

**LAB 3:**

**Parallel, Serial and USB interfacing with microcontroller and computer based system: continuing the exploration of interfacing techniques for sensors and actuators**

**MCTA 3202**

GROUP F

GROUP MEMBERS MATRIC NO

1. Muhammad Naufal bin Mohammad Bakri 2110333
2. Muhammad Khairul Aiman bin Mohamad Basari 2113571
3. Muhammad Nabil Ikhmal Bin Khairil Rizal 2114577
4. Muhammad Zakwan Zikri bin Irwan Affandi 2111033
5. Che Muhammad Danial Bin Che Suhaimi 2119075

DATE OF SUBMISSION :

8th November 2023

# **Table of Contents**

| CONTENT | PAGE | REMARK |
| --- | --- | --- |
| **PART A** | | |
| INTRODUCTION | 3-4 |  |
| MATERIALS AND EQUIPMENT | 4 |  |
| EXPERIMENTAL SETUP | 5 |  |
| METHODOLOGY | 5 |  |
| RESULT (OBSERVATION) | 6-8 |  |
| DISCUSSION | 8 |  |
| CONCLUSION | 9 |  |
| RECOMMENDATION | 9 |  |
| REFERENCES | 9 |  |
| APPENDICES | 10-14 |  |
| **PART B** | | |
| INTRODUCTION | 15-16 |  |
| MATERIALS AND EQUIPMENT | 16 |  |
| EXPERIMENTAL SETUP | 17 |  |
| METHODOLOGY | 17 |  |
| RESULT (OBSERVATION) | 17-19 |  |
| DISCUSSION | 19 |  |
| CONCLUSION | 20 |  |
| RECOMMENDATION | 20 |  |
| REFERENCES | 20 |  |
| APPENDICES | 21-23 |  |
| STUDENT’S DECLARATION | 24 |  |

# **PART A**

## **Abstract**

This lab report investigates the communication between an Arduino microcontroller and a Python program for data input. The objective of this experiment was to establish a bi-directional communication link between the Arduino and Python, allowing the exchange of data. The serial communication protocol was employed as the means to achieve this interactivity.

The setup involved connecting an Arduino board to a computer via USB and developing a Python script to communicate with it using the PySerial library. Data transmission was initiated from both ends, enabling data to be sent from the Arduino to the Python program and vice versa.

Various experiments were conducted to evaluate the performance of this communication method, including the successful transmission and reception of data in different formats, such as integers, strings, and sensor readings. The results showed reliable data transfer with minimal data loss and latency, making it a suitable solution for interfacing Arduino with Python.

Overall, this experiment demonstrated the successful establishment of communication between an Arduino microcontroller and Python, emphasizing the importance of serial communication in bridging the gap between hardware and software applications. This practical implementation holds great potential for numerous applications, including IoT projects, data logging, and real-time monitoring, offering a versatile and efficient way to input and output data between the two platforms.

## 

## **Introduction**

### Overview of the experiment's purpose and objectives

The primary objective of this experiment with the MPU-6050 is to measure and analyze motion and orientation data. From the obtained data, it is then being utilized to create a straightforward hand gesture recognition system by using Python script.

### Background information and relevant theory or concepts

Firstly, the MPU6050 is an Inertial Measurement Unit (IMU) that has a 6 axis measurement which is 3 acceleration axis and 3 orientation axis. This data can be accessed in real time giving the opportunity for us to do gesture reading.

