveness of using I2C for sensor communication.

The set baud rate of 9600 was appropriate, balancing data transmission speed and stability, essential for real-time applications.

2. Gesture Recognition Accuracy:

The system could reliably detect predefined gestures (left and right tilts). The thresholds for recognizing gestures were well-calibrated, allowing accurate and repeatable gesture detection.

This initial success suggests the MPU6050 sensor's potential for interactive applications, like motion-based control systems, where fast and reliable response to gestures is crucial.

3. Potential Improvements:

Data Filtering: Adding filtering to the accelerometer and gyroscope data could improve the gesture recognition's accuracy by minimizing noise.

Threshold Adjustment: Further calibration of thresholds could improve detection accuracy, particularly in applications where the sensor might encounter varied motion patterns.

Expansion of Gesture Library: Defining additional gestures would broaden the system's applicability. Integrating more complex gesture recognition algorithms could enable a richer set of responses.

4. Real-World Applications:

The MPU6050 sensor and this communication setup can be applied in projects requiring continuous motion tracking, such as wearable technology, sports analytics, or robotics.

With further refinements, this system could serve as the basis for more advanced applications that require precise and robust gesture recognition.

CONCLUSION

This experiment successfully demonstrated how to set up communication between MPU6050 sensor, Arduino board and PC to capture and transmit accelerometer and gyroscope data. Using I2C communication, the MPU6050 can interface with the Arduino, allowing the Arduino to read and send motion data via a serial connection to the PC. The Arduino code provides a steady stream of sensor readings at a baud rate of 9600, which are then received and displayed on the PC using a simple Python script. This setup confirms the functionality of the MPU6050 sensor in capturing motion data and the ease with which this data can be transmitted and displayed on external devices, paving the way for a variety of real-time monitoring and data analysis applications.

This setup demonstrates the MPU6050's potential for integration into larger projects that require continuous motion tracking, such as robotics, wearable technology, and sports performance analysis. The system also serves as a basis for further improvements in data processing, visualization and storage on the PC side, creating opportunities to leverage sensor data in various applications.

From the task given, This experiment was successful, demonstrating the ability of the MPU6050 sensor to detect and distinguish certain gestures when integrated with the Arduino platform. By processing accelerometer and gyroscope data, the system reliably identifies gestures, such as Gesture 1 (to the right) and Gesture 2 (to the left), based on defined thresholds. These results highlight the effectiveness of the MPU6050 in motion-based applications, providing accurate real-time feedback via serial communication to inform users of detected gestures. The code's modular structure allowed for easy customization, offering scalability to include additional gestures as needed. These successful gesture detection systems show promise for interactive applications, from user interfaces and wearable devices to robotics, where fast responses to physical gestures enhance the user experience.

RECOMMENDATION

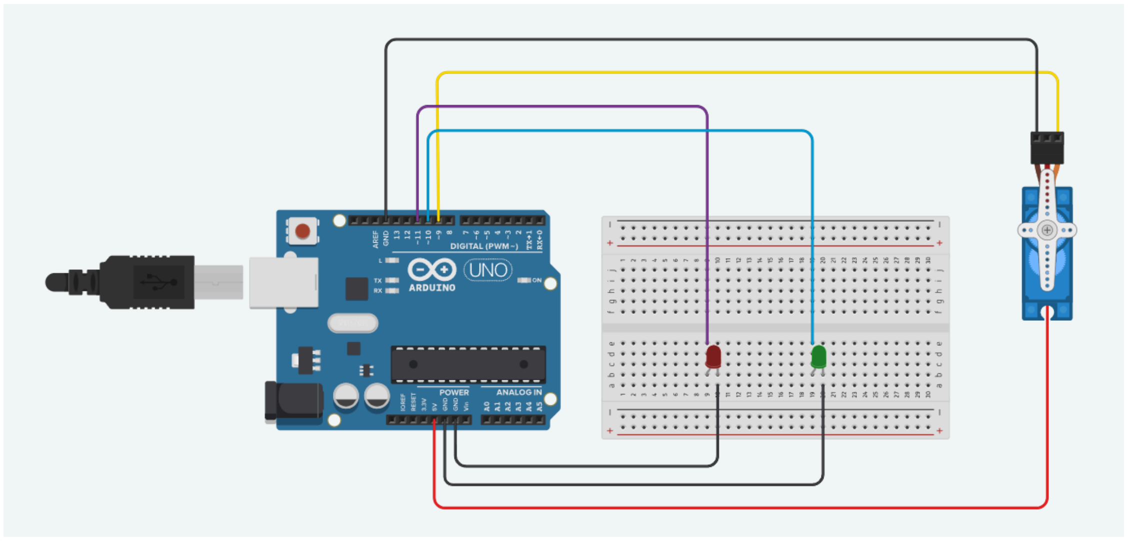
To further develop the system, several improvements can increase its flexibility and accuracy. First, we can add more servo motors to be more flexible (instead of just right and left, we can set it to go up and down also back and forward. In addition, refining the gesture recognition algorithm, potentially with machine learning techniques, can provide more robust detection, especially in settings with varying motion patterns. Implementing dynamic threshold adjustments will allow the system to adapt to different user environments and intensities. Extending the gesture library with additional motion definitions and actions as well will make the system more versatile and applicable across a wider range of applications.These improvements can make gesture detection systems more practical and effective in real-world applications.

