LABORATORY REPORT

LAB 3:
SERIAL AND USB INTERFACING WITH MICROCONTROLLER AND COMPUTER BASED SYSTEM (2): SENSORS AND ACTUATORS.

GROUP B

PROGRAMME: MECHATRONIC ENGINEERING

GROUP MEMBERS:	MATRIC NO:
1. MUHAMMAD AFIQ BIN AHMAD	2115203
2. HANIS BINTI MOHD IZANI	2020590
3. SARAH AISHAH BINTI SA'AID HAZLEY	2117600
4. NUR AYU NAZIERA BINTI ROSLI	2119202
5. NOORUL GHOUSIAH BINTI NOORDEEN SAHIB	2118298

DATE OF SUBMISSION:
Wednesday, 8th November 2023

ABSTRACT

This lab consists of two parts, that is, part A and part B. The objective of part A is to interface MPU6050 accelerometer and gyroscope sensor with Arduino, whereas the objective of part B is to use RFID card authentication as an input in order to control Arduino functions. For both parts, circuit connections are made, appropriate libraries are utilised, and serial communications between Python and Arduino are established. The results of both parts of the lab are satisfactory. The interfacing between Arduino and MPU6050 sensor was successful, and the RFID card authentication system is also integrated into the Arduino program successfully.

TABLE OF CONTENTS

Table of Contents

INTRODUCTION

Lab 3 is a laboratory experiment on the topic of serial and USB interfacing with microcontroller and computer-based system. The purpose of this lab is to study the procedure of interfacing various components with Arduino, in addition to utilizing the serial connection between Arduino and Python. This lab consists of two parts, that is, part A involving MPU6050 sensor and part B involving RFID system. The objective of part A is to interface MPU6050 accelerometer and gyroscope sensor with Arduino, whereas the objective of part B is to use RFID card authentication as an input in order to control Arduino functions.

MPU6050 sensor is an inertial measurement unit (IMU) that can be used in many applications including motion tracking, orientation and position detection. The MPU6050 is a Micro-Electro-Mechanical Systems (MEMS) which consists of a 3-axis accelerometer and 3-axis gyroscope to measure motion-related parameters of a system or object. This sensor also contains a Digital Motion Processor (DMP) to perform complex calculations. Apart from that, the sensor also has two auxiliary pins that can be used to interface external IIC modules.

Radio Frequency Identification (RFID) tags can be found in many industries such as in anti-theft measures and security access cards. MFRC522 is an RFID reader module that uses electromagnetic waves in the radio frequency spectrum for communication and data transfer. The RFID system consists of two components: the proximity coupling device (the RFID reader) and the proximity integrated card (the tags). The RFID reader consists of a reader or writer, and an antenna to emits high frequency electromagnetic waves, whereas the RFID tag consists of an antenna and the RFID chip which will store all the data. When the tag is triggered by an electromagnetic interrogation pulse from a nearby RFID reader, the tag will transmit data from the tag to the reader. This data will be analysed by the reader to identify the tag.