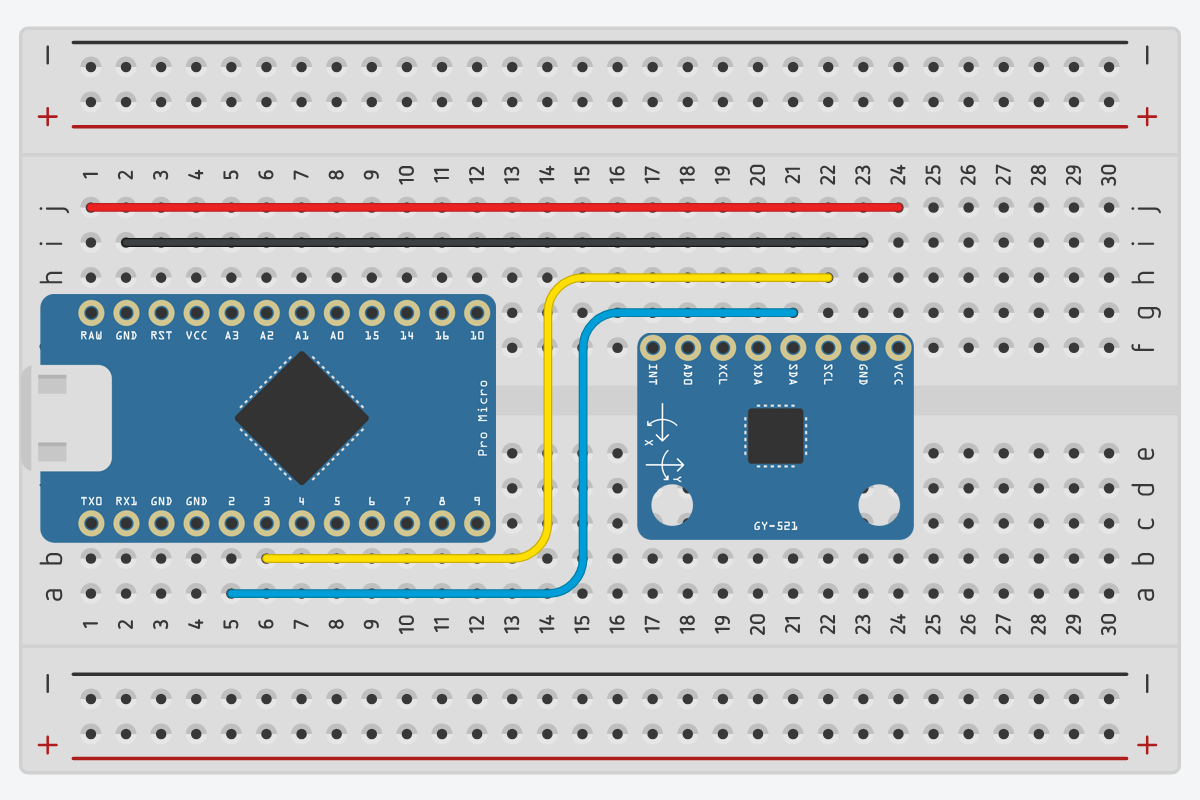
Secondly, in this experiment we have to make a connection between Arduino and the MPU6050 sensor. The way we are going to do it is by using the I2C protocol. This protocol uses two wires to establish a connection. The first wire is SCL which is the serial clock pin. The second one is the SDA; serial data pin where data is being sent between both devices.

## **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**

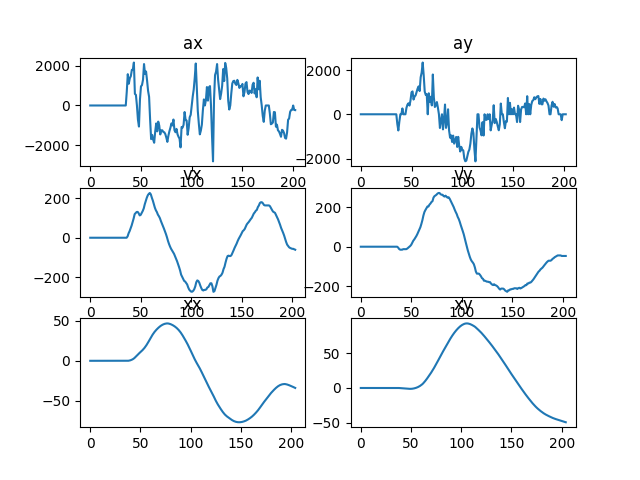


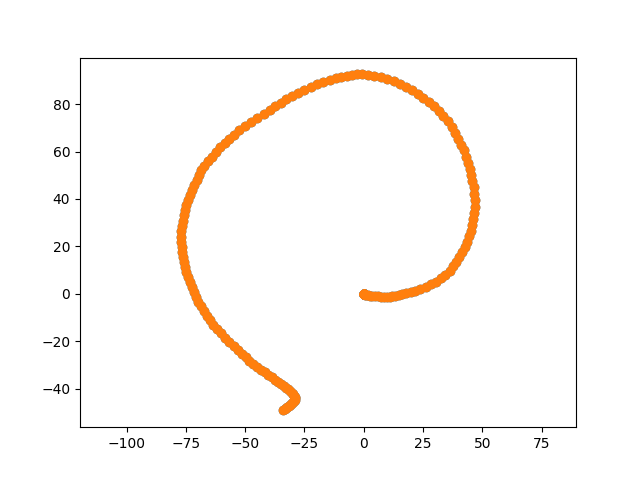
## **Methodology**

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.

## 

## **Data Collection**





<https://raw.githubusercontent.com/NotLafuan/GROUP-F-MCTA-3203/main/Week%204/imu/data.txt>

## **Data Analysis**

The scatter plot graph shows the path taken by the IMU throughout the data collection period. Since the raw data from the IMU is acceleration, we have to do integration on the data two times to get the displacement. These equations are used to do the calculation.

The first graph shows this integration process from acceleration to velocity to displacement.

## **Results**



<https://github.com/NotLafuan/GROUP-F-MCTA-3203/raw/main/Week%204/Group%20F%20Week%204%20Part%20A.mp4>

## **Discussion**

From our testing, we found out that the MPU6050 has noise in the data. We remove this noise while idle by setting the minimum value for the data to be registered. In our case, the maximum and minimum noise value on idle is around 200 and -200, so any data that is in this range is considered 0. Another way to solve this problem is by using a butterworth low pass filter but from our finding it does not have much impact on the displacement result. The idle value plays a more important role in making sure the data is accurate.