EXPERIMENT 4B - <u>RFID card authentication and controlling a servo motor with Python and Arduino,</u>

MATERIALS AND EQUIPMENT

- Arduino Mega 2560
- RFID card reader
- RFID card
- Servo motor
- Jumper wires
- Breadboard
- USB to VGA converter
- LEDs

EXPERIMENTAL SETUP



METHODOLOGY

1. Purpose: To authenticate RFID tags and control a servo motor using signals sent through Python and Arduino.

2. Device Setup: Set up the RFID reader to communicate with the Arduino via USB and connect the servo motor for controlled motion based on RFID input.

3. Programming: Write Arduino code to handle RFID data and motor control and a Python script to interface with the USB reader.

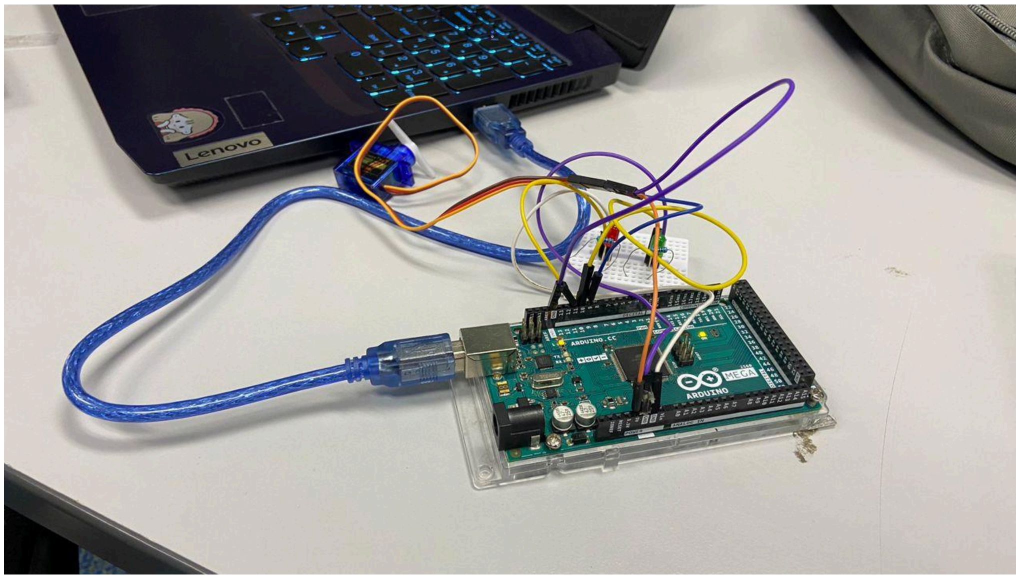
PROCEDURE

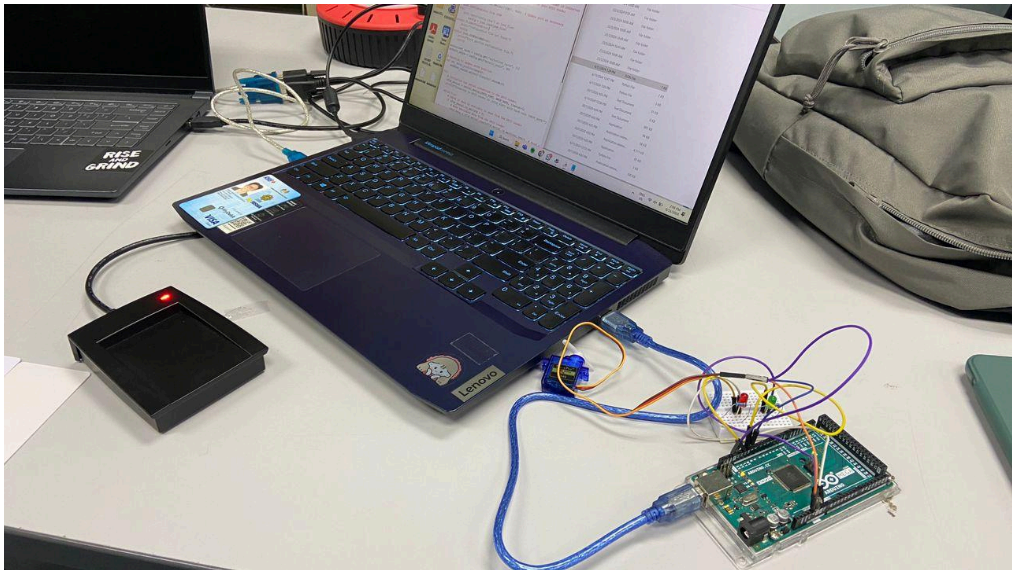
1. **Hardware setup:**
