LABORATORY REPORT

MECHATRONICS SYSTEM INTEGRATION

MCTA 3203

SEMESTER 2 2023/2024

WEEK 4

LAB: SERIAL COMMUNICATION

SUBMISSION DATE:

GROUP: 6

	NAME	MATRIC NUMBER
1.	MUHAMMAD AZIM ZUFAYRI BIN NORAZAM	2211973
2.	MUHAMMAD AMMAR FARIS BIN ARAFAS	2212645
3.	AHMAD ADAM BIN NORHELME	2219973
4.	AHMAD HAFIZULLAH BIN IBRAHIM	2216185

Table of Contents

Abstract

This module aims to demonstrate the effectiveness of the MPU6050 sensor as a small, low cost and easy to use device for the purpose of sensing motion and orientation data. The capability of this particular sensor gives it a versatile function for a variety of applications involving the aforementioned motion sensing ability. In this experiment, we will be using the sensor MPU6050 attached to an Arduino microcontroller, and a PC to communicate via python. The sensor will capture relatable data and send it to the Arduino microcontroller for further processing. Arduino will then send the data to PC via serial communication and the data is analyzed in real time through Python. With proper calibration and coding, we may create an algorithm that can understand certain gestures through motion capture and output a proper response to the input. This project gives a deep understanding about the potential of using even a cheap sensor like MPU6050 and Arduino to do simple yet quite profound tasks such as motion sensing.

Introduction

The use of MPU6050 sensor is significant in the environment of cheap, low-cost and simple setup. The combination of accelerometer & gyroscope measurement in one sensor makes this particular sensor very useful and preferred as a motion and orientation measurement. Furthermore, with the use of Arduino as a microcontroller, we can take the data and communicate to a PC via serial communication and further analyze it using Python. This experiment aims to demonstrate the versatility of Arduino and how we can use MPU6050 as an example of motion sensing and calibrating.

Materials and Equipment

- Arduino board
- MPU6050 sensor
- Computer with Arduino IDE and Python installed
- Jumper wires or breadboard wires to connect the sensors to arduino
- USB cable

Methodology:

Experimental Setup

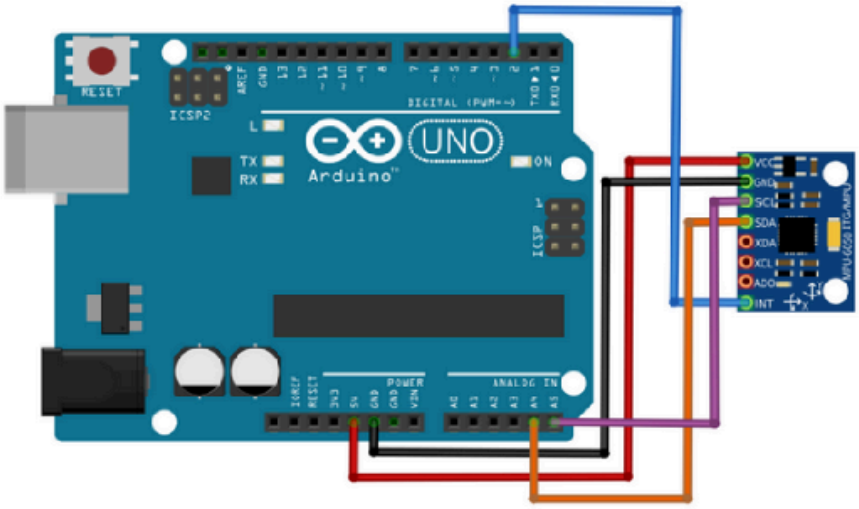


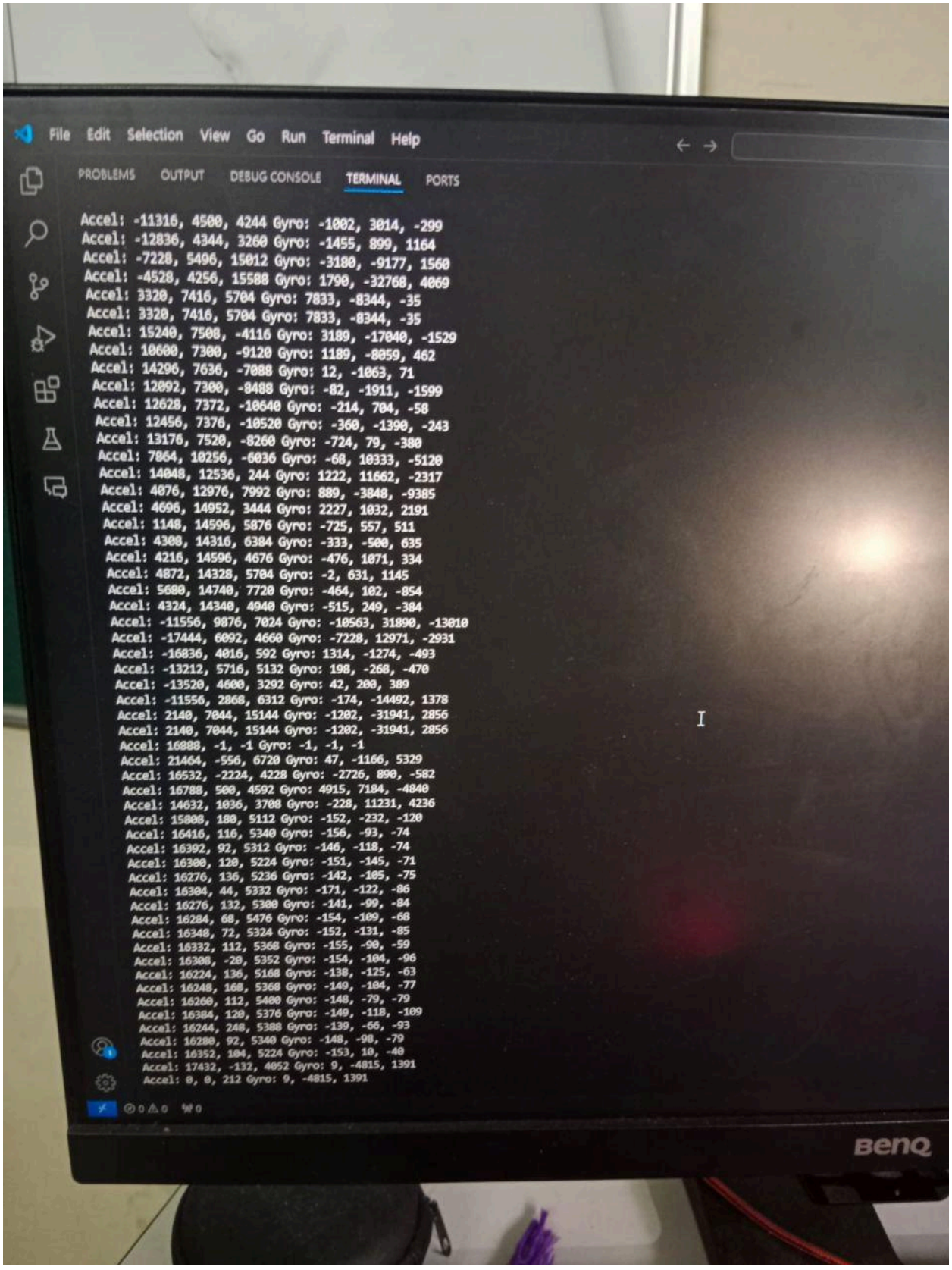
Fig 1

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.

