KULLIYYAH OF ENGINEERING (KOE)

MECHATRONICS SYSTEM INTEGRATION (MCTA3203)

SEMESTER 1, 24/25

SECTION 1

PROJECT REPORT WEEK 4

TITLE : SERIAL COMMUNICATION IMU & RFID READER

PREPARED BY :

NO	NAME	MATRIC NO
1	ARIF EMRULLAH BIN TAJUL ARIFFIN	2215359
2	AZLIYANA SYAHIRAH BINTI AZAHARI	2210620
3	DAMIA MAISARAH BINTI ZAWAWI	2217830
4	HUDA BINTI AB RAHMAN AL-QARI	2226676
5	IKMAL HAKIM BIN ZAKI	2125625

DATE OF SUBMISSION : 6TH NOVEMBER 2024

ABSTRACT

This experiment focuses on the fundamentals of serial and USB interfacing to integrate sensors and actuators with microcontroller-based systems, specifically examining the MPU6050 IMU sensor and an RFID card reader. The MPU6050, known for its compact design and combined accelerometer and gyroscope capabilities, offers a valuable source of motion and orientation data for various applications. By establishing a connection between the MPU6050 and a computer through an Arduino board, we demonstrated how data from the sensor can be processed and applied in different project scenarios, emphasising the sensor's versatility in embedded systems.

In addition to sensor integration, this experiment covers interfacing a computer with an RFID card reader via USB HID protocols, a common method for USB-connected devices. With Python code, we implemented RFID-based authentication to control a servo motor through Arduino, simulating a controlled access system. Together, these activities provide foundational skills in serial and USB communication for complex projects requiring integrated sensor and actuator systems, illustrating practical applications in computer-based automation and interactive systems.

TABLE OF CONTENTS

INTRODUCTION

In modern embedded systems, serial and USB interfacing techniques are essential for integrating sensors and actuators with microcontrollers and computer-based systems. This experiment focuses on interfacing an MPU6050 Inertial Measurement Unit (IMU) and an RFID card reader using an Arduino microcontroller, demonstrating foundational concepts in sensor data acquisition and device control. The MPU6050, a widely used sensor known for its compact size, affordability, and dual accelerometer-gyroscope capabilities, provides motion and orientation data, making it valuable for a variety of applications requiring precise measurements. By connecting the MPU6050 to a computer via an Arduino, this experiment illustrates how motion data can be transferred and utilised in real-time projects.

In addition to the IMU, this setup includes an RFID card reader connected via USB, requiring specific methods for Python code to communicate with it. Many RFID readers operate as USB Human Interface Devices (HID), necessitating libraries or modules to handle HID communication effectively. The experiment also incorporates a servo motor controlled by Arduino based on RFID card authentication, demonstrating how USB and serial communication can facilitate security-based actuator control. This setup serves as a practical introduction to more advanced digital interfacing, providing hands-on experience with core principles of embedded system design and interactive control applications.

