



الجامعة الإسلامية العالمية ماليزيا
INTERNATIONAL ISLAMIC UNIVERSITY MALAYSIA
يُونُسُ بَرَسِيَّتِي اِسْلَامُ اَنْتَا اِبْعَثْنَا مَلِيْسِيَا

Garden of Knowledge and Virtue

LAB REPORT

MCTA 3203

SECTION 1

GROUP D

EXPERIMENT 4A

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

| MATIC NO. | NAME |
|------------------|--------------------------------------|
| 2111633 | MUHAMMAD FARID BIN JAFRI |
| 2110765 | MUHAMMAD ARIF SYAZWAN BIN MOHD RIZAL |
| 2112969 | MUHAMMAD HAZIQ HAIKAL BIN SUAIB |
| 2115207 | MUHAMMAD AMIRUL SYAFIQ BIN RUSLANI |
| 2116281 | MUHAMMAD FADHLUL WAFI BIN AHMAD NAIM |

SUBMISSION DATE: 8/11/2023

INSTRUCTOR: DR. WAHJU SEDIONO

Abstract

The primary aim of this project is to develop a serial communication interface between the Python programming language and an Arduino microcontroller. This interface will facilitate the transmission of data from a potentiometer connected to the Arduino to a computer via a USB connection. This facilitates comprehension of the procedure for establishing serial connectivity and enables the utilisation of real-time data from the Arduino in Python programs. In order to establish serial communication, the Arduino will be programmed to provide potentiometer data in a continuous manner through a serial interface. Conversely, a Python script will utilise the pyserial package to create a serial connection and get the data transmitted by the Arduino. The project yielded significant results, demonstrating the successful development of a serial communication link between Python and an Arduino. This achievement facilitated the seamless real-time transfer of potentiometer readings. The project showcased the potential for establishing the dependable connection between Arduino and Python, enabling the utilisation of the acquired data for many purposes, including data visualisation through plotting.

In summary, this experiment showcases the effective development of a serial communication interface between the Python programming language and an Arduino microcontroller. This interface enables the interchange of data, specifically pertaining to potentiometer readings. The efficacy of this communication architecture for real-time data usage in Python programs is demonstrated through the configuration of both ends. The results of this project contribute to the establishment of a fundamental comprehension of the process of serial communication between microcontrollers and computers, hence creating opportunities for diverse applications and projects in subsequent endeavours.

Table of Contents

| | |
|-------------------------------------|-----------|
| Introduction..... | 4 |
| Materials and Equipment..... | 5 |
| Experimental Setup..... | 6 |
| Methodology..... | 7 |
| Results (Observation)..... | 9 |
| Discussions..... | 10 |
| Conclusion..... | 11 |
| Recommendations..... | 11 |
| References..... | 12 |
| Student's Declaration..... | 13 |

Introduction

The goal of this project is to set up a serial communication link between an Arduino and Python so that data can be sent via USB from the Arduino's potentiometer to a computer. This makes it easy to understand how to connect to a serial port and lets Python programs use real-time data from an Arduino.

The microcontroller Arduino is used to read data from devices and run hardware. An Arduino would be used to connect to the gear and sensors and send data to the computer since the computer itself can't get to them. Plus, Arduino boards are a lot less expensive than most computers, so if a motor or device fails and destroys an Arduino, it's not as bad as if it were a computer.

Python is a flexible computer language that is often used in data science. The pyserial tool lets a Python script read and write to a serial port. So, a Python script can be used to set up contact between a computer and an Arduino. Also, Python's flexibility in changing data lets you do complicated calculations with access to more computing power.

Materials and Equipment

Materials Needed:

1. Arduino Board
2. MPU6050 sensor
3. Jumper Wires
4. LED
5. 220 resistor
6. Breadboard

Equipment:

1. Computer with necessary software for programming and testing (Arduino IDE)
2. Screwdriver and pliers (for assembly and adjustments)

Experimental Setup

Hardware setup:

1. The MPU6050 sensor was connected to the Arduino board.
2. I2C communication was used; SDA and SCL pins of MPU6050 were connected to Arduino pins A4 and A5.
3. The power supply and ground of the MPU6050 were connected to Arduino's 5V and GND pins, respectively.
4. The Arduino board was connected to the PC via USB for data exchange.

