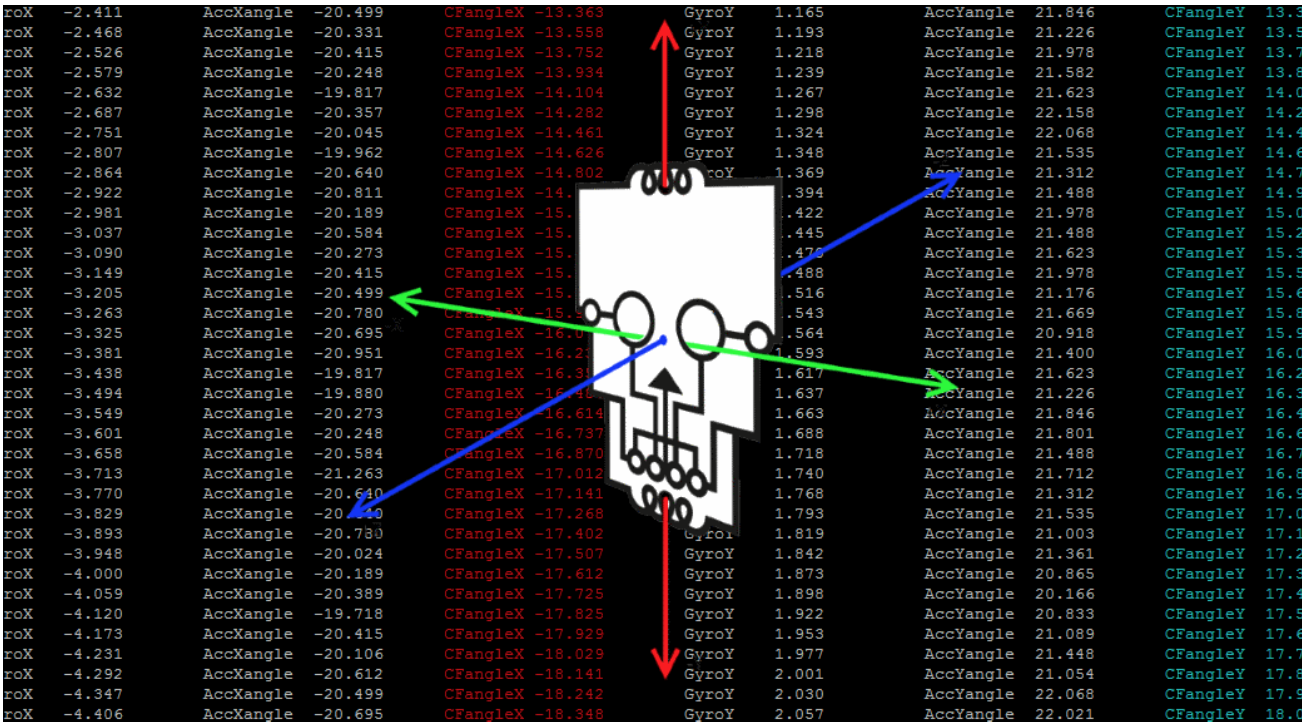
ozzmaker.com



Guide to interfacing a Gyro and Accelerometer with a Raspberry Pi

December 11, 2014 | Mark Williams | 50 Comments

This guide covers how to use an Inertial Measurement Unit (IMU) with a Raspberry Pi . This is an updated guide and improves on the old one found here.

In this guide I will explain how to get readings from the IMU and convert these raw readings into usable angles. I will also show how to read some of the information in the datasheets for these devices.

This guide focuses on the BerryIMU. However, the theory and principals below can be applied to any digital IMU, just some minor modifications need to be made. Eg Pololu MinIMU, Adafruit IMU and Sparkfun IMUs

Git repository here
The code can be pulled down to your Raspberry Pi with;

```
pi@raspberrypi ~ $ git clone http://github.com/mwilliams03/BerryIMU.git
```

A note about Gyros and Accelerometers

When using the IMU to calculate angles, readings from both the gyro and accelerometer are needed which are then combined. This is because using either on their own will result in inaccurate readings. And a special note about yaw.

Here is why;
Gyros – A gyro measures the rate of rotation, which has to be tracked over time to calculate the current angle. This tracking causes the gyro to drift. However, gyros are good at measuring quick sharp movements.

Accelerometers – Accelerometers are used to sense both static (e.g. gravity) and dynamic (e.g. sudden starts/stops) acceleration. They don't need to be tracked like a gyro and can measure the current angle at any given time. Accelerometers however are very noisy and are only useful for tracking angles over a long period of time.

