The L3G4200D 3-Axis Gyro

Table of Contents

Our sensor, a L3G4200D 3-Axis Gyro with voltage regulator carrier/breakout board (from Pololu) includes a high-precision ST L3G4200D 3-axis gyroscope. The L3G4200D sensor measures the angular rates of rotation (velocity) about the pitch (x), roll (y), and yaw (z) axes with a configurable sensitivity of ±250°/s, ±500°/s, or ±2000°/s. We will only be using the z-axis sensor.

The L3G4200 communicates with our microcontroller over an I2C serial interface. The board includes a 3.3 V linear regulator and integrated level-shifters that allows it to work over an input voltage range of 2.5–5.5 V. The IMU board uses a standard .1" header allowing it to plug into our solderless breadboard.

1.1 Terminology

Sensitivity

An angular rate gyroscope is a device that produces a positive-going digital output for counterclockwise rotation around the sensitive axis considered. Sensitivity describes the gain of the sensor and can be determined by applying a defined angular velocity to it. This value changes very little over temperature and time. Note: *Our gyros are mounted upside down on the breadboard so a clockwise rotation generates a positive output.*

Zero-rate level

Zero-rate level describes the actual output signal if there is no angular rate present. The zero-rate level of precise MEMS (Microelectromechanical system) sensors is, to some extent, a result of stress to the sensor and, therefore, the zero-rate level can slightly change after mounting the sensor onto a printed circuit board or after exposing it to extensive mechanical stress. This value changes very little over temperature and time.

Stability over temperature and time

The single driving mass design of the ST gyroscopes matches the MEMS mechanical mass and the ASIC interface, delivers a high level of stability over temperature and time.

2 Reading and Downloads

1. L3G4200D MEMS motion sensor: ultra-stable three-axis digital output gyroscope http://

and Kalman Filters (optional): http://tom.pycke.be/mav/71/kalman-filtering-of-imu-data

3 Lab Checklist

Here is your checklist for this lab.

- ☐ Learn how to connect and sample Gyro data
- ☐ Measure Gyro noise and drift at rest
- ☐ Compute angle as the Gyro is rotated
- ☐ Explore data filtering techniques
- ☐ Rover travels in a straight line using a closed-loop control algorithm. (Part 2)

4 How it Works

The purpose of a gyro is to maintain a sense of direction. A gyro does this by sensing changes in it's angular motion (angular velocity), and outputting a digital number that is proportional to its rotating speed. This angular velocity can then be integrated over time to give you the offset, in degrees, of the gyro from it's zeroed position. In other words, it will tell you the net rotation the gyro is experiencing over time.

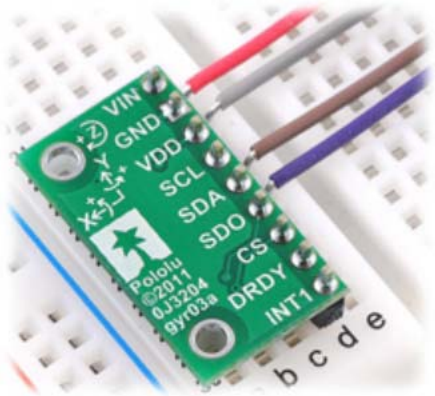
Our gyro can measure three axes referred to as roll (x axis), pitch (y axis), and yaw (z axis). For our rover application, we are only concerned with the yaw (z axis). In terms of the L3G4200D, angular velocity is measured in degrees per second. Our gyro returns a positive number for clockwise rotation, and a negative number for counterclockwise (note: L3G4200D MIMS chip is inverted)

A parameter of interest for a gyro is the sensitivity, which is found in the sensor's data sheet. It is the parameter linking your DN to your angular velocity (degrees/s).

EQ 1-1

Our L3G4200D gyro has three sensitivity settings ±250°/s, ±500°/s, or ±2000°/s. Our rover does not change direction very quickly so the lower the sensitivity the better.

