MCTA 3203 SYSTEM INTEGRATION LAB

WEEK 4:
SERIAL AND USB INTERFACING WITH MICROCONTROLLER AND COMPUTER BASED SYSTEM (2): SENSORS AND ACTUATORS

SECTION : 1
GROUP NUMBER : 7 (G)

LECTURER'S NAME : DR ZULKIFLI BIN ZAINAL ABIDIN

GROUP MEMBERS :

NO :		COURSE	MATRIC
1.	MUHAMMAD NAQIB AIMAN BIN YUSNI	BMCT	2117765
2.	MUHAMAD AZRI BIN ABDUL AZIZ	BMCT	2113537
3.	MUHAMMAD HAFIDZUDDIN HANIF DANIAL BIN NORIZAL	BMCT	2123651
4.	SHAREEN ARAWIE BIN HISHAM	BMCT	2116943

TABLE OF CONTENTS

Abstract

The MPU6050 is a versatile sensor renowned for its compact size, affordability, and seamless integration capabilities, rendering it indispensable for applications necessitating motion and orientation data. By amalgamating accelerometer and gyroscope measurements, it furnishes a rich fount of information suitable for diverse projects and devices. This abstract encapsulates the essential steps required to establish a connection between a personal computer and an MPU6050 Inertial Measurement Unit (IMU) using an Arduino board. The process entails configuring the Arduino board to serve as an intermediary, facilitating communication between the MPU6050 and the computer. Key steps include initialising the Arduino's serial communication, configuring the MPU6050's registers, calibrating sensor readings, and transmitting data packets to the computer. Throughout the experiment, meticulous attention is paid to ensuring seamless data transmission and accuracy, laying the groundwork for leveraging the MPU6050's capabilities in various applications.

Introduction

The objective of the experiment is to interact between arduino and PC using python and generate motion data from MPU6050 Inertial Measurement Unit (IMU). By using python script and arduino code that is suitable , we can gain the data from MPU6050.The Python script can store the data, display it, or carry out any additional processing that we have programmed it to do. By elucidating these key procedures, this experiment aims to equip enthusiasts and professionals with the knowledge to harness the capabilities of the MPU6050 sensor through Python, empowering them to innovate and create within the realm of motion sensing technology.

Material 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 colours.

Circuit & Experimental Setup

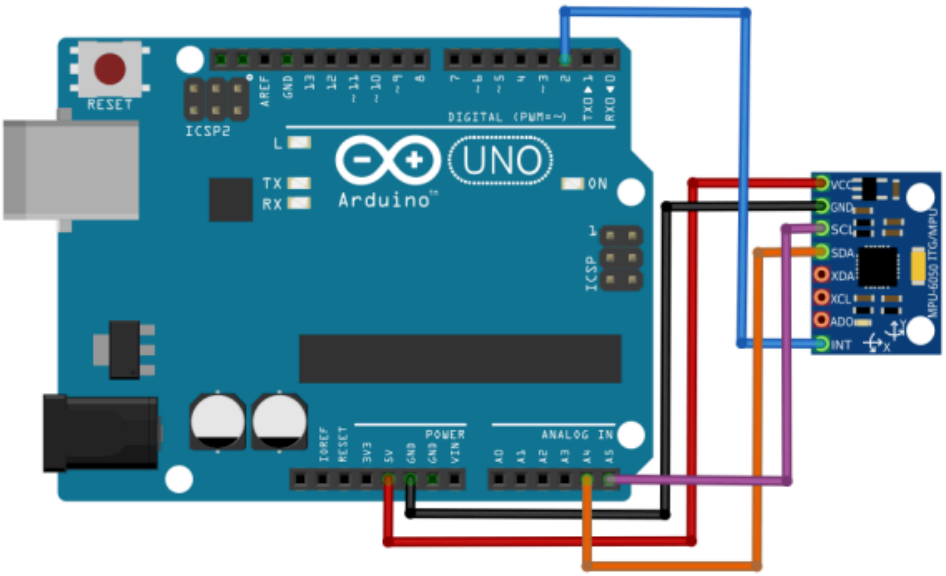

Fig. 1: Arduino-MPU6050 Connections

1. Connect the MPU6050 sensor to the Arduino board using the appropriate pins. The MPU6050
2. 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).
3. Connect the power supply and ground of the MPU6050 to the Arduino's 5V and GND pins.
4. Ensure that the Arduino board is connected to your PC via USB.