Accelerometers cannot measure yaw. To explain it simply, yaw is when the accelerometer is on a flat level surface and it is rotated clockwise or anticlockwise. As the Z-Axis readings will not change, we cannot measure yaw. A gyro and a magnetometer can help you measure yaw. This will be covered in a future guide.

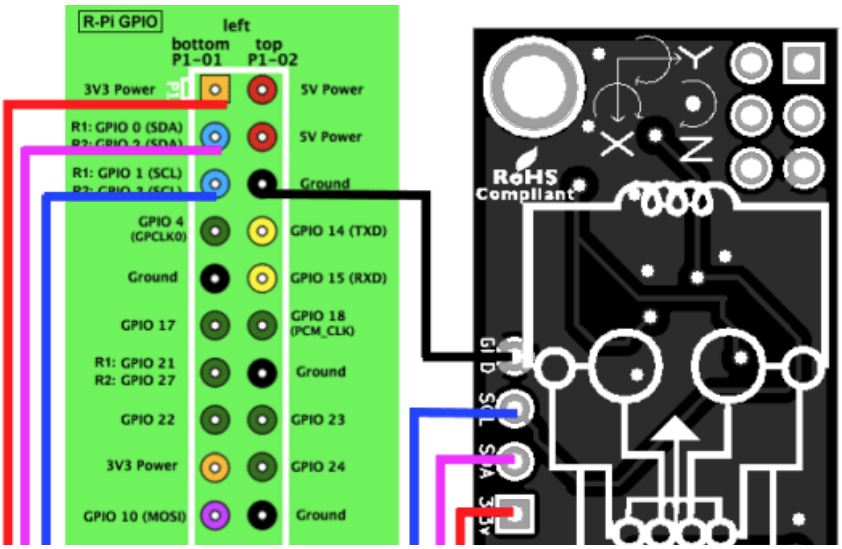
Here is an excellent tutorial about accelerometers and gyros.

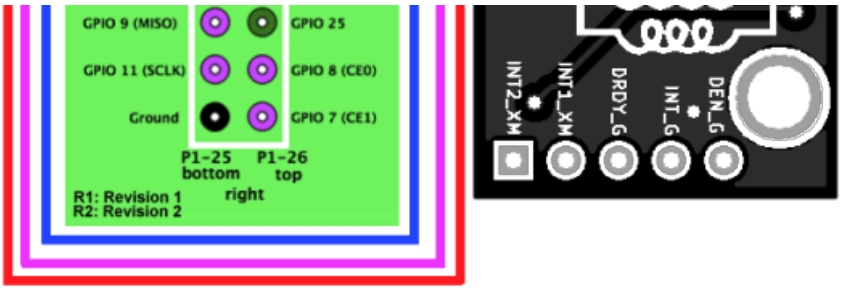
Setting up the IMU and I2C

The IMU used for this guid a BerryIMU which uses a LSM9DS0, which consists of a 3-axis gyroscope, a 3-axis accelerometer and a 3-axis magnetometer.
The datasheet is needed if you want to use this device;LSM9DS0

This IMU communicates via the I2C interface.

The image below shows how to connect the BerryIMU to a Raspberry Pi





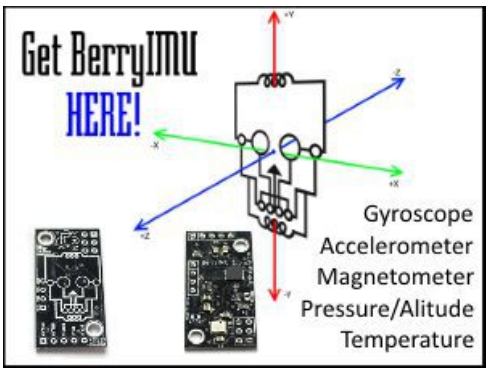
Once connected, you need to enable I2C on your Raspberry Pi and install the I2C-tools. This guide will show you how.
Once you have i2c enabled, you should see the addresses of the components on the IMU when using i2cdetect;

```
pi@raspberrypi ~ $ sudo /usr/sbin/i2cdetect -y 1
0 1 2 3 4 5 6 7 8 9 a b c d e f
00: – – – – – – – – – – – – –
10: – – – – – – – – – – – – – – 1e –
20: – – – – – – – – – – – – – – – –
30: – – – – – – – – – – – – – – – –
40: – – – – – – – – – – – – – – – –
50: – – – – – – – – – – – – – – – –
60: – – – – – – – – – – 6a – – – –
70: – – – – – – – 77
```

The gyro on the LSM9DS0 is using the address of 0x6a and the accelerometer and magnetometer are using 0x1e, this can be verified with page 33 on the datasheet.