5 Wire the Gyro



Install the breakout in the breadboard as shown in the figure above (chip inverted). When the sensor is installed upside down, it will give a positive value for clockwise rotation and a negative value for counterclockwise rotation about the z-axis.
Make the following connections with wires between the Arduino and the L3G4200D:

Arduino Uno/Duemilanove		L3G4200D Carrier
5V	⇨	VIN
GND	⇨	GND
Analog Pin 5	⇨	SCL
Analog Pin 4	⇨	SDA

6 Set up the Gyro for MATLAB

by Nic Flenghi

6.1 Step 1: Download Files

Download and Extract the files in the `Gyro_matlab_code.zip` files included in this folder. The extracted folder should include the following files.

1. `srv_gyro.pde`
2. `install_arduino.m`
3. `arduino.m`
4. `gyro_test.m`

Note: While some of the files names in the files downloaded have the same names as those found at Mathworks; Nic has modified them for our class.

6.2 Step 2: Upload the `srv_gyro.pde` Server

of commands can be found at the top of the `srv_gyro.pde` file.

Update by iBanana Spring '12
This particular step although short and simple looking, caused a lot of grief because when the file `srv_gyro.pde` was run, it called upon `L3G4200D.h`, and an error was displayed saying that the file `Arduino.h` did not exist. Hours of troubleshooting were spent trying to fix this error. After extensive research it was noted that the file `L3G4200D.h` (found in https://github.com/pololu/L3G4200D) is meant to be used within the Arduino IDE 1.0. So we saw that most likely this file was tweaked in order to work with the latest version of IDE. Now we had two choices (1) Install and work with Arduino IDE 1.0 or (2) use the older `L3G4200D.h` that worked with Arduino IDE 0022 provided in the Lab folder (see L3G4200D Read Me file). After we pasted this version of the file under Documents/Arduino, srv_gyro.pde finally uploaded.

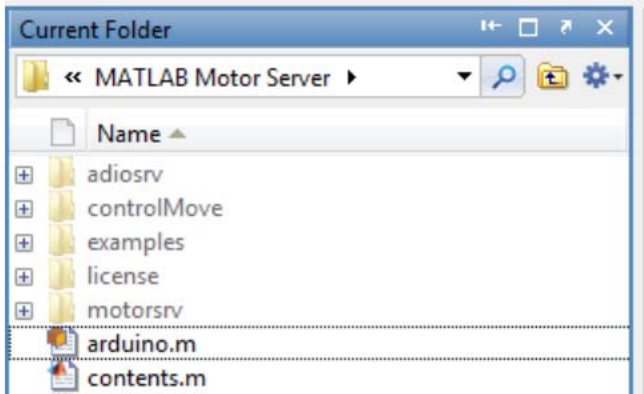
6.3 Step 3: Add `arduino.m` to MATLAB Path

Add the modified `arduino.m` file to your MATLAB path. You may do this manually, or by running `install_arduino.m`.

The `arduino.m` file takes care of sending and receiving commands and data from the Arduino server (`srv_gyro.pde`). It also gives us a collection of MATLAB functions to use instead of manually entering the alphanumeric codes. For example, `a.gyroRead('z')` sends the code `'Gz'` over the serial interface and listens for the value sent back from the board.

6.4 Step 4: Instantiate the MATLAB Arduino Class

Note: The MATLAB Motor Server folder must be open when performing commands such as instantiating the Arduino (Figure 7).



```
a = arduino('COMX');
```

Where `COMX` is the port the Arduino is connected to (this can be determined using the Arduino IDE or Device Manager).

6.5 Step 5: Run `gyro_test.m`

To read the z-axis of the gyro you simply need to enter:

```
a.gyroRead('z'); % current value of z, you can also use 'x' or 'y'
```

The count of the gyro is displayed ranging from +(2^{15}- 1) to -(2^{15}).

Run `gyro_test.m` to duplicate the functionality of the Arduino IDE example code (see Appendix A) and test your gyro is working properly.

6.6 Step 6: Modify `srv_gyro.pde` to Change Sensitivity

Unlike the Pololu library upon which this lab was developed, the Sparfun Library for its L3G4200D breakout board allows you to change the sensitivity of the gyro when it is instantiated. In this step you will implement a less elegent solution by manually modify our srv_gyro.pde code to set the sensitivity.

