I wanted to interface my Pi to a Six-Axis Gyro + Accelerometer sensor and the one I settled on was based on a MPU-6050 chip.  I went for this board mainly because I could get it cheap on eBay and wasn't worried about the cost if I broke it.

|  |
| --- |
| [http://1.bp.blogspot.com/-VD4tnn3Towo/UnkvTpt2WvI/AAAAAAAAA2o/0Aa5WlIVXn8/s1600/mpu-6050.jpg](http://www.ebay.co.uk/sch/i.html?_trksid=p2050601.m570.l1313.TR2.TRC0.A0.Xmpu-6050&_nkw=mpu-6050&_sacat=0&_from=R40) |
| [Found on eBay for a few quid](http://www.ebay.co.uk/sch/i.html?_trksid=p2050601.m570.l1313.TR2.TRC0.A0.Xmpu-6050&_nkw=mpu-6050&_sacat=0&_from=R40) |

**Set up (for Rasbian)**

It's an [I2C](http://en.wikipedia.org/wiki/I%C2%B2C) board so first you need to install the relevant Linux drivers, here's how.  Open the file for editing (needs sudo)

sudo vi /etc/modules

add the following lines to the bottom of the file, save it and reboot the Pi

i2c-bcm2708  
i2c-dev

Now check the blacklists file

sudo vi /etc/modprobe.d/raspi-blacklist.conf

and make sure that the following lines start with a # (a comment) if they are present, if not don't worry

#blacklist spi-bcm2708

#blacklist i2c-bcm2708

|  |
| --- |
| <http://3.bp.blogspot.com/-2604CaZam3k/UnlQY-50ejI/AAAAAAAAA3A/-UKwowAWSmA/s1600/IIC-to-6050.png> |
| Pin connections |

**Connecting the sensor**

To connect the sensor you need to use the GPIO pins on the Pi, the important pins are

* Pin 1 - 3.3V connect to VCC
* Pin 3 - SDA connect to SDA
* Pin 5 - SCL connect to SCL
* Pin 6 - Ground connect to GND

these need to be connect as shown in the image.

Once you have the board connected you can test to see if the Pi has detected it.  This is done with the following command to install the i2c tools

sudo apt-get install i2c-tools

and then either

sudo i2cdetect -y 0 (for a Revision 1 board like mine)

or

sudo i2cdetect -y 1 (for a Revision 2 board)

then you should see output showing any I2C devices that are attached and their addresses

0 1 2 3 4 5 6 7 8 9 a b c d e f

00: -- -- -- -- -- -- -- -- -- -- -- -- --

10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --

60: -- -- -- -- -- -- -- -- 68 -- -- -- -- -- -- --

70: -- -- -- -- -- -- -- --

This shows that the Pi has detected the sensor with an address of 0x68 (hexadecimal), this address is needed to interact with it.  Enter the following command and you should get an output of 0x68 on screen if everything is working properly.

sudo i2cget -y 0 0x68 0x75

This command talks to the device whose address is 0x68 (the sensor) and retrieves the value in the register 0x75 which has a default value of 0x68 the same value as the address.

To be able to read from the I2C using Python bus we need to install the smbus module

sudo apt-get install python-smbus

Now to some code, this is just simple test code to make sure the sensor is working

|  |  |  |
| --- | --- | --- |
| 01 | #!/usr/bin/python | |
| 02 |  |

|  |  |  |
| --- | --- | --- |
| 03 | import smbus | |
| 04 | import math |

|  |  |
| --- | --- |
| 05 |  |
| 06 | # Power management registers | |

|  |  |
| --- | --- |
| 07 | power\_mgmt\_1 = 0x6b |
| 08 | power\_mgmt\_2 = 0x6c |

|  |  |
| --- | --- |
| 09 |  |
| 10 | def read\_byte(adr): | |

|  |  |  |
| --- | --- | --- |
| 11 | return bus.read\_byte\_data(address, adr) | |
| 12 |  |

|  |  |
| --- | --- |
| 13 | def read\_word(adr): |
| 14 | high = bus.read\_byte\_data(address, adr) | |

|  |  |  |
| --- | --- | --- |
| 15 | low = bus.read\_byte\_data(address, adr+1) | |
| 16 | val = (high << 8) + low |