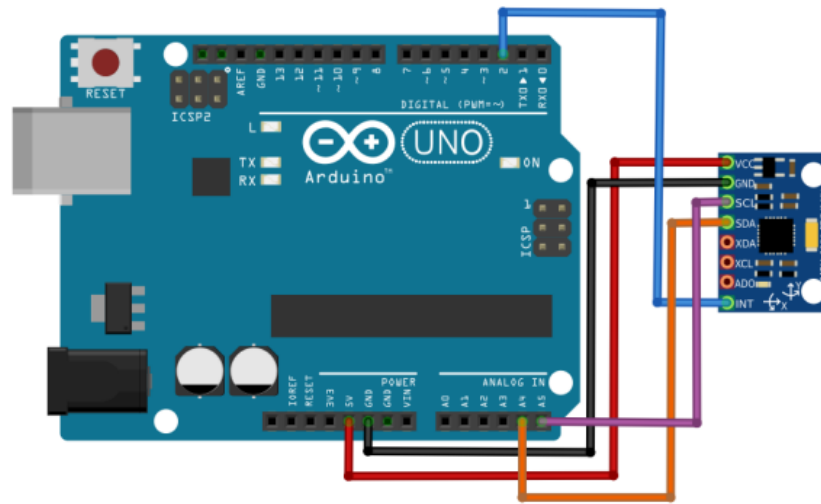


Figure 1

Software setup:

1. The Arduino IDE was used for writing and uploading code to the Arduino.
2. We wrote Arduino code to interface with the MPU6050 sensor, read data, and format it for transmission.
3. We used the PySerial library for Python to establish serial communication between the Arduino and the PC.
4. Python script was used for data reception and display on the PC's console.
5. Data parsing and processing were done within the Python script, if necessary, to interpret the data format sent by the Arduino.

Methodology

Procedures:

1. After successfully setting up the hardware and software, the Arduino code has been verified and uploaded into the Arduino microcontroller using the Arduino IDE.
2. The python code was saved into the computer file and run to observe the output.
3. The values for the MPU6050 were displayed on the python terminal and have been observed and recorded by group members.

Coding:

```

1 #include <Wire.h>
2 #include <MPU6050.h>
3
4 MPU6050 mpu;
5
6 void setup() {
7   Serial.begin(9600);
8   Wire.begin();
9   mpu.initialize();
10 }
11 void loop() {
12   int ax, ay, az, gx, gy, gz;
13   mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
14   Serial.print("Accel: ");
15   Serial.print(ax);
16   Serial.print(", ");
17   Serial.print(ay);
18   Serial.print(", ");
19   Serial.print(az);
20   Serial.print(" Gyro: ");
21   Serial.print(gx);
22   Serial.print(", ");
23   Serial.print(gy);
24   Serial.print(", ");
25   Serial.println(gz);
26   delay(300);
27 }

```

1. *Arduino IDE*

```
import serial

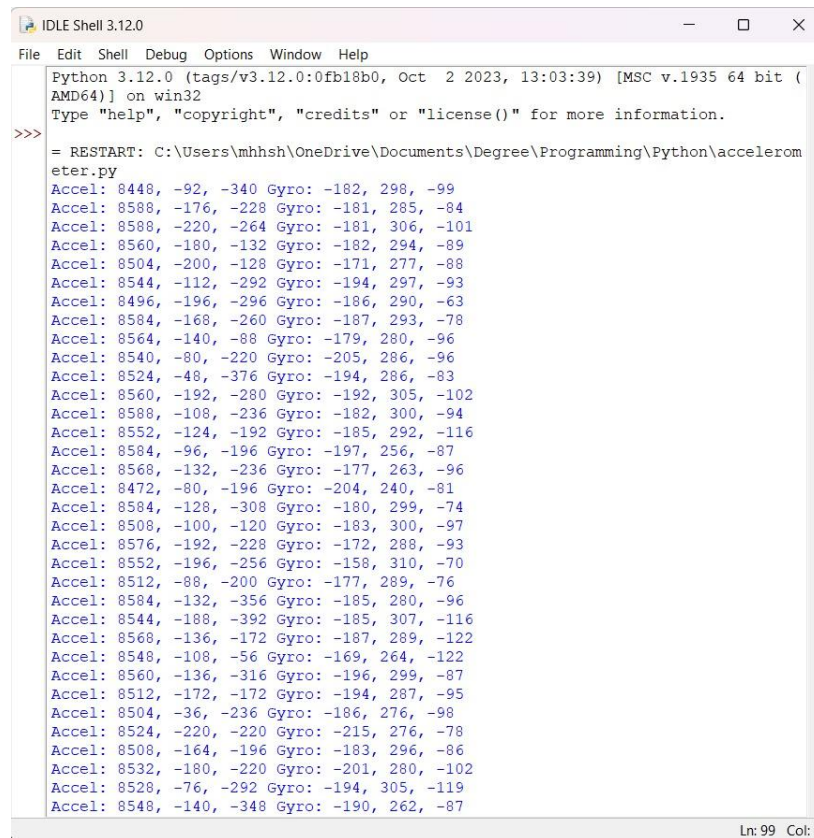
ser = serial.Serial('COM5', 9600)

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

2. *Python*

* These codes can be found on our GitHub.

Results (Observation)