Add the following code in the **`setup()`** section of the `srv_gyro.pde` code within the Arduino IDE. The sensitivity appears to default to 250, and is changed by writing to bits 4 and 5 in `L3G4200D_CTRL_REG4`. Read the STMicroelectronics L3G4200D data sheet for all the details. Since our rover moves slowly, we want keep the gyro at its default value - the lowest sensitivity (250 deg/s).

```
void setup() {
  /* initialize serial                                      */
  Serial.begin(115200);
  Wire.begin();
  gyro.enableDefault();
// set gyro sensitivity (250, 500, or 2000 deg/s) uncomment the
// appropriate command.
// 250 deg/s
    gyro.writeReg(L3G4200D_CTRL_REG4, 0b00000000);
// 500 deg/s
//  gyro.writeReg(L3G4200D_CTRL_REG4, 0b00010000);
// 2000 deg/s
//  gyro.writeReg(L3G4200D_CTRL_REG4, 0b00110000);
```

value tends to oscillate sporadically about zero. As we integrate angular velocity (deg/sec) to measure the angle (deg) any bias in this noise (plus or minus) can accumulate and lead to the gyro angle drifting while it's sitting still - *the zero-rate level*. Later in the lab we will filter out most of this by setting a threshold value. But first, lets run some experiments in MATLAB to quantify both the noise and zero-rate level of our gyro.

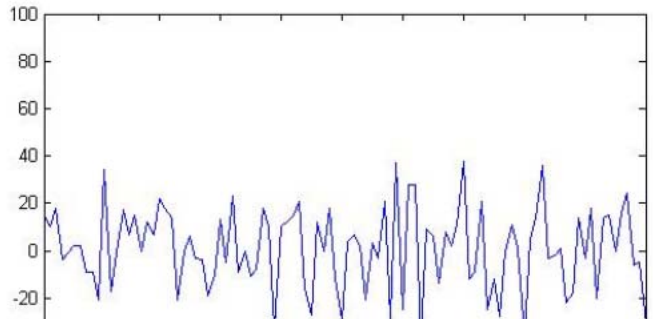
7.1 Gyro Noise at Rest Experiment

Place your rover on a solid table and be sure not to bump it during the test. Then, run the following code and observe the output from your sensor.

```
%% Gyro Noise at Rest
dc_offset = 0;
z = zeros(1,100);       % setup an array for 100 samples
x = 1:100;              % x-axis for plot
figure(1)               % bring plot window to front
for n = 1:100
  z(n) = a.gyroRead('z') - dc_offset;     % sample gyro z-axis value
  fprintf('Z: %6d, \n', z(n)) % print value to command window

  % live plot of data
  plot(x,z)
  axis([1 100 -100 100])
  drawnow                    % forces MATLAB to redraw figures
end

% Calculate max, min, and mean values.
fprintf('Max value: %d\n',max(z))
fprintf('Min value: %d\n',min(z))
fprintf('Mean value: %f\n',mean(z))
```



```
Max value: 38
Min value: -37
Mean value: 1.150000
```

Your results should be similar to those above. Though the plot may look relatively noisy, consider that our minimum and maximum DN ranges are -32,768 and 32,767 respectively. A value of around 30 to 50 in comparison is minuscule (less than 0.2 %) .

The gyro used as an example above almost zero drift.
In the next example a dc offset of 30 is observed.