PART A

MATERIALS AND EQUIPMENTS

1. Arduino Mega 2560 board
2. MPU6050
3. Jumper wires
4. Breadboard

EXPERIMENTS SETUP

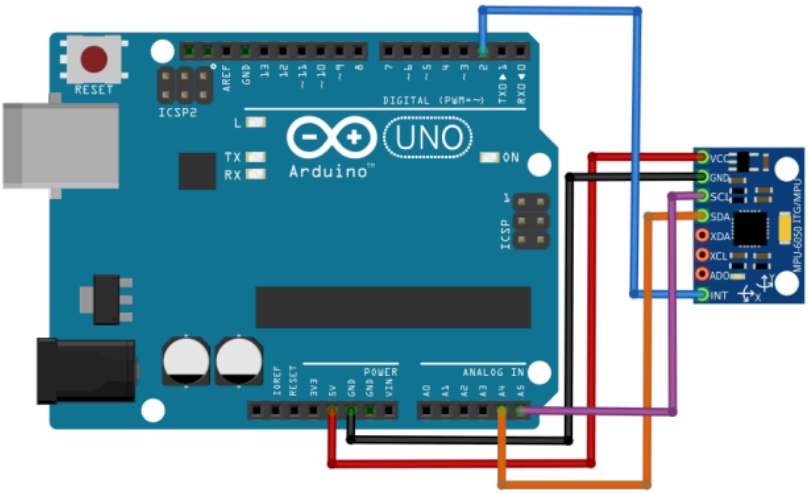


Fig. 1: Arduino-MPU6050 Connections

METHODOLOGY

1. Component Selection and Preparation:
    - The components used in this setup include an Arduino Uno microcontroller and an MPU6050 accelerometer and gyroscope sensor.
    - The Arduino Uno serves as the microcontroller to read data from the MPU6050 sensor using the I2C communication protocol.
2. Wiring the Connections:
    - I2C Communication Pins:
        - The SDA (Serial Data) pin on the MPU6050 was connected to the A4 pin on the Arduino Uno.
        - The SCL (Serial Clock) pin on the MPU6050 was connected to the A5 pin on the Arduino Uno.

PROCEDURES

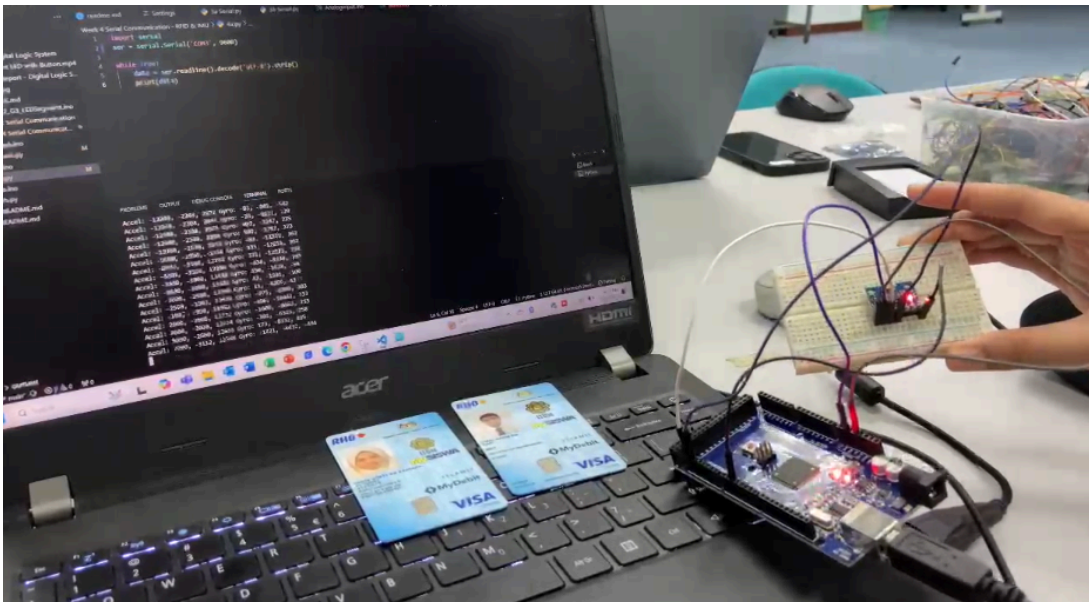
1. The MPU6050 sensor was connected to the Arduino board using the appropriate pins. The SDA and SCL pins of the MPU6050 were connected to the corresponding pins on the Arduino (typically A4 and A5 on most Arduino boards) to enable I2C communication.
2. The power supply and ground of the MPU6050 were connected to the Arduino's 5V and GND pins.
3. The Arduino board was connected to the PC via USB.