```

IDLE Shell 3.12.0
Python 3.12.0 (tags/v3.12.0:0fb18b0, Oct 2 2023, 13:03:39) [MSC v.1935 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\mhsh\OneDrive\Documents\Degree\Programming\Python\accelerometer.py
Accel: 8448, -92, -340 Gyro: -182, 298, -99
Accel: 8588, -176, -228 Gyro: -181, 285, -84
Accel: 8588, -220, -264 Gyro: -181, 306, -101
Accel: 8560, -180, -132 Gyro: -182, 294, -89
Accel: 8504, -200, -128 Gyro: -171, 277, -88
Accel: 8544, -112, -292 Gyro: -194, 297, -93
Accel: 8496, -196, -296 Gyro: -186, 290, -63
Accel: 8584, -168, -260 Gyro: -187, 293, -78
Accel: 8564, -140, -88 Gyro: -179, 280, -96
Accel: 8540, -80, -220 Gyro: -205, 286, -96
Accel: 8524, -48, -376 Gyro: -194, 286, -83
Accel: 8560, -192, -280 Gyro: -192, 305, -102
Accel: 8588, -108, -236 Gyro: -182, 300, -94
Accel: 8552, -124, -192 Gyro: -185, 292, -116
Accel: 8584, -96, -196 Gyro: -197, 256, -87
Accel: 8568, -132, -236 Gyro: -177, 263, -96
Accel: 8472, -80, -196 Gyro: -204, 240, -81
Accel: 8584, -128, -308 Gyro: -180, 299, -74
Accel: 8508, -100, -120 Gyro: -183, 300, -97
Accel: 8576, -192, -228 Gyro: -172, 288, -93
Accel: 8552, -196, -256 Gyro: -158, 310, -70
Accel: 8512, -88, -200 Gyro: -177, 289, -76
Accel: 8584, -132, -356 Gyro: -185, 280, -96
Accel: 8544, -188, -392 Gyro: -185, 307, -116
Accel: 8568, -136, -172 Gyro: -187, 289, -122
Accel: 8548, -108, -56 Gyro: -169, 264, -122
Accel: 8560, -136, -316 Gyro: -196, 299, -87
Accel: 8512, -172, -172 Gyro: -194, 287, -95
Accel: 8504, -36, -236 Gyro: -186, 276, -98
Accel: 8524, -220, -220 Gyro: -215, 276, -78
Accel: 8508, -164, -196 Gyro: -183, 296, -86
Accel: 8532, -180, -220 Gyro: -201, 280, -102
Accel: 8528, -76, -292 Gyro: -194, 305, -119
Accel: 8548, -140, -348 Gyro: -190, 262, -87
Ln: 99 Col: 0

```

Figure 2

As we move the MPU6050, the reading of the accelerometer and gyroscope value is shown in Python script. This changing value indicates an established serial communication between Arduino and Python. During the movement of the MPU6050, the accelerometer readings will change along its three axes (X, Y and Z) to represent change in acceleration. For instance, when we tilted the sensor to the right, the Y-axis accelerometer value increased. Similarly when we tilted the sensor forward, the reading of the Z-axis accelerometer also increased. Meanwhile, when we moved the MPU6050, the gyroscope value changed to represent the rotational speed around the three axes (X, Y and Z). For example, when we did a quick rotation around the Z-axis, the value of the gyroscope for the Z-axis increased, indicating change in angular velocity.

Link for the video:

<https://drive.google.com/file/d/1BvfQtJ1N1b9BwNqhTVFSsLvH8dStmfhV/view?usp=sharing>

Discussions

From the task in the manual, we were asked to create a straightforward hand gesture recognition system by capturing accelerometer and gyroscope data during the execution of predefined hand movements. We employed an algorithm to identify and categorise these gestures using the collected sensor data. Additionally, the paths of hand movement in an x-y coordinate system have been visualised. Below are the codes for Arduino and Python:

Arduino:

```

1 #include <Wire.h>
2 #include <MPU6050.h>
3
4 MPU6050 mpu;
5
6 int16_t ax, ay, az, gx, gy, gz;
7 int scalingFactor = 20;
8
9 void setup() {
10   Serial.begin(9600);
11   Wire.begin();
12   mpu.initialize();
13 }
14
15 void loop() {
16   mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
17   int x = map(ax, -17000, 17000, 0, scalingFactor);
18   int y = map(ay, -17000, 17000, 0, scalingFactor);
19   Serial.print(x);
20   Serial.print(" ");
21   Serial.println(y);
22   delay(300);
23 }

```

Python:

```

import serial
from time import sleep

ser = serial.Serial('COM5', 9600)
dt=0.1
count=0

if ser.is_open:
    print("Serial port is open")
    try:
        while True:
            data = ser.readline().decode('utf-8').strip()
            input_list = data.split()

            if len(input_list) == 2:
                dataX, dataY = map(int, input_list)

                if count != 1 and dataY < 8:
                    print("Object moved to the left")
                    count = 1
                elif count != 2 and dataY > 10:
                    print("Object moved to the right")
                    count = 2
                elif count != 3 and dataX < 8:
                    print("Object moved to the back")
                    count = 3
                elif count != 4 and dataX > 10:
                    print("Object moved to the front")
                    count = 4

                sleep(dt)
            except KeyboardInterrupt:
                pass
            finally:
                ser.close()
    else:
        print("Serial port is closed")

```

* These codes can be found on our GitHub.

```

Serial port is open
Object moved to the right
Object moved to the front
Object moved to the right
Object moved to the left
Object moved to the right
Object moved to the back
Object moved to the front
Object moved to the back

```

Output for the task

Conclusion

The implementation of a hand gesture recognition system using the MPU6050 accelerometer and gyroscope, coupled with an algorithm for gesture identification, has proven to be a successful and insightful lab experiment. The combination of sensor data collection and signal processing allowed for the recognition and categorization of predefined hand movements, demonstrating the potential of this technology for various applications. While challenges such as noise and complex gesture recognition exist, the experiment lays a foundation for future work, showcasing the promise of this technology in human-computer interaction, wearable devices, and assistive technology.

Recommendations

For the refinement and advancement of the hand gesture recognition system utilising the MPU6050, several key recommendations can be considered. Firstly, a dedicated effort should be invested in noise reduction techniques and the fine-tuning of calibration processes to augment the accuracy and reliability of the sensor data. Algorithm optimization remains crucial, particularly in the identification of complex or subtle hand movements, and continuous parameter adjustments are advised. Integrating machine learning methodologies could contribute to the system's adaptability and its ability to learn new gestures over time.

Additionally, prioritising the improvement of the user interface, incorporating real-time feedback mechanisms, and collaborating with wearable technology developers will collectively enhance the overall user experience and foster the integration of the system into various applications.

References

1. *Last Minute Engineers. (2022, October 29). In-depth: Interface MPU6050 Accelerometer & Gyroscope sensor with Arduino.*
<https://lastminuteengineers.com/mpu6050-accel-gyro-arduino-tutorial/>
2. *MPU6050 interfacing with Arduino Uno: Arduino. ElectronicWings. (n.d.).*
<https://www.electronicwings.com/arduino/mpu6050-interfacing-with-arduino-uno>

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 acknowledgment, and that the original work contained herein has not been undertaken 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 contributor 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: *arifsyazwan*

Name: Muhammad Arif Syazwan Bin Mohd Rizal
Matric Number: 2110765

Read ✓

Understand ✓

Agree ✓

Signature: *haziqhaikal*

Name: Muhammad Haziq Haikal bin Suaib
Matric Number: 2112969

Read ✓

Understand ✓

Agree ✓

Signature: *amirul*

Name: Muhammad Amirul Syafiq Bin Ruslani
Matric Number: 2115207

Read ✓

Understand ✓

Agree ✓

Signature: *Wafi*

Name: Muhammad Fadhlul Wafi Bin Ahmad Naim
Matric Number: 2116281

Read ✓

Understand ✓

Agree ✓

Signature: *faridjafri*

Name: Muhammad Farid Bin Jafri
Matric Number: 2111633

Read ✓

Understand ✓

Agree ✓