|  |  |  |
| --- | --- | --- |
| 17 | return val | |
| 18 |  |

|  |  |
| --- | --- |
| 19 | def read\_word\_2c(adr): |
| 20 | val = read\_word(adr) | |

|  |  |
| --- | --- |
| 21 | if (val >= 0x8000): |
| 22 | return -((65535 - val) + 1) | |

|  |  |
| --- | --- |
| 23 | else: |
| 24 | return val | |

|  |  |
| --- | --- |
| 25 |  |
| 26 | def dist(a,b): | |

|  |  |  |
| --- | --- | --- |
| 27 | return math.sqrt((a\*a)+(b\*b)) | |
| 28 |  |

|  |  |
| --- | --- |
| 29 | def get\_y\_rotation(x,y,z): |
| 30 | radians = math.atan2(x, dist(y,z)) | |

|  |  |  |
| --- | --- | --- |
| 31 | return -math.degrees(radians) | |
| 32 |  |

|  |  |
| --- | --- |
| 33 | def get\_x\_rotation(x,y,z): |
| 34 | radians = math.atan2(y, dist(x,z)) | |

|  |  |  |
| --- | --- | --- |
| 35 | return math.degrees(radians) | |
| 36 |  |

|  |  |
| --- | --- |
| 37 | bus = smbus.SMBus(0) # or bus = smbus.SMBus(1) for Revision 2 boards |
| 38 | address = 0x68       # This is the address value read via the i2cdetect command | |

|  |  |
| --- | --- |
| 39 |  |
| 40 | # Now wake the 6050 up as it starts in sleep mode | |

|  |  |  |
| --- | --- | --- |
| 41 | bus.write\_byte\_data(address, power\_mgmt\_1, 0) | |
| 42 |  |

|  |  |
| --- | --- |
| 43 | print "gyro data" |
| 44 | print "---------" |

|  |  |
| --- | --- |
| 45 |  |
| 46 | gyro\_xout = read\_word\_2c(0x43) | |

|  |  |
| --- | --- |
| 47 | gyro\_yout = read\_word\_2c(0x45) |
| 48 | gyro\_zout = read\_word\_2c(0x47) |

|  |  |
| --- | --- |
| 49 |  |
| 50 | print "gyro\_xout: ", gyro\_xout, " scaled: ", (gyro\_xout / 131) | |

|  |  |
| --- | --- |
| 51 | print "gyro\_yout: ", gyro\_yout, " scaled: ", (gyro\_yout / 131) |
| 52 | print "gyro\_zout: ", gyro\_zout, " scaled: ", (gyro\_zout / 131) |

|  |  |
| --- | --- |
| 53 |  |
| 54 | print | |

|  |  |
| --- | --- |
| 55 | print "accelerometer data" |
| 56 | print "------------------" |

|  |  |
| --- | --- |
| 57 |  |
| 58 | accel\_xout = read\_word\_2c(0x3b) | |

|  |  |
| --- | --- |
| 59 | accel\_yout = read\_word\_2c(0x3d) |
| 60 | accel\_zout = read\_word\_2c(0x3f) |

|  |  |
| --- | --- |
| 61 |  |
| 62 | accel\_xout\_scaled = accel\_xout / 16384.0 | |

|  |  |
| --- | --- |
| 63 | accel\_yout\_scaled = accel\_yout / 16384.0 |
| 64 | accel\_zout\_scaled = accel\_zout / 16384.0 |

|  |  |
| --- | --- |
| 65 |  |
| 66 | print "accel\_xout: ", accel\_xout, " scaled: ", accel\_xout\_scaled | |

|  |  |
| --- | --- |
| 67 | print "accel\_yout: ", accel\_yout, " scaled: ", accel\_yout\_scaled |
| 68 | print "accel\_zout: ", accel\_zout, " scaled: ", accel\_zout\_scaled |

|  |  |
| --- | --- |
| 69 |  |
| 70 | print "x rotation: " , get\_x\_rotation(accel\_xout\_scaled, accel\_yout\_scaled, accel\_zout\_scaled) | |

|  |  |
| --- | --- |
| 71 | print "y rotation: " , get\_y\_rotation(accel\_xout\_scaled, accel\_yout\_scaled, accel\_zout\_scaled) |