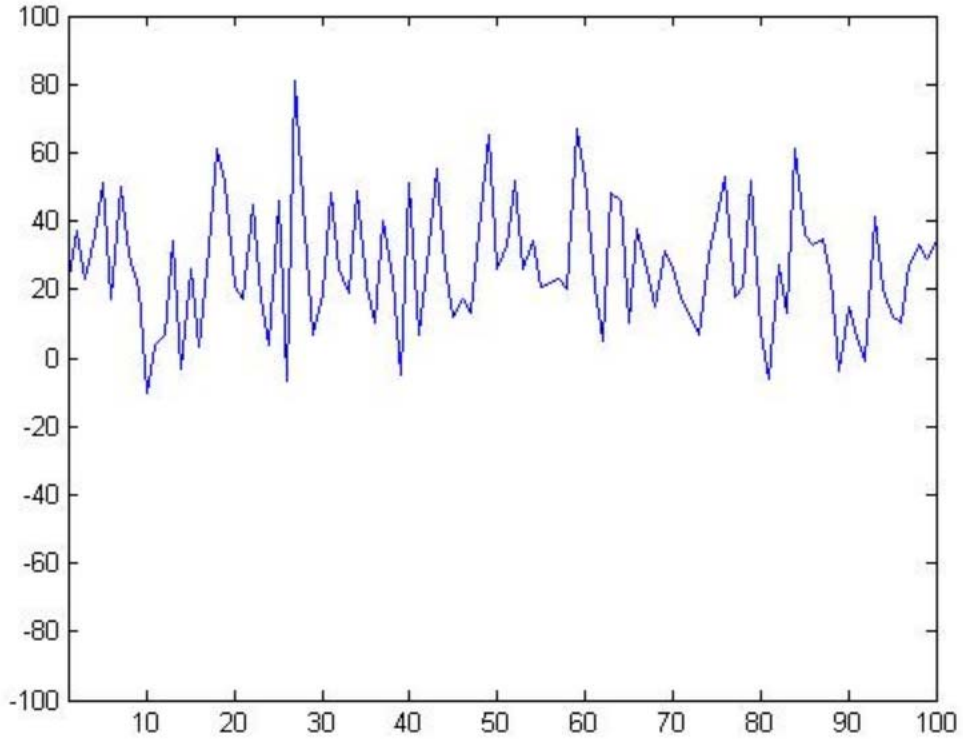


Figure 7.1b Gyro with DC offset

To correct for this offset simply change the constant dc_offset from 0 to 30.

```
dc_offset = 30;
```

Here is the output data with every point subtracted by 30. This simple code change has masked our DC

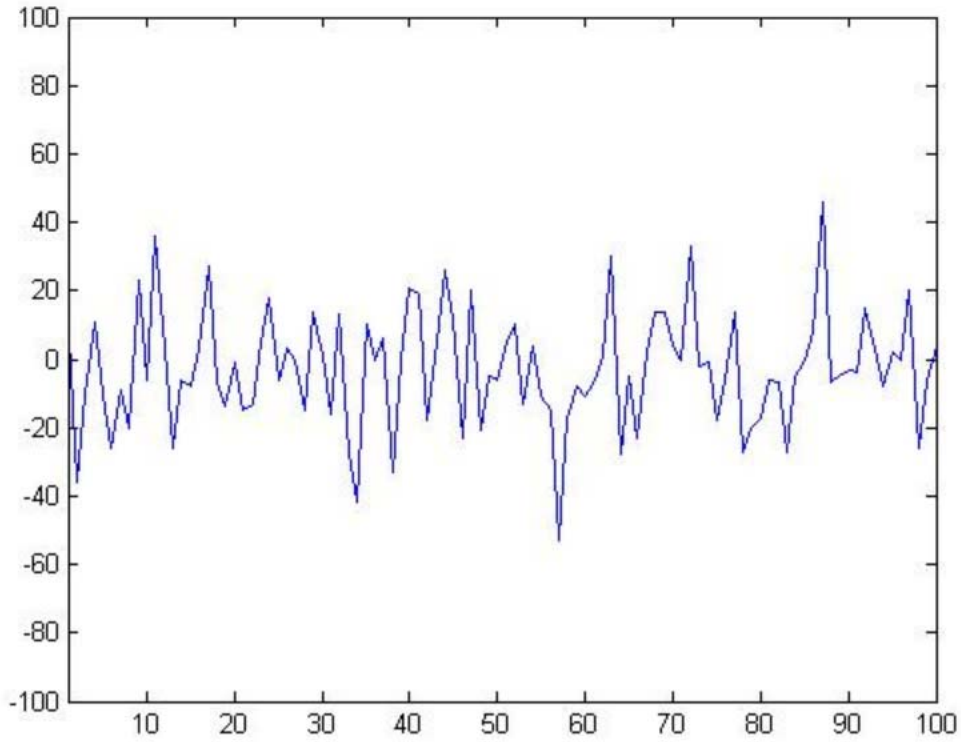


Figure 7.2c Gyro with DC offset programmatically removed

7.2 Gyro Drift at Rest (Zero-rate Level) Experiment

The value output by a gyroscope can change or "drift" over time. This phenomenon is noticed when at a rest position. Specifically, as the DN value oscillate randomly it may favor either the positive or negative side. Thus, when integrating the gyro, the drift error will be included in our current angle and accumulate over time. Before we run our experiment lets take a look at the code we will be running to calculate angle.

