IMU & AHRS algorithms

Francois

IMU & AHRS algorithms

The objective of this document is to provide a quick start to anyone interested in using IMUs.

Even though the code is simple and short, it is not a *plug and play* library. This document was created so you can understand what your code.

Hardware

You will need:

- Computer
- Arduino Uno R3
- IMU

I used IMU MinIMU-9 v3 from Polulu found here: https://www.pololu.com/product/2468

Follow tutorials and connect computer, Arduino and IMU

Software

You will need:

- Arduino IDE
- Arduino libraries for IMU sensors

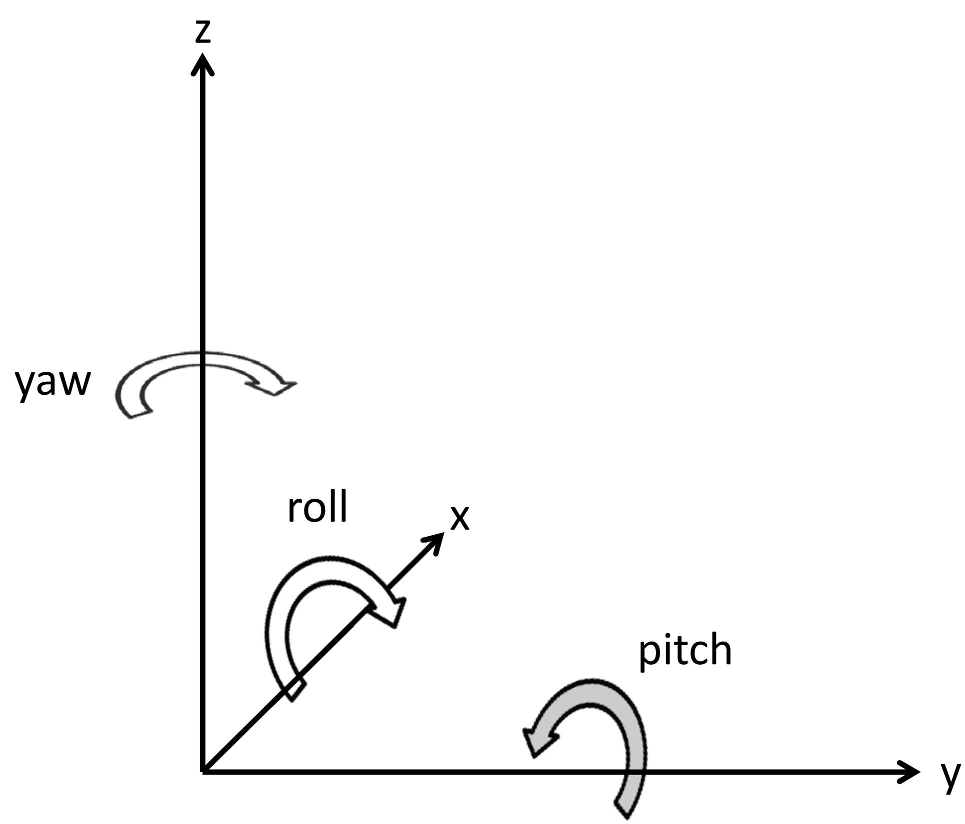
  https://www.pololu.com/product/2468/resources

