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.
- 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 loaded, the sensor library is included at line 1. Line 2 loads the I2C library.
- Line 4 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 6 is used to track the time and have the display updated sensor data once a second.
- Lines 8-13 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 12 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 15 – 41 is the loop function.
- Line 16 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 19-26 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

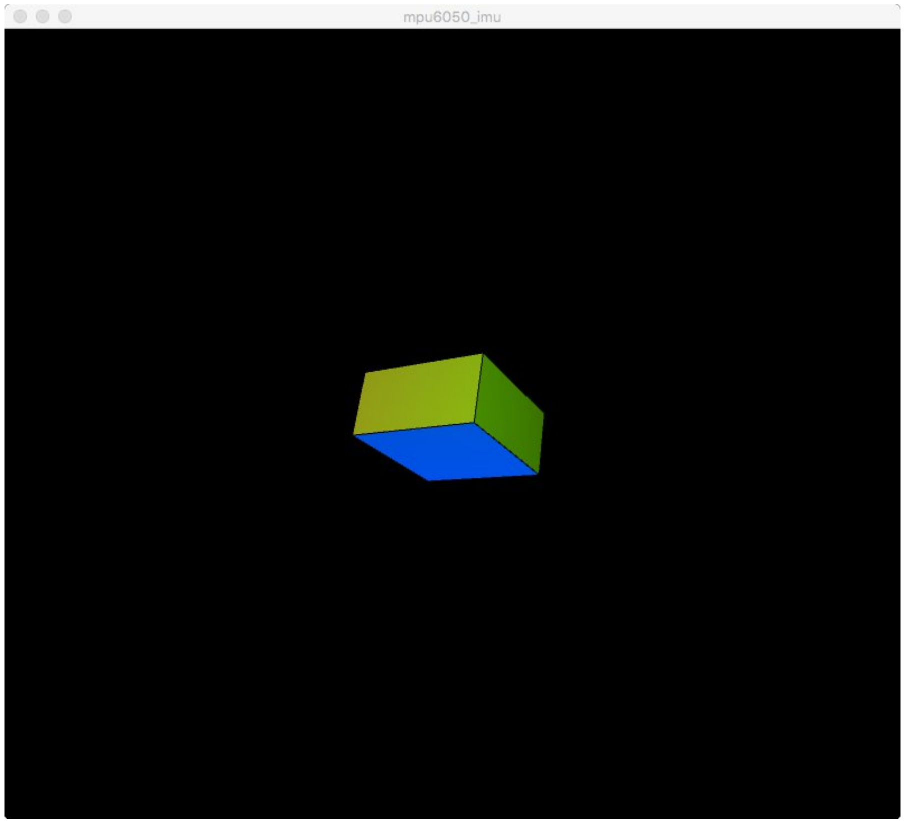
- Lines 28 and 29 return the sensor angle in the X and Y axis based on the accelerometer.
- mpu6050.getAccAngleX() returns an angle in degrees referenced to the Z and X axis.
- mpu6050.getAccAngleY() returns the angle in degrees referenced to the Z and Y axis.
- mpu6050.getGyroAngleX() returns the angle 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

- This next program will demonstrate how the sensor can be used to control the orientation of an object in a 3D rendered program.
- The demonstration will use a wireless connection using UDP packets. UDP packets allow for higher rates of data transmission. It is possible for some packets to be lost but for this demonstration, lost packets are not an issue.



IMU Demonstration SLATE Code

- The SLATE program will send UDP packets. Initialization is similar to the example program but only the angles for the three axis are sent.
- The code to the right is the setup portion. Most of it is the same as before. The addition is the WiFi configuration.

```
#include <ESP8266WiFi.h>
#include <WiFiUdp.h>
#include <MPU6050_tockn.h>
#include <Wire.h>

WiFiUDP client;

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);
  WiFi.mode(WIFI_AP);
  WiFi.softAP("UDPIMU");
}
```