CODING

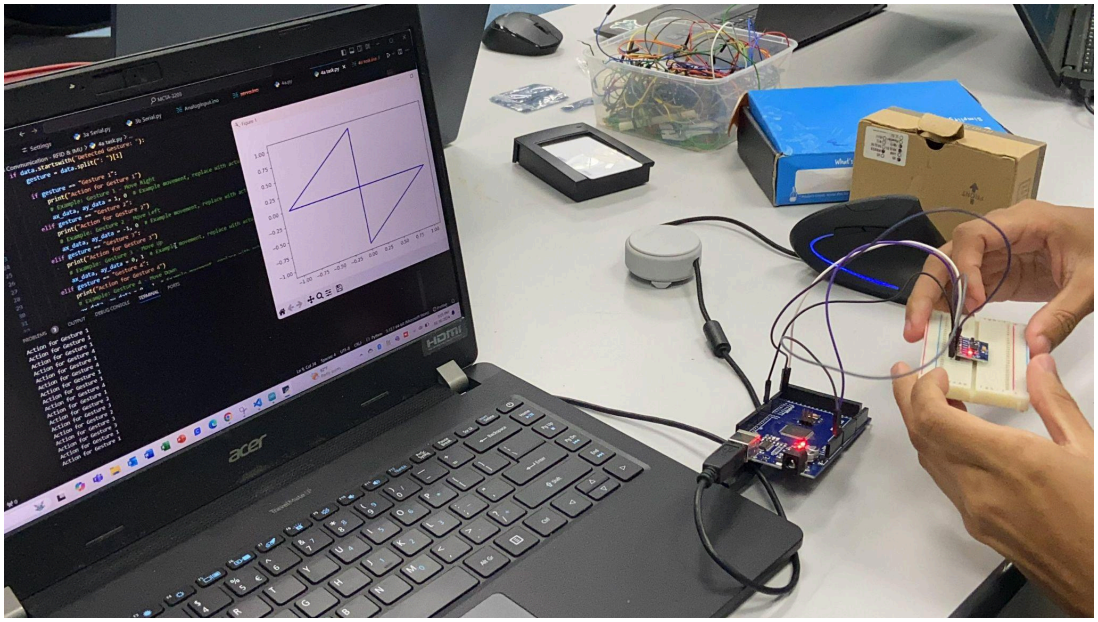
```
import serial
ser = serial.Serial('COM3', 9600)
```

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

RESULTS



In this experiment, the gyroscope measures changes in the orientation of the x, y and z axes. X axis is the rotation of left or right while y is the rotation of backward or upward and z axis is the rotation of twisting or spinning motion. When the gyroscope tilts to a different position, it detects the rate of rotation around each axis and updates these values in real time to the serial monitor, where it shows the x, y and z coordinates.



The picture shown above, where the graph simulates the coordinates of z, y and z. When the gyroscope tilts the output shows in the graph in real time and the coordinates can be seen in the serial monitor.

DISCUSSIONS

· Software

The Arduino uses the MPU6050 library to gather and process the accelerometer and gyroscope data from the MPU6050 sensor via I2C communication. The Python script reads the data gathered through the serial python library at the Arduino communication port. The code will read the data of the sensor as it is being tilted, pitched and such. This can be used to categorize and classify each movement uniquely.

· Electrical

The SDA and SCL pins on the MPU6050 connect to the Arduino's A4 and A5 pins for efficient data transfer. I2C is effective for this project as it allows multiple devices to communicate on a single set of wires. The MPU6050's VCC and GND connect to the Arduino's 5V and GND pins, ensuring a stable power supply essential for consistent sensor readings. The Arduino is connected to a PC via USB for continuous data streaming, allowing real-time data analysis and visualization.

