MCTA 3203 (MECHATRONICS SYSTEM INTEGRATION)
WEEK 4

TITLE:

Serial and USB interfacing with microcontroller and computer-based system (2): Sensors and actuators

SEMESTER 1, 24/25

SECTION 1 – GROUP 9

LECTURER:
DR. WAHYU

no	Name	Matric number
1	AZMI BIN BASHARUDIN	2211387
2	HAMZAH FAISAL	2219537
3	KHAIRULDDIN BIN ZULKIFLEI	2210527
4	MUHAMAD IZZUDIN BIN MUHAMAD	2219735
5	MUHAMMAD AIMAN BIN MOHD RODZI	2118813

DATE OF EXPERIMENT: 30 OCTOBER 2024
DATE OF SUBMISSION: 5 NOVEMBER 2024

ABSTRACT

This experiment involved using an MPU6050 sensor interfaced with an Arduino board to acquire accelerometer and gyroscope data, which was then transmitted to a PC via the serial port. With the Arduino code handling data acquisition and transmission, a Python script on the PC received and displayed the real-time sensor data. The setup allowed for data collection, visualization, and analysis, facilitating interaction with the sensor and enabling experimentation in various applications, from motion tracking to gesture recognition.

In the RFID-based access control experiment, an Arduino, and Python are used to validate RFID cards. The Arduino code distinguishes between authorized and unauthorized cards, using LEDs to signal access approval or denial. When an authorized card is detected, the green LED lights up, and a servo motor activates, while an unauthorized card causes the red LED to illuminate. This experiment offers a practical illustration of how RFID technology can be applied to access control systems, demonstrating the integration of hardware components for secure access management

TABLE OF CONTENT

1. INTRODUCTION

This experiment focused on the integration of sensors in an electronic environment. Two sensors were used in the experiment which is the MPU6050 sensor and an RFID sensor. An MPU6050 is a motion tracking device which contains a 3-axis accelerometer, 3-axis gyroscope, and Digital Motion Processor (DMP). The module used for this purpose is also referred to as an Inertial Measurement Unit (IMU). This module is used in the first part of the experiment. The first part of the experiment goals is to obtain the digital readings of the 6 gyroscope and accelerometer axis from the IMU. Then, the IMU is worn by the experimenter's hand and obtains the readings of some hand gestures and identifies them .

The second experiment revolves around the integration of an RFID and a servo motor. The aim here is to create an access control system that authenticates users using RFID card by integrating RFID card authentication and servo motor control, with the help of Python and Arduino. Python is used to communicate with the reader, receive data from RFID cards, and make access control decisions. Additionally, LEDs are incorporated into the setup for visual feedback

2. MATERIALS AND EQUIPMENT

PART A:

- Arduino board
- MPU6050 sensor
- Jumper wires
- USB cable
- Power supply
- LEDs

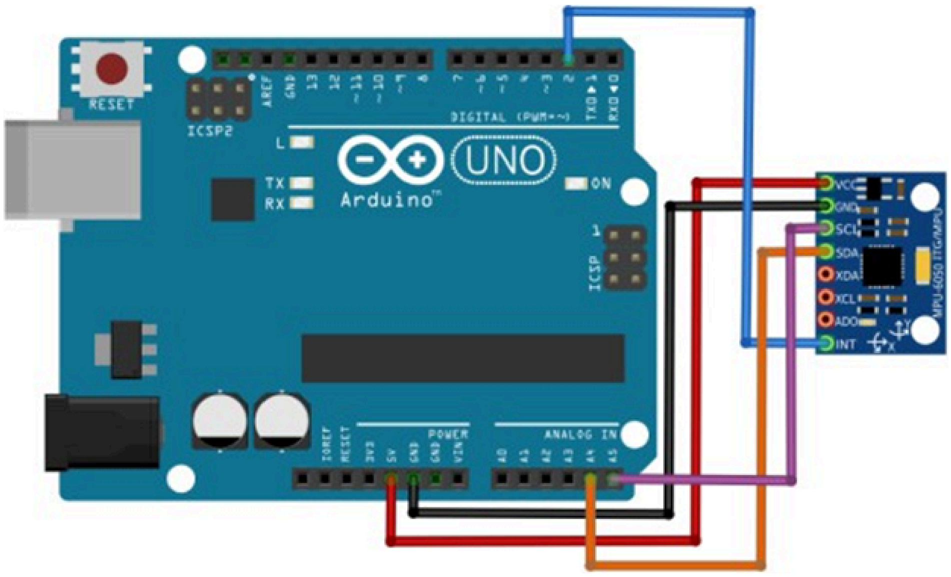
PART B:

- Arduino board
- RFID card reader with USB connectivity
- RFID tags or cards
- Servo Motor
- Jumper wires
- Breadboard
- LEDs
- USB cables
- Computer with Arduino IDE and Python installed
- Datasheets and Manuals
- Power Supply (optional)
- Mounting Hardware (for the servo)

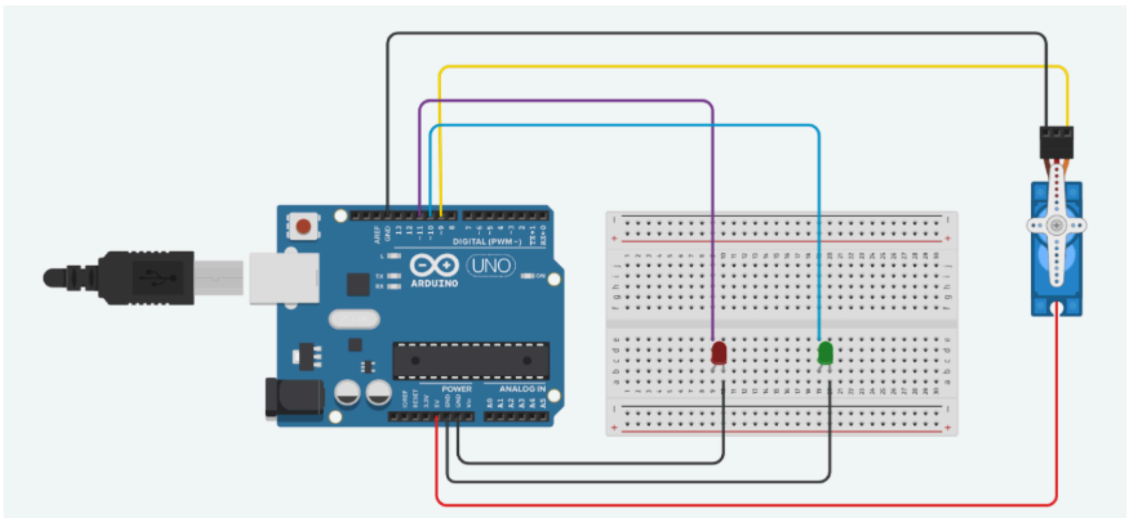
3. EXPERIMENT SETUP

PART A:

1. 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).
2. Connect the power supply and ground of the MPU6050 to the Arduino's 5V and GND pins.
3. Ensure that the Arduino board is connected to your PC via USB.



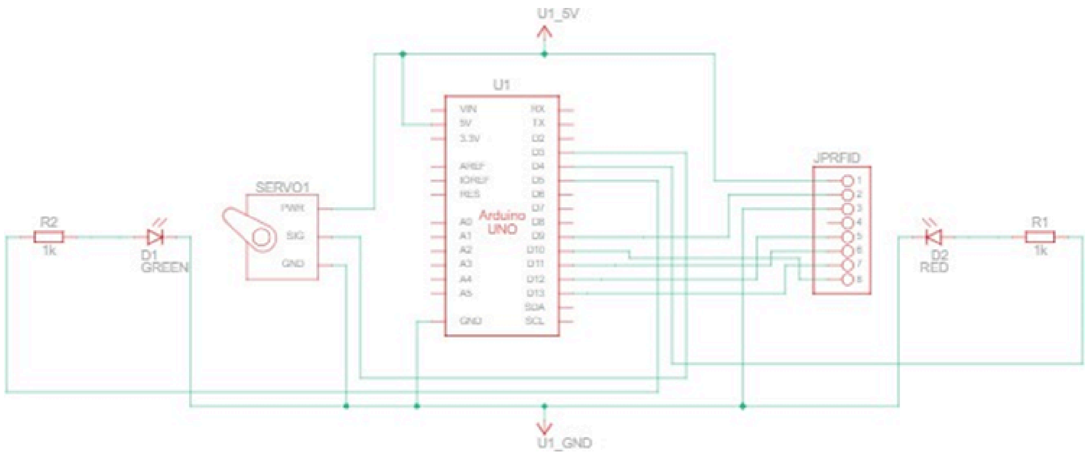
PART B:



```
     Arduino                          Servo Motor
  +-----------+             +----------------+
  |           |             |                |
  |    5V     |-----------| Power  (Red)     |
  |           |             |                |
  |    GND    |-----------| Ground (Brown)   |
  |           |             |                |
  |    D9     |-----------| Signal (Yellow)  |
  |           |             |                |
  +-----------+             +----------------+
```