When you run the code you will see output similar to this 

gyro data

---------

gyro\_xout: -92 scaled: -1

gyro\_yout: 294 scaled: 2

gyro\_zout: -104 scaled: -1

accelerometer data

------------------

accel\_xout: -3772 scaled: -0.230224609375

accel\_yout: -52 scaled: -0.003173828125

accel\_zout: 15408 scaled: 0.9404296875

x rotation: -13.7558411667

y rotation: -0.187818934829

### Accelerometer data

Let's have a look at the code in more detail.

|  |  |
| --- | --- |
| 1 | accel\_xout = read\_word\_2c(0x3b) |
| 2 | accel\_yout = read\_word\_2c(0x3d) |

|  |  |
| --- | --- |
| 3 | accel\_zout = read\_word\_2c(0x3f) |

These three lines read the raw X,Y & Z accelerometer values, the parameter in each call is the register within the sensor that holds the data.  The sensor has a number of registers which have different functionality as documented in this [datasheet](http://invensense.com/mems/gyro/documents/RM-MPU-6000A-00v4.2.pdf).  The registers we are interested in for the acceleromter data are 0x3b, 0x3d, 0x3f and these hold the raw data in 16 bit [two's complement](http://en.wikipedia.org/wiki/Two's_complement) format.  
  
The following code reads a word (16 bits) from a given register and converts it from two's complement

|  |  |
| --- | --- |
| 1 | def read\_word\_2c(adr): |
| 2 | val = read\_word(adr) | |

|  |  |
| --- | --- |
| 3 | if (val >= 0x8000): |
| 4 | return -((65535 - val) + 1) | |

|  |  |
| --- | --- |
| 5 | else: |
| 6 | return val | |

Once we have the raw data we need to scale it and then convert it into something useful like a rotation angle. Again from the data sheet we can see the default scaling we need to apply to the raw accelerometer values is 16384, so we divide the raw data by this value.

|  |  |
| --- | --- |
| 1 | accel\_xout\_scaled = accel\_xout / 16384.0 |
| 2 | accel\_yout\_scaled = accel\_yout / 16384.0 |

|  |  |
| --- | --- |
| 3 | accel\_zout\_scaled = accel\_zout / 16384.0 |

Now we have the values that gravity is exerting on the sensor in each of the three dimensions, from this we can calculate the rotations in the X & Y axes.

|  |  |
| --- | --- |
| 01 | def dist(a,b): |
| 02 | return math.sqrt((a\*a)+(b\*b)) | |

|  |  |
| --- | --- |
| 03 |  |
| 04 | def get\_x\_rotation(x,y,z): | |

|  |  |  |
| --- | --- | --- |
| 05 | radians = math.atan(x / dist(y,z)) | |
| 06 | return math.degrees(radians) |

|  |  |
| --- | --- |
| 07 |  |
| 08 | def get\_y\_rotation(x,y,z): | |

|  |  |  |
| --- | --- | --- |
| 09 | radians = math.atan(y / dist(x,z)) | |
| 10 | return math.degrees(radians) |

Here is an [excellent article](http://www.hobbytronics.co.uk/accelerometer-info) showing the details behind the maths for this.  What this gives us is the rotation angle in degrees for both the X & Y axes and is shown in the output.

x rotation: -13.755841166

y rotation: -0.187818934829

So in this instance the sensor is rotated by -13.7o around X and -0.1o around Y.

### Gyroscope data

In a similar manner we can read the data from the Gyroscope part of the sensor. This is done in the following code

|  |  |
| --- | --- |
| 1 | gyro\_xout = read\_word\_2c(0x43) |
| 2 | gyro\_yout = read\_word\_2c(0x45) |

|  |  |  |
| --- | --- | --- |
| 3 | gyro\_zout = read\_word\_2c(0x47) | |
| 4 |  |

|  |  |
| --- | --- |
| 5 | print "gyro\_xout: ", gyro\_xout, " scaled: ", (gyro\_xout / 131) |
| 6 | print "gyro\_yout: ", gyro\_yout, " scaled: ", (gyro\_yout / 131) |

|  |  |
| --- | --- |
| 7 | print "gyro\_zout: ", gyro\_zout, " scaled: ", (gyro\_zout / 131) |

So we read the values from the registers 0x43, 0x45 & 0x47, again we can see from the datasheet that these hold the raw gyro data. To scale these we divide by 131 to give the degrees per second rotation value. 