· **Hardware**

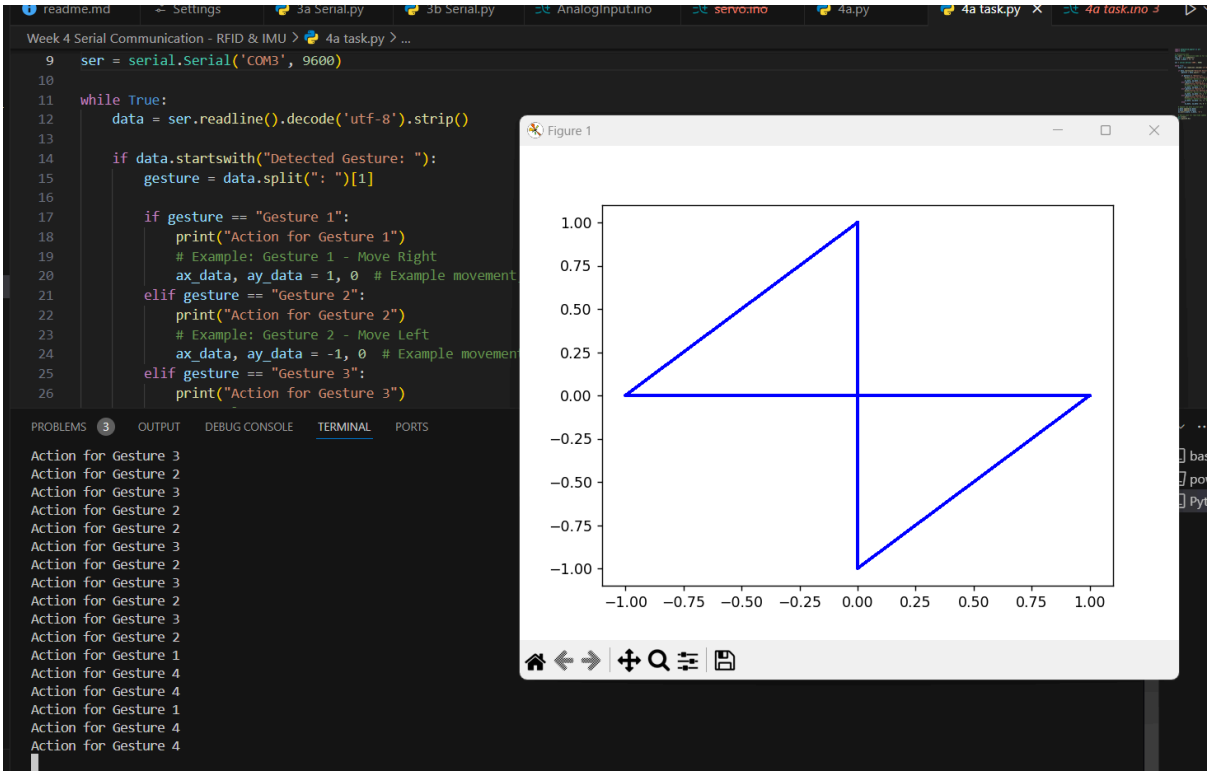
The **MPU6050** is a sensor that combines a **3-axis accelerometer** and **3-axis gyroscope** to track motion and orientation. It measures acceleration to detect changes in position and tilt, while the gyroscope tracks rotational speed around each axis, making it useful for applications that need precise movement data. With its **Digital Motion Processor (DMP)**, the MPU6050 can efficiently combine these readings, reducing the workload on a connected microcontroller, like an Arduino. It connects through an **I2C interface** to send motion data, making it popular in projects like robotics, wearable devices, and gaming controllers.

Questions:

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

Answers :

In the code below, we established four different gestures that detect four sets of movements: left, right, upward, and downward. Gesture 1 represents a right movement, gesture 2 represents left, gesture 3 is upward, and gesture 4 is downward. When the Python script runs and the gyroscope rotates to different positions, it provides real-time feedback to the serial monitor, indicating which gesture is detected, and translates the gesture into the graph below.




```
import matplotlib.pyplot as plt
import serial

# init plot
plt.ion()
fig, ax = plt.subplots()
x_data, y_data = [], []

ser = serial.Serial('COM3', 9600)
```

