



الجامعة الإسلامية العالمية ماليزيا
INTERNATIONAL ISLAMIC UNIVERSITY MALAYSIA
وَنَبْرَسَتِي: إِسْلَامٌ وَأَنْبَارٌ يَجْعَلُ مِلَّةً

MECHATRONICS SYSTEM INTEGRATION

EXPERIMENT 4A: IMU-Arduino Interfacing Using Serial
Communication

NAME	MATRIC NUMBER
MUHAMAD IMRAN HAZIQ BIN HAIZAM	2216429
TANISHA ROWSHAN SALEH	2228290
MUHAMMAD YUSOF IRFAN BIN ABIDIN	2210915
ARIESHA DALIELA HANIM BINTI AMIR	2316834

DATE OF SUBMISSION: 11/04/25

TABLE OF CONTENTS

Title	Page No.
Abstract	2
Introduction	2
Materials And Equipment	2 - 3
Experimental Set-Up	3 - 4
Methodology	4 - 5
Results	6
Discussion	7
Conclusion	7 - 8
Recommendation	8
Reference	8
Student Declaration Form	8 -11

Abstract

This experiment was conducted to better understand serial interfacing and its applications. For experiment 4A, an Arduino microcontroller was interfaced with an IMU (inertial measurement unit), which was used to measure and record hand movement. Using an Arduino microcontroller to interface with an MPU6050, this was successfully achieved. The objective of this experiment was to allow both an Arduino and Python IDE to record and display hand movements in numerical form, with regards to an xyz plane. The movements were displayed on the serial monitor of the Arduino IDE and the terminal of the Python IDE as positive and negative numbers, indicating the location of the hand on the plane.

Introduction

Experiment 4A was conducted in order to understand serial communication. Serial communication involves communication between devices, and in this case, it involved communication between an IMU and a computer, using an Arduino microcontroller as an interface. IMUs generally contain both accelerometers and gyroscopes, used to measure acceleration and angular velocity in three-dimensional space. IMUs are used for various purposes, namely navigation and motion tracking. Some areas of application are aviation, robotics, and the military.

Detecting the motion of a hand and its position can be useful in many situations, such as in physiotherapy where the motion, velocity and/or acceleration of a person's hand can often determine their current recovery state, or easily controlling home appliances with the flick of a hand. These are just a few of the numerous ways hand-controlled technology can ease lives.

Based on the knowledge of how IMU-Arduino interfacing works, it was expected to record and display the xyz coordinates of the hand on the serial monitor of the Arduino IDE and the terminal of the Python IDE, as it moved near the sensor.

Materials and Equipment

Experiment 4A

Materials Needed:

- 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 colours.

Experimental Set-Up

Hardware setup:

Hardware Setup:

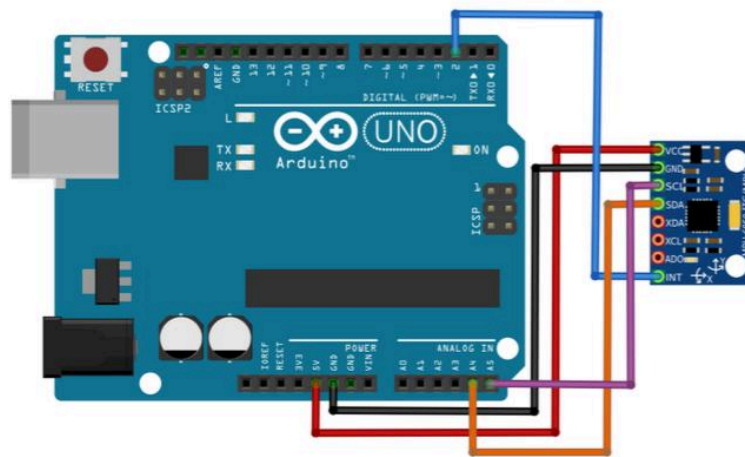
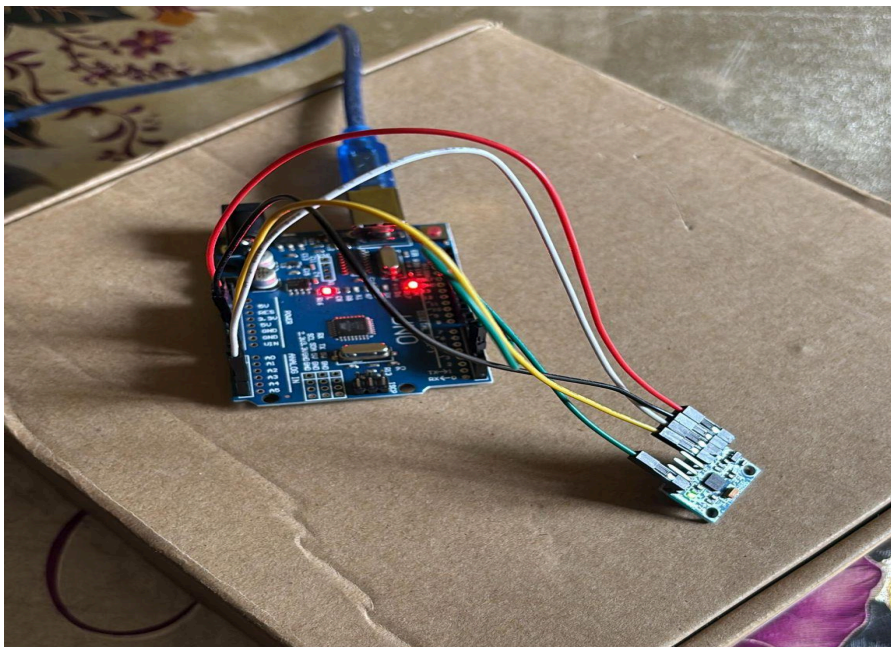