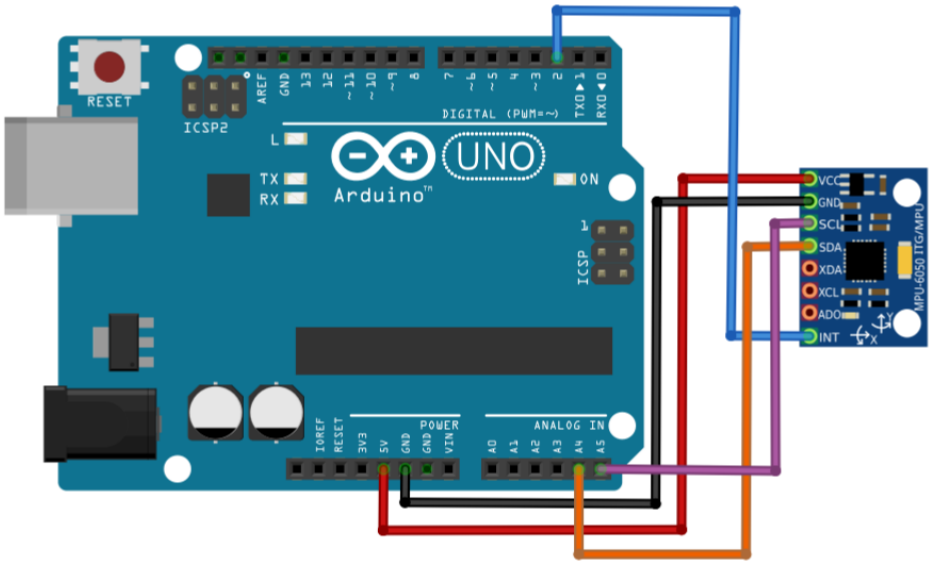
PART A

PROCEDURE

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 colors.

Experimental Setup



This is the Arduino's code that we use to read data from the MPU6050 sensor and send it to the PC via the serial port:

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

void setup() {
  Serial.begin(9600);
  Wire.begin();
 mpu.initialize();
}
void loop() {
  int ax, ay, az, gx, gy, gz;
  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);
}
```

This is the Python script we use to receive and print the data from the Arduino:

```
import serial

try:
    ser = serial.Serial('COM5', 9600)
    while True:
        data = ser.readline().decode('utf-8').strip()
        print(data)
except serial.SerialException as e:
    print(f"An error occurred: {e}")
except KeyboardInterrupt:
    print("Script terminated by user.")
finally:
    ser.close()
```

Methodology

1. The MPU6050 sensor is connected to the Arduino board using the appropriate pins. The SDA and SCL pins of the MPU6050 are connected to the A4 and A5 of the Arduino boards.

2. The power supply and ground of the MPU6050 are connected to Arduino's 5V and GND pins.

3. Then, the Arduino board is connected to the PC via USB.

This is the Arduino's code that we use for the task:

```
#include <Wire.h>
#include <MPU6050.h>
MPU6050 mpu;
const int threshold = 1000; // Adjust this threshold as needed
int previousGesture = -1;
void setup() {
 Serial.begin(9600);
 Wire.begin();
 mpu.initialize();
}

void loop() {
 int gesture = detectGesture();
 Serial.print("Detected Gesture: ");
 if (gesture == 1) {
 Serial.println("Gesture 1");
 // Perform an action for Gesture 1
 } else if (gesture == 2) {
 Serial.println("Gesture 2");
 // Perform an action for Gesture 2
```

```
 }
 // Add more gesture cases as needed
 previousGesture = gesture;

 // Read sensor data and send it to Python
 int ax, ay, az, gx, gy, gz;
 mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
 Serial.print(ax);
 Serial.print(",");
 Serial.println(ay);
 delay(100); // Adjust the delay as needed to control the data transmission
rate
}

