



MECHATRONICS SYSTEM INTEGRATION (MCTA 3203)

SECTION 2, SEMESTER 1, 2024/2025

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

ACTIVITY REPORT

Week : 4 (a)

No	Name (Group 6)	Matric No.
1	MUHAMMAD BASIL BIN ABDUL HAKIM	2215315
2	MUHAMMAD SYAFIQ BIN NOR AZMAN	2213187
3	MUHAMMAD HAFIZ HAMZAH BIN FANSURI	2212803
4	MUHAMMAD AMMAR ZUHAIR BIN NOR AZMAN SHAH	2110259
5	MUHAMMAD RAZIQ BIN KAHARUDDIN	2120225

Table of content

Table of content.....	2
Abstract.....	3
Material and equipment.....	4
Experimental Setup and Methodology.....	6
Results.....	7
Discussions.....	8
Question.....	15
Recommendation.....	17
Conclusion.....	17
Student Declaration.....	18

Abstract

This project aims to create a serial communication interface between an Arduino microcontroller and Python, allowing data transfer over USB. Specifically, the interface gathers data from a potentiometer connected to the Arduino and sends it to a computer. The Arduino is programmed to continuously send potentiometer data over a serial port, while a Python script uses the PySerial library to receive this data. This setup enables real-time data usage within Python programs, making it easy to visualize and process Arduino data directly on a computer.

Research during the project has demonstrated the feasibility of a stable serial connection between Arduino and Python, which opens up many applications for data visualization and analysis. By simplifying the integration of Arduino data into Python, this interface provides a framework for future projects, offering a foundation for serial communication between microcontrollers and computers. The successful configuration highlights the effectiveness of this system for using live data in various applications, paving the way for advanced projects in data-driven fields.

Material and equipment

- 1) 1, Arduino UNO microcontroller board
- 2) Jumper wires (male-to-male)
- 3) 1, Breadboard
- 4) 1, MPU 6050 sensor
- 5) 2, LEDs
- 6) 1, Resistor, 1k Ohm
- 7) Computer with Arduino IDE and Python installed

Experimental Setup and Methodology

- 1) The MPU6050 sensor was connected to the Arduino board.
- 2) I2C communication was employed, with the SDA and SCL pins of the MPU6050 connected to the Arduino pins A4 and A5, respectively.
- 3) The power supply and ground of the MPU6050 were linked to the Arduino's 5V and GND pins, correspondingly.
- 4) For data exchange, the Arduino board was connected to the PC via USB.

