Mechatronics System Integration (MCTA3203)

Week **4a**: Serial and USB interfacing with microcontroller and computer based system (2): Sensors and actuators.

The MPU6050's small size, low cost, and ease of integration make it a versatile sensor for a variety of applications that require motion and orientation data. Its ability to combine accelerometer and gyroscope measurements provides a valuable source of information for a wide range of projects and devices. To establish a connection between a personal computer and an IMU (Inertial Measurement Unit) MPU6050 through an Arduino board, the process involves a series of key steps:

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.

Hardware Setup:

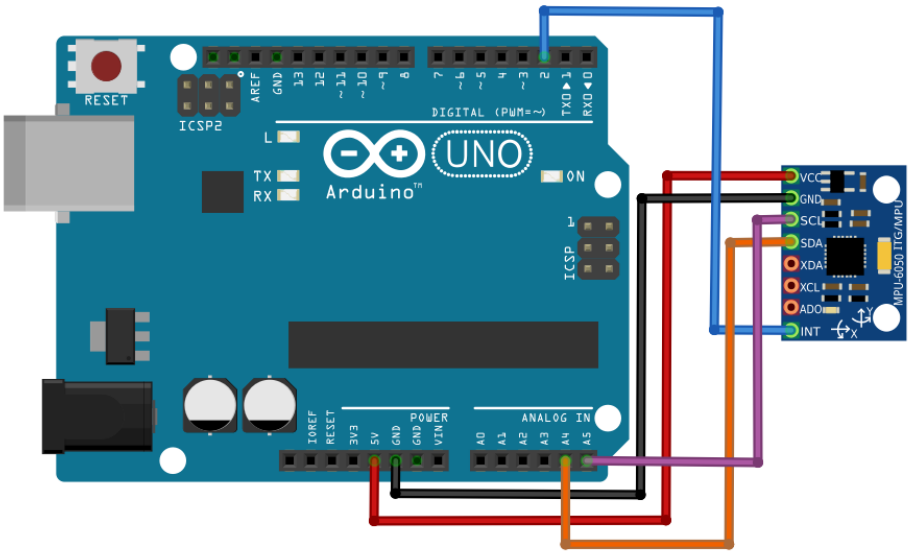

Fig. 1: Arduino-MPU6050 Connections

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.

Arduino Code:

Write the Arduino code to read data from the MPU6050 sensor and send it to the PC via the serial port. Here's a *basic* example Arduino sketch:

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

MPU6050 mpu;

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

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

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

Upload this code to your Arduino board. It reads accelerometer and gyroscope data from the MPU6050 sensor and sends it to the PC via the serial port at a baud rate of 9600.

Python Code on PC:

On your PC, you can use Python to receive and process the data from the Arduino. You'll need the *pyserial* library to communicate with the Arduino. Install it using pip if you haven't already:

```
pip install pyserial
```

Here's a simple Python script to *receive* and *print* the data from the Arduino:

```
import serial

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

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

Run the Setup:

With the Arduino code uploaded and the Python script running on your PC, you should see the accelerometer and gyroscope data from the MPU6050 sensor printed to your PC's console.

This setup allows your PC to communicate with the Arduino, read sensor data from the MPU6050, and process the data as needed. You can further enhance the Python script to save the data, visualize it, or perform any other desired processing.

Task

Create a straightforward hand gesture recognition system by capturing accelerometer and gyroscope data during the execution of predefined hand movements. Employ an algorithm to identify and categorize these gestures using the collected sensor data. Additionally, visualize the *paths* of hand movement in an x-y coordinate system.

Hint: Try the following codes.

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();

  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
}
```

Python Code:

```
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
```