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

    if data.startswith("Detected Gesture: "):
        gesture = data.split(": ")[1]

        if gesture == "Gesture 1":
            print("Action for Gesture 1")
            # Gesture 1 - Move Right
            ax_data, ay_data = 1, 0
        elif gesture == "Gesture 2":
            print("Action for Gesture 2")
            # Gesture 2 - Move Left
            ax_data, ay_data = -1, 0
        elif gesture == "Gesture 3":
            print("Action for Gesture 3")
            # Gesture 3 - Move Up
            ax_data, ay_data = 0, 1
        elif gesture == "Gesture 4":
            print("Action for Gesture 4")
            # Gesture 4 - Move Down
            ax_data, ay_data = 0, -1
        else:
            ax_data, ay_data = 0, 0  # No movement or gesture detected

    # Append new data to plot path
    x_data.append(ax_data)
    y_data.append(ay_data)
    ax.plot(x_data, y_data, 'b-')

    # Redraw plot for real-time update
    plt.draw()
    plt.pause(0.01)
```

CONCLUSION

In experiment 4a we established a successful configuration between MPU6050 and arduino to demonstrate how a gyroscope can accurately measure and report changes in orientation along the x,y and z axes by detecting the angular velocity in the MPU6050 module. When tilted or rotated, the gyroscope provides real time data on each axis which is displayed in the serial monitor. While in the task given where the coordinates data needs to be translated into a graph, we successfully established.

RECOMMENDATIONS

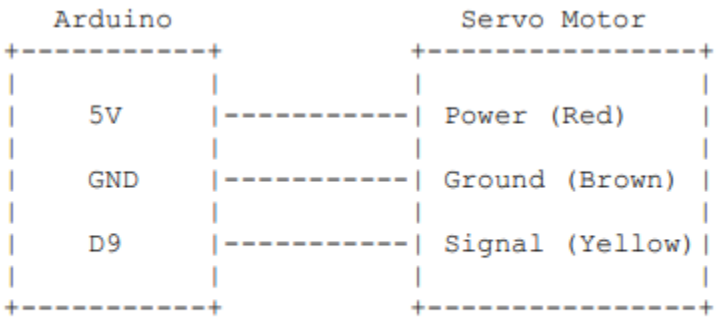