Fig. 1: Arduino-MPU6050 Connections



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.

Methodology

Code (Arduino)

```
#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();

  if (gesture != previousGesture) {
    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;
    }
}

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
}

```

Code (Python)

```

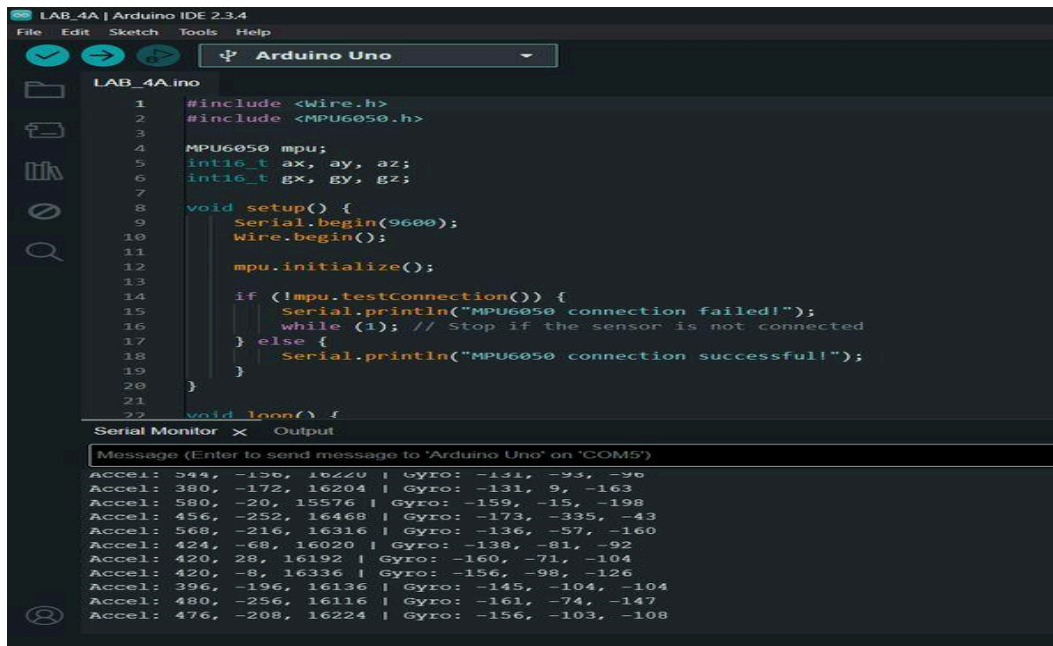
import serial

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

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