7.2.1 Angle Measurement Program

The following discussion of the Angle Measurement program used in the Gyro Drift Experiment will not make any sense unless you **read and understand** the How it Works section of this document.

The program will start a timer in MATLAB then ask the gyro for its current angular velocity. If the DN returned is larger than the at rest threshold, the timer is stopped and the change in angle

retrieve a value from the gyro (around 50 ms), we're not implementing any further delay at this point. The timer simply keeps track of this time as accurately as possible. Note: the gyro should be able to operate at significantly higher frequencies when running directly from the Arduino, which should improve accuracy.

```
%% Angle Measurement
% To stop measuring, rotate your rover 360 degrees or use Ctrl+C in the command
window.
dc_offset = 0;
gyro_drift = 0;
ang = 0;
sensitivity = 250;  % adjust for your gyro, should be 250.
threshold = 0;      % Threshold DN from stationary test

timer = tic;            % start timer
while abs(ang) < 360    % program stops after 360 degree turn
  for n = 1:20
   v = a.gyroRead('z') - dc_offset;    % read z axis, requires srv_gyro.pde on board
   if abs(v) > threshold   % apply drift threshold to raw data (DN)
    dt = toc(timer);       % dt = elapsed time since timer started
    timer = tic;
    ang = ang + (v/32768)*sensitivity*dt - gyro_drift; % integrate angle from
velocity
   end
  end
  % display angle after every 20 loops, trying to conserve processing time
  clc
  fprintf('Angle: %2.4f\n',ang)
end
```

7.2.2 Observe Drift

Copy the "Angle Measurement" code above and set an appropriate threshold level as determined from the Gyro Noise at Rest experiment. Leave your rover perfectly still and watch the calculated angle measurement; it should jump up and down a little bit, but stay very near zero.

7.2.3 Correct for Drift

If, after a few seconds the observed angle starts to steadily increase or decrease, you may want to consider changing the constant gyro_drift from zero to something like 0.0018. This value is used in the angle calculation (integrator).

because it measures both the drift and shows the noise graph with one run through.

```
%% Angle Measurement
% To stop measuring, rotate your rover 360 degrees or use Ctrl+C in the command
%window.
dc_offset = 0;
gyro_drift = 0;
ang = 0;
sensitivity = 300; % adjust for your gyro, should be 250.
threshold = 0; % Threshold DN from stationary test
timer = tic; % start timer
z = zeros(1,100); % setup an array for 100 samples
x = 1:100; % x-axis for plot
figure(1) % bring plot window to front
while abs(ang) < 360 % program stops after 360 degree turn
for n = 1:100
z(n) = a.gyroRead('z')-dc_offset; % sample gyro z-axis value
% live plot of data
plot(x,z)
axis([1 100 -100 100])
drawnow % forces MATLAB to redraw figures
v = a.gyroRead('z'); % read z axis, requires srv_gyro.pde on board
if abs(v) > threshold % apply drift threshold to raw data (DN)
dt = toc(timer); % dt = elapsed time since timer started
timer = tic;
ang = ang + (v/32768)*sensitivity*dt-gyro_drift; % integrate angle from velocity
end
end
% display angle after every 20 loops, trying to conserve processing time
clc
fprintf('Angle: %2.4f\n',ang)
fprintf('Max value: %d\n',max(z))
fprintf('Min value: %d\n',min(z))
fprintf('Mean value: %f\n',mean(z))
end
```

8 Measure Rotation

Print the Compass Drawing provided in Appendix B. Try running the Angle Measurement program and watching the value as you slowly rotate your rover over a fixed angle.