For future iterations of the experiment, the gyroscope can be improved by adding gesture calibration. By implementing gesture calibration it could help to ensure that each gesture can be detected consistently. This could involve capturing initial sensor values as reference to account for any offset or tilt in the gyroscope's starting position. Another recommendation is to implement error handling. By including error handling in the script to address any disconnections or sensor issues. This will make it easier to detect whether the code and gyroscope are working.

PART B

MATERIALS AND EQUIPMENTS

1. Arduino Mega 2560
2. RFID card reader with USB connectivity
3. RFID tags or cards that can be used for authentication
4. Servo Motor
5. Jumper wires
6. Breadboard
7. LEDs of red and green
8. USB cables
9. Power Supply (optional)

EXPERIMENTS SETUP



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

METHODOLOGY

1. Wiring the Connections:
    - The Signal (Yellow) wire of the servo motor was connected to Digital Pin 9 (D9) on the Arduino. This pin will send the PWM signal to control the position of the servo.

PROCEDURES

1. Connect the servo to Arduino Mega 2560 which the power (red) wire was connected to the 5V pin, the ground (brown) wire was connected to the GND pin and the signal (yellow) wire was connected to Digital Pin 9 (D9).
2. The RFID reader was connected to the computer via USB
3. Upload the Arduino code to Arduino Mega 2560.
4. Create Python code to interact with the USB RFID reader,
5. Set up Python code to check if detected RFID tags were on an authorized list. If a card was recognized, an "A" signal was sent to the Arduino to move the servo. If not, a "D" signal was sent.
6. Testing was conducted by scanning both authorized and unauthorized RFID cards near the reader, verifying that the servo motor moved accordingly based on the signals received.

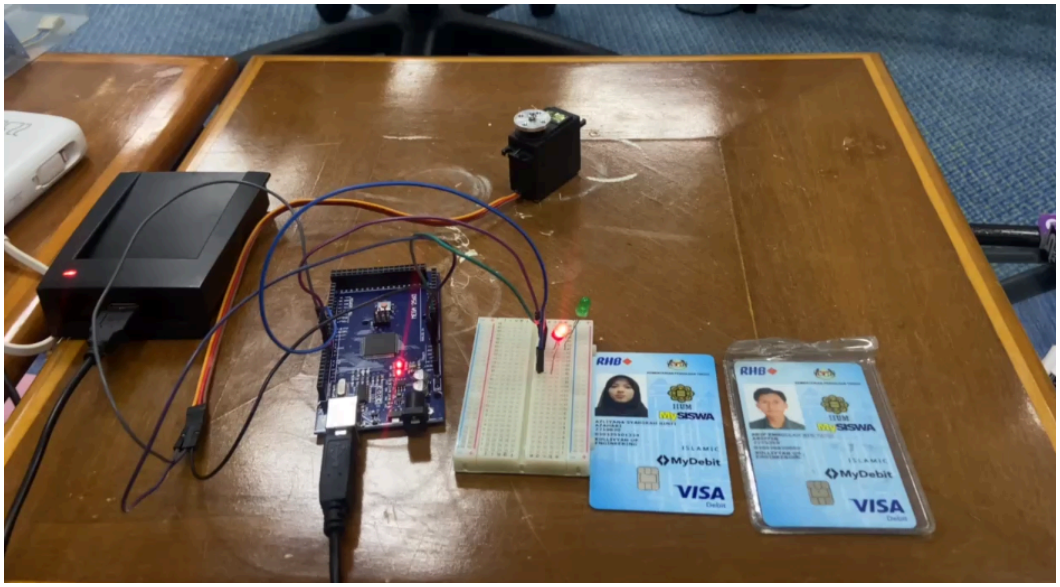
CODING

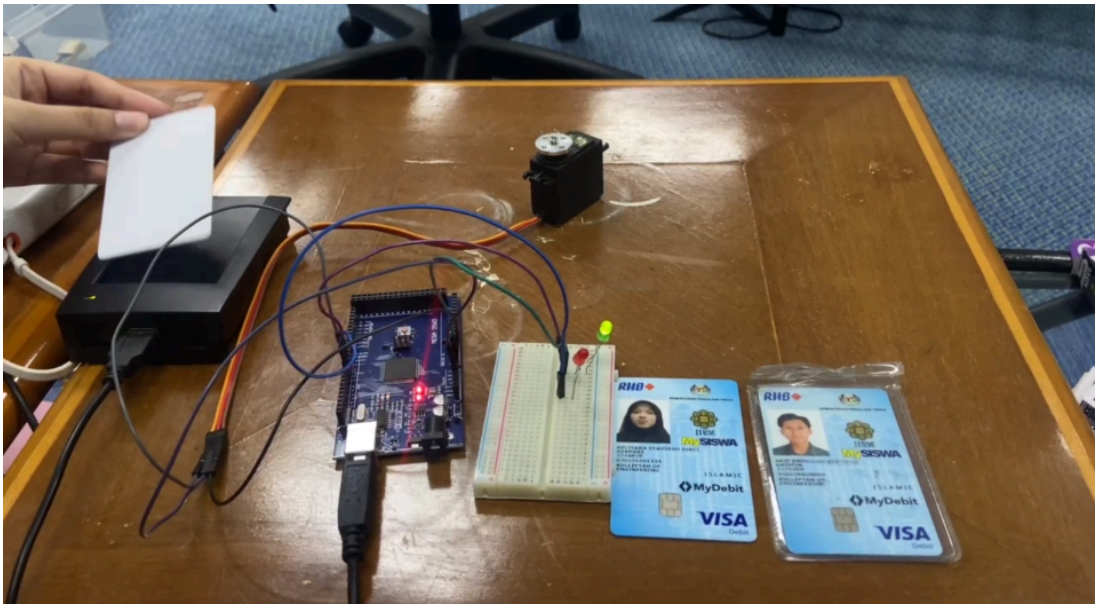
```
import serial
import time