gyro\_xout: -92 scaled: -1

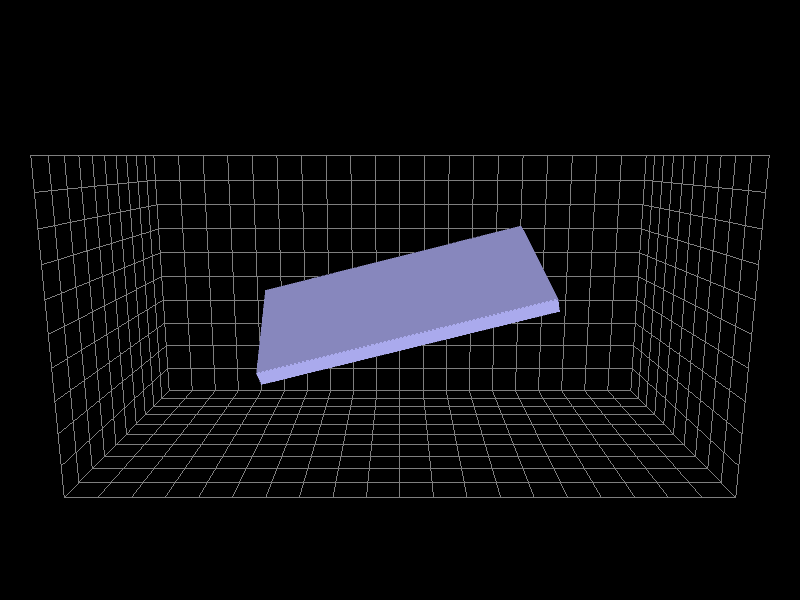
gyro\_yout: 294 scaled: 2

gyro\_zout: -104 scaled: -1

The output in my case show the gyro wasn't moving when I took reading.

### Final thoughts

The code I present here is very basic and should be extended to handle errors and allow the sensor to be configured with different sensitivity levels. I've done this in my application and embedded it into a web server. This allows me to make a simple http request to the Raspberry Pi and get a reading from the sensor.  
  
To help me test and visualise the data better I've written some simple OpenGL code to graphically represent the sensor's orientation in 3D space. 

[](http://4.bp.blogspot.com/-1dv_jsrjdtw/UnrEzCuGYYI/AAAAAAAAA3o/SZKUROP1g58/s1600/3d-view.png)

This OpenGL code runs on my Linux desktop machine and queries the Pi periodically to get the data and renders the above image.

### Using a simple web server to serve up the data

Let's start by setting up a simple server based on [web.py](http://webpy.org/), which is installed via apt-get

sudo apt-get install python-webpy

Now create a directory to put the code in and create a simple test program

mkdir webpy  
cd webpy  
vi server.py

Use the following as a test

|  |  |  |
| --- | --- | --- |
| 01 | #!/usr/bin/python | |
| 02 | import web |

|  |  |
| --- | --- |
| 03 |  |
| 04 | urls = ( | |

|  |  |  |
| --- | --- | --- |
| 05 | '/', 'index' | |
| 06 | ) |

|  |  |
| --- | --- |
| 07 |  |
| 08 | class index: | |

|  |  |
| --- | --- |
| 09 | def GET(self): |
| 10 | return "Hello, world!" | |

|  |  |
| --- | --- |
| 11 |  |
| 12 | if \_\_name\_\_ == "\_\_main\_\_": | |

|  |  |  |
| --- | --- | --- |
| 13 | app = web.application(urls, globals()) | |
| 14 | app.run() |

Save the and then set it as executable with

chmod +x server.py

and then run the code

./server.py

 you will see something like this showing the server is waiting for a request (pressing Ctrl+C will stop the server)

http://0.0.0.0:8080/

Now point your browser at **http://ip-address-of-your-pi:8080** and it will show a web page with the content of **Hello, world!**.  We can make use of this to read data from a remote machine, in my case my Linux desktop.  
  
