DOCUMENTATION

PRODUCT: **GY-521 THREE AXIS ACCELEROMETER/GYROSCOPE BREAKOUT BOARD (MPU 6050 CHIP)**

MANUFACTURER: **INVENSENSE INC.**

MICROSCALE EMBEDDED

1. INTRODUCTION

The Invensense GY521 Sensor Board contains a very accurate (and tiny) sensor chip – MPU6050 – that contains an accelerometer and a gyroscope inbuilt in the sensor based on MEMS (Micro Electro Mechanical Systems) technology. The MEMS Technology uses silicon wafers to fabricate capacitive circuits that emulate an accelerometer and a gyroscope (and other mechanical parts like have holes, cavity, channels, cantilevers, membranes, etc.) generating currents proportional to the magnitude of acceleration (linear or angular) induced on the sensor.

An accelerometer is a device that is able to measure linear acceleration – motion along an axis, and also the rate at which linear velocity is changing with time. It is able to measure acceleration forces either static (e.g. the force of gravity) or dynamic (e.g. forces induced on the sensor by motion or vibration). The gyroscope, on the other hand, is a device capable of measuring rotational velocity (or angular rate) – or simply, the rate at which the angle of a body is changing – on the given axis of consideration. In order to obtain the angle of tilt, the angular rate can be integrated over a sampled time – although there tends to be an accumulation of error in the angular data as time progresses. For a three-axis gyroscope, the outputs would be in three forms:

- Yaw: Angular rate on the vertical axis.
- Pitch: Angular rate on the horizontal axis.
- Roll: Angular rate on the front to back axis (most likely y-axis but dependent on orientation).

2. PRODUCT OVERVIEW

The GY-521 breakout board has an onboard sensor chip – MPU-6050 – that contains an accelerometer, a gyroscope, an onboard temperature sensor and terminals to connect to an external sensor using I2C serial protocol (e.g. a magnetometer) through the auxiliary I2C terminals on the breakout board. The I2C serial protocol is the main mode of communication between the device and the microcontroller, or the device and an external sensor.

These inbuilt sensors – the accelerometer and the gyroscope – have three axes, and can sense acceleration and rotation on the three axes, giving output through the 16-bit Analog-to-digital converter on the device. It features three 16-bit analog-to-digital converters (ADCs) for digitizing the gyroscope outputs and three 16-bit ADCs for digitizing the accelerometer outputs. For precision tracking of both fast and slow motions, the parts feature a user-programmable gyroscope full-scale range of ±250, ±500, ±1000, and ±2000°/sec (dps), and a user-programmable accelerometer full-scale range of ±2g, ±4g, ±8g, and ±16g. The real acceleration (in g) and angular rate (in degrees per second) data can be gotten from the ADC values by dividing appropriately with the sensitivity values of the accelerometer and the gyroscope respectively.

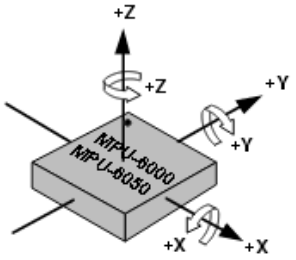
NOTE: 1g = 9.81m/s^2 approximately

2.1. PRODUCT CHARACTERISTICS

- Standard I2C communications protocol with a frequency of 400KHz
- Built-in 16bit ADC
- 16-bit data output
- Tri-Axis Gyro with a sensitivity up to 131 LSBs/dps and a full-scale range of ± 250, 500, 1000, 2000°/s (dps)
- Tri-Axis Accelerometer with a programmable full-scale range of ± 2, ± 4, ± 8, ± 16g
- Digital-output temperature sensor
- 10,000 g shock tolerant
- Internal Digital Motion Processing™ (DMP™) engine supports 3D Motion Processing and gesture recognition algorithms
- Auxiliary master I2C bus for reading data from external sensors (e.g., magnetometer)

2.2. ORIENTATION OF AXES

The orientation of the axes of the sensor board is given in the figure below; with the orientation of rotation going clockwise around any of the considered axes when the view is from the origin.



2.3. ADC OUTPUT

2.3.1. GYROSCOPE

The gyroscope tells how much the angular velocity is changing with time. It gives a measure of how fast a certain rotation occurs along any of the axes. It produces a value when rotating about any axis, and this value increases as the angular velocity increases (that is as the speed of turn increases). As stated previously, MEMS technology is used in implementing a gyroscope using Coriolis Effect (An effect that is observed in a rotating body, such that object was thrown from the body so as to follow a straight path ends up following a curved path instead, in the direction of the rotation).