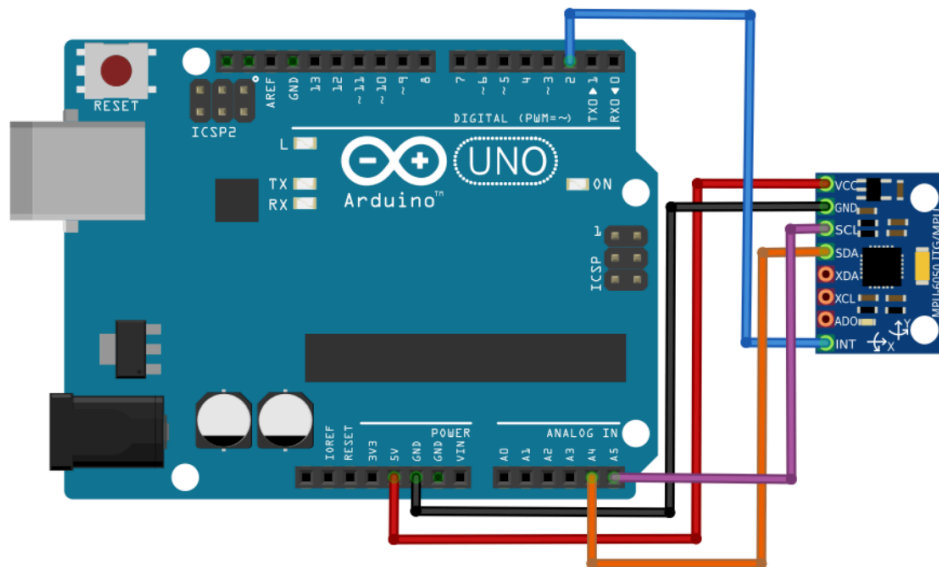


Fig. 1: Arduino-MPU6050 Connections

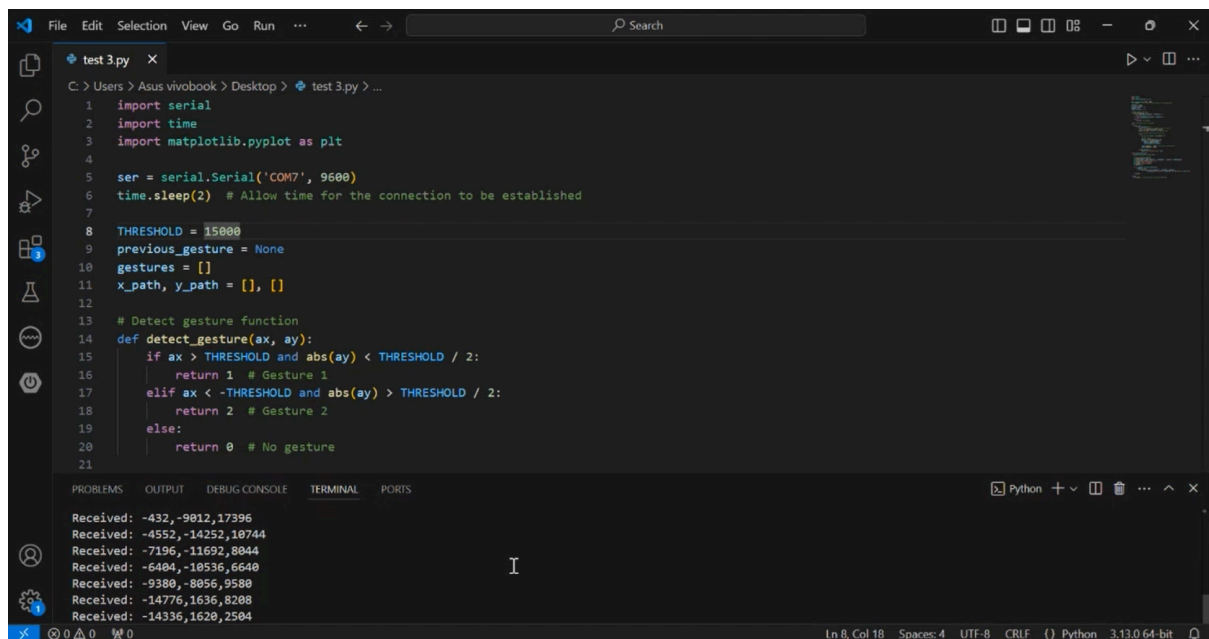
Software Setup:

- 1) Code was authored and uploaded to the Arduino utilizing the Arduino IDE.
- 2) Arduino code was developed to facilitate communication with the MPU6050 sensor, retrieve data, and format it for transmission.
- 3) The PySerial package for Python was employed to establish a serial connection between the Arduino and the PC.
- 4) Data reception and presentation on the PC console were achieved through a Python script.
- 5) Within the Python script, data processing and parsing were conducted as needed to interpret the format of the data transmitted by the Arduino.

Results

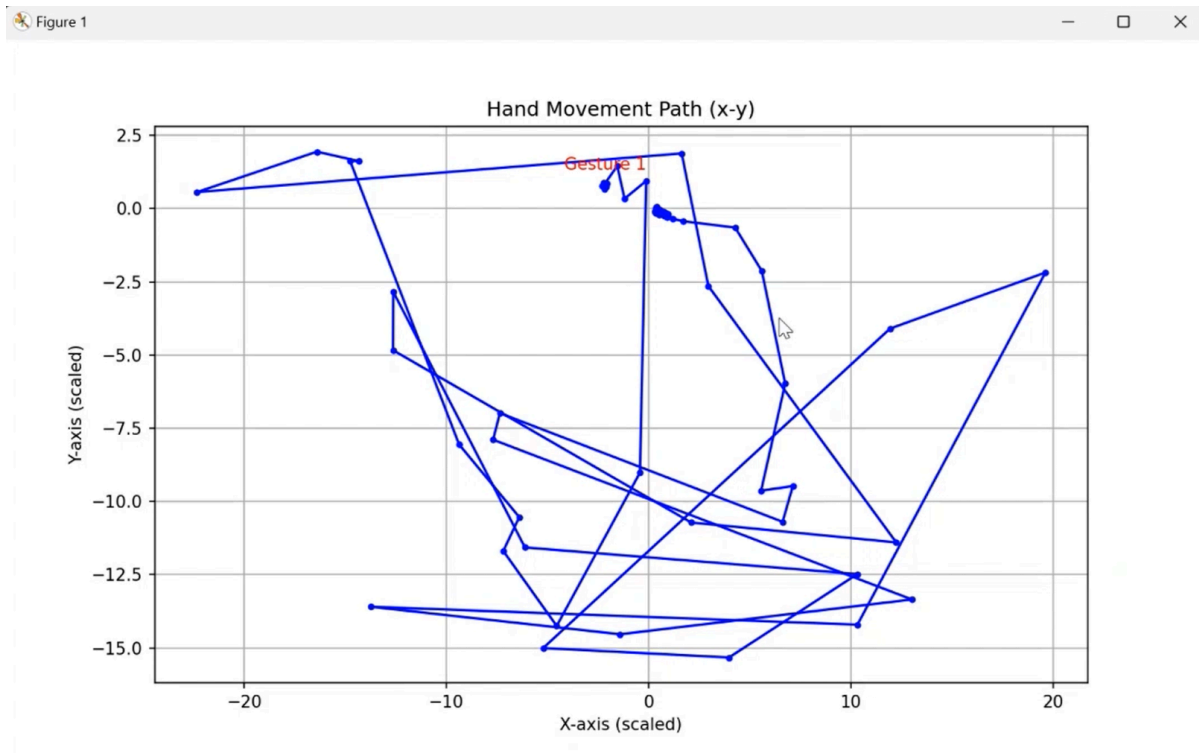
In this experiment, we managed to develop a simple hand gesture recognition system. Using data obtained from the MPU6050 sensor to record the motion movement of the hand, the data was then transmitted into the Arduino board that had been programmed in the Arduino IDE.