IMU Demonstration SLATE Code

- The loop function updates the sensor values.
- It then creates a string using snprintf() with the three angles separated by a comma.
- The last three lines send the UDP packet.
- **client.beginPacket()** identifies the target IP address and port number.
- **client.write()** fills the packet with the string.
- **client.endPacket()** sends the packet.

```
void loop() {
  if(millis() - timer > 100) {
    mpu6050.update();
   timer = millis();
    snprintf(buf,64,"%f,%f,%f\n",
         mpu6050.getAngleX(),
         mpu6050.getAngleY(),
         mpu6050.getAngleZ());
    Serial.print(buf);

client.beginPacket("192.168.4.2",3000);
    client.write(buf);
    client.endPacket();
  }
}
```

IMU Demonstration SLATE Code

- The data is sent 10 times a second. the rate is controlled by the value 100 in the highlighted line. It is the number of milliseconds between samples. It can be changed.

```
void loop() {
  if(millis() - timer > 100) {
    mpu6050.update();
   timer = millis();
    snprintf(buf,64,"%f,%f,%f\n",
         mpu6050.getAngleX(),
         mpu6050.getAngleY(),
         mpu6050.getAngleZ());
    Serial.print(buf);

client.beginPacket("192.168.4.2",3000);
    client.write(buf);
    client.endPacket();
  }
}
```

IMU Demonstration Processing Code

- The processing code receives the UDP packets.
- First, select **UDP** in the **Sketch Import Library** menu. If you do not have the UDP library, select **Add Library** in the **Import Library** menu. Locate the UDP library at the bottom of the list. Internet access is required.
- The UDP object p is declared. A global string message is created. It will hold the data received from the SLATE.
- In the setup() function, the window size is set. The extra argument P3D directs the program to use 3D acceleration.

```
import hypermedia.net.*;

UDP p;

String message;

void setup() {
  size(800, 700, P3D);
  p = new UDP(this,
3000);
  p.listen(true);
}
```

IMU Demonstration Processing Code

- Next the UDP port is configured. All UDP packets are to go to port 3000.
- p.listen(true) tells the program to monitor for UDP packets.

```
import hypermedia.net.*;

UDP p;

String message;

void setup() {
  size(800, 700, P3D);
  p = new UDP(this,
3000);
  p.listen(true);
}
```

IMU Demonstration Processing Code

- UDP packets are captured with the **receive()** function below.
- The **receive()** function is an event function. It only executes when a UDP packet is received. The function copies the information received into the **message** string.

```
void draw() {
  if (message != null) {
    spotLight(255, 0, 0, 500, 0, 0, -1, 0, 0, 50,
1);
    spotLight(0, 255, 0, 0, 500, 0, 0, -1, 0, 50,
1);
    spotLight(0, 0, 255, 0, 0, 500, 0, 0, -1, 50,
2);
    camera(50, 500, 50, 0, 0, 0, 0, 0, 1);
    String[] dat = message.split(",");
    float x = float(dat[0])/360.0 * 6.28;
    float y = float(dat[1]) /360.0 * 6.28;
    float z = float(trim(dat[2])) / 360.0 * 6.28;
    println(x + " " + y + " " + z);
    background(0); background to black
    pushMatrix();
    rotateY(y);
    rotateX(x);
    rotateZ(z);
    fill(250, 250, 250);
    box(100, 100, 50);
    popMatrix();
  }
}

void receive( byte[] data, String ip, int port )
{
  message = new String( data );
}
```

IMU Demonstration Processing Code

- The draw() function renders the 3D object. First, the message string is checked to see if there is any data.
- If there is data, the scene is created. Three spot lights are created with colors red, green and blue pointing to the center of the scene from three directions.

```
void draw() {
  if (message != null) {
    spotLight(255, 0, 0, 500, 0, 0, -1, 0, 0, 50, 1);
    spotLight(0, 255, 0, 0, 500, 0, 0, -1, 0, 50, 1);
    spotLight(0, 0, 255, 0, 0, 500, 0, 0, -1, 50, 2);
    camera(50, 500, 50, 0, 0, 0, 0, 0, 1);
    String[] dat = message.split(",");
    float x = float(dat[0])/360.0 * 6.28;
    float y = float(dat[1]) /360.0 * 6.28;
    float z = float(trim(dat[2])) / 360.0 * 6.28;
    println(x + " " + y + " " + z);
    background(0); background to black
    pushMatrix();
    rotateY(y);
    rotateX(x);
    rotateZ(z);
    fill(250, 250, 250);
    box(100, 100, 50);
    popMatrix();
  }
}

void receive( byte[] data, String ip, int port ) {
  message = new String( data );
}
```

