MPU-6050 6 Axis IMU

Introduction

- The MPU-6050 is a six degree of freedom Inertial Measurement Unit. It consists of a 3-axis accelerometer and a 3-axis rate gyroscope.
- The accelerometer has a settable sensitivity range of 2Gs to 16 Gs.
- The rate gyroscope has a settable sensitivity range of 250 degrees per second to 5000 degrees per second.
- In this lesson, the rate gyro will be used to determine orientation. The library includes functions for calculating the angle of the sensor after it is calibrated. A processing program will be used to graphically demonstrate the orientation of the sensor.

Connect the MPU-6050

- Connect the MPU-6050 to the I2C bus on the SLATE board.
- Connect the sensor VIN to the SLATE 3.3V.
- Connect the sensor GND to the SLATE GND.
- Connect the sensor SCL to the SLATE D5 SCL.
- Connect the sensor SDA to the SLATE D4 SDA.

MPU-6050 Library

- Download the library from www.stensat.org. This library was modified to work properly on the ESP8266 SLATE board.
- In the Arduino IDE, select the **Sketch** menu and then select **Include Library**. Select **Add .ZIP file**. Locate the file and select it. It will be added to your library.
- In the **File** menu, select **Examples** then locate **MPU6050_tockn**. Select **GetAllData**. The program will open in a window.
  - Before compiling, make the following changes:
  - Line 11, Wire.begin(); ==> Wire.begin(4,5);
  - Line 12, mpu6050.begin(); ==> mpu6050.begin(ACCEL_2G,GYRO_500);
  - Change line 10 baud rate from 9600 to 115200.
- Compile and upload the program.
- When done uploading, open the Serial monitor and make sure the baud rate is set to 115200.
- When the program starts, it will spend about 3 seconds calibrating. Make sure the sensor is not moving during this time. It will calibrate right after the code finishes uploading.

The Code

- In the program, the sensor library is included at line 2. Line 3 loads the I2C library.
- Line 5 creates a sensor object. The argument is Wire which tells the library to use the I2C interface. This is done to allow multiple I2C buses to be used. Only one is used here.
- The timer variable in line 7 is used to track the time and have the display updated sensor data once a second.
- Lines 9-14 is the setup function. The serial interface is configued then the I2C interface. Next the sensor is configured with the accelerometer set to 2G range and the gyro set to 500 degrees per second rotation rate range.
- Line 13 calls a library function to calibrate the gyroscope. The gyroscope has what is called a DC offset or constant offset. This is an error that all sensors have and can be measured with the sensor not moving. The library subtracts the offset from all measurements.

The Code

- Lines 16 – 46 is the loop function.
- Line 19 determines if a second has passed. If so, the reset of the code is executed.
- Line 17 is the function that collects the sensor data. The results are kept in the library variables.
- Lines 22-40 display the sensor results Notice that the values displayed are function calls. **mpu6050.getTemp()** will return the temperatuer in Celcius. **mpu6050.getAccx()** will return the X-axis accelerometer value in Gs and so on. Notice the values are in floating point and processed from the raw values.

The Code

- Lines 31 and 32 return the sensor angle in the X and Y axis based on the accelerometer.
- **mpu6050.getAccAngleX()** returns an angle in degrees referenced to the Z and X axis.
- **mpu6050.getAccAngleY()** returns the angle in degrees referenced to the Z and Y axis.
- **mpu6050.getGyroAngleX()** returns the angle calculated by the accumulation of the rate gyro around the X axis.
- **mpu6050.getGyroAngleY()** returns the angle calculated by the accumulation of the rate gyro around the Y axis.
- **mpu6050.getGyroAngleZ()** returns the angle calculated by the accumulation of the rate gyro around the Z axis.

The Code

- **mpu6050.getAngleX()** provides the angle around the X axis based on the combination of the accelerometer and gyro.
- **mpu6050.getAngleY()** provides the angle around the Y axis based on the combination of the accelerometer and gyro.
- **mpu6050.getangleZ()** provides the angle around the Z axis based on the combination of the accelerometer and gyro.
- These three functions provide the best orientation value of the sensor and can be used to indicate the orientation of any device it is connected.

IMU Demonstration SLATE Code

- This program is a simpler version of the example program where only the X,Y,Z angles are sent over the USB port.
- Enter this program in the Arduino IDE and upload to the SLATE.
- The python program that plotted the X,Y,Z values of the accelerometer in the SYST101 lesson will be modified to plot the angles from the MPU6050.

```
#include <MPU6050_tockn.h>
#include <Wire.h>

MPU6050 mpu6050(Wire);
long timer = 0;
char buf[64];

void setup() {
  Serial.begin(115200);
  Wire.begin(4,5);
  mpu6050.begin(ACCEL_2G,GYRO_500);
  mpu6050.calcGyroOffsets(true);
}

void loop() {
  mpu6050.update();
  Serial.print(mpu6050.getAngleX());
  Serial.print(“ “);
  Serial.print(mpu6050.getAngleY());
  Serial.print(“ “);
  Serial.println(mpu6050.getAngleZ());
  delay(10);
}
```

Python Plotting Program

- The python code is a copy of the code for plotting the accelerometer data. Minor changes to the code are shown here.
- Line 9, the range is changed to support +/- 180 degrees range.
- Line 16 needs to be set to the COM port of the SLATE.
- Line 20 changes the plot title text.
- Line 22 changes the Y axis label.

```
1   import matplotlib.pyplot as plt
2   import matplotlib.animation as animation
3   import serial
4
5   fig = plt.figure()
6   ax = fig.add_subplot(1, 1, 1)
7
8   x_len = 200
9   y_range = [-200, 200]
10
11 xs = list(range(0, 200))
12 xa = [0] * x_len
13 ya = [0] * x_len
14 za = [0] * x_len
15 ax.set_ylim(y_range)
16 s = serial.Serial('COM5',115200)
17 line, = ax.plot(xs, xa,label='X')
18 line2, = ax.plot(xs,ya,label='Y')
19 line3, = ax.plot(xs,za,label='Z')
20 plt.title('MPU6050')
21 plt.xlabel('Samples')
22 plt.ylabel('Degrees')
23 ax.legend()
```

Python Plotting Program

- The rest of the code is not changed.
- Refer to page 93 of the SYST 101 document for details of how the program works.

```python

def animate(i, xa,ya,za):
    a = s.readline()
    b = a.decode('utf-8')
    c = b.split(' ')
    if len(c) == 3:
        xa.append(float(c[0]))
        ya.append(float(c[1]))
        za.append(float(c[2]))
        xa = xa[-x_len:]
        ya = ya[-x_len:]
        za = za[-x_len:]
        line.set_ydata(xa)
        line2.set_ydata(ya)
        line3.set_ydata(za)
        return line,line2,line3,
    else:
        return line,line2,line3,
ani = animation.FuncAnimation(fig,animate,
    fargs=(xa,ya,za,),
    interval=1,
    blit=True)
plt.show()  # show the figure
```

Testing

- Upload the Arduino Code to the SLATE. Check to see if the data is properly being generated. All angles should be near zero if the board is laying flat on a level table.
- Close the serial monitor window and run the python program.
- Rotate the board on the table and see how the Z-axis data changes. Rotate more than 360 degrees. Notice the plot goes out of range. Rotate in the opposite direction. The Z-axis plot should return back into the plot area.
- Do the same for the other two axis. Notice they only go to +/- 180 degrees. The accelerometer is being used with the gyro to maintain orientation information. The accelerometers are using gravity as a reference to down. For the Z-axis, there is no reference. A magnetometer could be used as a reference for the Z-axis using the earth’s magnetic field.