Using Python, the data can be visualised on the graph to view the hand movement path in the x-y coordinate system. The integration between Arduino and Python allows the data collected from the sensor to be easily understood to many people by visualisation on the graph.



```
File Edit Selection View Go Run ... Search
test 3.py x
C:\Users\Asus vivobook\Desktop> test 3.py > ...
1 import serial
2 import time
3 import matplotlib.pyplot as plt
4
5 ser = serial.Serial('COM7', 9600)
6 time.sleep(2) # Allow time for the connection to be established
7
8 THRESHOLD = 15000
9 previous_gesture = None
10 gestures = []
11 x_path, y_path = [], []
12
13 # Detect gesture function
14 def detect_gesture(ax, ay):
15     if ax > THRESHOLD and abs(ay) < THRESHOLD / 2:
16         return 1 # Gesture 1
17     elif ax < -THRESHOLD and abs(ay) > THRESHOLD / 2:
18         return 2 # Gesture 2
19     else:
20         return 0 # No gesture
21
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Received: -432,-9812,17396
Received: -4552,-14252,10744
Received: -7196,-11692,8044
Received: -6404,-10536,6640
Received: -9380,-8056,9580
Received: -14776,1636,8208
Received: -14336,1620,2504
Ln 8, Col 18 Spaces: 4 UTF-8 CRLF Python 3.13.0 64-bit
```

Python codes



Graph obtained from Python

Discussions

Hardware Discussion

a) Arduino Uno

Arduino Uno is a microcontroller that has been embedded with electrical components that can be used in many engineering projects. It can be used to control external electrical components such as LED, servo motor and 7-segment display using codes that will be uploaded into the Arduino.

b) Jumper wire

Jumper wires are used to connect electrical components to let current flow through them.

c) MPU 6050

MPU6050 is an integrated 6-axis motion-tracking device. The sensor is embedded with a 3-axis gyroscope and a 3-axis accelerometer. MPU6050 is used to print its Cartesian coordinate location in the x,y and z axes while the gyroscope prints out the orientation of the sensor.

Software Discussion

ARDUINO CODE

```
#include <Wire.h>

#include <MPU6050.h>

MPU6050 mpu;

const int threshold = 10000;

int previousGesture = -1;

void setup() {

    Serial.begin(9600);

    Wire.begin();

    mpu.initialize();

    if (!mpu.testConnection()) {

        Serial.println("MPU6050 connection failed!");

        while (1); //Stop execution if sensor fails to initialize
    }
}
```

```

    } else {

        Serial.println("MPU6050 connected successfully.");

    }

}

void loop() {

    int gesture = detectGesture();

    if (gesture != previousGesture) { //Detect change in gesture

        Serial.print("Detected Gesture: ");

        if (gesture == 1) {

            Serial.println("Gesture 1"); //Gesture 1

        } else if (gesture == 2) {

            Serial.println("Gesture 2"); //Gesture 2

        }

        previousGesture = gesture;

    }

    delay(200);

}

int detectGesture() {

    int16_t ax, ay, az, gx, gy, gz;

    //Get motion data from the MPU6050

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

```

```

//Print sensor values to monitor data

Serial.print(ax); Serial.print(",");

Serial.print(ay); Serial.print(",");

Serial.println(az);


//Define gesture recognition logic

if (ax > threshold && abs(ay) < threshold / 2) {

    return 1; //Gesture 1 detected

} else if (ax < -threshold && abs(ay) > threshold / 2) {

    return 2; //Gesture 2 detected

}

return 0; //No gesture detected
}

```

1. Global Variables and Objects

In this function, we declare the libraries that we are going to use to tell the Arduino that we are using the MPU6050 library so it can communicate with the sensor. We also declare the variables globally so we can keep calling the same variables in other functions without needing to redeclare the variables.