    - Connect the servo's power, ground, and signal wire to the Arduino's 5V output, ground, and one of the PWM pins respectively.
    - The RFID reader is connected and powered via a USB connection using a VGA-to-USB converter.
2. **Coding for Arduino:**
    - Program the Arduino to receive signals from Python and be able to control the angle of the servo motor.
3. **Python Script setup:**
    - Install the *pyusb* library in order for Python to recognise the USB HID devices.
    - Find the RFID reader's vendor ID and product ID.
4. **Testing:**
    - Run both of the programs and place the RFID card near the reader, the card's UID will be sent from the Arduino to the Python script, which will then display it in the terminal.

RESULTS

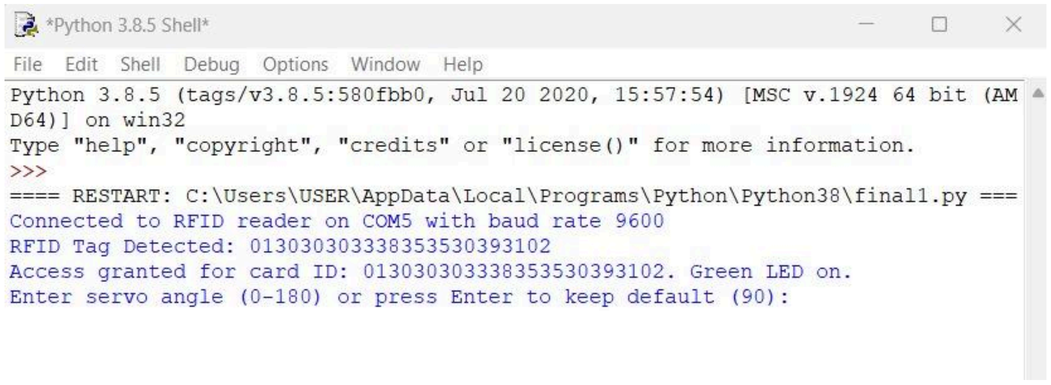
Reading of an RFID card by the RFID reader:

Observing the servo motor's motion once the RFID card is detected:

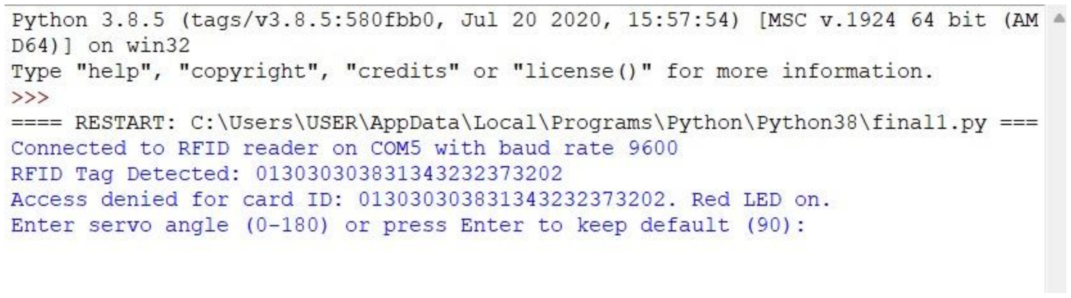




After running the coding and tapping the RFID card on the reader, the program successfully grants or denies the specific tag registered. The following outcome can be observed: when access is granted, the green LED is turned on, while when access is denied, the red LED is turned on.



```
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
==== RESTART: C:\Users\USER\AppData\Local\Programs\Python\Python38\final1.py ===
Connected to RFID reader on COM5 with baud rate 9600
RFID Tag Detected: 013030303338353530393102
Access granted for card ID: 013030303338353530393102. Green LED on.
Enter servo angle (0-180) or press Enter to keep default (90):
```



```
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
==== RESTART: C:\Users\USER\AppData\Local\Programs\Python\Python38\final1.py ===
Connected to RFID reader on COM5 with baud rate 9600
RFID Tag Detected: 013030303831343232373202
Access denied for card ID: 013030303831343232373202. Red LED on.
Enter servo angle (0-180) or press Enter to keep default (90):
```

DISCUSSION

This experiment demonstrated successful integration of an RFID-based access control system with an Arduino and Python setup, where recognized RFID tags trigger specific actions such as LED lighting and servo motor control.

1. Functionality and Responsiveness:

The system effectively identified RFID cards and accurately triggered responses based on access permissions. This was confirmed through the LED feedback: a green LED for authorized access and a red LED for unauthorized attempts.

The servo motor responded promptly to the RFID card scan, rotating to the designated angle upon detecting an authorized card, which showcases the system's potential for real-time access control applications.

2. Reliability of Serial Communication:

Communication between the RFID reader and Arduino was stable, and the data transmission from Arduino to Python was reliable over multiple trials. The use of the Python script for reading and displaying RFID data also enhanced the system's flexibility, enabling easier adjustments of access parameters and user permissions.

The integration with the servo motor, which was controlled via PWM signals from the Arduino, proved effective. This further validates the Arduino's capability to handle multiple I/O operations concurrently without noticeable latency**.**

3. Challenges and Limitations:

Distance Sensitivity: The RFID reader had a limited detection range, requiring the RFID card to be in close proximity for successful reading. This limitation may affect usability in scenarios where greater detection distance is desired.

Interference and Noise: The setup occasionally experienced interference from surrounding electronics, potentially due to USB power limitations or noise from other components. Adding decoupling capacitors or shielding the RFID reader may help mitigate these issues in future iterations.

Servo Motor Precision: While the servo motor successfully rotated to the specified position upon access, precision and speed could be improved for applications requiring finer control or faster response times**.**

4. Potential Enhancements:

Enhanced Authentication: Incorporating multiple forms of user authentication (e.g., PIN entry, biometrics) could increase the system's security level, making it suitable for more sensitive access control applications.