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

- Connect the servo's ground wire (usually brown or black) to one of the ground (GND) pins on the Arduino.
- Connect the servo's signal wire (usually orange or yellow) to one of the PWM pins on the Arduino (e.g., pin 9).
- Ensure that you have a common ground connection between the Arduino and the servo motor to complete the circuit.
- 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.



4. METHODOLOGY

PART 4A: MPU6050 sensor

This experiment involved connecting an MPU6050 sensor to an Arduino board and reading its accelerometer and gyroscope data. The Arduino sent the data to a PC via serial port at a baud rate of 9600. Python software on the PC presented sensor data, enabling interaction, data collection, and analysis for the experiment.

Arduino Code :

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

MPU6050 mpu;

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

    if (mpu.testConnection()) {
        Serial.println("MPU6050 connected successfully.");
    } else {
        Serial.println("MPU6050 connection failed.");
    }
}

void loop() {
    int16_t ax, ay, az, gx, gy, gz;
    mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);


    Serial.print("AX: "); Serial.print(ax);
    Serial.print(" AY: "); Serial.print(ay);
    Serial.print(" AZ: "); Serial.print(az);
    Serial.print(" GX: "); Serial.print(gx);
    Serial.print(" GY: "); Serial.print(gy);
    Serial.print(" GZ: "); Serial.println(gz);

    delay(100);
}
```

Python Script :

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

ser = serial.Serial('COM6', 9600)
time.sleep(2)
x_positions = [0]
y_positions = [0]
gestures = []
THRESHOLD_X = 5000
THRESHOLD_Y = 5000
dt = 0.1

def detect_gesture(ax, ay, gx, gy):

   if abs(ax) > THRESHOLD_X and abs(gy) > THRESHOLD_Y:
       return "Wave"
   elif ax < -THRESHOLD_X and ay > THRESHOLD_Y:
       return "Swipe Left"
   elif ax > THRESHOLD_X and ay > THRESHOLD_Y:
       return "Swipe Right"
   return "Detecting"


def update_plot(frame):
   global x_positions, y_positions


   line = ser.readline().decode().strip()
   if "AX" in line and "AY" in line:
       try:
           ax = int(line.split("AX: ")[1].split(" ")[0])
           ay = int(line.split("AY: ")[1].split(" ")[0])
           gx = int(line.split("GX: ")[1].split(" ")[0])
           gy = int(line.split("GY: ")[1].split(" ")[0])
           gesture = detect_gesture(ax, ay, gx, gy)
           gestures.append(gesture)
           new_x = x_positions[-1] + ax * (dt ** 2)
           new_y = y_positions[-1] + ay * (dt ** 2)
           x_positions.append(new_x)
           y_positions.append(new_y)
           if len(x_positions) > 100:
               x_positions.pop(0)
               y_positions.pop(0)
           ax_plot.clear()
           ax_plot.plot(x_positions, y_positions, marker='o')
```

```
            ax_plot.set_xlim(-10000, 10000)
            ax_plot.set_ylim(-10000, 10000)
            ax_plot.set_xlabel("X Position")
            ax_plot.set_ylabel("Y Position")
            ax_plot.set_title(f"Hand Movement: {gesture}")
        except ValueError:
            print("Error parsing data")


fig, ax_plot = plt.subplots()
ani = FuncAnimation(fig, update_plot, interval=10, save_count=10000)
plt.show()
ser.close()
```

PART 4B: RFID READER & SERVO MOTOR

The RFID experiment uses Arduino and Python programming to verify the Data Identifier (DI) of RFID cards. The Arduino code distinguishes between approved and unauthorized cards and controls LEDs accordingly: green for access-granted cards, and red for refused cards. A servo motor activates and moves upon access being granted. This experiment demonstrates RFID-based access control and how its components work together to give or refuse access depending on card recognition.