2. *Void setup()* Function

In this function, we set up the electrical components that we are using and the code will run once. We initialise the baud rate of 9600 bits/sec which will allow communication between the computer for debugging and logging. We also initialise the MPU 6050 to communicate the sensor with Arduino. A test command is also implemented to make sure our Arduino is properly connected with the MPU6050. If there are any connection errors, a warning will be printed out.

3. *Void Loop()* Function

This function will run repeatedly once it reaches the end of the function. In this function, we will receive inputs from the *detectGesture()* function to print out the gesture we receive from the MPU6050. This function will check if the next gesture is the same as the previous and print out the new gesture if the system detects a different gesture. We also put delays to make sure the reading does not get messed up.

4. *detectGesture()* Function

In this function, we will tell the MPU6050 to take readings. First, we declare the variables *ax*, *ay*, *az*, *gx*, *gy* and *gz* and the Arduino will store the inputs in the variables through the command *mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz)* respectively and print out the coordinates. Next, we run an *if-else* statement to know if there are any changes in the orientation of the MPU6050 that will be called in the *void loop()* function.

PYTHON CODE

```
import serial

import time

import matplotlib.pyplot as plt

ser = serial.Serial('COM7', 9600)

time.sleep(2)  #Allow time for the connection to be established

THRESHOLD = 15000

previous_gesture = None

gestures = []

x_path, y_path = [], []

#Detect gesture function

def detect_gesture(ax, ay):

    if ax > THRESHOLD and abs(ay) < THRESHOLD / 2:

        return 1  #Gesture 1
```

```
elif ax < -THRESHOLD and abs(ay) > THRESHOLD / 2:

    return 2 #Gesture 2

else:

    return 0 #No gesture


#Read and process data from Arduino
try:

    while True:

        if ser.in_waiting > 0:

            #Read line from Arduino, decode and strip newline

            line = ser.readline().decode().strip()

            print(f"Received: {line}") # Debugging output

            #Parse accelerometer data from the line

            try:

                ax, ay, az = map(int, line.split(","))

                #Detect gesture

                gesture = detect_gesture(ax, ay)

                if gesture != previous_gesture:

                    gestures.append(gesture)

                    previous_gesture = gesture

            x_path.append(ax / 1000) #Scaling for visualization

            y_path.append(ay / 1000)
```

```

        except ValueError:

            print("Error parsing line:", line)

except KeyboardInterrupt:

    print("Interrupted! Plotting path...")

    # Plot x-y path of movement

    plt.figure(figsize=(10, 6))

    plt.plot(x_path, y_path, marker='o', linestyle='-', color='b',
markersize=3)

    plt.title('Hand Movement Path (x-y)')

    plt.xlabel('X-axis (scaled)')

    plt.ylabel('Y-axis (scaled)')

    plt.grid(True)

    for i, gesture in enumerate(gestures):

        if gesture != 0:

            plt.annotate(f'Gesture {gesture}', (x_path[i], y_path[i]),

                        textcoords="offset points", xytext=(0, 10),
ha='center', color='red')

    plt.show()

finally:

    ser.close() #Close the serial connection when done

```

1. Import Library

First, we import libraries in Python to enable serial connection, time-related functions and plot related graphs.

2. Setup Serial Connection

In this part, we make sure that Python is connected to the same port that the Arduino is using to the computer. In this experiment, we connect it to port 7. We also put delays to give time for the programme to initialise the connection.

3. Initialise Variables

Next, we initialise variables for Python to store the necessary information in the variables.

4. *def detect_gesture()* Function

In this function, we receive inputs from the MPU6050 and return values based on the *if-else* statement.

5. Loop

In the first *try* section, we will continuously check in the serial buffer if there are any data from the Arduino and print the received line by line from the MPU6050. The received line is based on outputs in IDE. It is important to make sure that the output arrangement in Python is the same as in IDE to avoid errors.