## 

## **Conclusion**

In this experiment, the main objective is to measure and analyze motion and orientation data. We successfully utilize the obtained data to create a simple hand gesture recognition system by using Python script. The MPU6050 sensor has proven to be a valuable component in understanding and measuring motion and orientation in various applications. Throughout this lab, we explored the fundamental principles behind the MPU6050 sensor, its components, and its working mechanism.

Through conducting this experiment, the serial communication established is robust and very reliable.Although certain constraints, such as noise sensitivity and drift over time, the MPU6050 continues to be an affordable and easily obtainable option for motion detection uses. By being aware of its strengths and weaknesses, engineers and researchers may modify their experimental settings to maximize efficiency.

## **Recommendations**

Improving the orientation estimate accuracy of the MPU6050 experiment entails combining sensor fusion techniques, such a complementing filter or Kalman filter. These algorithms address the limits of individual sensors by integrating data from the MPU6050's accelerometer and gyroscope. This successfully reduces drift and improves the accuracy of orientation data. This change calls for carefully calibrating the sensor setup, developing a reliable data logging system for thorough analysis, and modifying the code to include the selected method. The investigation should involve extensive testing to show the huge improvement possible with this strategy by comparing the accuracy of orientation estimations before and after sensor fusion.

## **References**

<https://www.arduino.cc/reference/en/language/functions/communication/serial/>

<https://www.arduino.cc/reference/en/language/functions/communication/wire/>

<https://docs.arduino.cc/learn/communication/wire>

<https://github.com/electroniccats/mpu6050>

<https://pyserial.readthedocs.io/en/latest/>

<https://cdn.sparkfun.com/datasheets/Sensors/Accelerometers/RM-MPU-6000A.pdf>

<https://www.nxp.com/docs/en/application-note/AN3397.pdf>

## **Appendices**

### Code Snippets

Arduino Code:

#include <Wire.h>

#include <MPU6050.h>

MPU6050 mpu;

int16\_t ax, ay, az, gx, gy, gz;

uint32\_t ax\_0 = 0, ay\_0 = 0;

void Calibrate()

{

unsigned int count1;

count1 = 0;

do

{

mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

ax\_0 = ax\_0 + ax; // Accumulate Samples

ay\_0 = ay\_0 + ay;

count1++;

} while (count1 != 0x0400); // 1024 times

ax\_0 = ax\_0 >> 10; // division between 1024

ay\_0 = ay\_0 >> 10;

}

void setup()

{

Serial.begin(9600);

Wire.begin();

mpu.initialize();

Calibrate();

}

void loop()

{

mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

Serial.print("Accel: ");

Serial.print(ax - (int)ax\_0);

Serial.print(" ");

Serial.print(ay - (int)ay\_0);

Serial.print(" ");

Serial.print(az);

Serial.print(" Gyro: ");

Serial.print(gx);

Serial.print(" ");

Serial.print(gy);

Serial.print(" ");

Serial.println(gz);

delay(10); // Adjust the delay as needed

}

Python Code:

imu.py

import serial

import time

time.sleep(3)

ax\_lst, ay\_lst, az\_lst, gx\_lst, gy\_lst, gz\_lst = [], [], [], [], [], []

with serial.Serial('COM11', 9600) as ser:

ax\_lst, ay\_lst, az\_lst, gx\_lst, gy\_lst, gz\_lst = [], [], [], [], [], []

while True:

try:

data = ser.readline().decode('utf-8').strip()

\_, ax, ay, az, \_, gx, gy, gz = data.split()

ax, ay, az, gx, gy, gz = \

float(ax), float(ay), float(az), float(gx), float(gy), float(gz)

ax\_lst.append(ax)

ay\_lst.append(ay)

az\_lst.append(az)

gx\_lst.append(gx)

gy\_lst.append(gy)

gz\_lst.append(gz)

print(f'Accel: {ax:7.2f} {ay:7.2f} {az:7.2f} Gyro: {gx:7.2f} {gy:7.2f} {gz:7.2f}')

except ValueError:

continue

except KeyboardInterrupt:

break

with open('data.txt', 'w+') as f:

text = ''

for ax, ay, az, gx, gy, gz in zip(ax\_lst, ay\_lst, az\_lst, gx\_lst, gy\_lst, gz\_lst):

text += f'{ax} {ay} {az} {gx} {gy} {gz}\n'

f.write(text)

plotter.py

import matplotlib.pyplot as plt

ax\_lst, ay\_lst, az\_lst, gx\_lst, gy\_lst, gz\_lst = [], [], [], [], [], []

with open('data.txt', 'r') as f:

while line := f.readline().strip():

ax, ay, az, gx, gy, gz = line.split()

ax\_lst.append(float(ax))

ay\_lst.append(float(ay))

az\_lst.append(float(az))

gx\_lst.append(float(gx))

gy\_lst.append(float(gy))

gz\_lst.append(float(gz))