Quoted from the datasheet with the address highlighted;
"The SDO/SA0 pins (SDO_XM/SA0_XM or SDO_G/SA0_G) can be used to modify the least significant bit of the device address. If the SA0 pin is connected to the voltage supply, LSb is '1' (ex. address 0011101b) else if SA0 pad is connected to ground, the LSb value is '0' (ex. address 0011110b)."

We will not be covering the magnetometer in this post.



Reading and Writing to the Accelerometer and Gyroscope

We read and write values from different register in the above addresses to do different things. Eg Turn the Gyro on, set the sensitivity level, read accelerometer values. etc..
The git repository includes a header file for the LSM9DS0 which lists all the registers used by this sensor.

We use ioctl calls to read and write information to the I2C bus.

Open I2C Bus

First, you need to open the I2C bus device file, as follows;

```
1  char filename[20];
2  sprintf(filename, "/dev/i2c-%d", 1);
3  file = open(filename, O_RDWR);
4  if (file<0) {
5          printf("Unable to open I2C bus!");
6          exit(1);
7  }
```

As we are using sprintf, you will need to include the header file stdio.h

Selecting the Gyro or Accelerometer
Before we can start writing and reading values from the sensors, we need, we need to select the address of the sensor we want to write to.

```
1  if (ioctl(file, I2C_SLAVE, <em>device_address</em>) < 0) {
2          printf("Error: Could not select magnetometer\n");
3  }
```

device_address is the address of the sensor you want to select. E.g. 0x6a for the LSM9DS0.
I2C_SLAVE is defined in i2c-dev.h

You will need to include the file control headers in your program, fcntl.h.
You will also need to include the headers for the i2c bus – linux/i2c-dev.h

If we only had one sensor, we would only need to do this once at the start of the program. As we have two, we need to select each of the different sensors just before we want to read or write a value.

We should write a function to do this;

```
1  void selectDevice(int file, int addr)
2  {
3   char device[3];
4
5   if (ioctl(file, I2C_SLAVE, addr) < 0) {
6   printf("Failed to select I2C device.");
7   }
8  }
```

When we call the function, we will pass the file pointer and the address of the sensor we want to select. Like this;

```
1  selectDevice(file,ACC_ADDRESS);
```

Enable the Accelerometer

```
// Enable accelerometer.
        writeAccReg(CTRL_REG1_XM, 0b01100111); //  z,y,x axis enabled, continu
        writeAccReg(CTRL_REG2_XM, 0b00100000); // +/- 16G full scale
```

To enable to accelerometer, we need to write a byte to the CTRL_REG1_XM(0x20) register on the LSM9DS0. This can be see on page 55 of the datasheet, where you will also find other values that can be used.

After this, we then need to write a byte to the CTRL_REG2_XM(0x21) register to set our sensitivity. This information is on page 56 of the datasheet.

Enable the Gyro

Enabling the gyro is very similar to the accelerometer.

```
// Enable Gyro
        writeGyrReg(CTRL_REG1_G, 0b00001111); // Normal power mode, all axes e
        writeGyrReg(CTRL_REG4_G, 0b00110000); // Continuos update, 2000 dps fu
```

The values for CTRL_REG1_G and CTRL_REG4_G can be found on page 41 and 44 of the datasheet.

Reading the Values from the Accelerometer and Gyro

Once you enable the accelerometer and gyro, you can start reading the raw values from these sensors.
Reading the accelerometer

```
void readACC(int  *a)
{
        uint8_t block[6];
        selectDevice(file,ACC_ADDRESS);
        readBlock(0x80 | OUT_X_L_A, sizeof(block), block);

        *a = (int16_t)(block[0] | block[1] << 8);
        *(a+1) = (int16_t)(block[2] | block[3] << 8);
        *(a+2) = (int16_t)(block[4] | block[5] << 8);
}
```

The above function will read the raw values from the accelerometer, here is a description of what is happening;

1. An array of 6 bytes is first created to store the values.
2. The I2C slave address is selected for the accelerometer, by passing the accelerometer address of ACC_ADDRESS or 0x1E to the selectDevice() function.
3. Using the readBlock() function from i2c-dev.h, we read 6 bytes starting at OUT_X_L_A (0x28). This is shown on

page 61 of the datasheet.