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

#define RST_PIN 9
#define SS_PIN 10
#define GREEN_LED 7
#define RED_LED 8
#define SERVO_PIN 6

MFRC522 mfrc522(SS_PIN,
RST_PIN);
Servo myServo;
const int bufferSize = 512;
String jsonData = R"({
  "uids": [
    "E2801105200072DAF3A70ACE",
    "123456789ABCDEF012345678"
  ],
```

```
"servo_position": {
    "recognized": 90,
    "unrecognized": 0
  }
})";

DynamicJsonDocument
doc(bufferSize);
const int eepromStartAddress = 0;
String byteToHexString(byte *buffer,
byte bufferSize) {
  String result = "";
  for (byte i = 0; i < bufferSize; i++) {
    if (buffer[i] < 0x10) result += "0";
    result += String(buffer[i], HEX);
  }
  result.toUpperCase();
  return result;
}
```

```
bool isRecognizedUID(String uid) {
  JsonArray uids =
doc["uids"].as<JsonArray>();
  for (String storedUID : uids) {
    if (uid == storedUID) return true;
  }
  return false;
}


void saveToEEPROM() {
  String jsonString;
  serializeJson(doc, jsonString);

  for (unsigned int i = 0; i <
jsonString.length(); i++) {

EEPROM.write(eepromStartAddress
+ i, jsonString[i]);
  }
  EEPROM.write(eepromStartAddress
+ jsonString.length(), '\0');
}

void loadFromEEPROM() {
  char jsonBuffer[bufferSize];
  int i = 0;
  char ch;


  while ((ch =
EEPROM.read(eepromStartAddress +
i)) != '\0' && i < bufferSize - 1) {
    jsonBuffer[i++] = ch;
  }
  jsonBuffer[i] = '\0';
```

```
 DeserializationError error =
deserializeJson(doc, jsonBuffer);
  if (error) {

    Serial.println("Failed to load JSON
from EEPROM. Using default.");
    Serial.print("Error: ");
    Serial.println(error.c_str());

    deserializeJson(doc, jsonData);
    saveToEEPROM();
    Serial.println("Default JSON saved
to EEPROM.");
  } else {
    Serial.println("JSON successfully
loaded from EEPROM.");
  }
}
// Register a new UID
void registerUID(String uid) {
  JsonArray uids =
doc["uids"].as<JsonArray>();
  uids.add(uid);
  saveToEEPROM();
  Serial.println("New UID registered: "
+ uid);
}
void setServoAngle(int angle) {
  myServo.write(angle);
  delay(500);
}
void setup() {
  Serial.begin(9600);
  SPI.begin();
  mfrc522.PCD_Init();
  myServo.attach(SERVO_PIN);

  pinMode(GREEN_LED, OUTPUT);
  pinMode(RED_LED, OUTPUT);

  loadFromEEPROM();

  Serial.println("Ready to scan RFID
cards.");
}
```

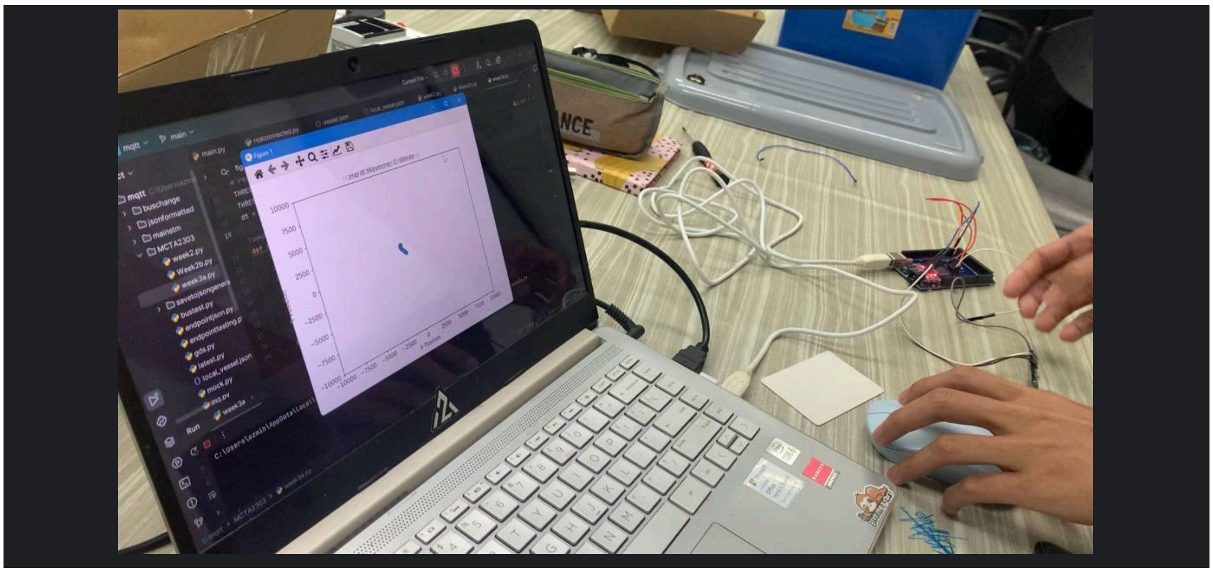
```
void loop() {
  if
(!mfrc522.PICC_IsNewCardPresent
() ||
!mfrc522.PICC_ReadCardSerial())
{
    return;
  }