Results


File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Accel: -11316, 4500, 4244 Gyro: -1002, 3014, -299
Accel: -12836, 4344, 3260 Gyro: -1455, 899, 1164
Accel: -7228, 5496, 15012 Gyro: -3180, -9177, 1560
Accel: -4528, 4256, 15588 Gyro: 1790, -32768, 4069
Accel: 3320, 7416, 5704 Gyro: 7833, -8344, -35
Accel: 3320, 7416, 5704 Gyro: 7833, -8344, -35
Accel: 15240, 7508, -4116 Gyro: 3189, -17040, -1529
Accel: 10600, 7300, -9120 Gyro: 1189, -8059, 462
Accel: 14296, 7636, -7088 Gyro: 12, -1063, 71
Accel: 12092, 7300, -8488 Gyro: -82, -1911, -1599
Accel: 12628, 7372, -10640 Gyro: -214, 704, -58
Accel: 12456, 7376, -10520 Gyro: -360, -1390, -243
Accel: 13176, 7520, -8260 Gyro: -724, 79, -380
Accel: 7864, 10256, -6036 Gyro: -68, 10333, -5120
Accel: 14048, 12536, 244 Gyro: 1222, 11662, -2317
Accel: 4076, 12976, 7992 Gyro: 889, -3848, -9385
Accel: 4696, 14952, 3444 Gyro: 2227, 1032, 2191
Accel: 1148, 14596, 5876 Gyro: -725, 557, 511
Accel: 4308, 14316, 6384 Gyro: -333, -500, 635
Accel: 4216, 14596, 4676 Gyro: -476, 1071, 334
Accel: 4872, 14328, 5704 Gyro: -2, 631, 1145
Accel: 5680, 14740, 7720 Gyro: -464, 102, -854
Accel: 4324, 14340, 4940 Gyro: -515, 249, -384
Accel: -11556, 9876, 7024 Gyro: -10563, 31890, -13010
Accel: -17444, 6092, 4660 Gyro: -7228, 12971, -2931
Accel: -16836, 4016, 592 Gyro: 1314, -1274, -493
Accel: -13212, 5716, 5132 Gyro: 198, -268, -470
Accel: -13520, 4600, 3292 Gyro: 42, 200, 389
Accel: -11556, 2868, 6312 Gyro: -174, -14492, 1378
Accel: 2140, 7044, 15144 Gyro: -1202, -31941, 2856
Accel: 2140, 7044, 15144 Gyro: -1202, -31941, 2856
Accel: 16888, -1, -1 Gyro: -1, -1, -1
Accel: 21464, -556, 6720 Gyro: 47, -1166, 5329
Accel: 16532, -2224, 4228 Gyro: -2726, 890, -582
Accel: 16788, 500, 4592 Gyro: 4915, 7184, -4840
Accel: 14632, 1036, 3708 Gyro: -228, 11231, 4236
Accel: 15808, 180, 5112 Gyro: -152, -232, -120
Accel: 16416, 116, 5340 Gyro: -156, -93, -74
Accel: 16392, 92, 5312 Gyro: -146, -118, -74
Accel: 16300, 120, 5224 Gyro: -151, -145, -71
Accel: 16276, 136, 5236 Gyro: -142, -105, -75
Accel: 16304, 44, 5332 Gyro: -171, -122, -86
Accel: 16276, 132, 5300 Gyro: -141, -99, -84
Accel: 16284, 68, 5476 Gyro: -154, -109, -68
Accel: 16348, 72, 5324 Gyro: -152, -131, -85
Accel: 16332, 112, 5368 Gyro: -155, -90, -59
Accel: 16308, -20, 5352 Gyro: -154, -104, -96
Accel: 16224, 136, 5168 Gyro: -138, -125, -63
Accel: 16248, 168, 5368 Gyro: -149, -104, -77
Accel: 16260, 112, 5400 Gyro: -148, -79, -79
Accel: 16384, 120, 5376 Gyro: -149, -118, -109
Accel: 16244, 248, 5388 Gyro: -139, -66, -93
Accel: 16280, 92, 5340 Gyro: -148, -98, -79
Accel: 16352, 104, 5224 Gyro: -153, 10, -40
Accel: 17432, -132, 4052 Gyro: 9, -4815, 1391
Accel: 0, 0, 212 Gyro: 9, -4815, 1391
BenQ

Discussion

Components:

- MPU6050 Sensor: This sensor combines an accelerometer and a gyroscope, providing data about motion and orientation.
- Arduino Board: Acts as the intermediary between the sensor and the computer, facilitating data collection and transmission.
- Computer: Runs Python scripts to receive, process, and act upon the data received from the Arduino.
- USB Cable and Power Supply: Connects the Arduino to the computer and provides power to both the Arduino and the MPU6050.
- Hardware Setup:
  The connections involve linking the MPU6050 sensor to the Arduino board and ensuring power and ground connections are established. This setup enables the Arduino to read data from the sensor and transmit it to the computer via USB.

Arduino Code:

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

MPU6050 mpu;

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

void loop() {
  int16_t ax, ay, az, gx, gy, gz; // Define variables to store sensor readings
  mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz); // function call

  Serial.print("Accel: ");
  Serial.print(ax);
```

```
  Serial.print(", ");
  Serial.print(ay);
  Serial.print(", ");
  Serial.print(az);
  Serial.print(" Gyro: ");
  Serial.print(gx);
  Serial.print(", ");
  Serial.print(gy);
  Serial.print(", ");
  Serial.println(gz);

  delay(250); // Adjust the delay as needed
}
```

Python Code:

```
import serial
ser = serial.Serial('COM5', 9600)
while True:
    data = ser.readline().decode('utf-8').strip()
    print(data
```

Sensor Selection:

The MPU6050 was chosen for this task due to several key features that make it suitable for gesture recognition applications:

- Combination of Accelerometer and Gyroscope: The MPU6050 integrates both an accelerometer and a gyroscope in a single chip, providing comprehensive motion sensing capabilities.
- Small Size and Low Cost: Its compact size and affordability make it accessible for hobbyist projects and prototyping.
- I2C Communication: The sensor communicates via the I2C protocol, which is widely supported by microcontrollers like the Arduino, simplifying interfacing.

- High Accuracy: The MPU6050 offers relatively high accuracy and stability in measuring motion and orientation, crucial for precise gesture recognition tasks.
- Data Processing:
  Accelerometer and gyroscope data provide complementary information about motion:

- Accelerometer: Measures linear acceleration along the sensor's axes, indicating changes in velocity. This data is crucial for detecting gestures involving movement, such as swipes or shakes.

- Gyroscope: Measures angular velocity or rotational motion around the sensor's axes, providing information about orientation changes. Gyroscope data helps identify gestures involving rotations or twists, like turning a knob or flipping a device. Algorithms such as threshold-based detection, machine learning classifiers, or signal processing techniques like Fourier transforms can be employed to analyze this data effectively and identify specific gestures.

- Gesture Definition:
  In the Arduino code, gestures are defined and identified based on threshold values and conditions applied to the accelerometer data. Threshold values determine the sensitivity of gesture detection, distinguishing between normal movements and intentional gestures. For instance, crossing a certain threshold of acceleration along specific axes might indicate a swipe or shake gesture. These thresholds and conditions need to be carefully tuned to ensure accurate and reliable gesture recognition without false positives or negatives.

  Python Processing: The Python script receives data from the Arduino, decodes it, and interprets gesture information. It then executes corresponding actions based on the recognized gestures. This script can be expanded to perform more complex actions by integrating additional logic, such as:

- Implementing a state machine to recognize sequential gestures or gestures combined with button presses.

- Integrating with other systems, such as controlling IoT devices, virtual reality environments, or interactive installations.
- Adding error handling and feedback mechanisms to improve system robustness and user experience.

Task Week 4

Arduino code:

```
#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();
  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
  } else if (gesture == 3) {
    Serial.println("Gesture 3");
    // Perform an action for Gesture 3
  }
  // Add more gesture cases as needed
  previousGesture = gesture;
  // Read sensor data and send it to Python
  int ax, ay, az, gx, gy, gz;
```

```
  mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
  Serial.print(ax);
  Serial.print(",");
  Serial.println(ay);
  delay(500); // Adjust the delay as needed to control the data transmission rate
}
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
  } else if (ax < threshold && ay > threshold) {
    return 3; // Gesture 3
  }
  // Add more gesture conditions as needed
  return 0; // No gesture detected
}
```

Conclusion

In conclusion, the MPU6050 sensor combined with an Arduino and Python scripts offers a straightforward approach to hand gesture recognition. By capturing accelerometer and gyroscope data, the system identifies predefined gestures based on threshold values. This setup allows for real-time monitoring and response to hand movements. While the provided system is basic, it can be expanded for more complex actions and applications. Gesture recognition technology holds promise for improving user interactions and accessibility across various domains

Recommendations

Calibration: Account for variations in hand gestures among different users by implementing calibration routines or adaptive learning mechanisms. Personalizing the system's gesture recognition capabilities can improve user experience and accuracy.

Fine-tune Threshold Values: Experiment with different threshold values in the Arduino code to ensure accurate detection of hand gestures. Adjusting these thresholds can improve the system's sensitivity and specificity, reducing false positives and negatives.

Consider Environmental Factors: Evaluate the system's performance under different environmental conditions, such as lighting or background noise. Make adjustments to the algorithm and sensor settings to ensure robust operation in various contexts.

Acknowledgement

A special thanks goes out to Dr. Wahju Sediono, Dr. Ali Sophian, Dr. Zulkifli Bin Zainal Abidin, our teaching assistants and our peers for their invaluable help and support in finishing this report. Their advice and guidance have greatly improved the quality and understanding of the project. Their commitments are greatly appreciated.

Student Declaration

We, hereby declare that this project is entirely our own work except the documents that were given as references. Any external sources utilized for reference or inspiration have been properly cited and credited.

azim

__Muhammad Azim Zufayri Bin Norazam__

ammar

__Muhammad Ammar Faris Bin Arafas__

hafiz

__Ahmad Hafizullah Bin Ibrahim__

adam

__Ahmad Adam Bin Norhelme__