# X

for i, ax in enumerate(ax\_lst):

if ax <= 200 and ax >= -200:

ax\_lst[i] = 0

vx\_lst = [0]

for i, ax in enumerate(ax\_lst):

vx\_lst.append(vx\_lst[-1] + ax \* 0.01)

xx\_lst = [0]

for i, vx in enumerate(vx\_lst):

xx\_lst.append(xx\_lst[-1] + vx \* 0.01)

# Y

for i, ay in enumerate(ay\_lst):

if ay <= 200 and ay >= -200:

ay\_lst[i] = 0

vy\_lst = [0]

for i, ay in enumerate(ay\_lst):

vy\_lst.append(vy\_lst[-1] + ay \* 0.01)

xy\_lst = [0]

for i, vy in enumerate(vy\_lst):

xy\_lst.append(xy\_lst[-1] + vy \* 0.01)

plt.subplot(3, 2, 1)

plt.title('ax')

plt.plot(ax\_lst)

plt.subplot(3, 2, 3)

plt.title('vx')

plt.plot(vx\_lst)

plt.subplot(3, 2, 5)

plt.title('xx')

plt.plot(xx\_lst)

plt.subplot(3, 2, 2)

plt.title('ay')

plt.plot(ay\_lst)

plt.subplot(3, 2, 4)

plt.title('vy')

plt.plot(vy\_lst)

plt.subplot(3, 2, 6)

plt.title('xy')

plt.plot(xy\_lst)

plt.show()

cum\_x, cum\_y = [], []

for x,y in zip(xx\_lst,xy\_lst):

cum\_x.append(x)

cum\_y.append(y)

plt.clf()

plt.axis('equal')

plt.scatter(cum\_x, cum\_y)

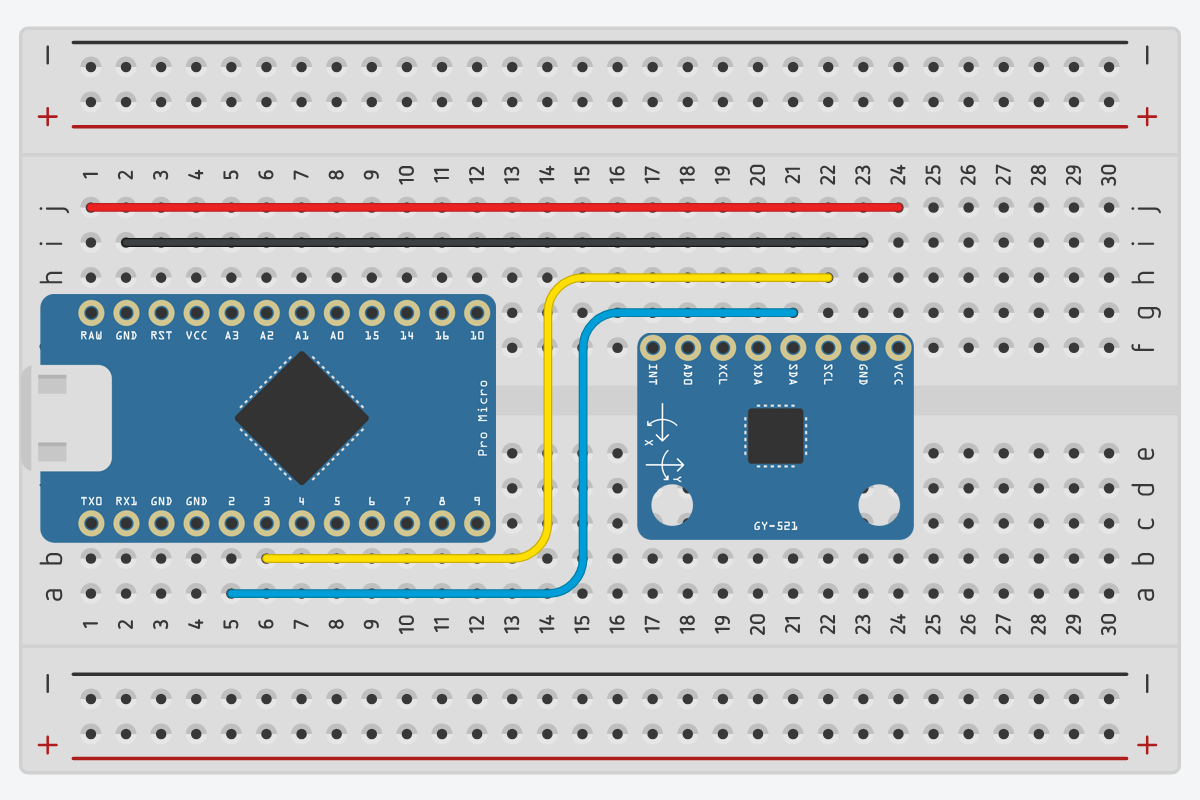
plt.pause(0.000001)

plt.axis('equal')

plt.scatter(xx\_lst, xy\_lst)

plt.show()

### Circuit Diagram



# **PART B**

## **Abstract**

This lab report investigates the communication between an Arduino microcontroller and a Python program for data input. The objective of this experiment was to establish a bi-directional communication link between the Arduino and Python, allowing the exchange of data. The serial communication protocol was employed as the means to achieve this interactivity.