My R Pi- server ip 192.168.43.211

Linux machine client ip- 192.168.43.172

### Adding the sensor code to the server

Replace the contents of server.py with

|  |  |  |
| --- | --- | --- |
| 01 | #!/usr/bin/python | |
| 02 | import web |

|  |  |  |
| --- | --- | --- |
| 03 | import smbus | |
| 04 | import math |

|  |  |
| --- | --- |
| 05 |  |
| 06 | urls = ( | |

|  |  |  |
| --- | --- | --- |
| 07 | '/', 'index' | |
| 08 | ) |

|  |  |
| --- | --- |
| 09 |  |
| 10 | # Power management registers | |

|  |  |
| --- | --- |
| 11 | power\_mgmt\_1 = 0x6b |
| 12 | power\_mgmt\_2 = 0x6c |

|  |  |
| --- | --- |
| 13 |  |
| 14 | bus = smbus.SMBus(0) # or bus = smbus.SMBus(1) for Revision 2 boards | |

|  |  |  |
| --- | --- | --- |
| 15 | address = 0x68       # This is the address value read via the i2cdetect command | |
| 16 |  |

|  |  |
| --- | --- |
| 17 |  |
| 18 | def read\_byte(adr): | |

|  |  |  |
| --- | --- | --- |
| 19 | return bus.read\_byte\_data(address, adr) | |
| 20 |  |

|  |  |
| --- | --- |
| 21 | def read\_word(adr): |
| 22 | high = bus.read\_byte\_data(address, adr) | |

|  |  |  |
| --- | --- | --- |
| 23 | low = bus.read\_byte\_data(address, adr+1) | |
| 24 | val = (high << 8) + low |

|  |  |  |
| --- | --- | --- |
| 25 | return val | |
| 26 |  |

|  |  |
| --- | --- |
| 27 | def read\_word\_2c(adr): |
| 28 | val = read\_word(adr) | |

|  |  |
| --- | --- |
| 29 | if (val >= 0x8000): |
| 30 | return -((65535 - val) + 1) | |

|  |  |
| --- | --- |
| 31 | else: |
| 32 | return val | |

|  |  |
| --- | --- |
| 33 |  |
| 34 | def dist(a,b): | |

|  |  |  |
| --- | --- | --- |
| 35 | return math.sqrt((a\*a)+(b\*b)) | |
| 36 |  |

|  |  |
| --- | --- |
| 37 | def get\_y\_rotation(x,y,z): |
| 38 | radians = math.atan2(x, dist(y,z)) | |

|  |  |  |
| --- | --- | --- |
| 39 | return -math.degrees(radians) | |
| 40 |  |

|  |  |
| --- | --- |
| 41 | def get\_x\_rotation(x,y,z): |
| 42 | radians = math.atan2(y, dist(x,z)) | |

|  |  |  |
| --- | --- | --- |
| 43 | return math.degrees(radians) | |
| 44 |  |

|  |  |
| --- | --- |
| 45 |  |
| 46 | class index: | |

|  |  |
| --- | --- |
| 47 | def GET(self): |
| 48 | accel\_xout = read\_word\_2c(0x3b) | |

|  |  |
| --- | --- |
| 49 | accel\_yout = read\_word\_2c(0x3d) |
| 50 | accel\_zout = read\_word\_2c(0x3f) |

|  |  |
| --- | --- |
| 51 |  |
| 52 | accel\_xout\_scaled = accel\_xout / 16384.0 | |

|  |  |
| --- | --- |
| 53 | accel\_yout\_scaled = accel\_yout / 16384.0 |
| 54 | accel\_zout\_scaled = accel\_zout / 16384.0 |

|  |  |
| --- | --- |
| 55 |  |
| 56 | return str(get\_x\_rotation(accel\_xout\_scaled, accel\_yout\_scaled, accel\_zout\_scaled))+" "+str(get\_y\_rotation(accel\_xout\_scaled, accel\_yout\_scaled, accel\_zout\_scaled)) | |

|  |  |
| --- | --- |
| 57 |  |
| 58 |  |

|  |  |  |
| --- | --- | --- |
| 59 | if \_\_name\_\_ == "\_\_main\_\_": | |
| 60 |  |

|  |  |  |
| --- | --- | --- |
| 61 | # Now wake the 6050 up as it starts in sleep mode | |
| 62 | bus.write\_byte\_data(address, power\_mgmt\_1, 0) |

|  |  |
| --- | --- |
| 63 |  |
| 64 | app = web.application(urls, globals()) | |

|  |  |
| --- | --- |
| 65 | app.run() |

The server has to be run as sudo so you have permissions to read from the I2C

sudo ./server.py

Connecting via your browser will now return the X & Y rotation values e.g.