The gyroscope uses Coriolis Effect to transform an angular velocity into a displacement. The Coriolis force acts perpendicular to the rotation axis and to the velocity of the body in the rotating frame. The displacement induces a change in capacitance between the mass and the housing (made through MEMS tech.), thus transforming the angular motion rate input to the gyroscope into an electrical output; which in turn serves as input to the 16-bit ADC for each axis of the gyroscope.

The user-set full scale range (e.g. ± 250 °/sec) of the gyroscope maps to signed values of the 16-bit output, i.e. from -32768 to +32767 (2^{16} values = 65536 values). The angular rate data

can be gotten from the gyroscope by dividing appropriately by the sensitivity value of the gyroscope dependent on the user set full scale range:

± 250 °/sec - 131 LSB/(°/s)

± 500 °/sec - 65.5 LSB/(°/s)

± 250 °/sec - 32.8 LSB/(°/s)

± 250 °/sec - 16.4 LSB/(°/s)

2.3.2. ACCELEROMETER

The accelerometer can be used to sense the orientation of gravity for static and dynamic bodies since all bodies on earth are subject to gravity. It can also be used to measure axial acceleration, detect vibration (or shock) levels.

When an acceleration occurs in given direction, there tends to be an "Inertial Force" equivalent to the acceleration force which the accelerometer uses to measure the induced acceleration. This Inertial Force is best seen in a moving vehicle with a body (say a human). When the moving vehicle accelerates (or increases its velocity), the body in the vehicle feels a force that pushes it backwards opposite to the direction of the acceleration, that force is called the Inertial Force. Hence, the accelerometer would give a positive output when the inertial force is in the direction of the positive axis of the device. Take for example: when the sensor board is placed on a flat surface, the only force acting on the device is that of gravity; the ADC output would be positive for the z-axis (Check orientation of axes on sensor board) because the inertial force (Not the gravitational force) is in the direction of the positive axis.

The user-set full-scale range (e.g. ±2g) of the accelerometer maps to signed values of the 16-bit output, i.e. from -32768 to +32767 (2^{16} values = 65536 values).

2.3.3. TEMPERATURE

An on-chip temperature sensor and ADC are used to measure the MPU-6050 die temperature. The readings from the ADC can be read from the FIFO or the Sensor Data registers. The range of temperature that can be read is -40 to 85°C.

To compute the temperature reading use the following formula:

Temperature (in °C) = (Signed Value from ADC)/340 + 36.53

2.4. FUNCTIONS/APPLICATIONS

The device can be used as a stand-alone temperature sensor, accelerometer or gyroscope or at best a combination of the three. The onboard chip – MPU-6050 – can be integrated into a smartphone to detect tilt or orientation of the device using the accelerometer.

It is also used in modern vehicles as a trigger for airbags. It can be used in quadcopters for stability (usually, a combination of the output of the accelerometer and the gyroscope would give a better stability than using them individually). It can also be used in unmanned ground vehicles (UGV) to give it a sense of orientation.

3. PRODUCT APPLICATION SAMPLE – DRIVING LEDS WITH THE MPU-6050 ACCELEROMETER

3.1. OVERVIEW

This application uses the accelerometer of the GY521 sensor board to drive six LEDS corresponding to the positive and the negative parts of the three axes of the board. Any acceleration force detected along the direction of any of the axes whether positive or negative would drive the LED corresponding to that direction of the axis. For example, the LED indicating +z direction would always be ON so far the device is placed horizontally and facing upwards because of gravity.

3.2. HARDWARE

3.2.1. PART LIST

- Six LEDs
- Invensense GY521 Sensor board
- Arduino Uno
- Six 470Ω Resistors

3.2.2. CIRCUIT

The sensor board and the Arduino are connected through I2C serial communication, with the sensor board supplying values to the Arduino through the connection, after the 3D data has been processed by the onboard Digital Motion Processor of the sensor chip (the MPU-6050 chip). The LEDs are hooked up to the Arduino PWM pins (pins 3, 5, 6, 9, 10 & 11) to produce analog signals that show the extent of acceleration at any axis.