The setup involved connecting an Arduino board to a computer via USB and developing a Python script to communicate with it using the PySerial library. Data transmission was initiated from both ends, enabling data to be sent from the Arduino to the Python program and vice versa.

Various experiments were conducted to evaluate the performance of this communication method, including the successful transmission and reception of data in different formats, such as integers, strings, and sensor readings. The results showed reliable data transfer with minimal data loss and latency, making it a suitable solution for interfacing Arduino with Python.

Overall, this experiment demonstrated the successful establishment of communication between an Arduino microcontroller and Python, emphasizing the importance of serial communication in bridging the gap between hardware and software applications. This practical implementation holds great potential for numerous applications, including IoT projects, data logging, and real-time monitoring, offering a versatile and efficient way to input and output data between the two platforms.

## **Introduction**

### Overview of the experiment's purpose and objectives

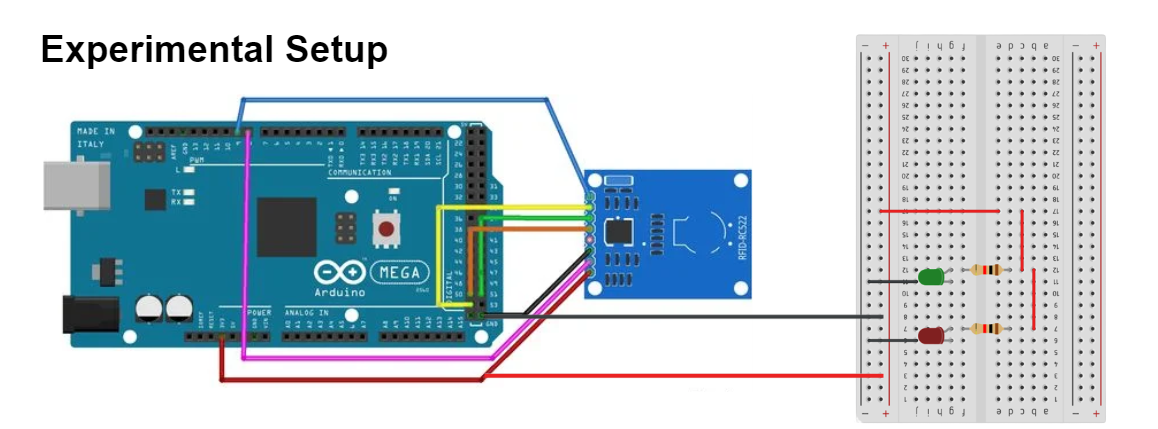
The primary objective of the experiment is to demonstrate the integrated communication between Arduino microcontroller , Python programming and an RFID reader. The Arduino works as a hardware to interface and read RFID tag information and creates a communication channel between Arduino hardware and Python program to process the data received, and then send a feedback for the Arduino to complete the action that has been specified in the codes. Its aim is to demonstrate the integration of multiple technologies allowing RFID data to be retrieved and processed through Python highlighting the potential efficiencies in different branches of systems.

### Background information and relevant theory or concepts

The Unit Identifier (UID) is firstly scanned or tapped on the RFID reader. This information is then transferred to the serial port for the python to read and then execute an if-else function. When the program has identified the type of data received, it will then gives a special character output either F or A. It is then sent to the serial port for the Arduino to read the special character that has been received and if it is an ‘F’ it will light up the LED on port number 13. However, if it is the letter ‘A’ it will light up the LED on port number 12.