4. The values are expressed in 2's complement (MSB for the sign and then 15 bits for the value) so we need to combine;
   block[0] & block[1] for X axis
   block[2] & block[3] for Y axis
   block[4] & block[5] for Z axis

Reading the gyro

```
void readGYR(int *g)
{
    uint8_t block[6];

        selectDevice(file,GYR_ADDRESS);

    readBlock(0x80 | OUT_X_L_G, sizeof(block), block);

        *g = (int16_t)(block[0] | block[1] << 8);
        *(g+1) = (int16_t)(block[2] | block[3] << 8);
        *(g+2) = (int16_t)(block[4] | block[5] << 8)
}
```

The above function reads the values from the gyro. It is very similar to the readACC() function.
Once we have the raw values, we can start converting them into meaningful angle values.

Convert the Raw Values to Usable Angles

Gyro

```
//Convert Gyro raw to degrees per second
rate_gyr_x = (float) gyrRaw[0] * G_GAIN;
rate_gyr_y = (float) gyrRaw[1]  * G_GAIN;
rate_gyr_z = (float) gyrRaw[2]  * G_GAIN;
```

For the gyro, we need to work out how fast it is rotating in degrees per second(dps). This is easily done by multiplying the raw value by the sensitivity level that was used when enabling the gyro. G_GAIN = 0.07, which is based off a sensitivity level of 2000 dps. Looking at the table on page 13 of the datasheet, we can see that we need to multiple it by 0.07 to get degrees per second. The table shows it as 70 mdps (milli degrees per second). Another example: Looking at the table, if we chose a sensitivity level of 250dps when enabling the gyro, then we would have to multiply the raw values by 0.00875.

We now need to track this gyro overtime.

```
//Calculate the angles from the gyro
```

```
2 gyroXangle+=rate_gyr_x*DT;
3 gyroYangle+=rate_gyr_y*DT;
4 gyroZangle+=rate_gyr_z*DT;
```

DT = loop period. I use a loop period of 20ms. so DT = 0.02.
gyroXangle = is the current X angle calculated from the gyro X data.
I have DT set to 20ms. This is how long it takes to complete one cycle of the main loop. This loop period has to be constant and accurate, otherwise your gyro will drift more then it should.

Accelerometer

```
1 //Convert Accelerometer values to degrees
2
3 AccXangle = (float) (atan2(accRaw[1],accRaw[2])+M_PI)*RAD_TO_DEG;
4 AccYangle = (float) (atan2(accRaw[2],accRaw[0])+M_PI)*RAD_TO_DEG;
```

The angles can be calculated by using trigonometry and the raw values from the other accelerometer axis. This is done using the Atan2 function to return the principal value of the tangent of the other angles, expressed in radians. We add π to the radians so that we get a result between 0 and 2. We then convert the radians to degrees by multiplying the radians by 57.29578 (180/π).
M_PI = 3.14159265358979323846
RAD_TO_DEG = 57.29578 , 1 radian = 57.29578 degrees

You will need to include the math header file in your program, math.h and when compiling you will also need to link the math library. '-lm'

Combining Angles from the Accelerometer and Gyro

Once you have the both the angles from both sensors, you will have to combined them to overcome the gyro drift and the accelerometer noise.
We can do this by using a filter which will trust the gyro for short periods of time and the accelerometer for longer periods of time.

There are two filers you could use, the Kalman Filter or the Complementary Filter. We will used the Complementary filter as it is simple to understand and less CPU intensive. The Kalman filter is way to far complex and would be very processor intensive.

Complementary filter;

Current angle = 98% x (current angle + gyro rotation rate) + (2% * Accelerometer angle)

or

```
1 CFangleX=AA*(CFangleX+rate_gyr_x*DT) +(1 - AA) * AccXangle;
2 CFangleY=AA*(CFangleY+rate_gyr_y*DT) +(1 - AA) * AccYangle;
```

CFangleX & CFangleY are the final angles to use.

Timing

The gyro tracking will not be accurate if you can not run a well timed loop.

The code includes the mymillis() function for timing.