  String uid =
byteToHexString(mfrc522.uid.uidB
yte, mfrc522.uid.size);
  Serial.println("Scanned UID: " +
uid);
  if (isRecognizedUID(uid)) {
     digitalWrite(GREEN_LED,
HIGH);
    digitalWrite(RED_LED, LOW);
    int angle =
doc["servo_position"]["recognized"
];
    setServoAngle(angle);
    Serial.println("UID recognized.
Servo set to angle: " +
String(angle));
  } else {

    digitalWrite(GREEN_LED,
LOW);
    digitalWrite(RED_LED, HIGH);
    int angle =
doc["servo_position"]["unrecognize
d"];
    setServoAngle(angle);
    Serial.println("UID
unrecognized. Servo set to angle: "
+ String(angle));
    registerUID(uid);
    Serial.println("UID has been
registered automatically.");
  }
 delay(2000);
  digitalWrite(GREEN_LED,
LOW);
  digitalWrite(RED_LED, LOW);
}
```

5. DATA COLLECTION

<u>PART 4A: MPU6050 sensor</u>



For this experiment, acceleration and rotational velocity data are continually recorded in three dimensions (x, y, and z) via the MPU6050 sensor. This data is collected to help understand the motion and direction of the gestures detected. Additionally, this data is essential for real-time monitoring and knowing the behaviour of the sensor in response to physical movements.

<u>PART 4B: RFID READER & SERVO MOTOR</u>

As for this experiment, the RFID reader scans RFID cards to get their unique identification or UID. User authentication requires this UID. Next, the system determines the authenticity of the RFID card that has been presented, indicating the status by the illumination of LEDs (green indicating permitted access and red indicating unauthorised access). Despite the constraints in the experiment, ideally, the servo motor should also move as part of the data-collecting process.

6. DATA ANALYSIS

PART 4A: MPU6050 sensor

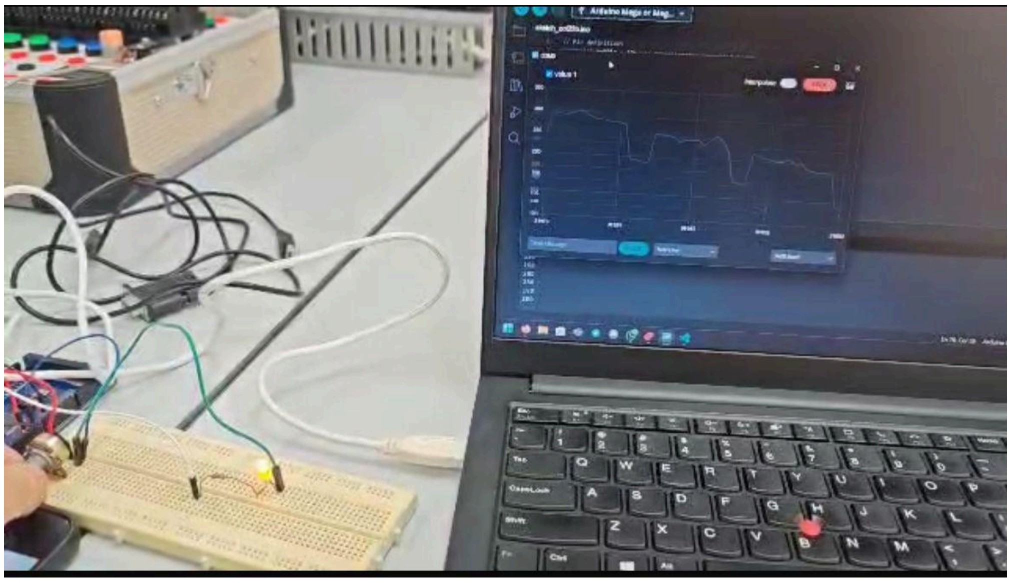
When a new gesture is detected, the Arduino sends a message, and Python displays the detected gesture (for example, "Action for Gesture: 1"). Essentially, when the MPU module is held upward (the component is upward, the label is downward, and the wire connection is exactly in the front), it will read as gesture 1, indicating that it detects a different motion in the x-axis. Meanwhile, if the MPU module is pushed or tilted to the right, it will be read as gesture 2 (different in the y-axis). However, the reading displayed in Python is not as rapid as the reading presented on the Arduino's serial monitor. When using the Python code, the MPU module must be jerked or tapped harder for it to read the motion.

PART 4B: RFID READER & SERVO MOTOR

Green LED illuminates indicating the RFID reader identifies an authorised UID is scanned and access is granted otherwise red LED will turn on when a different UID card is used which led to denied access. The servo motor is a critical component for physically granting access, but it's not behaving as expected

7. RESULT