Take note of the following:

- I2C serial communication pins for the Arduino may vary; I used a clone board that had separate pins for the I2C communication (SDA and SCL pins). However, the standard Arduino board uses pins A4 & A5 (SCL to A5, SDA to A4) for I2C communication.
- Pin2 (External Interrupt Pin) of the Arduino is connected to the INTerrupt pin of the sensor (pin 8).
- The ADO pin of the device is connected to ground to set the address to 0x68 – the default address of the device.

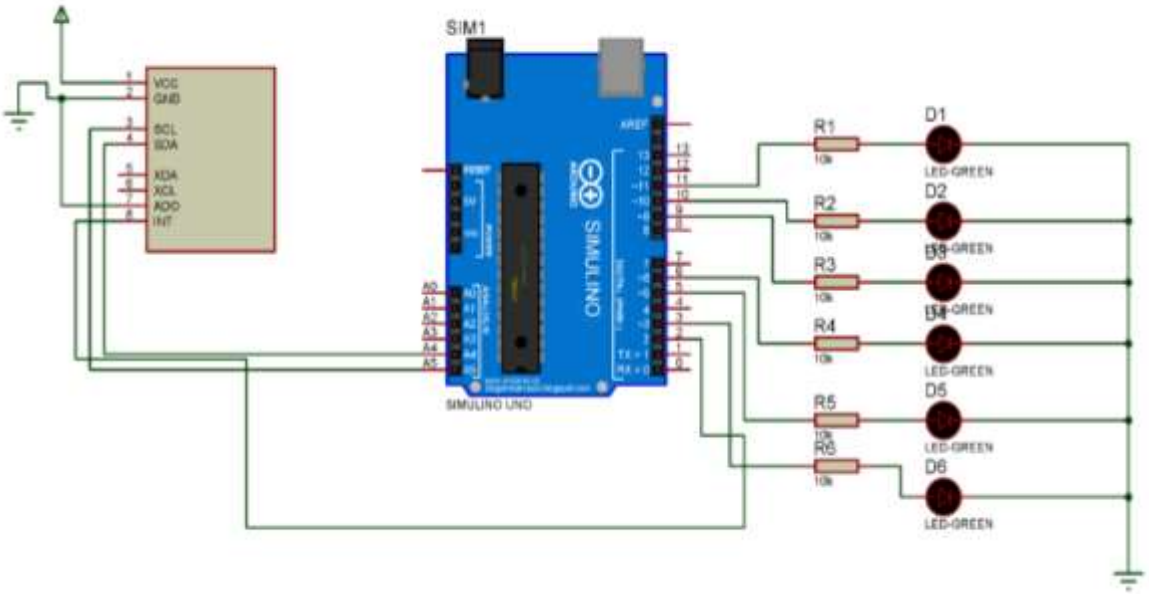
3.2.3. PIN CONNECTION SUMMARY

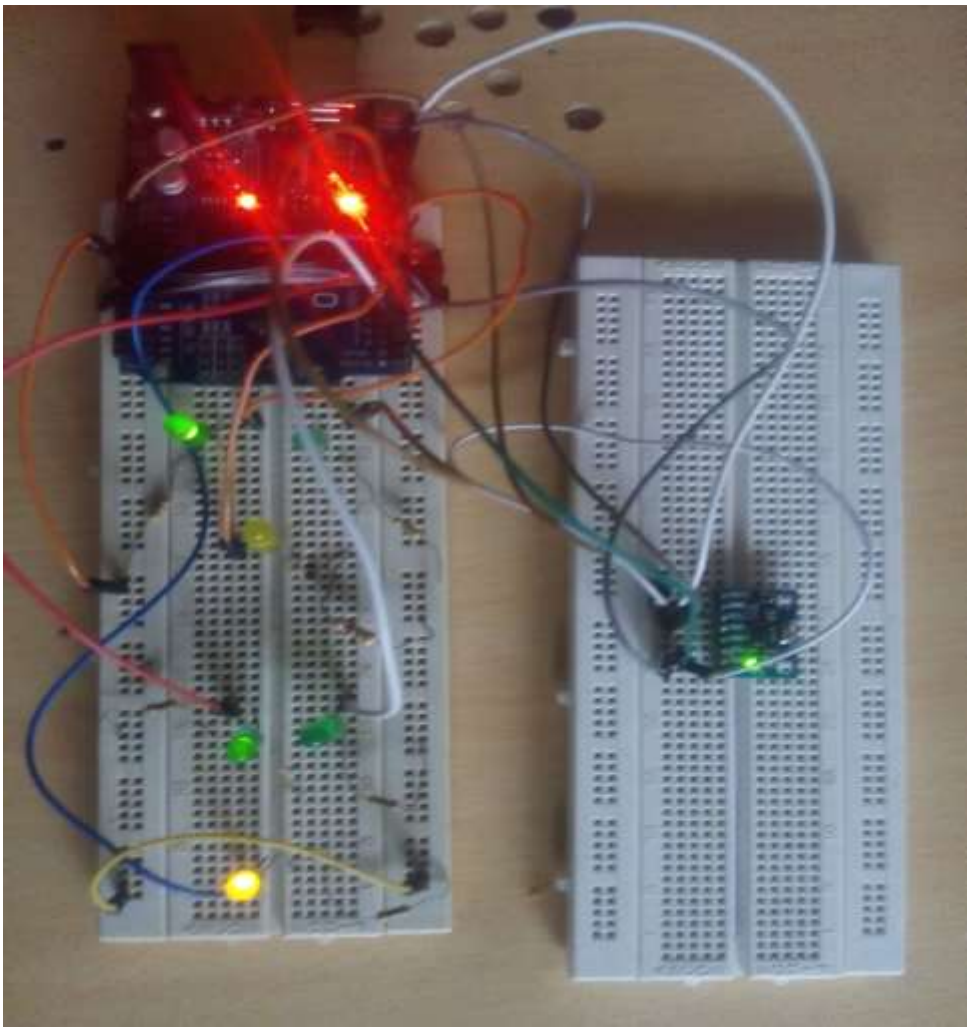
ARDUINO D2	INT
ARDUINO A4	SDA
ARDUINO A5	SCL
GROUND	ADO

The Arduino PWM digital pins: 3, 5, 6, 9, 10, 11 are connected to the various LEDs.

3.2.4. PIN-OUT DESCRIPTION

- **VCC:** This is the input voltage terminal that takes the voltage of 5V. Connecting a voltage of 3V3 (3.3V) to VCC might cause irregular output values. This is so because the board contains an on-board voltage regulator to regulate the input voltage to 3.3V, hence, sending a 3.3V input would not be enough for normal operation.
- **GND:** This is connected to the GROUND terminal.
- **SCL:** This is the serial CLOCK line for the I2C serial communication.
- **SDA:** This is the serial DATA line of the I2C serial communication.
- **XDA:** Auxiliary serial DATA line for connection to the external sensor through I2C.
- **XCL:** Auxiliary serial CLOCK line for connection to the external sensor through I2C.