At the start of each loop, we get the current time;

```
startInt = mymillis();
```

then at the end of the loop, we make sure that at least 20ms has passed before continuing;

```
//Each loop should be at least 20ms.
while(mymillis() - startInt < 20)
{
        usleep(100);
}
```

If your loop takes longer than 20ms to run, 45ms for example. Jut update the above value to make sure that every loop cycle runs at the same speed and set this value in DT for the gyro tracking.

Convert readings -/+ 180 Degrees

```
AccXangle -= (float)180.0;
if (AccYangle > 90)
      AccYangle -= (float)270;
else
       AccYangle += (float)90;
```

The above code will convert the values for the X and Y axis on the accelerometer so that the value is 0 when the IMU is upright. This isn't needed, however I like all axis to be 0 when the device is upright.

Display the values in the Console

The code below will print the readings to the console in with color coding to make the values easier to read.

```
printf ("GyroX  %7.3f \t AccXangle \e[m %7.3f \t \033[22;31mCFangleX %7.3f\03
```

Compile the Code

```
pi@raspberrypi ~ $ gcc -o gyro_accelerometer_tutorial01 -l rt gyro_accelerometer_tutorial01.c -lm
```

Run the Program

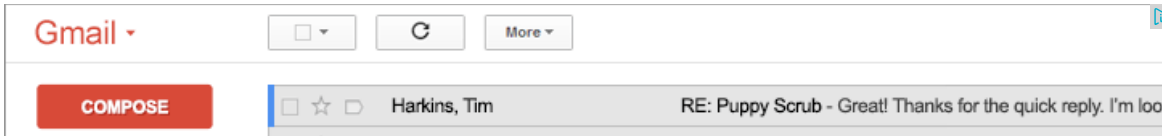
```
pi@raspberrypi ~ $ sudo ./gyro_accelerometer_tutorial01
```

Guides and Tutorials

In this order;

Guide to interfacing a Gyro and Accelerometer with a Raspberry Pi
Guide to interfacing a Gyro and Accelerometer with a Raspberry Pi – Kalman Filter
Create a Digital Compass with the Raspberry Pi – Part 1 – "The Basics"
Create a Digital Compass with the Raspberry Pi – Part 2 – "Tilt Compensation"
Create a Digital Compass with the Raspberry Pi – Part 3 – "Calibration"

These cover both BerryIMU and displaying graphics with SDL;
How to Create an Inclinometer using a Raspberry Pi and an IMU



SHARE THIS:

Twitter Facebook 3

ACCELEROMETER BERRYIMU FEATURED1 GYROSCOPE

50 THOUGHTS ON "GUIDE TO INTERFACING A GYRO AND ACCELEROMETER WITH A RASPBERRY PI"

Pingback: Create an Inclinometer Using a Raspberry Pi | Hackaday

ru4mj12 (@ru4mj12)
DECEMBER 24, 2014 AT 5:27 PM

Hi,

Regarding the filtering,
Current angle = 98% x (current angle + gyro rotation rate) + (2% * Accelerometer angle)

It looks like the current angle is weighed heavily by the gyro reading and weighs the accelerometer reading very little.. but how does this in principle filter the drift of the gyro?

★ Mark Williams
DECEMBER 24, 2014 AT 5:58 PM

This works as the gyro only reports rate of turn. If the IMU was not moving and it is at a 90 degress angle, the

gryo would report '0' (Zero) and the accelerometer would report 90 degress.

Paul
JANUARY 17, 2015 AT 4:23 PM

awesome project. I bought some parts from adafruit. Seems its using the same gyro as yours here, but when I run gyro_accelerometer_tutorial01 , I get

Failed to write byte to I2C Gyr

this is writing the 0x20 register in the enableIMU() function. (I added some debug)

I2cdetect shows valid devices, and this code – https://github.com/mpolaczyk/Gyro-L3GD20-Python/blob/master/L3GD20.py seems to work.
Any hints you can give me on whats going on would be really appreciated, as Im looking forward to having the off road inclinometer in my truck

Thanks, Paul

Paul
JANUARY 17, 2015 AT 4:40 PM

well, that was easy to fix.
#define GYR_ADDRESS 0x6B

was needed for my device.

AJ
SEPTEMBER 7, 2015 AT 10:50 AM

I am getting the same error as Paul:
Failed to write byte to I2C Gyr

Yet Paul says his fix was to add:
#define GYR_ADDRESS 0x6B

Where does he add this line?
Do I add it to "gyro_accelerometer_tutorial01.c", then compile?