int detectGesture() {
 int ax, ay, az, gx, gy, gz;
 mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
 // Perform gesture recognition here based on sensor data
 // Define conditions to recognize specific gestures
 if (ax > threshold && ay < threshold) {
 return 1; // Gesture 1
 } else if (ax < -threshold && ay > threshold) {
 return 2; // Gesture 2
 }
 // Add more gesture conditions as needed
 return 0; // No gesture detected
}
```

This is the python's code that we used to visualize data live:

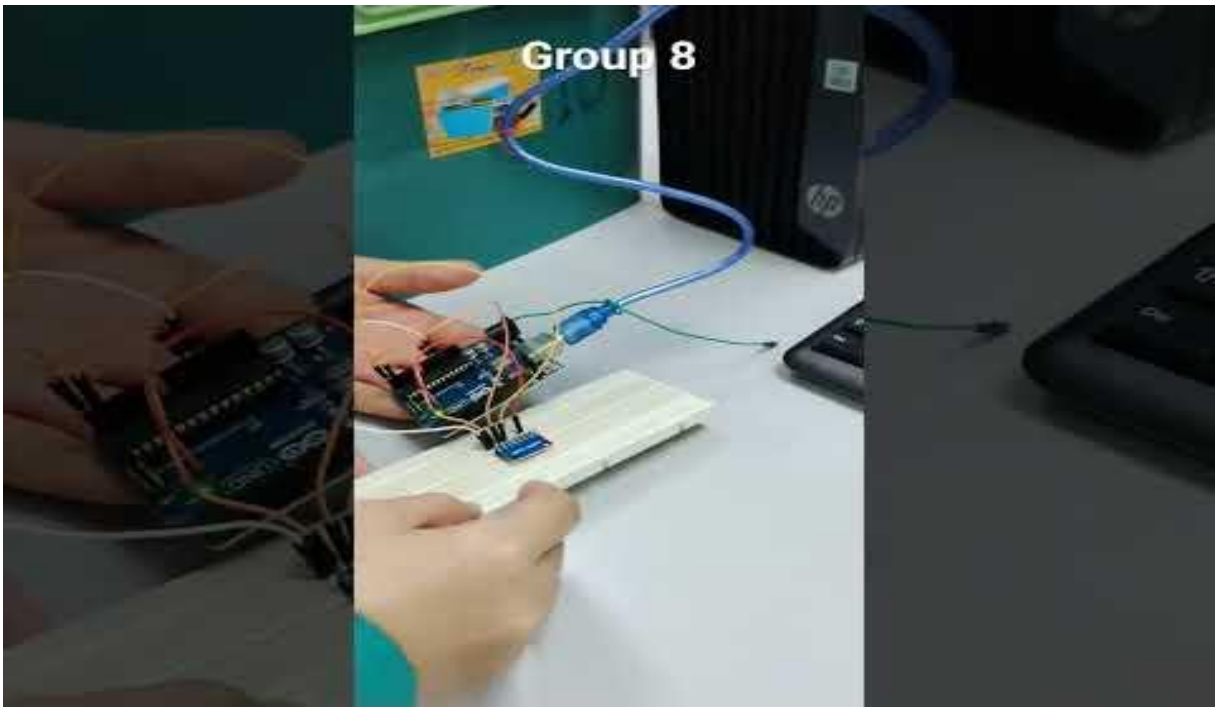
```
import serial
import matplotlib.pyplot as plt

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

# Create empty lists to store sensor data
accel_x = []
accel_y = []

# Create a live graph
plt.ion()
fig, ax = plt.subplots()
ax.set_xlim(-20000, 20000)
ax.set_ylim(-20000, 20000)
line, = ax.plot([], [], 'b-')
plt.title("Accelerometer Data")
```

```
while True:
    data = ser.readline().decode('utf-8').strip()
    if data.startswith("Detected Gesture: "):
        gesture = data.split(": ")[1]
        if gesture == "Gesture 1":
            # Perform an action for Gesture 1
            print("Action for Gesture 1")
        elif gesture == "Gesture 2":
            # Perform an action for Gesture 2
            print("Action for Gesture 2")
        # Add more gesture actions as needed
    else:
        # Parse sensor data
        try:
            accel_data = [int(val) for val in data.split(',')]
            ax, ay = accel_data  # Separate ax and ay
            accel_x.append(ax)
            accel_y.append(ay)

            # Update the graph
            line.set_data(accel_x, accel_y)
            plt.pause(0.01)  # Adjust the pause value as needed for real-time updates