-28.7291281627 -39.4833542336

### 3D visualisation

I'm using a Linux desktop and that is all I have tested this simple code on, I've no idea if it works on Windows or Macs and it certainly won't run on the Pi itself.  I'm no OpenGL guru so this code is just hacked together to get something visible.

Setting up OpenGL and pygame

sudo apt-get install python-opengl  
sudo apt-get install python-pygame

Now save the following to a file (in my case level.py) and run it

|  |  |  |
| --- | --- | --- |
| 001 | #!/usr/bin/python | |
| 002 |  |

|  |  |
| --- | --- |
| 003 | import pygame |
| 004 | import urllib |

|  |  |
| --- | --- |
| 005 | from OpenGL.GL import \* |
| 006 | from OpenGL.GLU import \* | |

|  |  |
| --- | --- |
| 007 | from math import radians |
| 008 | from pygame.locals import \* | |

|  |  |
| --- | --- |
| 009 |  |
| 010 | SCREEN\_SIZE = (800, 600) | |

|  |  |
| --- | --- |
| 011 | SCALAR = .5 |
| 012 | SCALAR2 = 0.2 | |

|  |  |
| --- | --- |
| 013 |  |
| 014 | def resize(width, height): | |

|  |  |  |
| --- | --- | --- |
| 015 | glViewport(0, 0, width, height) | |
| 016 | glMatrixMode(GL\_PROJECTION) |

|  |  |
| --- | --- |
| 017 | glLoadIdentity() |
| 018 | gluPerspective(45.0, float(width) / height, 0.001, 10.0) | |

|  |  |  |
| --- | --- | --- |
| 019 | glMatrixMode(GL\_MODELVIEW) | |
| 020 | glLoadIdentity() |

|  |  |  |
| --- | --- | --- |
| 021 | gluLookAt(0.0, 1.0, -5.0, | |
| 022 | 0.0, 0.0, 0.0, |

|  |  |  |
| --- | --- | --- |
| 023 | 0.0, 1.0, 0.0) | |
| 024 |  |

|  |  |
| --- | --- |
| 025 | def init(): |
| 026 | glEnable(GL\_DEPTH\_TEST) | |

|  |  |  |
| --- | --- | --- |
| 027 | glClearColor(0.0, 0.0, 0.0, 0.0) | |
| 028 | glShadeModel(GL\_SMOOTH) |

|  |  |
| --- | --- |
| 029 | glEnable(GL\_BLEND) |
| 030 | glEnable(GL\_POLYGON\_SMOOTH) | |

|  |  |  |
| --- | --- | --- |
| 031 | glHint(GL\_POLYGON\_SMOOTH\_HINT, GL\_NICEST) | |
| 032 | glEnable(GL\_COLOR\_MATERIAL) |

|  |  |  |
| --- | --- | --- |
| 033 | glEnable(GL\_LIGHTING) | |
| 034 | glEnable(GL\_LIGHT0) |

|  |  |  |
| --- | --- | --- |
| 035 | glLightfv(GL\_LIGHT0, GL\_AMBIENT, (0.3, 0.3, 0.3, 1.0)); | |
| 036 |  |

|  |  |
| --- | --- |
| 037 | def read\_values(): |
| 038 | link = "[http://192.168.1.65:8080](http://192.168.1.65:8080/)" # Change this address to your settings | |

|  |  |  |
| --- | --- | --- |
| 039 | f = urllib.urlopen(link) | |
| 040 | myfile = f.read() |

|  |  |  |
| --- | --- | --- |
| 041 | return myfile.split(" ") | |
| 042 |  |

|  |  |
| --- | --- |
| 043 | def run(): |
| 044 | pygame.init() | |

|  |  |  |
| --- | --- | --- |
| 045 | screen = pygame.display.set\_mode(SCREEN\_SIZE, HWSURFACE | OPENGL | DOUBLEBUF) | |
| 046 | resize(\*SCREEN\_SIZE) |

|  |  |
| --- | --- |
| 047 | init() |
| 048 | clock = pygame.time.Clock() | |

|  |  |  |
| --- | --- | --- |
| 049 | cube = Cube((0.0, 0.0, 0.0), (.5, .5, .7)) | |
| 050 | angle = 0 |