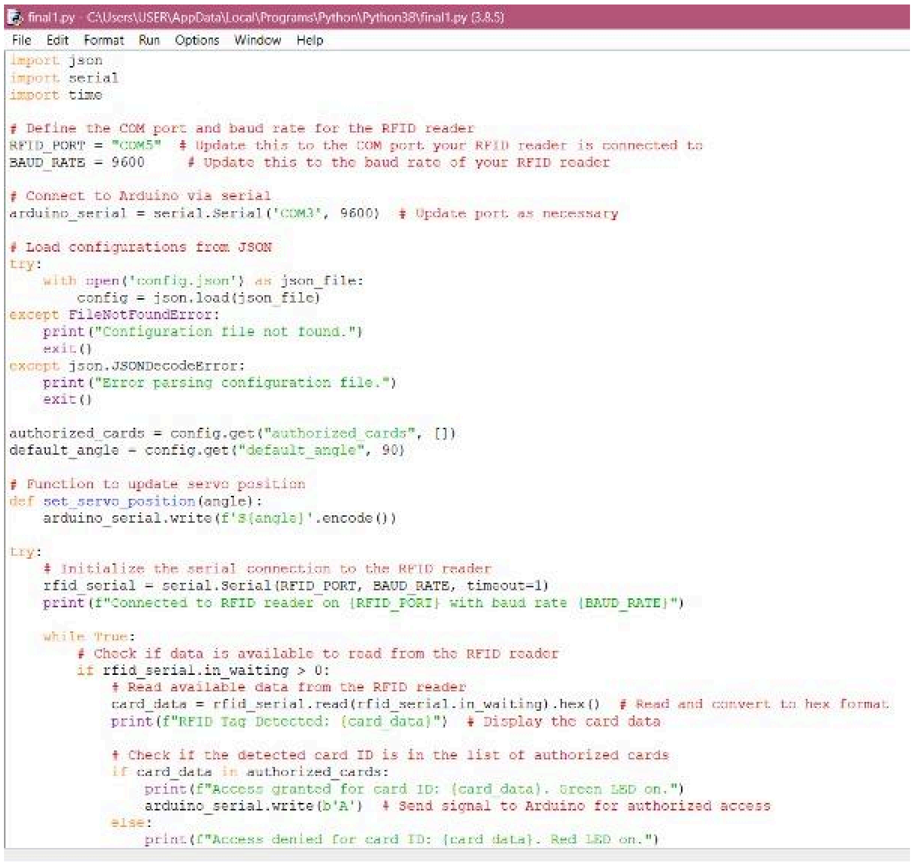
Dynamic Access Control: Creating a database to store RFID tags and user access privileges would enable dynamic updates without needing to reprogram the Arduino each time permissions change. This could be implemented by linking the system to a server or using additional local storage.

Expansion of Control Options: Besides controlling a servo motor, this setup could be expanded to activate other devices (e.g., door locks, alarms) based on user access level, further broadening its applicability in security and automation.

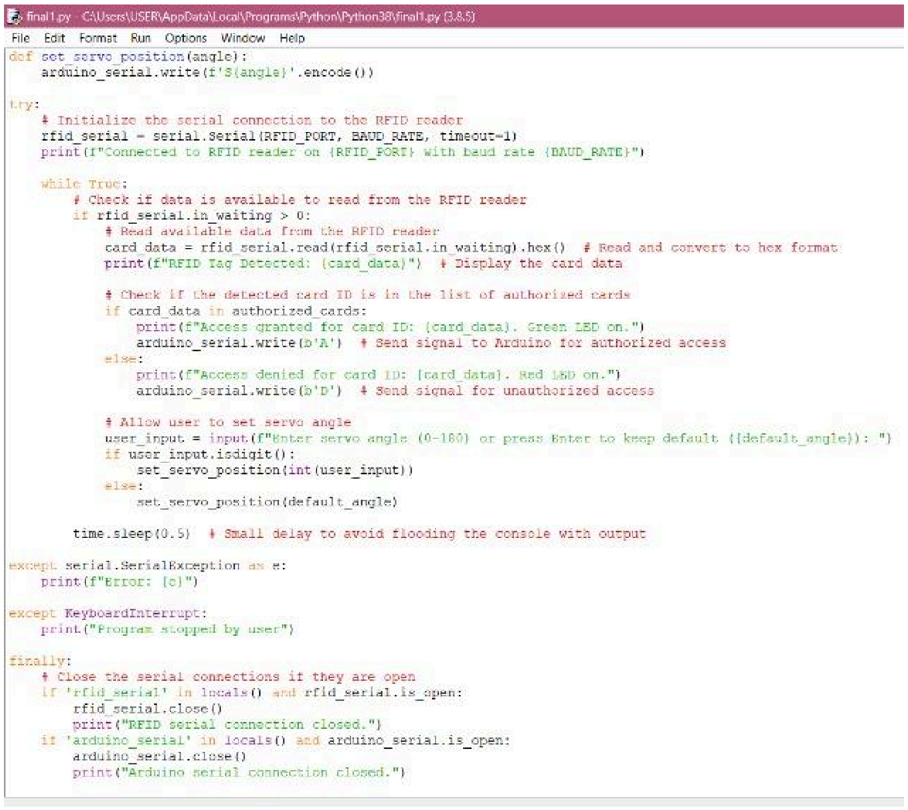
5. Applications and Future Uses:

The successful implementation of RFID-based access control, combined with actuator control, has numerous real-world applications, such as secure entry systems, automated check-in/check-out processes, and inventory management systems where item access is restricted.