        except ValueError:
            pass

plt.ioff()
plt.show()
```

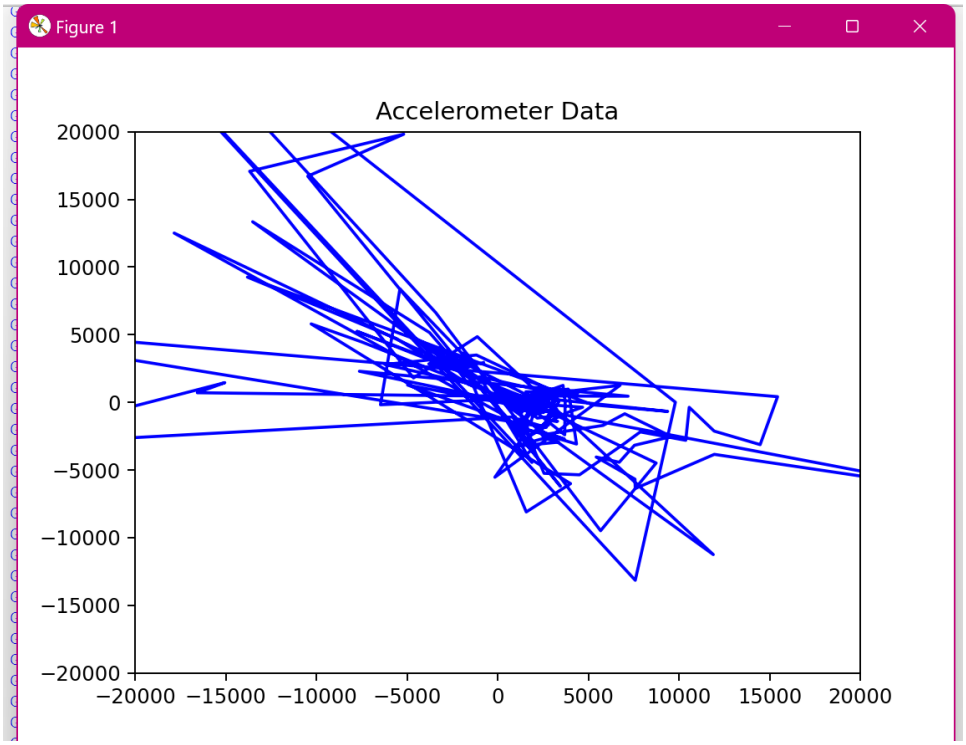
RESULTS

Video's link for MPU6050 sensor reading using Arduino and python:

https://youtu.be/8BGNDNRpwvg?si=cyIqRk8dAvkJYIZa



Real Time Accelerometer Data plotted in graph using matplotlib



Video link for live graph using matplotlib:

https://youtube.com/shorts/OGARBcgo1M4

DISCUSSION

The MPU6050 sensor is a great tool for motion tracking, position estimation and orientation sensing. The Arduino code provided in the report is used to read and print the accelerometer and gyroscope data in the Serial Monitor. The combination of accelerometer and gyroscope data allows for more accurate motion analysis.

PART B

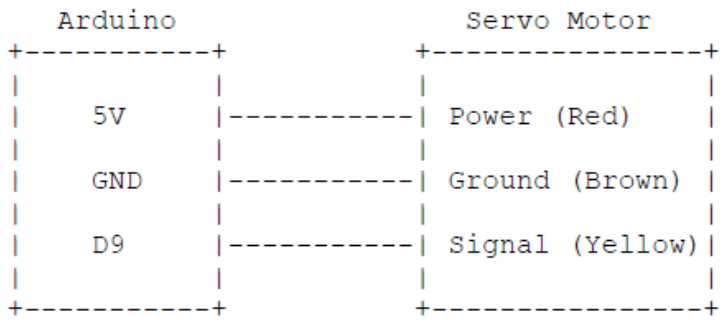
PROCEDURE

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 colours
- USB cables to connect the Arduino board and the RFID reader to computer.
- Computer with Arduino IDE and Python installed
- Datasheets and Manuals
- Power Supply (optional)
- Mounting Hardware (for the servo)

Experimental Setup