# serial port
arduino_port = 'COM7'
rfid_port = 'COM6'
baud_rate = 9600

# Authorized card IDs (replace these with actual card IDs)
authorized_cards = ["0008089233", "♥"]  # Example: ['4B8D', '7A4E']

def main():
    arduino = serial.Serial(arduino_port, baud_rate, timeout=1)
    time.sleep(2)
```

```
    rfid = serial.Serial(rfid_port, baud_rate, timeout=1)
    time.sleep(2)

    last_card_time = time.time()
    while True:
        current_time = time.time()

        if rfid.in_waiting > 0:  # Check if the RFID reader has data
            card_id = rfid.readline().decode('utf-8').strip()  # Read
the card ID
            card_id = card_id[1:]  # Remove the first character
(control character)
            print(f"Card ID: {card_id} (ASCII: {[ord(c) for c in
card_id]})")

            # Reset the last card detection time
            last_card_time = current_time

            # Check for empty card ID
            if not card_id:
                print("No card detected. Skipping...")
                continue  # Skip to the next iteration of the loop

            if card_id in authorized_cards:
                print("Access granted.")
                arduino.write(b'A')  # Send 'A' to Arduino to allow
servo control
            else:
                print("Access denied.")
                arduino.write(b'D')  # Send 'D' to Arduino to disallow
control
        else:
            # Check if 5 seconds have passed since the last card was
detected
            if current_time - last_card_time > 5:
                print("No card detected for 5 seconds. Sending 'D' to
Arduino.")
                arduino.write(b'D')  # Send 'D' to Arduino to disallow
control
                last_card_time = current_time

if __name__ == "__main__":
```

```
    main()
```

RESULTS





DISCUSSIONS

- **Software**

The program code is used to control real-time access with RFID cards. The code used is to show that the RFID card reader unique ID's can be used to communicate with Arduino via serial communications and perform an action, which in this case is the Servo. A problem we discovered with the RFID Reader we used is that we are unable to use the Vendor and Product ID from the information provided by the Device Manager program on Windows. An alternative approach that we took was reading the Card ID through the COM serial instead by using the serial python library. This meant that we had our Python code read from two COM ports, Arduino (COM7) and RFID (COM6). After thorough testing, we discovered that it works perfectly fine.

- **Electrical**

The RFID reader is connected via USB for power and a VGA cable for the communication line. The servo motor receives PWM signals from Arduino pin D9 for controlling the angle precisely. The Arduino is also connected via USB which provides both communication and power. Additional LEDs are also added to the Arduino, red and green LEDs, each connected to digital pins as indicators.

- **Hardware**

RFID card reader is a device used to read data stored on RFID (Radio Frequency Identification) cards or tags, commonly for identification, access control, and tracking purposes. RFID cards contain a small chip and antenna that communicate with the reader when brought within close proximity. The reader emits radio waves that power the card's chip, enabling it to send back stored information such as a unique ID number. Many RFID readers connect to computers or microcontrollers, like Arduino, via USB or serial interfaces. They are often used in security systems for entry access, inventory management, and contactless payments, as they provide a fast, secure way to identify and authenticate objects or individuals.

An RFID card is a contactless card that uses Radio Frequency Identification technology to transmit data wirelessly. It contains a small chip that stores information, such as a unique ID number, and an antenna that allows it to communicate with an RFID reader. When the card is placed near the reader, it receives power from the reader's radio waves and sends back its stored information. RFID cards are commonly used for access control, like unlocking doors, and for contactless payments, offering a quick and convenient way to identify users or objects without physical contact.

Questions:

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

Answers :

In the code below, we connected a servo motor, a red LED, and a green LED to the Arduino to demonstrate RFID detection functionality. Initially, the red LED lights up to indicate that no RFID card is detected, and the servo motor remains stationary. Once an RFID card is detected, the red LED turns off, the green LED lights up, and the servo motor rotates to signify successful detection. This setup provides clear visual feedback and motorized action to indicate the presence of a recognized RFID card.