Sensor integration for real-time gesture detection, ‘Hand Gesture Recognition’ using MPU6050 was successfully established using the connection between the sensor and Arduino board. The communication link between the two allows access to the motion and orientation data from the sensor which is then successfully displayed in python, using code that utilised the accelerometer and gyroscope measurements from the sensor. In the second part of the experiment, USB integration for RFID card readers and the computer also was successfully established. The program correctly grants or rejects the specific tag recorded once the coding is run and the RFID card is tapped on the reader. The green LED turns on when access is allowed, and the red LED turns on when access is refused. This is the result that can be seen.The integration that was achieved by arduino and Python integration allowed the access control system to authenticate users using RFID cards. In conclusion, the experiment successfully met its goals, offering practical insight and knowledge into setting up it. Thus demonstrating the potential and versatility of sensor integration and USB interfacing.



8. DISCUSSION

PART 4A: MPU6050 sensor

Arduino

The Arduino code initializes the MPU6050 sensor and reads its accelerometer and gyroscope data. The data was then transferred to the computer using the serial port. The Arduino code includes a gesture detection system that uses preset accelerometer thresholds to identify specific hand movements. Recognized hand gestures will be relayed to the PC.

Python

Python was used to receive and process the data transmitted by the Arduino.
The 'Pyserial' library facilitates communication between Arduino and Python. The Python code interprets input and responds to specified gestures, as demonstrated by the Arduino code.

MPU6050 sensor

Hand motion data was collected using the MPU6050 sensor, which includes a gyroscope and accelerometer. Using Arduino, initialize the sensor, set a threshold value, and receive real-time sensor data. Sensor data was used to identify and categorize gestures as one of two types.

TASK:

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 categorise these gestures using the collected sensor data. Additionally, visualise the paths of hand movement in an x-y coordinate system.

To construct and visualize a graph in Python, we should utilize the matplotlib module. When gestures are identified, data and readings are displayed in the Python script graph. The MPU6050 sensor's gesture detection should provide the desired results.

<u>PART 4B: RFID READER & SERVO MOTOR</u>

Arduino

The Arduino code in the technique section serves as a physical bridge between RFID and Python, facilitating communication. Arduino is designed for basic access control using LEDs, a servo motor, and an RFID reader. Access is given with a specific card UID, but denied for other UIDs. The code adds a visual indicator, such as a green LED, that lights up when the RFID reader finds a recognized UID and turns red when an unrecognised card is read.

Servo motor error

Once access to a certain UID card is allowed, the Servo motor should spin or move.
The Arduino code aims to provide a basic example for controlling servos. However, the servo could not move or turn since it required 5V.

EXPECTED:

If the UID card is granted access, the servo motor will turn and the green LED will illuminate. If the UID card is refused entry, the servo motor stops and the red LED illuminates.