6. Parse and Process Accelerometer Data

After that, we will call *def detect_gesture()* to check if there are any changes in the gesture by comparing it using the *if* function. We also insert *except ValueError* to print out a warning if the programme detects a parsing line error.

7. Graph

This part of the programming is run when there are interruptions when Python is run again. The graph will be plotted based on the values that we received and printed out on Python's terminal.

8. *Finally*

This command tells Python to close the serial connection once the programme is finished or interrupted.

Electrical discussion

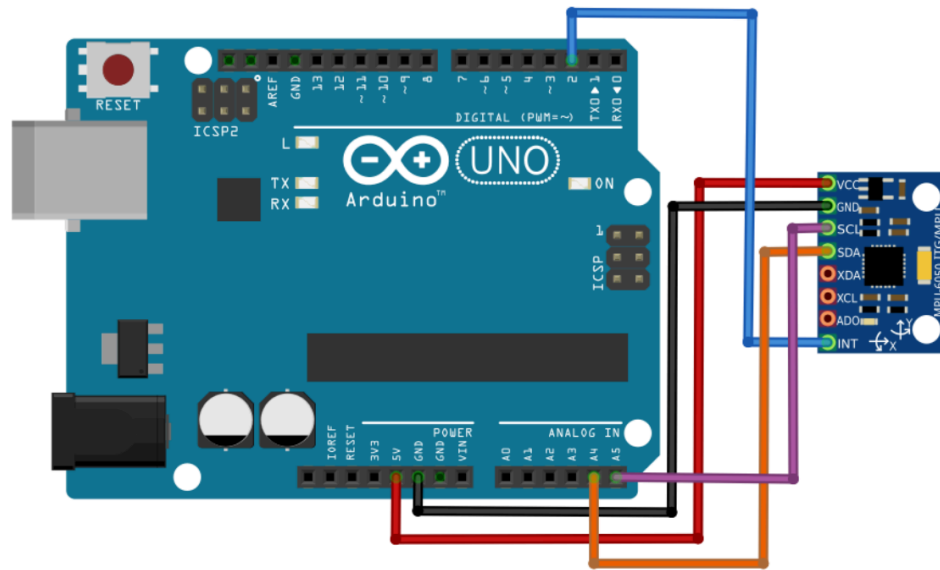


Fig. 1: Arduino-MPU6050 Connections

VCC and GND on the MPU6050 are connected to the 5V and GND pin on the Arduino respectively while the SCL is connected to the pin A4 and SDA to A5. Finally, the INT pin is connected to pin 2.

Recommendation

The accuracy of the MPU 6050 sensor can be improved by applying data filtering such as applying a low-pass filter to reduce noise in the accelerometer and gyroscope readings. This will ensure a smoother and more accurate visualisation of hand movement by removing sudden spikes and jitters in the data that are not part of the actual gesture.

This experiment can be beneficial for future usage such as being able to create biomechanical prosthetic limbs that are able to collect real time data through sensors for optimisation based on the user preference and needs.


Conclusion


A productive and instructive laboratory experiment has been conducted through the utilization of the MPU6050 accelerometer and gyroscope in tandem with an algorithm designed for gesture identification, culminating in the development of a hand gesture recognition system. The integration of sensor data acquisition and signal processing capabilities to discern and categorize predetermined hand gestures exemplifies the diverse range of potential applications inherent in this technology. Despite persisting challenges associated with noise mitigation and the detection of intricate gestures, the experiment lays the groundwork for future research initiatives, elucidating the considerable potential of this technology within domains such as wearables, assistive technologies, and human-computer interaction.


Student 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 taken 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's contributors to the report. We therefore, agreed unanimously that this report shall be submitted for marking and this final printed report have been verified by us.

NAME: Muhammad Basil bin Abdul Hakim	READ	✓
MATRIC NO: 2215315	UNDERSTAND	✓
SIGNATURE: 	AGREE	✓

NAME: Muhammad Syafiq Bin Nor Azman	READ	✓
MATRIC NO: 2213187	UNDERSTAND	✓
SIGNATURE 	AGREE	✓

NAME:Muhammad Hafiz Hamzah Bin Fansuri	READ	✓
MATRIC NO:2212803	UNDERSTAND	✓
SIGNATURE : 	AGREE	✓

NAME: Muhammad Ammar Zuhair Bon Nor Azman Shah	READ	✓
MATRIC NO: 2110259	UNDERSTAND	✓
SIGNATURE: 	AGREE	✓

NAME: MUHAMMAD RAZIQ BIN KAHARUDDIN	READ	✓
MATRIC NO: 2120225	UNDERSTAND	✓
SIGNATURE	AGREE	✓