★ Mark Williams
SEPTEMBER 7, 2015 AT 11:37 AM

No… change it in LSM9DS0.h then compile

Nick Twigg

MARCH 24, 2015 AT 7:39 AM

Have or are you guys working on a Python implementation of this?

★ Mark Williams
MARCH 25, 2015 AT 3:13 PM

We have just released some python code. See here for more details: http://ozzmaker.com/2015/03/25/python-code-for-berryimu-accelerometer-gyroscope-magnetometer-pressure-sensor/

Steven
APRIL 1, 2015 AT 7:34 AM

Hi,

Great tutorial, I am fairly new to using i2c and to programming in general so this site has helped me out a lot. I cloned the Berry IMU library and am now using the gyro_accelermoter_turtorial03_Kalman_filter directory.

I'm having some problems getting the Gyro and acclerometer to work in tandem though.
I got the gyro working, I'm using the same one you are, an L3GD20H, but can't get my accemerometer module, a sparkfun ADXL345, to work. I keep getting the message

"failed to write byte to 12C Acc"

I think this is because the ADXL345 uses an address of 0x53 which is not the same as the LSM9DS0 's accelerometer address 0x1E.

To try to solve this I went into the LSM9DS0.h and changed the ACC_ADDRESS to 0x53 but got the same error message. I also tried messing around with the accelerometer enable portion of the post but didn't get anywhere.

Do you have any ideas?

★ Mark Williams
APRIL 1, 2015 AT 6:56 PM

you can check the i2c addresses of connected devices by using `sudo i2cdetect -y 1`
what addresses do you have?

Steven
APRIL 2, 2015 AT 4:44 AM

It turns out that I connected the CS pin on the ADXL low instead of high and forgot to recheck the i2c addresses to make sure that the device was enabled. It works now, thanks!

Tyler
APRIL 15, 2015 AT 6:30 AM

Great tutorial this works great! I was wondering if you knew whether or not configuring I2C will affect the GPIO pins? I configured I2C and implemented this code and I also placed some SPI. However, after doing this, a simple code that should turn on my GPIO pins no longer worked. Have you heard of anything like this?

Pingback: Examine different accelerometer and gyroscopes to work with the raspberry pi | RoadStar

Peter
APRIL 21, 2015 AT 8:05 PM

Hi,

Thanks for this tutorial, it really helped me getting everything set up and running.

I have one question:
why when using i2c_smbus_read_block_data you send command (0x80 | OUT_X_L_A) instead of just register address (OUT_X_L_A)?

I know it won't work with just address (I checked), but I'd like to know why.

★ Mark Williams
APRIL 22, 2015 AT 5:22 AM

This is to do with the seven bit nature of i2c
Have a look at '6.1.1 i2c operations' in the datasheet, this is explained well;
http://ozzmaker.com/wp-content/uploads/2014/12/LSM9DS0.pdf

Peter
APRIL 22, 2015 AT 9:05 PM

Thank you very much, somehow I didn't notice this when reading through datasheet. Now everything is clear.

madeinoz
APRIL 26, 2015 AT 2:08 PM

Thanks for the tutorial, clearly explained how to fuse the sensor data in a simple way!

Pandrei
APRIL 28, 2015 AT 1:37 AM

A smoothing filter probably should be added to the gyro output and remove the initial gyro offert from the output. In the current code, when still the gyro output is not necessarily 0, which in turn adds a offset to the resulting angle.

Daniel
MAY 10, 2015 AT 9:20 AM

Amazing explanation, Great tutorial this works grea with an sparkfun imu LSM9DS0 6dof. Do you have an example how can I write the output data in a SD card? thanks a lot

★ Mark Williams
MAY 10, 2015 AT 9:34 AM

just redirect the output to a file
```
sudo ./gyro_accelerometer_tutorial01 > textfile.csv
```

★ Mark Williams
MAY 16, 2015 AT 3:53 AM

The code is C and Python

★ Mark Williams
MAY 17, 2015 AT 11:32 AM

you need to run it with sudo
```
sudo ./gyro_accelerometer_tutorial01
```

cameron
JUNE 4, 2015 AT 10:40 AM

So my ACCX and ACCY angles aren't updating when I run the example berry_imu python code. I would prefer to not have to scrap all of my motor controller and motion detection python code and rewrite it all in C.

★ Mark Williams
JUNE 4, 2015 AT 11:19 AM

Does it work with the example C code?

Siddharth Swarnkar
JUNE 7, 2015 AT 3:14 AM

Can someone please what is " ioctl(file,I2C_SLAVE,addr) " acctually doing.