- **ADO:** This sets the address of the device used.
- **INT:** This pin is used for interrupts





3.3. SOFTWARE

Problem Statement: To show the magnitude of acceleration along any axis with LEDs.

3.3.1. ALGORITHM

These steps are involved for this algorithm:

- Obtain the output of the sensor through I2C serial communication.
- Check whether the values obtained are negative or positive.
- Map the values obtained to the range of Arduino PWM values.
- Turn on the LEDs accordingly.

3.3.2. CODE

The code was written with the Arduino IDE using the I2C library, the MPU6050 library written by Jeff Rowberg, and also the Arduino Wire library for I2C serial communication. These libraries allow for easy collation of data from the device. The device is initialized, offsets set (the

offsets are gotten using the MPU6050_Calibration sketch), the full-scale range of the accelerometer (or gyroscope) set and then the digital low pass filter initialized in the setup() function of the Arduino.

After the setup(), we define three functions:

i. **mapOutput():** This takes the 16-bit acceleration values from the accelerometer and maps it to the PWM range (0 – 255) of the Arduino.
ii. **LightPosLED():** This lights up the LEDs designated for the positive axis.
iii. **LightNegLED():** This lights up the LEDs designated for the negative axis.

In the loop() function, we use an array to hold the output from ADC so as traverse through the values and check if they are positive or negative values in order to light up the corresponding LEDs.