```
import serial
import time
import json

arduino_port = 'COM7'  # serial com arduino
rfid_port = 'COM6'      # serial com rfid
baud_rate = 9600
```

```
# JSON for card data and set servo angle
config_data = '''{
    "authorized_cards": ["0008089233", "♥"],
    "servo_angle": 180
}'''
config = json.loads(config_data)

def main():
    # init serial com to arduino
    arduino = serial.Serial(arduino_port, baud_rate, timeout=1)
    time.sleep(2)

    # init serial com to rfid
    rfid = serial.Serial(rfid_port, baud_rate, timeout=1)
    time.sleep(2)

    last_card_time = time.time()  # init the last detected card time

    while True:
        current_time = time.time()

        if rfid.in_waiting > 0:  # check if the RFID reader has data
            card_id = rfid.readline().decode('utf-8').strip()  # read
card id
            card_id = card_id[1:]
            print(f"Card ID: {card_id} (ASCII: {[ord(c) for c in
card_id]})")

            # reset card detect last time
            last_card_time = current_time

            # check empty card id
            if not card_id:
                print("No card detected. Skipping...")
                continue

            if card_id in config['authorized_cards']:
                print("Access granted.")
                arduino.write(f"A{config['servo_angle']}\n".encode())
# send angle to Arduino
            else:
                print("Access denied.")
```

```
                arduino.write(b'D')  # send 'D' to arduino
        else:
            # check 5 seconds
            if current_time - last_card_time > 5:
                print("No card detected for 5 seconds. Sending 'D' to
Arduino.")
                arduino.write(b'D')  # send 'D' to arduino
                last_card_time = current_time  # reset time

if __name__ == "__main__":
    main()
```

CONCLUSION

In conclusion, We successfully established experiment 4B and the task given. The setup effectively demonstrates the integration of the RFID reader with visual and motorized feedback on the arduino. The red and green LEDs provide clear indications and immediate visual cues, while moto rotation indicates the success of RFID detection. This combination of components are ways to showcase RFID functionality.

RECOMMENDATIONS

To further enhance the RFID detection system, the RFID can be improved by UID based control whereas it allows the system to respond differently depending on which UID is detected. such as setting unique servo angles or using additional LEDs to indicate different access levels.

For future iterations of the experiment, the gyroscope can be improved by adding gesture calibration. By implementing gesture calibration it could help to ensure that each gesture can be detected consistently. This could involve capturing initial sensor values as reference to account for any offset or tilt in the gyroscope's starting position. Another recommendation is to implement error handling. By including error handling in the script to address any disconnections or sensor issues. This will make it easier to detect whether the code and gyroscope are working.

ACKNOWLEDGEMENT

We would like to express our gratitude to Dr. Wahju Sediono, Dr. Ali Sophian, Dr. Zulkifli Bin Zainal Abidin, for providing the necessary resources and facilities to conduct this and support throughout the duration of the project. Additionally, we extend our appreciation to all individuals who contributed to the success of this lab report through their valuable insights and feedback.

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 that no further improvement on the reports is needed from any of the individual contributors to the report.

Signature: 	**Read**	/
Name: Arif Emrullah Bin Tajul Arifin	**Understand**	/
Matric No: 2215359	**Agree**	/

Signature: 	**Read**	/
Name: Azliyana Syahirah Binti Azahari	**Understand**	/

Matric No: 2210620	**Agree**	/

Signature: 	**Read**	/
Name: Damia Maisarah Binti Zawawi	**Understand**	/
Matric No: 2217830	**Agree**	/

Signature: 	**Read**	/
Name: Ikmal Hakim Bin Zaki	**Understand**	/
Matric No: 2125625	**Agree**	/

Signature: 	**Read**	/
Name: Huda binti Ab Rahman Al-Qari	**Understand**	/
Matric No: 2226676	**Agree**	/