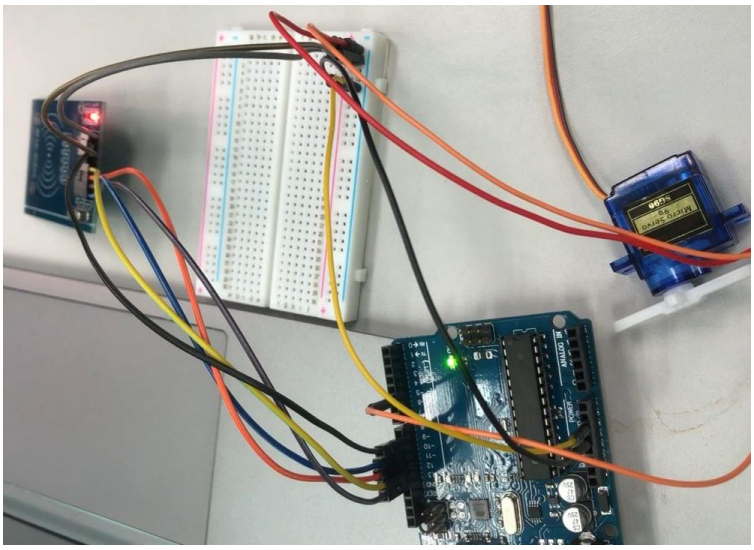
Servo Motor Wiring:

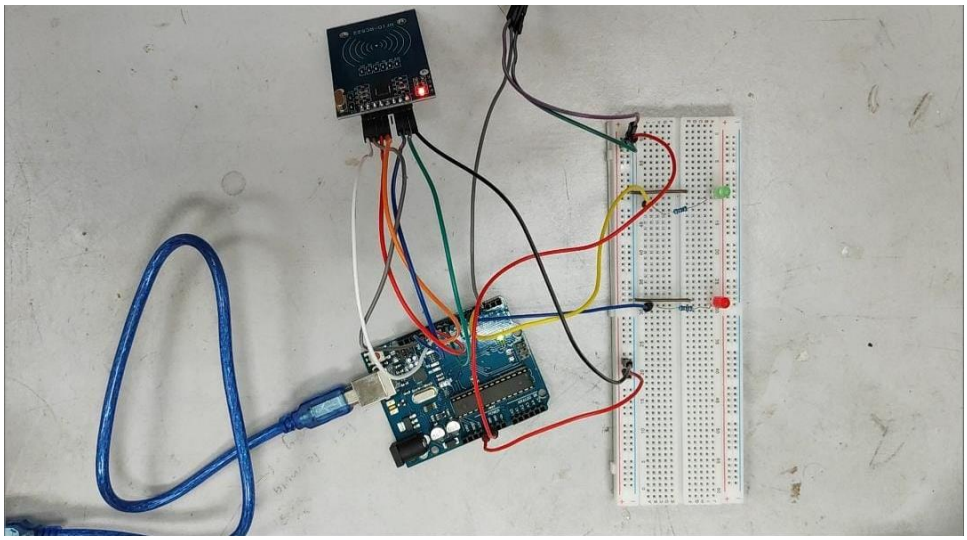


Methodology

1. The servo’s power wire is connected to the 5V output on the Arduino.
2. The servo's ground wire (usually brown or black) is connected to one of the ground (GND) pins on the Arduino.
3. Then the servo's signal wire (usually orange or yellow) is connected to the PWM pins on the Arduino.
4. Make sure that we have a common ground connection between the Arduino and the servo motor to complete the circuit.

Connection:



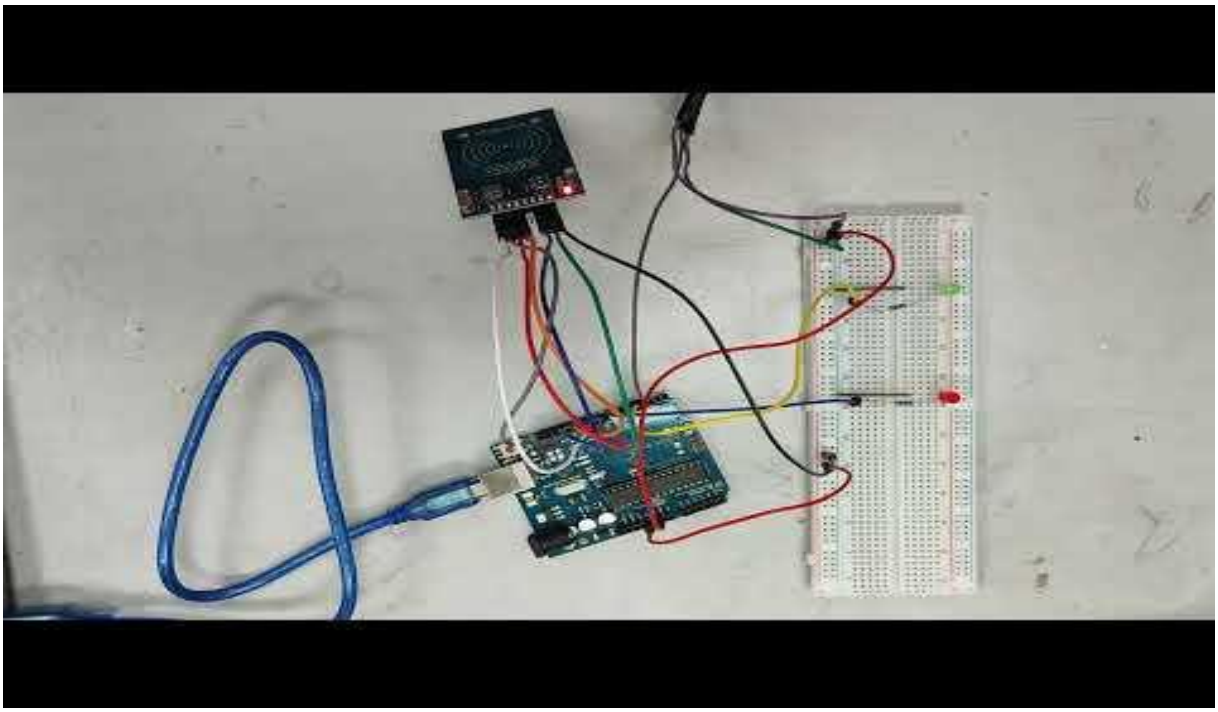


RESULTS

This is a two-part experiment. In the first, we utilised RFID to detect a valid card, the servo will spin, and a green LED will light up if the correct card is identified. A red LED will turn on and the servo will stop rotating if a card that is either unregistered or recognised is read. The Arduino IDE is the only program used in that section of the experiment.

Video link below:

https://youtu.be/-p2DtacKh2c?si=JKpA0sCZI0j2Wkce



The Coding:

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