```

Results

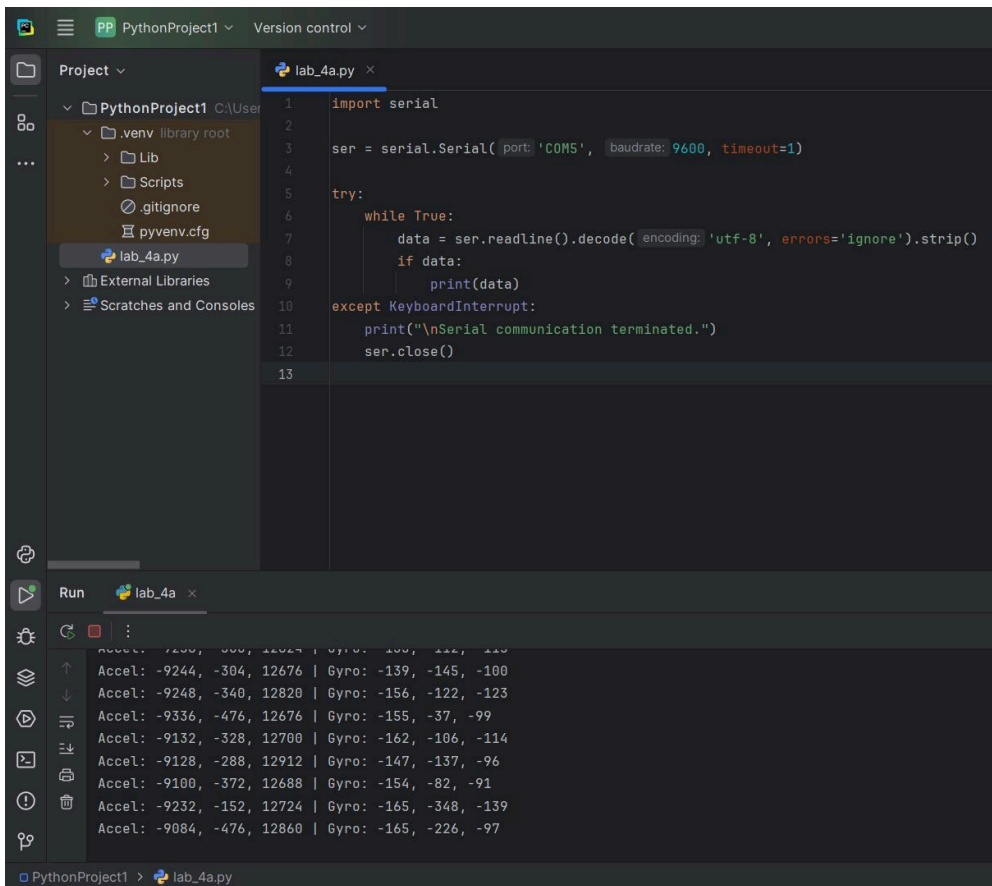


The screenshot shows the Arduino IDE 2.3.4 interface. The top menu bar includes File, Edit, Sketch, Tools, and Help. The target board is set to Arduino Uno. The file LAB_4A.ino is open, displaying the following code:

```
1 #include <Wire.h>
2 #include <MPU6050.h>
3
4 MPU6050 mpu;
5 int16_t ax, ay, az;
6 int16_t gx, gy, gz;
7
8 void setup() {
9   Serial.begin(9600);
10  Wire.begin();
11
12  mpu.initialize();
13
14  if (!mpu.testConnection()) {
15    Serial.println("MPU6050 connection failed!");
16    while (1); // Stop if the sensor is not connected
17  } else {
18    Serial.println("MPU6050 connection successful!");
19  }
20 }
21
22 void loop() {
```

The Serial Monitor is open, showing the output of the program. The first line is "MPU6050 connection successful!". Subsequent lines show acceleration and gyroscope data in the format "Accel: x, y, z | Gyro: x, y, z".

```
Message (Enter to send message to 'Arduino Uno' on 'COM5')
Accel: 344, -135, 15220 | Gyro: -131, -93, -96
Accel: 380, -172, 16204 | Gyro: -131, 9, -163
Accel: 580, -20, 15576 | Gyro: -159, -15, -198
Accel: 456, -252, 16468 | Gyro: -173, -335, -43
Accel: 568, -216, 16316 | Gyro: -136, -57, -160
Accel: 424, -68, 16020 | Gyro: -138, -81, -92
Accel: 420, 28, 16192 | Gyro: -160, -71, -104
Accel: 420, -8, 16336 | Gyro: -156, -98, -126
Accel: 396, -196, 16136 | Gyro: -145, -104, -104
Accel: 480, -256, 16116 | Gyro: -161, -74, -147
Accel: 476, -208, 16224 | Gyro: -156, -103, -108
```



The screenshot shows a Python IDE interface. The top bar includes a menu icon, "PythonProject1", and "Version control". The project explorer on the left shows the file structure, including lab_4a.py. The main editor displays the following Python code:

```
1 import serial
2
3 ser = serial.Serial(port='COM5', baudrate=9600, timeout=1)
4
5 try:
6     while True:
7         data = ser.readline().decode(encoding='utf-8', errors='ignore').strip()
8         if data:
9             print(data)
10 except KeyboardInterrupt:
11     print("\nSerial communication terminated.")
12     ser.close()
13
```