Coding

Arduino Coding:

```
#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 coding :

```
import serial
import time
import matplotlib.pyplot as plt

def process_data_line(line):
    try:
        _, ax, ay, az, _, gx, gy, gz = line.split()
         ax, ay, az, gx, gy, gz = float(ax), float(ay), float(az), float(gx),
float(gy), float(gz)
        return ax, ay, az, gx, gy, gz
    except ValueError:
        return None

time.sleep(3)
ax_lst, ay_lst, az_lst, gx_lst, gy_lst, gz_lst = [], [], [], [], [], []

with serial.Serial('COM8', 9600) as ser:
    while True:
        try:
            data = ser.readline().decode('utf-8').strip()
            sensor_data = process_data_line(data)
            if sensor_data:
                ax, ay, az, gx, gy, gz = sensor_data
                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}')
                time.sleep(0.1)  # Add a delay of 0.1 seconds
        except KeyboardInterrupt:
            break

with open('data.txt', 'w+') as f:
    for ax, ay, az, gx, gy, gz in zip(ax_lst, ay_lst, az_lst, gx_lst, gy_lst,
gz_lst):
        f.write(f'{ax} {ay} {az} {gx} {gy} {gz}\n')

# Plotting
plt.figure(figsize=(12, 8))

# Process data from file
ax_lst, ay_lst, az_lst, gx_lst, gy_lst, gz_lst = [], [], [], [], [], []
with open('data.txt', 'r') as f:
    for line in f:
        ax, ay, az, gx, gy, gz = map(float, line.split())
        ax_lst.append(ax)
        ay_lst.append(ay)
        az_lst.append(az)
        gx_lst.append(gx)
```

```
        gy_lst.append(gy)
        gz_lst.append(gz)

# Plot acceleration and velocity
plt.subplot(2, 2, 1)
plt.plot(ax_lst)
plt.title('Acceleration X')

plt.subplot(2, 2, 2)
plt.plot(ay_lst)
plt.title('Acceleration Y')

plt.subplot(2, 2, 3)
plt.plot(gx_lst)
plt.title('Gyro X')

plt.subplot(2, 2, 4)
plt.plot(gy_lst)
plt.title('Gyro Y')

plt.tight_layout()
plt.show()
```

Results

The output from the experiment showcases readings from the MPU6050 sensor, denoting accelerometer values in the x, y, and z directions, printed serially via the Arduino Uno to the PC using python. Each line of data , followed by the respective acceleration values along the x, y, and z axes, typically measured in units like metres per second squared (m/s^2) or gravitational units (g). A delay that can be adjustable between readings controls the data transmission rate. This setup enables real-time monitoring and analysis of motion-related data, offering insights into various applications such as motion tracking, gesture recognition, or orientation sensing.

Discussion

For real-time motion monitoring, the MPU6050 sensor is integrated with an Arduino Uno and Python. This provides a flexible and approachable method for gathering and analysing data. Users can instantly observe motion patterns by serially transferring accelerometer data to a PC. This allows for the usage of motion tracking, gesture recognition, and orientation sensing, among other uses. The data transfer rate may be adjusted according to need, and Python scripts make it easier to parse and visualise the data, providing opportunities for more in-depth research and interpretation. Although effective, scaling up this setup for larger datasets or more complicated applications may pose issues with memory constraints and data

processing performance. To ensure smooth operation and precise analysis, these difficulties might be solved by introducing parallel processing strategies, optimising code efficiency.

Conclusion

In conclusion, the experiment showcases the potential of the MPU6050 sensor, Arduino Uno, and Python integration for real-time motion monitoring and analysis. Through serial data transmission and Python processing, the setup offers a flexible and efficient means of capturing and interpreting accelerometer data. The applications span various fields, from sports analytics to robotics, highlighting the versatility and relevance of real-time motion analysis in contemporary technology. While challenges may arise when scaling up the setup for more extensive applications, addressing these hurdles through optimization and resource allocation can unlock even greater potential for understanding and leveraging motion-related data in diverse contexts.

Student's Declaration

Declaration:

We certify that this project/assignment is entirely our own work, except where we have given fully documented references to the work of others, and that the material in this assignment has not previously been submitted for assessment in any formal course of study.

azri

_____

Name: Muhamad Azri bin Abdul Aziz

hafidzuddin

_____

Name: Muhammad Hafidzuddin Hanif Danial bin Norizal

arawie

_____

Name: Shareen Arawie bin Hisham

naqib

_____

Name: Muhammad Naqib Aiman bin Yusni

Date: 18/3/2024