OBSERVED:

If the UID card is granted access, the servo motor will turn and the green LED will light up. If the UID card is denied entrance, the servo motor stops and the red LED turns on.

TASK:

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 unrecognised card is read. Incorporate structured JSON data handling within your code for better organisation and flexibility. Add some options for the user to freely set the angle position of the servo.

Green and red LEDs are added to alert the user. Using the Arduino and Python code provided in the methodology section, the green LED lights up when the card UID is granted access, while the red LED lights up when the card UID is incompatible and access is denied.

9. CONCLUSION

PART 4A: MPU6050 sensor

We successfully integrated the MPU6050 sensor with an Arduino board to capture motion and orientation data. Our hand gesture recognition system uses the sensor's accelerometer and gyroscope measurements to identify gestures based on preset thresholds. The data is then transmitted to the computer via serial port and Python. The Python script provided real-time data in response to user motions.

PART 4B: RFID READER & SERVO MOTOR

We successfully linked an RFID card reader with an Arduino board for user authentication in an access control system. The Arduino-python communication was required for authenticating and accessing the card to read its serial number. Two LEDs (red and green) were added to the connection: 'green' for card recognition or serial number reading, and 'red' for card unrecognition. The access control system was successful, however the servo motor required a separate power source.

10. RECOMMENDATION

The need for improvement is always needed to make sure the accuracy, reliability, and adaptability of the experiment results. Researchers can enhance their techniques and cut down on possible error sources through continuous improvement. The findings become more reliable as a result, yielding more exact and accurate outcomes. Below are a few recommendations that we can improve in the future.

1. Enhance gesture detection: For part 4A, we can consider refining the gesture recognition system. Instead of only detecting two simple gestures which is gesture and gesture 2, we should expand the system so that it can detect more complex gestures using additional sensors if needed by updating the Arduino and Python codes.

2. Testing and calibration: In the future, we should ensure that the sensor is calibrated properly in the experiment so that the sensor data is accurate. For part 4A, testing and calibration are also important to maintain and improve the reliability of the system and its results.

3. Servo motor troubleshooting: In part 4B, the servo motor is not moving as we expected. It is critical to address the servo motor issue in the experiment so that it will not happen in the future. For example, we need to ensure that the power supply is prepared if the servo needs it, the connections are correct and secure, and the code to control the movement of the servo motor is written properly.