## **Materials and Equipment**

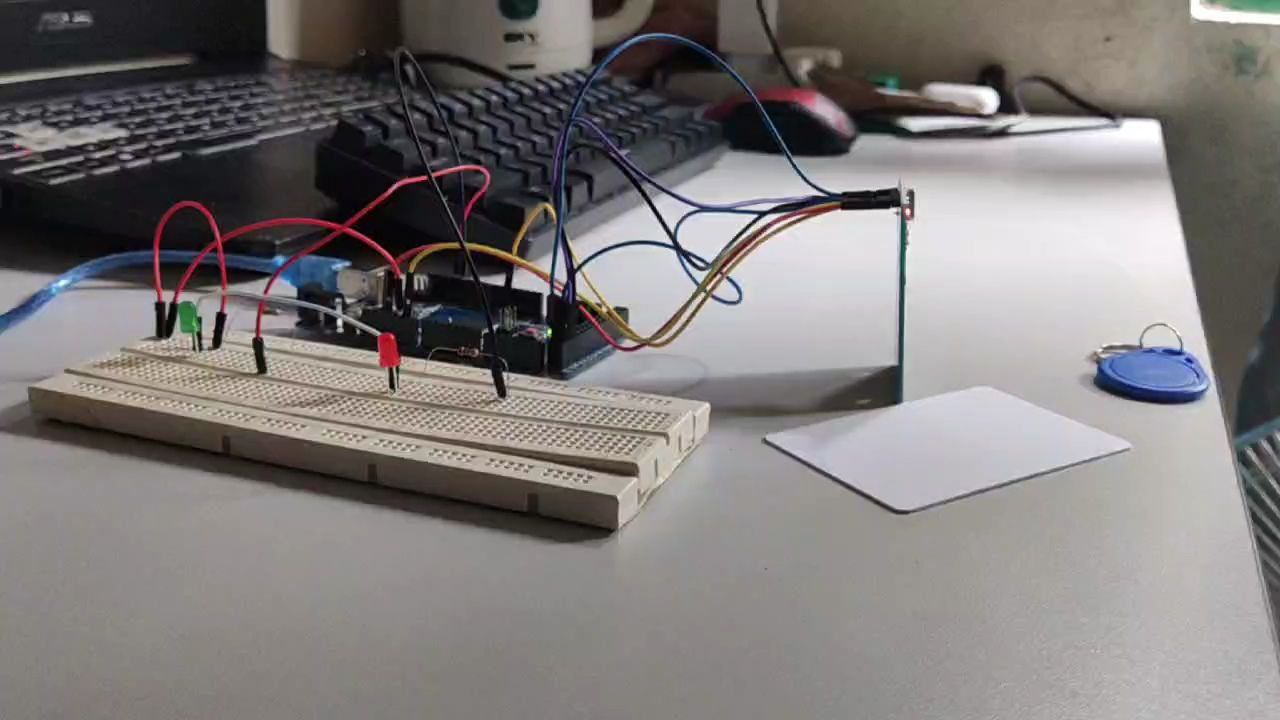
* Arduino board
* RFID card reader with USB connectivity
* RFID tags or cards that can be used for authentication
* Jumper wires
* Breadboard
* LEDs of various colors
* USB cables to connect the Arduino board and the RFID reader to your computer.
* Computer with Arduino IDE and Python installed
* Datasheets and Manuals: Make sure you have the datasheets or manuals for the RFID reader, servo motor, and any other components you are using. Most of them can be downloaded from the internet. Before starting the experiment, carefully read the documentation for each component and understand the electrical and mechanical requirements. Also, consider safety protocols and guidelines to ensure a safe working environment in the lab.



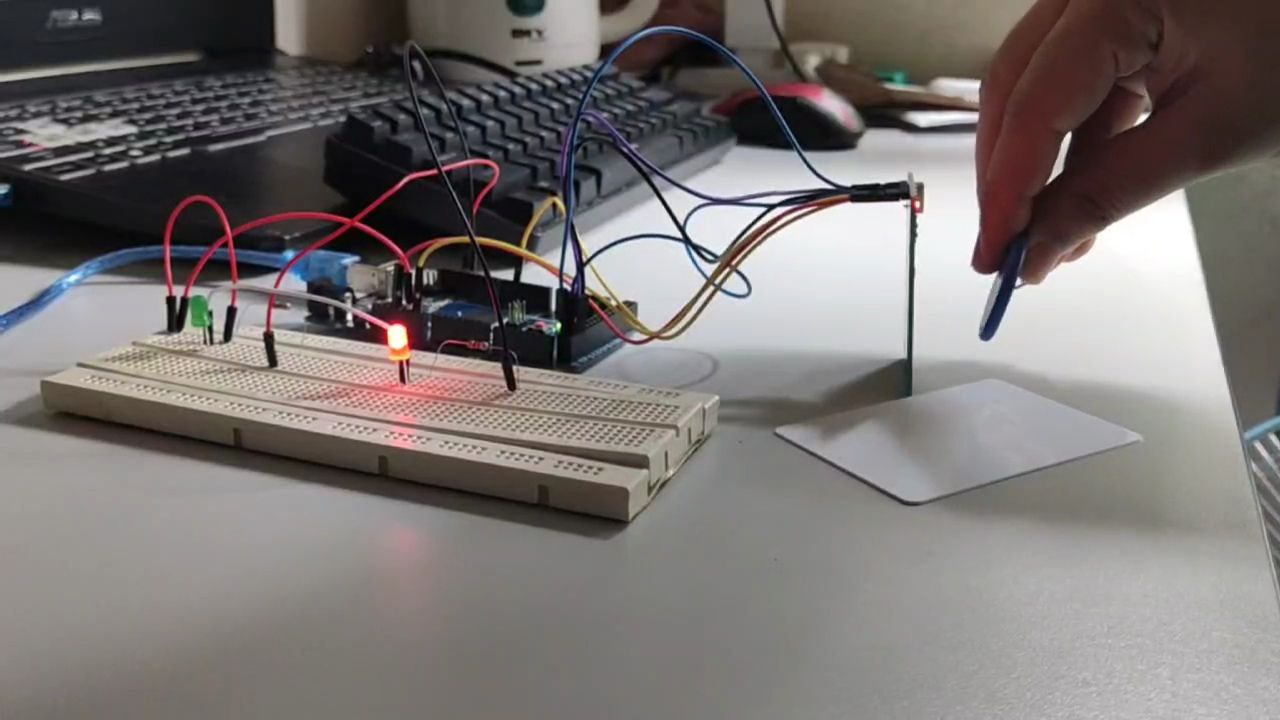
## **Methodology**

1. The circuit is assembled as per the experimental setup.
2. Arduino code was uploaded to the provided Arduino Uno.
3. The port on the Python script was changed to match the Arduino’s port.
4. The Python script was run on the computer.
5. A RFID card is placed on the sensor.
6. Changes in LEDs state are collected.