3.4. POINTS TO NOTE/OBSERVATIONS

- Using a battery as voltage input gives a more stable output than the voltage from a power supply connected to mains or from the Arduino 5V supply.
- The raw values of the accelerometer are usually very unstable and vary widely even when static, however, this can be reduced by setting the inbuilt Digital Low Pass filter using the code:

```
mpu.setDLPFMode(value)
//where value can have values from 0 – 7, but specifications recommend
//values from 1-6. Check Register Map – Under Register 26.
```

- The board contains a 3V3 voltage regulator hence, it is better to connect it to a 5v for proper operation, as 3V3 would give it a low power.
- The sensor gives values even when not in motion, this is due to external vibrations and electrical noise.
- Multiple sensors can be connected to the Arduino by connecting their various ADO pins to the output pins of Arduino and selecting them as when needed. To reduce the number of pins used, a shift register can be used.
- For more information on the product especially on the onboard chip – MPU6050, check out the following links:
  - http://store.invensense.com/datasheets/invensense/MPU-6050_DataSheet_V3%204.pdf
  - http://store.invensense.com/Datasheets/invensense/RM-MPU-6000A.pdf
  - http://www.i2cdevlib.com/docs/html/class_m_p_u6050.html#a7c0146d45537e4bd7a0d4c1c476fdab7

4. CODE SNAPSHOT

accelWithLedv1.2

```
// accelerometerWithLed v1.2

// Include the MPU 6050 LIBRARY and other required libraries
#include <MPU6050.h>
#include <I2Cdev.h>
#include <Wire.h>

//Define arduino PWM pins for the axial values of the sensor (x, y & z)
#define positiveX 3
#define negativeX 5
#define positiveY 6
#define negativeY 9
#define positiveZ 10
#define negativeZ 11

// Define MPU object and using the three axis output of the accelerometer
MPU6050 mpu;
int16_t ax, ay, az;

void setup() {
  // put your setup code here, to run once:
  // Initialize the sensor
  mpu.initialize();

  //Define pins as output
  pinMode(positiveX, OUTPUT);
  pinMode(negativeX, OUTPUT);
  pinMode(positiveY, OUTPUT);
  pinMode(positiveY, OUTPUT);
  pinMode(negativeY, OUTPUT);
  pinMode(positiveZ, OUTPUT);
  pinMode(negativeZ, OUTPUT);

   /*Offsets according to the MPU6050_calibration sketch used to adjust reading of the device
    Given in the format: ax ay az gx gy gz
    -1306       532     1189    43      -19     -42

    The calibration sketch can be run again to get values that is specific to device
  */
  //Set the offset of the device
  mpu.setXAccelOffset(-1306);
  mpu.setYAccelOffset(532);
  mpu.setZAccelOffset(1189);

 // Use a larger Full scale reading for Accelerometer and Gyroscope
 // 0 -> +/-2g, 1 -> +/-4g, 2 -> +/-8g, 3 -> +/-16g
 // Uncomment next line to set scale range
 // mpu.setFullScaleRange(3)
 mpu.setDLPFMode(6); // Set Low pass filter of the accelerometer and gyroscope
}

int mapOutput(long posValue) {
  // Takes any of the axial values of the accelerometer and maps it to the PWM range
  return map(posValue, 0, 20000, 0, 255);
}

void lightNegLED(int count, int value) {
```

```
  // Light the LEDs that corresponds to negative x, y or z.
  if (count == 0) {
    analogWrite(negativeX, value);
  }
  else if (count == 1) {
    analogWrite(negativeY, value);
  }
  else {
    analogWrite(negativeZ, value);
  }
}

void lightPosLED(int count, int value) {
  // Light the LEDS that corresponds to positive x, y, z.
  if (count == 0) {
    analogWrite(positiveX, value);
  }
  else if (count == 1) {
    analogWrite(positiveY, value);
  }
  else {
    analogWrite(positiveZ, value);
  }
}

int value[3];

// Holds the current PWM values for the current output

void loop() {
  // put your main code here, to run repeatedly:
  mpu.getAcceleration(&ax, &ay, &az);

  long accelMotion[3] = {ax, ay, az}; // Put the current output into an array

  // Loop through each value
  for (int i = 0; i < 3; i++) {
    if (accelMotion[i] < 0) {
      // Check if it is negative

      long accelValue = 0 - accelMotion[i]; // Make them positive so as to work with them

      value[i] = mapOutput(accelValue); //Map them to the arduino PWM range

      lightNegLED(i, value[i]); //Light the corresponding LEDs
    }
    else {
      value[i] = mapOutput(accelMotion[i]); //Map output to arduino PWM range

      lightPosLED(i, value[i]);//Light the corresponding LEDs
    }
  }
}
```

<

Code formatted for the Arduino forum has been copied to the clipboard.