The Run console at the bottom shows the output of the program, displaying acceleration and gyroscope data in the format "Accel: x, y, z | Gyro: x, y, z".

```
Accel: 7200, 000, 12024 | Gyro: 100, 112, 110
Accel: -9244, -304, 12676 | Gyro: -139, -145, -100
Accel: -9248, -340, 12820 | Gyro: -156, -122, -123
Accel: -9336, -476, 12676 | Gyro: -155, -37, -99
Accel: -9132, -328, 12700 | Gyro: -162, -106, -114
Accel: -9128, -288, 12912 | Gyro: -147, -137, -96
Accel: -9100, -372, 12688 | Gyro: -154, -82, -91
Accel: -9232, -152, 12724 | Gyro: -165, -348, -139
Accel: -9084, -476, 12860 | Gyro: -165, -226, -97
```

Discussion

The experiment demonstrated successful serial communication between an MPU6050 Inertial Measurement Unit (IMU) sensor and a personal computer via an Arduino microcontroller. This setup allowed for real-time data acquisition of accelerometer and gyroscope values, which formed the foundation for a simple hand gesture recognition system. The implementation illustrates how sensor data can be transmitted from embedded systems to desktop environments for further processing, classification, and visualization.

The MPU6050 sensor was effectively interfaced with the Arduino using I2C communication. The use of the Wire.h and MPU6050.h libraries simplified the process of initializing the sensor and acquiring motion data. Data was continuously sent over a serial connection at a baud rate of 9600 and successfully read by a Python script using the Pyserial library. This serial link enabled seamless communication and opened up opportunities for real-time signal processing on the PC.

One of the key highlights of the experiment was the basic gesture recognition algorithm implemented in the Arduino sketch. By applying threshold conditions to the accelerometer and gyroscope readings, distinct gestures could be identified, such as "Gesture 1" and "Gesture 2." While this approach is rudimentary and highly dependent on precise threshold tuning, it effectively demonstrated the concept of gesture recognition using IMU data.

On the PC side, the Python script responded dynamically to gesture events sent from the Arduino, highlighting potential use cases in human-computer interaction (HCI), robotics, and assistive technologies. For instance, each detected gesture could be mapped to a unique function, such as controlling an actuator or triggering a visual indicator.

However, this system also has several limitations. The gesture detection algorithm lacks robustness and scalability, as it only recognizes predefined motion thresholds and doesn't account for noise, sensor drift, or complex hand movements. A more advanced approach, such as machine learning-based classification, would offer improved accuracy and adaptability for a wider range of gestures. Moreover, real-time visualization of hand movement paths in an x-y coordinate system was mentioned but not implemented in the given code, representing a potential area for enhancement.

Conclusion

To summarize, this experiment was a success. Interfacing the IMU with the Arduino board was achieved accordingly, and tracking the motion of the hands was successful. Implementing this on a larger scale with additional modifications to support scalability could prove to be extremely helpful in many aspects, such

as in physiotherapy where hand movement can be used to track the recovery state of patients, or controlling home appliances without having to manually turn switches on. However, while this system proved itself to be useful, it lacked robustness, as it wasn't able to account for noise, sensor drift or complex hand movements.

Recommendations

- 1) Calibrate the sensor properly to prevent sensor drift and noise
- 2) Implement an AI system to detect complex hand movements

References

<https://www.pnisenor.com/understanding-imu-modules-guide-to-inertial-measurement-units/>

Student Declaration Form

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 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 that no further improvement on the reports is needed from any of the individual contributors to the report.

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

Signature: 	Read	/
Name: Muhamad Yusuf Irfan bin Abidin	Understand	/
Matric No: 2210915	Agree	/
Contribution : Discussion, Experimental Set-Up, Methodology		

Signature: Tanisha	Read	/
Name: Tanisha Rowshan Saleh	Understand	/
Matric No: 2228290	Agree	/
Contribution : Abstract, Introduction		

Signature: 	Read	/
Name: Muhamad Imran Haziq bin Haizam	Understand	/
Matric No: 2216429	Agree	/
Contribution : Results		

Signature: 	Read	/
Name: Ariesha Daliela Hanim Binti Amir	Understand	/
Matric No: 2316834	Agree	/
Contribution : Conclusion, Recommendation		