## **Data Collection**



i



ii

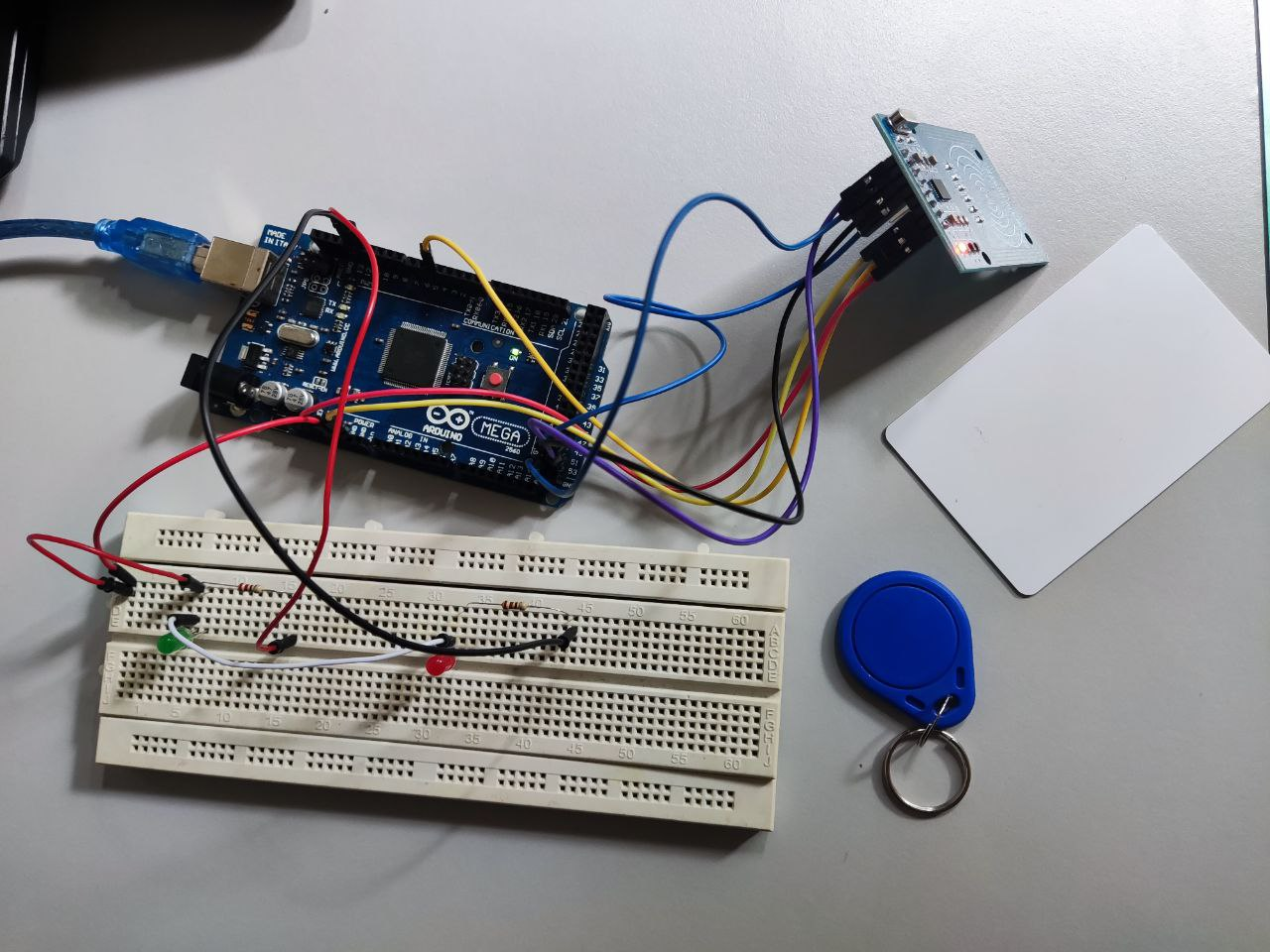


iii

## **Data Analysis**

Every RFID card and tag is equipped with its own unique UID (Unique Identifier). When the RFID reader captures the UID, it is presented on the serial monitor and recorded for reference. In this scenario, both the RFID tag and card utilize their UIDs. The UID data is transmitted through the serial port, which reads the UID and responds by sending specific characters back to the serial port for Arduino to process.This will then activate the LED for some specific RFID card and tag.