|  |  |
| --- | --- |
| 051 |  |
| 052 | while True: | |

|  |  |
| --- | --- |
| 053 | then = pygame.time.get\_ticks() |
| 054 | for event in pygame.event.get(): | |

|  |  |  |
| --- | --- | --- |
| 055 | if event.type == QUIT: | |
| 056 | return |

|  |  |  |
| --- | --- | --- |
| 057 | if event.type == KEYUP and event.key == K\_ESCAPE: | |
| 058 | return |

|  |  |
| --- | --- |
| 059 |  |
| 060 | values = read\_values() | |

|  |  |
| --- | --- |
| 061 | x\_angle = values[0] |
| 062 | y\_angle = values[1] |

|  |  |
| --- | --- |
| 063 |  |
| 064 | glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT) | |

|  |  |
| --- | --- |
| 065 |  |
| 066 | glColor((1.,1.,1.)) | |

|  |  |
| --- | --- |
| 067 | glLineWidth(1) |
| 068 | glBegin(GL\_LINES) | |

|  |  |
| --- | --- |
| 069 |  |
| 070 | for x in range(-20, 22, 2): | |

|  |  |  |
| --- | --- | --- |
| 071 | glVertex3f(x/10.,-1,-1) | |
| 072 | glVertex3f(x/10.,-1,1) |

|  |  |
| --- | --- |
| 073 |  |
| 074 | for x in range(-20, 22, 2): | |

|  |  |
| --- | --- |
| 075 | glVertex3f(x/10.,-1, 1) |
| 076 | glVertex3f(x/10., 1, 1) |

|  |  |
| --- | --- |
| 077 |  |
| 078 | for z in range(-10, 12, 2): | |

|  |  |
| --- | --- |
| 079 | glVertex3f(-2, -1, z/10.) |
| 080 | glVertex3f( 2, -1, z/10.) |

|  |  |
| --- | --- |
| 081 |  |
| 082 | for z in range(-10, 12, 2): | |

|  |  |
| --- | --- |
| 083 | glVertex3f(-2, -1, z/10.) |
| 084 | glVertex3f(-2,  1, z/10.) |

|  |  |
| --- | --- |
| 085 |  |
| 086 | for z in range(-10, 12, 2): | |

|  |  |
| --- | --- |
| 087 | glVertex3f( 2, -1, z/10.) |
| 088 | glVertex3f( 2,  1, z/10.) |

|  |  |
| --- | --- |
| 089 |  |
| 090 | for y in range(-10, 12, 2): | |

|  |  |
| --- | --- |
| 091 | glVertex3f(-2, y/10., 1) |
| 092 | glVertex3f( 2, y/10., 1) |

|  |  |
| --- | --- |
| 093 |  |
| 094 | for y in range(-10, 12, 2): | |

|  |  |
| --- | --- |
| 095 | glVertex3f(-2, y/10., 1) |
| 096 | glVertex3f(-2, y/10., -1) | |

|  |  |
| --- | --- |
| 097 |  |
| 098 | for y in range(-10, 12, 2): | |

|  |  |
| --- | --- |
| 099 | glVertex3f(2, y/10., 1) |
| 100 | glVertex3f(2, y/10., -1) | |

|  |  |
| --- | --- |
| 101 |  |
| 102 | glEnd() | |

|  |  |
| --- | --- |
| 103 | glPushMatrix() |
| 104 | glRotate(float(x\_angle), 1, 0, 0) | |

|  |  |  |
| --- | --- | --- |
| 105 | glRotate(-float(y\_angle), 0, 0, 1) | |
| 106 | cube.render() |

|  |  |
| --- | --- |
| 107 | glPopMatrix() |
| 108 | pygame.display.flip() | |

|  |  |
| --- | --- |
| 109 |  |
| 110 | class Cube(object): | |

|  |  |
| --- | --- |
| 111 |  |
| 112 | def \_\_init\_\_(self, position, color): | |

|  |  |  |
| --- | --- | --- |
| 113 | self.position = position | |
| 114 | self.color = color |

|  |  |
| --- | --- |
| 115 |  |
| 116 | # Cube information | |

|  |  |  |
| --- | --- | --- |
| 117 | num\_faces = 6 | |
| 118 |  |

|  |  |  |
| --- | --- | --- |
| 119 | vertices = [ (-1.0, -0.05, 0.5), | |
| 120 | (1.0, -0.05, 0.5), |