IMU Demonstration Processing Code

- The camera is positioned and points to the center of the scene with coordinates at (0,0,0).
- Next the message string is split into three elements for X, Y and Z.
- The elements are converted to floating point values. These values are the angles.
- The z value includes the trim function which removes the line feed from the string.

```
void draw() {
  if (message != null) {
    spotLight(255, 0, 0, 500, 0, 0, -1, 0, 0, 50, 1);
    spotLight(0, 255, 0, 0, 500, 0, 0, -1, 0, 50, 1);
    spotLight(0, 0, 255, 0, 0, 500, 0, 0, -1, 50, 2);
    camera(50, 500, 50, 0, 0, 0, 0, 0, 1);
    String[] dat = message.split(",");
    float x = float(dat[0])/360.0 * 6.28;
    float y = float(dat[1]) /360.0 * 6.28;
    float z = float(trim(dat[2])) / 360.0 * 6.28;
    println(x + " " + y + " " + z);
    background(0); background to black
    pushMatrix();
    rotateY(y);
    rotateX(x);
    rotateZ(z);
    fill(250, 250, 250);
    box(100, 100, 50);
    popMatrix();
  }
}

void receive( byte[] data, String ip, int port ) {
  message = new String( data );
}
```

IMU Demonstration Processing Code

- Next, the display is cleared to black.
- The object which is a rectangular box is then rotated in all axis to the angles generated by the sensor.
- The box is colored white so the colors from the lights can be seen clearly.
- The box is then rendered.

```
void draw() {
  if (message != null) {
    spotLight(255, 0, 0, 500, 0, 0, -1, 0, 0, 50, 1);
    spotLight(0, 255, 0, 0, 500, 0, 0, -1, 0, 50, 1);
    spotLight(0, 0, 255, 0, 0, 500, 0, 0, -1, 50, 2);
    camera(50, 500, 50, 0, 0, 0, 0, 0, 1);
    String[] dat = message.split(",");
    float x = float(dat[0])/360.0 * 6.28;
    float y = float(dat[1]) /360.0 * 6.28;
    float z = float(trim(dat[2])) / 360.0 * 6.28;
    println(x + " " + y + " " + z);
    background(0); background to black
    pushMatrix();
    rotateY(y);
    rotateX(x);
    rotateZ(z);
    fill(250, 250, 250);
    box(100, 100, 50);
    popMatrix();
  }
}

void receive( byte[] data, String ip, int port ) {
  message = new String( data );
}
```

IMU Demonstration Processing Code

- The draw() function executes at about 60 Hz which is the typical video refresh rate.
- If the message string is not updated, the current value is processed and used again.
- Run the program and move the sensor around.

```
void draw() {
  if (message != null) {
    spotLight(255, 0, 0, 500, 0, 0, -1, 0, 0, 50, 1);
    spotLight(0, 255, 0, 0, 500, 0, 0, -1, 0, 50, 1);
    spotLight(0, 0, 255, 0, 0, 500, 0, 0, -1, 50, 2);
    camera(50, 500, 50, 0, 0, 0, 0, 0, 1);
    String[] dat = message.split(",");
    float x = float(dat[0])/360.0 * 6.28;
    float y = float(dat[1]) /360.0 * 6.28;
    float z = float(trim(dat[2])) / 360.0 * 6.28;
    println(x + " " + y + " " + z);
    background(0); background to black
    pushMatrix();
    rotateY(y);
    rotateX(x);
    rotateZ(z);
    fill(250, 250, 250);
    box(100, 100, 50);
    popMatrix();
  }
}

void receive( byte[] data, String ip, int port ) {
  message = new String( data );
}
```