I found that my gyro was consistently measuring lower than it should be (i.e. rotating to 90 degrees measured only 77). If you experience similar results, try adjusting the sensitivity value (in my case approx. 310 worked well) and rerunning the program. - Nik

1. Copy the Filtering code below.
2. Run the program leaving the gyro at rest.
3. Observe the noise from the gyro in the top graph. The bottom graph implements a simple 5-point moving average filter to smooth the data. You may wish to modify the filter parameters by changing the highlighted sections of code.

```
%% DN Filtering
figure(1)
samples = 300;
filt = 5;
val1 = zeros(1,samples + filt);
val2 = zeros(1,samples + filt);
v = zeros(1,samples + filt);
x = 1:samples+filt;          % x-axis for plot
sensitivity = 310;  % adjust for your gyro, should be 250.
threshold = 0;      % Threshold DN from stationary test

timer = tic;            % start timer
for n = filt:samples+filt

v(n) = a.gyroRead('z');    % read z axis, requires srv_gyro.pde on board

if abs(v(n)) > threshold   % apply drift threshold to raw data (DN)

% implement 5-point moving averager
    val1(n) = (1/filt)*(v(n) + v(n-1) + v(n-2) + v(n-3) + v(n-4));
    val2(n) = v(n);

else
    val1(n) = 0;
    val2(n) = 0;
end % if


% display angle after every 20 loops, trying to conserve processing time


subplot(2,1,1)
plot(x,val1)
axis([1 samples+filt -50 50])
title('With Filtering')

subplot(2,1,2)
plot(x,val2)
axis([1 samples+filt -50 50])
title('Without Filtering')
drawnow
```

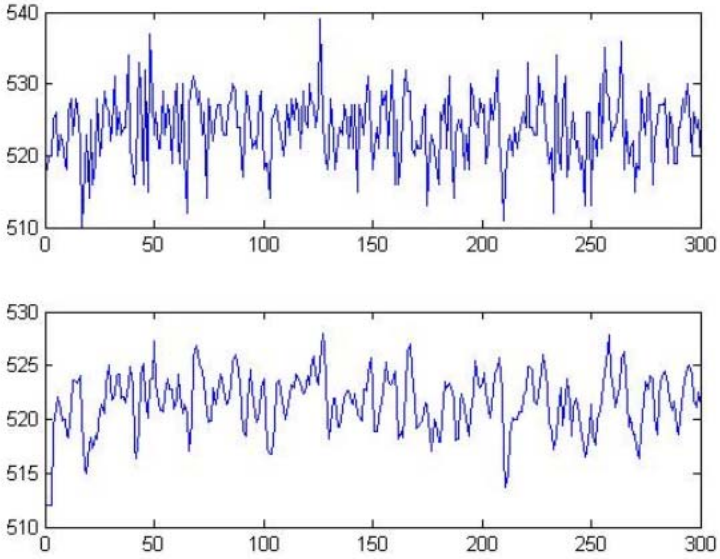
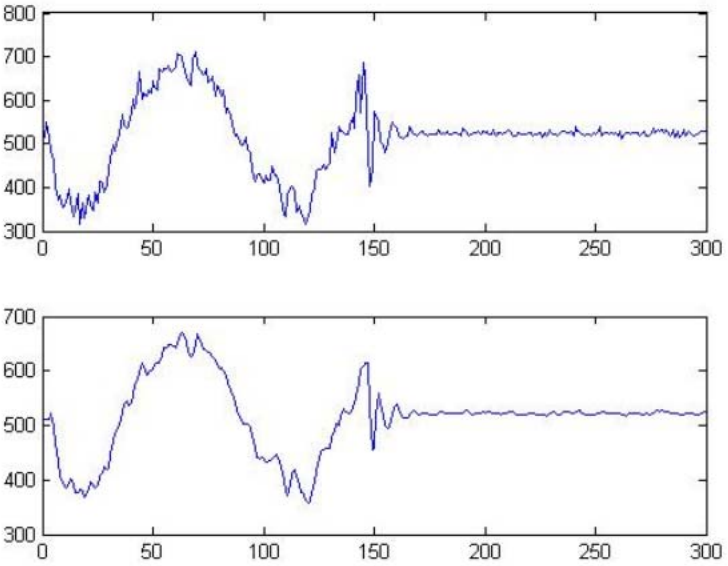


Figure 4.