Additionally, this setup serves as a foundation for smart home applications, allowing easy integration of multiple devices to create a cohesive and intelligent security solution.

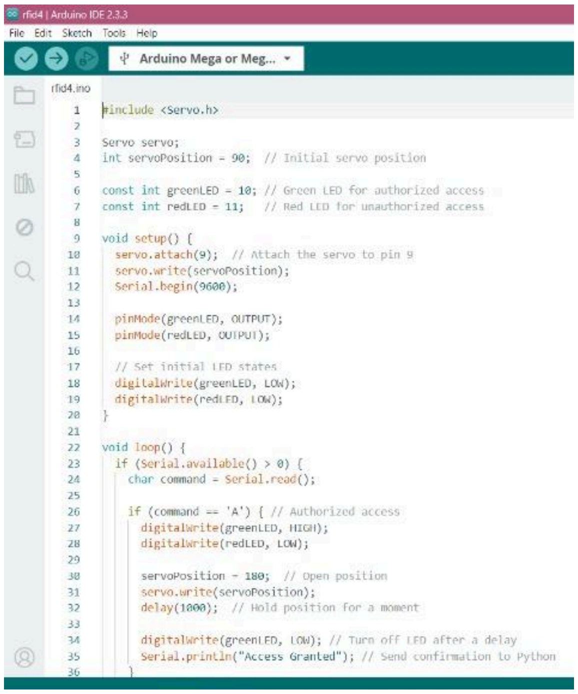

final1.py - C:\Users\USER\AppData\Local\Programs\Python\Python38\final1.py (3.8.5)
File Edit Format Run Options Window Help

```
import json
import serial
import time

# Define the COM port and baud rate for the RFID reader
RFID_PORT = "COM5"  # Update this to the COM port your RFID reader is connected to
BAUD_RATE = 9600     # Update this to the baud rate of your RFID reader

# Connect to Arduino via serial
arduino_serial = serial.Serial('COM3', 9600)  # Update port as necessary

# Load configurations from JSON
try:
    with open('config.json') as json_file:
        config = json.load(json_file)
except FileNotFoundError:
    print("Configuration file not found.")
    exit()
except json.JSONDecodeError:
    print("Error parsing configuration file.")
    exit()

authorized_cards = config.get("authorized_cards", [])
default_angle = config.get("default_angle", 90)

# Function to update servo position
def set_servo_position(angle):
    arduino_serial.write(f'S{angle}'.encode())

try:
    # Initialize the serial connection to the RFID reader
    rfid_serial = serial.Serial(RFID_PORT, BAUD_RATE, timeout=1)
    print(f"Connected to RFID reader on {RFID_PORT} with baud rate {BAUD_RATE}")

    while True:
        # Check if data is available to read from the RFID reader
        if rfid_serial.in_waiting > 0:
            # Read available data from the RFID reader
            card_data = rfid_serial.read(rfid_serial.in_waiting).hex()  # Read and convert to hex format
            print(f"RFID Tag Detected: {card_data}")  # Display the card data

            # Check if the detected card ID is in the list of authorized cards
            if card_data in authorized_cards:
                print(f"Access granted for card ID: {card_data}. Green LED on.")
                arduino_serial.write(b'A')  # Send signal to Arduino for authorized access
            else:
                print(f"Access denied for card ID: {card_data}. Red LED on.")
```