Is it putting I2C_slave=addr or what?
because here

http://www.cs.fsu.edu/~baker/devices/lxr/http/source/linux/include/linux/i2c-dev.h?v=2.6.25#L41

I2C_SLAVE is defined as 0x0703, but that need not be the address.

Christian
JULY 8, 2015 AT 5:26 AM

So when you get to the Open I2C Bus part where are you entering this code into? I am really lost when I get to this part I am not very fimilar with Raspberry Pi but i need to get it to give me angles thanks!

★ Mark Williams
JULY 8, 2015 AT 12:26 PM

This article explains the code at http://github.com/mwilliams03/BerryIMU.git
If you just want the angles, pull down the code, compile and then run

Christian
JULY 9, 2015 AT 5:51 AM

Thanks! That really helped! I keep getting an i2c error that says: unable to open i2c bus (followed the link for setting up the i2c)

Is the i2c needed for the C and pythons program? When I used the python program i got an error stating:

File “berryIMU.py”, line 126, in
writeACC(CTRL_REG1_XM, 0b01100111)#z,y,x axis enabled, continuous update, 100 Hz data rate
File “berryIMU,py”, line 38, in writeACC
bus.write_byte_data(ACC_ADDRESS , register, value)
IOError: [Errno 5] Input/output error

Sorry for all the comments i am not very good with the Rasberry Pi!

★ Mark Williams
JULY 9, 2015 AT 12:48 PM

Do you see the IMU?
What do you get when you do `sudo i2cdetect -y 1`

Christian
JULY 10, 2015 AT 1:48 AM

No I do not see the IMU when I do sudo i2cdetect -y 1, what should i do from here?

Another side question, could this be used with an Arduino since it has c code right?
Thanks!!!

★ Mark Williams
JULY 10, 2015 AT 12:05 PM

I would double check your connections. Are you using the jumper cables or have you soldered on the female header and have it connected to the GPIO pins?
It can definitely be used on an Arduino. However the code would have to be changed to support Arduino specific functions, like i2c read and write. The math will work

Robert
JULY 9, 2015 AT 6:42 AM

In your demo video a graphical BerryIMU was displayed on a screen, rotating relative to its real-time position. How could I go about setting up a graphical representation of the acc/gyro components?

★ Mark Williams
JULY 9, 2015 AT 12:34 PM

that was done using David's code;
https://github.com/DavidEGrayson/minimu9-ahrs

Robert
JULY 10, 2015 AT 6:27 AM

Regarding the minimu9-ahrs visualization, is there a way to shrink the size and dimensions of the displayed 3d image? Is there some sort of configuration file where that can be modified?

★ Mark Williams
JULY 10, 2015 AT 12:12 PM

This is the actual code that does the displaying, you need to pipe the output of the above code into this one;
https://github.com/DavidEGrayson/ahrs-visualizer.
Have a look in asset folder in the above link. It contains the image files you can maybe re-size.

gilles
JULY 15, 2015 AT 6:54 AM

Hi,

Really nice tutorial,
Is there is any c++ I2C lib or a way to compile the sensor.c in a c++ projet, I have some error with like O_RDWR missing.

cheers.

Robert
JULY 21, 2015 AT 2:56 AM

I had to link libstdc++ in the .c file, if that helps..

AJ
SEPTEMBER 8, 2015 AT 11:01 AM

Hello Mark,
Thank you for taking the time to create this, write it up, and answer questions!

With your help (in the above comments), I was able to modify LSM9DS0.h with the correct address in my i2c bus.
I used i2cdetect -y 1 and I saw that it was at 6b, so I entered "0x6B".
The other values are believe are right, as they are the 0x1E.

My problem is that I have zero on the AccXangle, and a static -90.000 in the AccYangle.

In addition to that, the values keep climbing for ever.
For example, the values of GyroX keep climbing (they suprassed 370.000, and I Ctrl+C then).

Do I have something wrongly configured?
Or is something amiss?

Thank you very much for your help!!