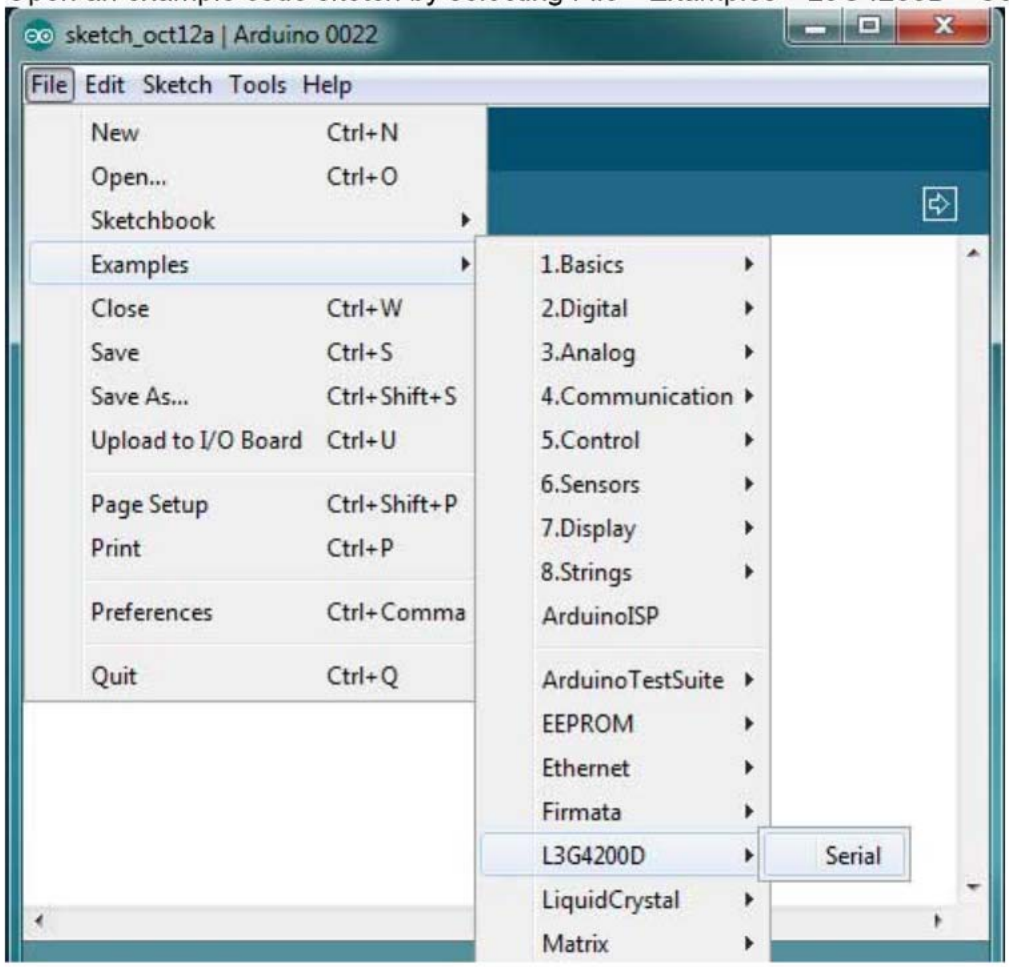
4. Repeat Steps 2-3 except rotate the gyro but keep below the threshold of 50°/sec. A graph similar to the following figure (Figure 5) is displayed.



Download the archive from "GitHub":https://github.com/pololu/L3G4200D, decompress it, and drag the "L3G4200D" folder to your arduino-00xx/libraries directory. Then restart the Arduino environment, so the libraries can be updated to include the L3G4200D library and its examples.

Step by Step Instructions

Open an example code sketch by selecting File→Examples→L3G4200D→Serial



Serial

```
#include <Wire.h>
#include <L3G4200D.h>

L3G4200D gyro;
```

```
  gyro.read();

  Serial.print("G ");
  Serial.print("X: ");
  Serial.print((int)gyro.g.x);
  Serial.print(" Y: ");
  Serial.print((int)gyro.g.y);
  Serial.print(" Z: ");
  Serial.println((int)gyro.g.z);

  delay(100);
}
```

This program continuously reads the gyro, communicating the readings over the serial interface. You can display the readings with the Arduino Serial Monitor.
Example output:

```
G X: 188 Y: -10 Z: -47
G X: 138 Y: -40 Z: -26
G X: 110 Y: -55 Z: 4
```

Appendix B Compass Drawing