|  |  |
| --- | --- |
| 121 | (1.0, 0.05, 0.5), |
| 122 | (-1.0, 0.05, 0.5), | |

|  |  |  |
| --- | --- | --- |
| 123 | (-1.0, -0.05, -0.5), | |
| 124 | (1.0, -0.05, -0.5), |

|  |  |
| --- | --- |
| 125 | (1.0, 0.05, -0.5), |
| 126 | (-1.0, 0.05, -0.5) ] | |

|  |  |
| --- | --- |
| 127 |  |
| 128 | normals = [ (0.0, 0.0, +1.0),  # front | |

|  |  |
| --- | --- |
| 129 | (0.0, 0.0, -1.0),  # back |
| 130 | (+1.0, 0.0, 0.0),  # right | |

|  |  |  |
| --- | --- | --- |
| 131 | (-1.0, 0.0, 0.0),  # left | |
| 132 | (0.0, +1.0, 0.0),  # top |

|  |  |  |
| --- | --- | --- |
| 133 | (0.0, -1.0, 0.0) ]  # bottom | |
| 134 |  |

|  |  |  |
| --- | --- | --- |
| 135 | vertex\_indices = [ (0, 1, 2, 3),  # front | |
| 136 | (4, 5, 6, 7),  # back |

|  |  |  |
| --- | --- | --- |
| 137 | (1, 5, 6, 2),  # right | |
| 138 | (0, 4, 7, 3),  # left |

|  |  |
| --- | --- |
| 139 | (3, 2, 6, 7),  # top |
| 140 | (0, 1, 5, 4) ]  # bottom | |

|  |  |
| --- | --- |
| 141 |  |
| 142 | def render(self): | |

|  |  |  |
| --- | --- | --- |
| 143 | then = pygame.time.get\_ticks() | |
| 144 | glColor(self.color) |

|  |  |
| --- | --- |
| 145 |  |
| 146 | vertices = self.vertices | |

|  |  |
| --- | --- |
| 147 |  |
| 148 | # Draw all 6 faces of the cube | |

|  |  |  |
| --- | --- | --- |
| 149 | glBegin(GL\_QUADS) | |
| 150 |  |

|  |  |
| --- | --- |
| 151 | for face\_no in xrange(self.num\_faces): |
| 152 | glNormal3dv(self.normals[face\_no]) | |

|  |  |  |
| --- | --- | --- |
| 153 | v1, v2, v3, v4 = self.vertex\_indices[face\_no] | |
| 154 | glVertex(vertices[v1]) |

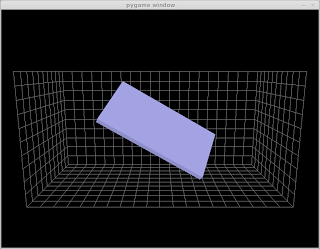
|  |  |
| --- | --- |
| 155 | glVertex(vertices[v2]) |
| 156 | glVertex(vertices[v3]) |

|  |  |  |
| --- | --- | --- |
| 157 | glVertex(vertices[v4]) | |
| 158 | glEnd() |

|  |  |
| --- | --- |
| 159 |  |
| 160 | if \_\_name\_\_ == "\_\_main\_\_": | |

|  |  |
| --- | --- |
| 161 | run() |

Remember to change the URL line **038** to your specific value (the address you used earlier to test the server). When you run it a window will open showing the orientation of the sensor, rotating the sensor will update the display.

[](http://4.bp.blogspot.com/--cu3ozZ8104/Un9jMpZiUGI/AAAAAAAAA5o/gK9u02hRM_k/s1600/level-blog.png)

You'll notice that when the sensor isn't being physically moved the noisy data is causing it to wobble.

Other Readings

https://www.youtube.com/watch?v=M9lZ5Qy5S2s

http://playground.arduino.cc/Main/MPU-6050

http://playground.arduino.cc/Main/Gyro

<http://forum.arduino.cc/index.php?topic=58048.0>

https://www.sunfounder.com/learn/sensor-kit-v2-0-for-raspberry-pi-b-plus/lesson-32-mpu6050-gyro-acceleration-sensor-sensor-kit-v2-0-for-b-plus.html