```
#define SS_PIN 10 // MOSI of RFID
#define RST_PIN 9 // RST of RFID
#define LED_G 4    // Define green LED pin
#define LED_R 5    // Define red LED
MFRC522 mfrc522(SS_PIN, RST_PIN); // Create MFRC522 instance.
Servo myServo; // Define servo name


void setup()
{
  Serial.begin(9600); // Initiate a serial communication
  SPI.begin(); // Initiate SPI bus
  mfrc522.PCD_Init(); // Initiate MFRC522
  myServo.attach(3); // Servo pin
  myServo.write(90); // Servo start position
  pinMode(LED_G, OUTPUT);
  pinMode(LED_R, OUTPUT);
  Serial.println("Put your card to the reader...");
  Serial.println();
}


void loop()
{
  // Look for new cards
  if (!mfrc522.PICC_IsNewCardPresent())
  {
    return;
  }
  // Select one of the cards
  if (!mfrc522.PICC_ReadCardSerial())
  {
    return;
  }
  // Show UID on the serial monitor
  Serial.print("UID tag :");
  String content = "";
  byte letter;
  for (byte i = 0; i < mfrc522.uid.size; i++)
  {
    Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
    Serial.print(mfrc522.uid.uidByte[i], HEX);
    content.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " "));
    content.concat(String(mfrc522.uid.uidByte[i], HEX));
  }
  Serial.println();
  Serial.print("Message : ");
  content.toUpperCase();
  if (content.substring(1) == "83 B2 1A 11")
  {
    Serial.println("Authorized access");
    Serial.println();
```