## **Results**



<https://github.com/NotLafuan/GROUP-F-MCTA-3203/raw/main/Week%204/Group%20F%20Week%204%20Part%20B.mp4>

## **Discussion**

From our testing, we found out that it is better to register an RFID card without changing a JSON file directly. It is due to a few reasons which is it will bring a higher ease of use. The reason beind this is that a modified JSON file directly can increase the risk of getting errors especially to people who aren’t familiar with the JSON syntax. By directly register an RFID card it will simplifies the task and minimizes amount of errors.

Other than that, it will enhance the security features of the product by ensuring that only the authorized individuals are allowed access to registering RFID cards.

## **Conclusion**

In this experiment, the main objective is to demonstrate the integrated communication between Arduino microcontroller, Python programming and an RFID reader. RFID card reader identification experiment has provided valuable insights into the practical application of RFID technology for secure and efficient identification systems.

The results of the trials showed how fast and dependable RFID technology is for correctly detecting and authenticating RFID cards. RFID is a great option for applications like inventory management, attendance monitoring, and access control because of its benefits, which include contactless functioning and simplicity of integration into other systems.

However, it's important to recognise such problems, including signal interference and security issues, and take the necessary steps to solve them. All things considered, the RFID card reader identification experiment emphasises how flexible and useful RFID technology is in practical situations, highlighting how it may improve security and expedite procedures in a range of applications.

## **Recommendations**

In order to enhance the outcome of the experiment, we can try to make the system more robust by trying on a large volume of RFID tags and making sure that it would be resilient with the accuracy of the outcomes. A more robust product from the experiment would help making it more reliable and resilient if it were to be commercialised to the public such as for LRT entrance system and others.

Other than that, we can also improve the User Interface by providing graphical feedbacks and representations using 7 segment displays to create a more user-friendly experience. To make it a vast system that can store data for the ease of repeating users, connect the system created to cloud services for data storage, analysis and accessibility. This will create a broader access to the RFID to be remotely monitored.

## **References**

<https://www.arduino.cc/reference/en/language/functions/communication/serial/>

<https://www.arduino.cc/reference/en/language/functions/communication/spi/>

<https://github.com/miguelbalboa/rfid>

<https://pyserial.readthedocs.io/en/latest/>

## 

## **Appendices**

### Code Snippets

Arduino code

#include <SPI.h>

#include <MFRC522.h>

#define SS\_PIN 53

#define RST\_PIN 5

MFRC522 rfid(SS\_PIN, RST\_PIN); // Instance of the class

MFRC522::MIFARE\_Key key;

// Init array that will store new NUID

byte nuidPICC[4];

char data;

void setup()

{

Serial.begin(9600);

SPI.begin(); // Init SPI bus

rfid.PCD\_Init(); // Init MFRC522

pinMode(12, OUTPUT);

pinMode(13, OUTPUT);

for (byte i = 0; i < 6; i++)

{

key.keyByte[i] = 0xFF;

}

}

void loop()

{

// Look for new cards

if (!rfid.PICC\_IsNewCardPresent())

return;

// Verify if the NUID has been readed

if (!rfid.PICC\_ReadCardSerial())

return;

for (byte i = 0; i < 4; i++)

{

nuidPICC[i] = rfid.uid.uidByte[i];

}

printHex(rfid.uid.uidByte, rfid.uid.size);

Serial.println();

rfid.PICC\_HaltA();

rfid.PCD\_StopCrypto1();

if (Serial.available() > 0)

data = Serial.read();

if (data == 'f')

{

digitalWrite(13, HIGH);

delay(2000);

digitalWrite(13, LOW);

}

if (data == 'a')

{

digitalWrite(12, HIGH);

delay(2000);

digitalWrite(12, LOW);

}

}

void printHex(byte \*buffer, byte bufferSize)

{

for (byte i = 0; i < bufferSize; i++)

{

Serial.print(buffer[i] < 0x10 ? " 0" : " ");

Serial.print(buffer[i], HEX);

}

}

Python code

import serial

# Open a serial connection to the Arduino

arduino = serial.Serial('COM6', baudrate=9600, timeout=1)

while True:

arduino\_data = arduino.readline().decode('utf-8').strip()

if arduino\_data:

print(f'Received data from Arduino: {arduino\_data}')

if arduino\_data == '35 B2 15 AD':

arduino.write('f'.encode())

elif arduino\_data == '2D 83 06 31':

arduino.write('a'.encode())

### 

### 

### Circuit Diagram

# 

# **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 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: Naufal Read

Name: Naufal Understand

Matric Number: 2110333 Agree

Contribution: Testing and Debugging

Signature: Aiman Read

Name: Aiman Understand

Matric Number: 2113571 Agree

Contribution : Programming

Signature: Nabil Read

Name: Nabil Understand

Matric Number: 2114577 Agree

Contribution: Circuit Design

Signature: Zakwan Read

Name: Zakwan Understand

Matric Number: 2111033 Agree

Contribution: Data Analysis

Signature: Che Read

Name: Che Understand

Matric Number: 21119075 Agree

Contribution : Discussion