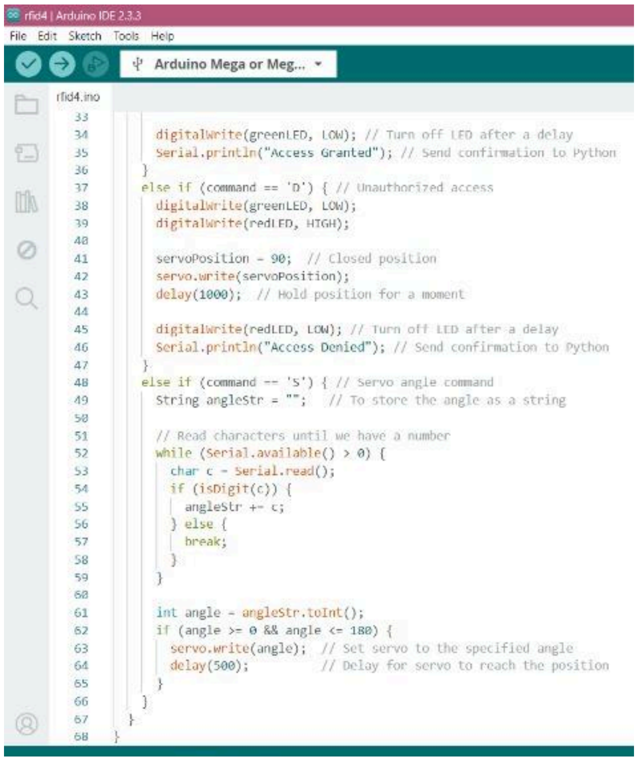

final1.py - C:\Users\USER\AppData\Local\Programs\Python\Python38\final1.py (3.8.5)
File Edit Format Run Options Window Help

```python
def set_servo_position(angle):
    arduino_serial.write(f'S{angle}'.encode())

try:
    # Initialize the serial connection to the RFID reader
    rfid_serial = serial.Serial(RFID_PORT, BAUD_RATE, timeout=1)
    print(f"Connected to RFID reader on {RFID_PORT} with baud rate {BAUD_RATE}")

    while True:
        # Check if data is available to read from the RFID reader
        if rfid_serial.in_waiting > 0:
            # Read available data from the RFID reader
            card_data = rfid_serial.read(rfid_serial.in_waiting).hex()  # Read and convert to hex format
            print(f"RFID Tag Detected: {card_data}")  # Display the card data

            # Check if the detected card ID is in the list of authorized cards
            if card_data in authorized_cards:
                print(f"Access granted for card ID: {card_data}. Green LED on.")
                arduino_serial.write(b'A')  # Send signal to Arduino for authorized access
            else:
                print(f"Access denied for card ID: {card_data}. Red LED on.")
                arduino_serial.write(b'D')  # Send signal for unauthorized access

            # Allow user to set servo angle
            user_input = input(f"Enter servo angle (0-180) or press Enter to keep default ({default_angle}): ")
            if user_input.isdigit():
                set_servo_position(int(user_input))
            else:
                set_servo_position(default_angle)

        time.sleep(0.5)  # Small delay to avoid flooding the console with output

except serial.SerialException as e:
    print(f"Error: {e}")

except KeyboardInterrupt:
    print("Program stopped by user")

finally:
    # Close the serial connections if they are open
    if 'rfid_serial' in locals() and rfid_serial.is_open:
        rfid_serial.close()
        print("RFID serial connection closed.")
    if 'arduino_serial' in locals() and arduino_serial.is_open:
        arduino_serial.close()
        print("Arduino serial connection closed.")
```

- **Software**



```cpp
#include <Servo.h>

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

const int greenLED = 10; // Green LED for authorized access
const int redLED = 11;   // Red LED for unauthorized access

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

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

  // Set initial LED states
  digitalWrite(greenLED, LOW);
  digitalWrite(redLED, LOW);
}

void loop() {
  if (Serial.available() > 0) {
    char command = Serial.read();

    if (command == 'A') { // Authorized access
      digitalWrite(greenLED, HIGH);
      digitalWrite(redLED, LOW);

      servoPosition = 180;  // Open position
      servo.write(servoPosition);
      delay(1000);  // Hold position for a moment

      digitalWrite(greenLED, LOW); // Turn off LED after a delay
      Serial.println("Access Granted"); // Send confirmation to Python
    }
```



```

      digitalWrite(greenLED, LOW); // Turn off LED after a delay
      Serial.println("Access Granted"); // Send confirmation to Python
    }
    else if (command == 'D') { // Unauthorized access
      digitalWrite(greenLED, LOW);
      digitalWrite(redLED, HIGH);

      servoPosition = 90;  // Closed position
      servo.write(servoPosition);
      delay(1000);  // Hold position for a moment

      digitalWrite(redLED, LOW); // Turn off LED after a delay
      Serial.println("Access Denied"); // Send confirmation to Python
    }
    else if (command == 'S') { // Servo angle command
      String angleStr = "";   // To store the angle as a string

      // Read characters until we have a number
      while (Serial.available() > 0) {
        char c = Serial.read();
        if (isDigit(c)) {
          angleStr += c;
        } else {
          break;
        }
      }

      int angle = angleStr.toInt();
      if (angle >= 0 && angle <= 180) {
        servo.write(angle);  // Set servo to the specified angle
        delay(500);          // Delay for servo to reach the position
      }
    }
  }
}
```