```
    delay(500);
    digitalWrite(LED_G, HIGH);
    myServo.write(180);
    delay(500);
    myServo.write(0);
    digitalWrite(LED_G, LOW);
  }
  else
  {
    Serial.println("Access denied");
    digitalWrite(LED_R, HIGH);
    delay(1000);
    digitalWrite(LED_R, LOW);
  }
}
```

For the second part, we integrated the same Rfid but this time using python and Arduino. The function of this circuit is almost the same as the previous experiment except the serial monitor will print different messages depending on the cards' address.

https://youtube.com/shorts/eH4Y1Z81fQ0?si=jN19zMsK5SxqCk6x

The code:

```
import serial
import json


with open("people.json", "r") as f:
    VIP = json.load(f)


with serial.Serial('COM9', 9600) as ser:
    print("RFID Data ready!")


    try:
        while True:
            data = ser.readline().decode()
            if data.startswith("Card UID:"):
                card = data.split(":")[1].strip()
                if card in VIP:
                    print(VIP[card]["Name"])
                    ser.write("A".encode())
                else:
                    print("Hello?")
                    ser.write("B".encode())
```

```
    except KeyboardInterrupt:
        ser.close()
        print("Serial Close")
```

The Json code:

```
{
    "83 B2 1A 11": {
        "Name":"Fiq"
    },
    "35 B2 15 AD": {
        "Name":"Nis"
    }
}
```

DISCUSSION

For this experiment, to integrate the RFID we need to download the Arduino Library MFRC522.h and need to find out the address of the RFID card. With this information we can program this to light up the LED and move the servo motor.

CONCLUSION

Firstly, the serial communication between Python and Arduino improves the functionality of programs to the next levels by adding a layer of complexity to codes. Secondly, MPU6050 accelerometer gyroscope sensor is a great choice of a module with low cost and high accuracy for projects involving the measurement of motion-related parameters such as acceleration and velocity. Finally, MFRC522 RFID reader module is a suitable module that can be used for a much easier method of communication and data transfer compared to barcode and QR code, as the tag does not need to be within the RFID reader's line of sight for it to be functional.

RECOMMENDATIONS

REFERENCES

APPENDICES

Jobit Joseph (2022, May 16). *How Does the MPU6050 Accelerometer & Gyroscope Sensor Work and Interfacing It With Arduino*. Circuit Digest. Retrieved November 5, 2023, from https://circuitdigest.com/microcontroller-projects/interfacing-mpu6050-module-with-arduino

Jobit Joseph (2022, May 18). *Interfacing RFID Reader With Arduino*. Circuit Digest. Retrieved November 5, 2023, from https://circuitdigest.com/microcontroller-projects/interfacing-rfid-reader-module-with-arduino

ACKNOWLEDGEMENT

We would like to express our gratitude to Dr. Wahju Sediono, one of the lecturers of this course, mechatronic system integration 1 (MCTA 3203) for his guidance while the experiment is being carried out.

STUDENT'S DECLARATION

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 person.

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 contributors to the report.

We, therefore, agreed unanimously that this report should be submitted for marking and this final printed report has been verified by us.

Signature: *Ghousiah* Name: Noorul Ghousiah Binti Noordeen Sahib Matric Number: 2118298 Contribution: abstract, introduction, conclusion	Read Understand Agree	/ / /
Signature: *hanis* Name: hanis Matric Number: 2020590 Contribution:	Read Understand Agree	/ / /
Signature: *Afiq* Name: Muhammad Afiq bin Ahmad Matric Number: 2115203 Contribution: Materials setup, methadology	Read Understand Agree	/ / /
Signature: *ayunaziera* Name: Nur Ayu Naziera Binti Rosli Matric Number: 2119202 Contribution: material setup methodology (part A), result	Read Understand Agree	/ / /

Signature: *Sarah* Name: Sarah Aishah binti Sa'aid Hazley Matric Number: 2117600 Contribution: Discussion and Results	Read	/
	Understand	/
	Agree	/