Daniel
SEPTEMBER 27, 2015 AT 8:16 AM

Hi. Is the same result running the script from sudo ./gyro_accelerometer_tutorial01 than from the python in BerryIMU/python-LSM9DS0-gryo-accel-compass $ berryIMU.py

And another question is
When I watch results from the sudo ./gyro_accelerometer_tutorial01 in screen everthing is ok, but when I ouput to a *.csv or *.txt file,, some errors appears in between the data

Example:

"Loop Time 20 GyroX -113.579 AccXangle [m -179.498 [22;31mCFangleX -173.466[0m GyroY -219.676 AccYangle 178.223 [22;36mCFangleY 177.599 [0m"

any suggestions?

thanks in advace

mwilliams03
SEPTEMBER 27, 2015 AT 4:26 PM

The data should be the same for both the python and the C code.
The “errors” you are seeing is the ASCII escapes sequences use to output the data in different colours on the terminal.
Replace the printf statement in gyro_accelerometer_tutorial01.c with this;
printf (” GyroX %7.3f \t AccXangle %7.3f \t CFangleX %7.3f \t GyroY %7.3f \t AccYangle %7.3f \t CFangleY %7.3f\t\n”,gyroXangle,AccXangle,CFangleX,gyroYangle,AccYangle,CFangleY);

Daniel
SEPTEMBER 27, 2015 AT 11:09 PM

Thank you. tutorial01, 02, y 03 are working fine without ascii escapes due to colours in terminal. But, when I tried to run berryIMU.py it gives me the following

pi@raspberrypi ~/BerryIMU/python-LSM9DS0-gryo-accel-compass $ sudo python berryIMU.py
Traceback (most recent call last):
File “berryIMU.py”, line 126, in
writeACC(CTRL_REG1_XM, 0b01100111) #z,y,x axis enabled, continuos update, 100Hz data rate
File “berryIMU.py”, line 38, in writeACC
bus.write_byte_data(ACC_ADDRESS , register, value)
IOError: [Errno 5] Input/output error

any suggestion?

thanks again

★ Mark Williams
SEPTEMBER 27, 2015 AT 11:17 PM

Are you using a BerryIMU?
What is the output of `sudo i2cdetect -y 1`?

Daniel
SEPTEMBER 28, 2015 AT 1:46 AM

No. I am using the LSM9DS0
output of sudo i2detect -y 1 is

0 1 2 3 4 5 6 7 8 9 a b c d e f
00: – – – – – – – – – – – – – – – –
10: – – – – – – – – – – – – – – 1d – –
20: – – – – – – – – – – – – – – – –
30: – – – – – – – – – – – – – – – –
40: – – – – – – – – – – – – – – – –

50: – – – – – – – – – – – – – – – –
60: – – – – – – – – – – – – 6b – – – –
70: – – – – – – – –

★ Mark Williams
SEPTEMBER 28, 2015 AT 11:08 AM

edit LSM9DS0.py and change the address values to match your device.

Daniel
SEPTEMBER 30, 2015 AT 1:40 AM

it works now perfect ,using both c and or py. Thanks a lot!!

Daniel
OCTOBER 2, 2015 AT 10:25 PM

Is it possible to change the output rate of data that we can see in screen or and of csv? I mean, I want to calculate then average speeds, maximum angle of inclination, etc etc, but the number of rows in the current outputs is huge. Thanks again

★ Mark Williams
OCTOBER 3, 2015 AT 9:00 AM

yes, that is easy. Just remove the unneeded values from the print statement;
in C:

```
printf (" GyroX %7.3f \t AccXangle \e[m %7.3f \t \033[22;31mCFangleX
%7.3f\033[0m\t GyroY %7.3f \t AccYangle %7.3f \t \033[22;36mCFangleY %7.3f\t
\033[0m\n",gyroXangle,AccXangle,CFangleX,gyroYangle,AccYangle,CFangleY);
```

Python;

```
print ("\033[1;34;40mACCX Angle %5.2f ACCY Angle %5.2f\033[1;31;40m\tGRYX Angle
%5.2f GYRY Angle %5.2f GYRZ Angle %5.2f \033[1;35;40m \tCFangleX Angle %5.2f
\033[1;36;40m CFangleY Angle %5.2f \33[1;32;40m HEADING %5.2f \33[1;37;40m
tiltCompensatedHeading %5.2f\033[0m " % (AccXangle,
AccYangle,gyroXangle,gyroYangle,gyroZangle,CFangleX,CFangleY,heading,tiltCompensat
edHeading))
```

Vinicius Oliveira
OCTOBER 24, 2015 AT 4:03 AM

How did you find the main loop duration period (DT) ??

★ Mark Williams

OCTOBER 24, 2015 AT 4:19 PM

DT can be anything really… it only has to be constant as this value is used later in the code to work out the angle of the gyro