- **Hardware**

Firstly, we have the RFID (Radio-Frequency Identification) reader connected and powered by the laptop. Its communication is handled through a USB cable with its purpose in receiving the tagged signal of an RFID card. Next, we have the servo motor that is directly connected to the Arduino microcontroller. The Arduino is programmed to interpret signals and control the motor's motion, whilst the servo motor acts as the output device. Similar to the servo motor, LEDs are connected to the Arduino by the breadboard with jumper wires to transmit and confirm the feedback signal.

- **Electrical**

To go into further detail, we have the Arduino as the central microcontroller which bridges the connection between the LED and servo motor. The servo motor is connected directly to the 5 V output pin, ground pin, and PWM Pin 9 for the signal. As for the LED, we use the PWM Pin 10 and 11 to connect to the anode and cathode respectively. Lastly, we have three different USB connections to the laptop. One is to power and upload the program to the Arduino. The other two are to power and VGA--to-USB converter and transmit the signals from the RFID reader to the laptop.

CONCLUSION

In conclusion, the successful completion of this project, along with incremental improvements, will demonstrate a solid understanding of serial and USB interfaces, as well as effective microcontroller control in embedded systems. By linking the RFID reader to a computer via USB and using Python to communicate with an Arduino-controlled servo motor, this setup achieves a functional access control system with visual feedback and custom controls.

The addition of green and red LEDs for visual indicators improves the user experience by providing clear feedback on access status, while JSON data handling introduces a structured and flexible approach to managing authorized card and servo settings. Allows a user-defined servo angle to adjust the system, making it more versatile and customizable. Overall, this improvement showcases efficiency in sensor-actuator integration, modular code design and effective user interface practices. This improved system serves as a valuable foundation for further projects in automation, IoT, and real-world embedded applications.

RECOMMENDATION

To improve project functionality and user experience, consider including data logging, audio feedback, refined servo control and modular code refactoring. Data logging will create a record of each access attempt, capturing the card's UID, access status and timestamp. This information can be stored locally or in a cloud database, offering a valuable history of access events that can improve security and provide insight into usage patterns. Adding audio feedback with a buzzer will improve user interaction by providing immediate audible signals such as a short beep for access granted and a longer beep for access denied helping users understand access status without relying solely on visual signals. Advanced servo control can also improve functionality by allowing smooth and gradual movement between angles, rather than quick snaps to a specific position. This will make the system more versatile and refined, especially in scenarios where precise or smooth motion is preferred. Together, these improvements will significantly increase the robustness, usability and adaptability of the project.

REFERENCES

RFID FAQ. (n.d.). HID Global. Retrieved November 5, 2024, from https://www.hidglobal.com/rfid-faq

Sensors Modules Mpu6050 Gyroscope Accelerometer Temperature Senso... (n.d.). ElectronicWings. Retrieved November 5, 2024, from https://www.electronicwings.com/sensors-modules/mpu6050-gyroscope-accelerometer-temperature-sensor-module

ACKNOWLEDGEMENT

We would like to express our gratitude to Dr. Wahju Sediono, Dr. Ali Sophian, Dr. Zulkifli bin Zainal Abidin, my teaching assistant, and my peers for their invaluable help and support in finishing this project. Their advice, feedback, and experience have greatly influenced the level of quality and understanding of this work.

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 has 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: [illegible]	**Read**	/
Name: AKMAL ZARIF BIN NAJIB	**Understand**	/
Matric No: 2211971	**Agree**	/

Signature: [illegible]	**Read**	/
Name: AMIR FARHAN BIN GHAFFAR	**Understand**	/
Matric No: 2115617	**Agree**	/

Signature: [illegible]	**Read**	/
Name: AMIRAH HUDA BINTI JAMALULAIL ASRI	**Understand**	/
Matric No: 2210776	**Agree**	/

Signature: [illegible]	**Read**	/
Name: AMIRUL AIZAD BIN AMIR	**Understand**	/
Matric No: 2211263	**Agree**	/

Signature: [illegible]	**Read**	/
Name: ANAS BIN BADRULZAMAN	**Understand**	/
Matric No: 2219945	**Agree**	/