4. User feedback: In part 4B we can improve the user experience by adding audio feedback to the user. Instead of only using LEDs as indicators, we can use buzzers to indicate feedback to the user whether their card UID is granted or denied.

5. Power Supply for servo motor: We should consider using a separate and steady power supply for the servo motor. Servo motors can consume a substantial amount of electricity, therefore a dedicated power supply helps prevent interference with other components and assure reliable performance.

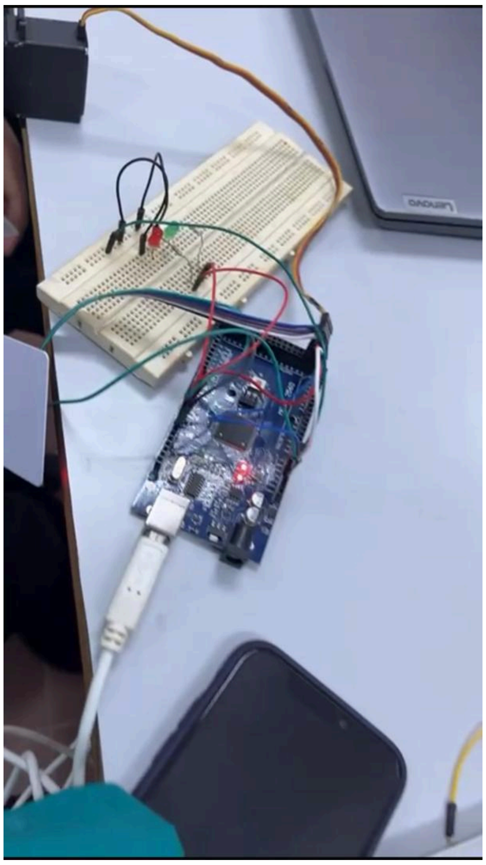
By implementing these recommendations, we can improve the system's reliability and efficiency. Hence, it will reduce the errors that might occur in the future.

11. REFERENCES

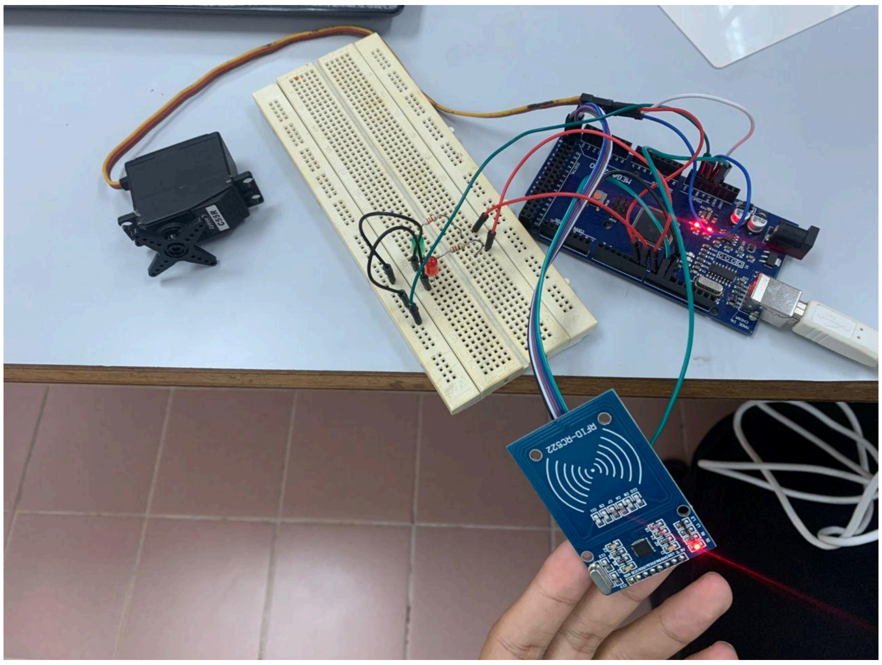
- Kumar, S., Das, M. R., Kushalkar, R., Venkat, N., Gourshete, C., & Moudgalya, K. M. (2021, June). Microcontroller Programming with Arduino and Python. https://www.researchgate.net/profile/Sudhakar-Kumar-2/publication/353849431_Microcontroller_programming_with_Arduino_and_Python/links/61177e891ca20f6f861ecc9c/Microcontroller-programming-with-Arduino-and-Python.pdf

- Zulkifli. (2014). Mechatronics Interfacing Lab Manual, (Rev. ed.). Unpublished Class Materials

12. APPENDICES



Final version for part A


Final version of part B

13. ACKNOWLEDGEMENT

We want to thank everyone who helped, guided, and supported us throughout this endeavour. First and foremost, we thank Assoc Prof Dr. Zulkifli Bin Zainal Abidin, Dr. Ali Sophian, and also Dr. Wahyu Sediono for providing extensive guidance and supervision during the experiment. Their comments, suggestions, and excitement helped us understand Arduino programming.

Our fellow group members deserve special recognition for their teamwork and support. Our conversations, information sharing, and problem-solving sessions dramatically improved our understanding of the experiment's topics and the overall learning experience. Our group members' joint efforts not only enhanced our learning experience, but also greatly contributed to the effective completion of this project.

14. STUDENTS 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 no further improvement on the reports is needed from any of the individual's contributors to the report.

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

SIGNATURE	[illegible]	
NAME	Azmi bin basharudin	**Read** ✓
MATRIC NUMBER	2211387	**Understand** ✓
CONTRIBUTION	Introduction, material and equipment	**Agreed** ✓

SIGNATURE	[illegible]	
NAME	Hamzah faizal	**Read** ✓
MATRIC NUMBER	2219537	**Understand** ✓
CONTRIBUTION	Experiment setup, methodology, data collection	**Agreed** ✓

SIGNATURE	[illegible]	
NAME	Khairulddin bin zulkiflei	**Read** ✓
MATRIC NUMBER	2210527	**Understand** ✓
CONTRIBUTION	Data analysis, result, discussion	**Agreed** ✓

SIGNATURE	[illegible]	
NAME	Muhamad Izzudin Bin Muhamad	**Read** ✓
MATRIC NUMBER	2219735	**Understand** ✓
CONTRIBUTION	Conclusion, recommendation, references	**Agreed** ✓

SIGNATURE	aiman	
NAME	Muhammad aiman bin mohd rodzi	**Read** ✓
MATRIC NUMBER	2118813	**Understand** ✓
CONTRIBUTION	Appendices, acknowledgement, students declaration	**Agreed** ✓