- Excel
- PLX-DAQ add-in for Excel

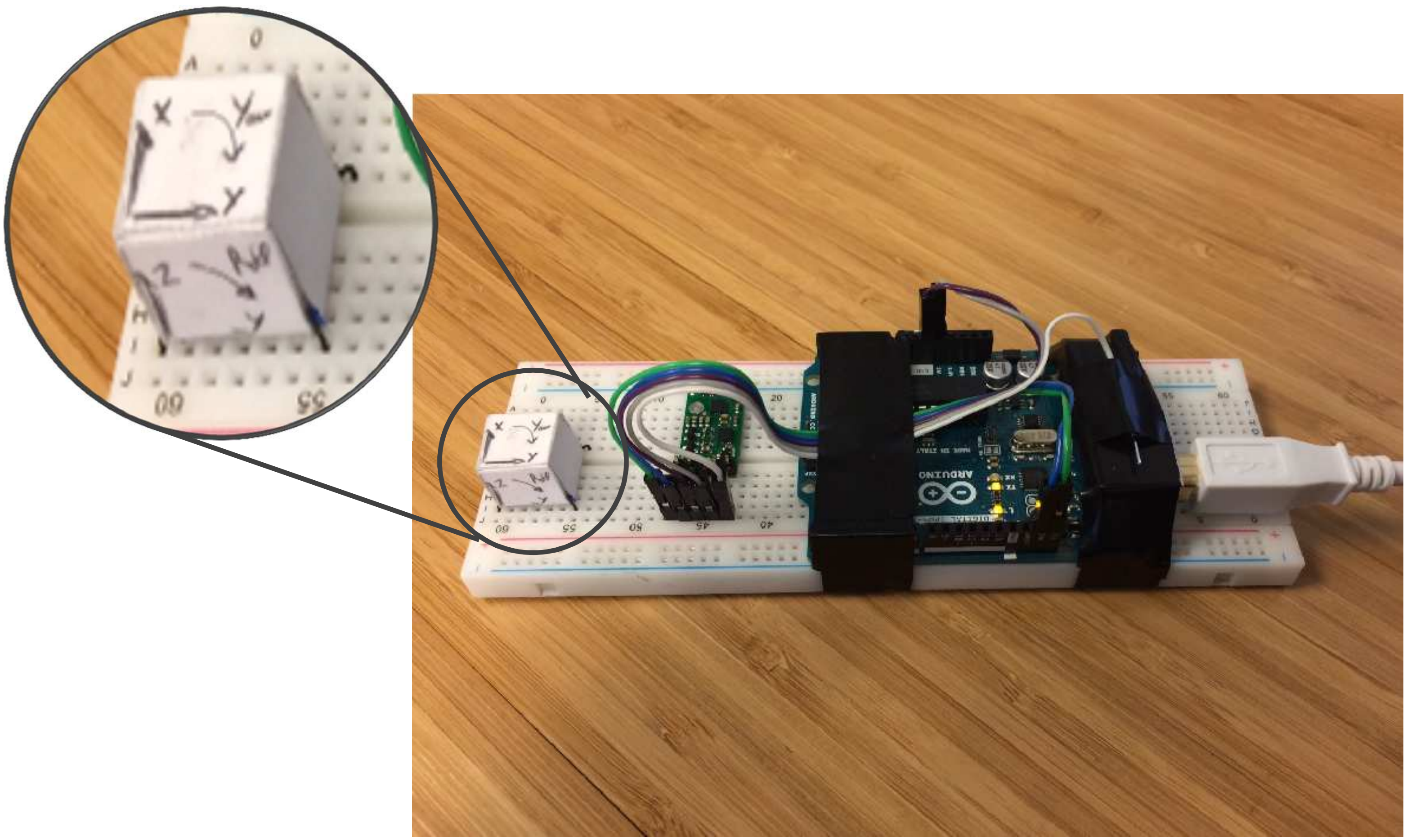
  http://www.parallax.com/downloads/plx-daq

IMU

Lets start with a reference

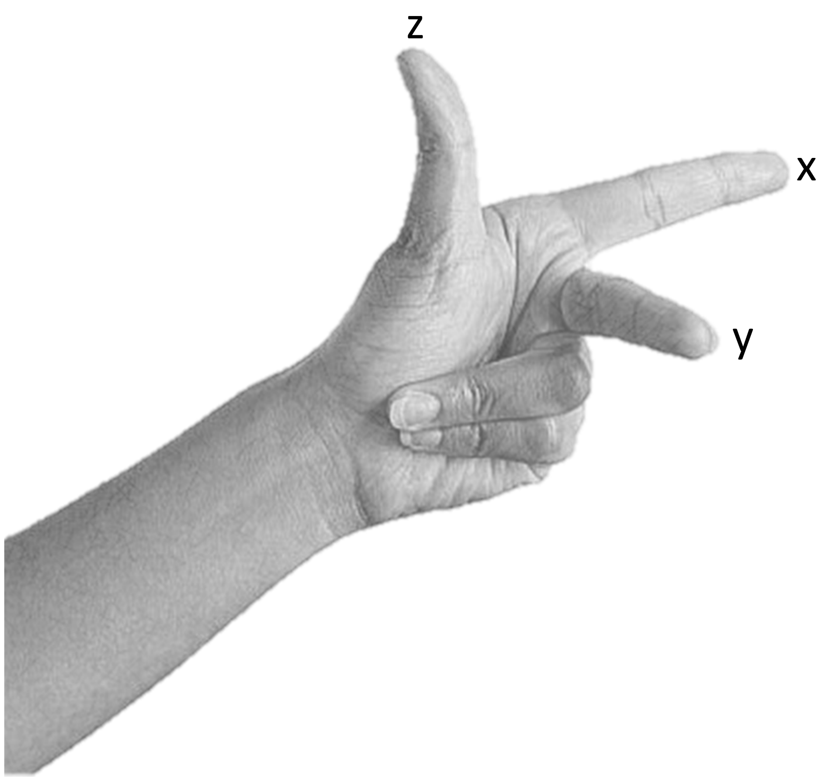


This is the most important step



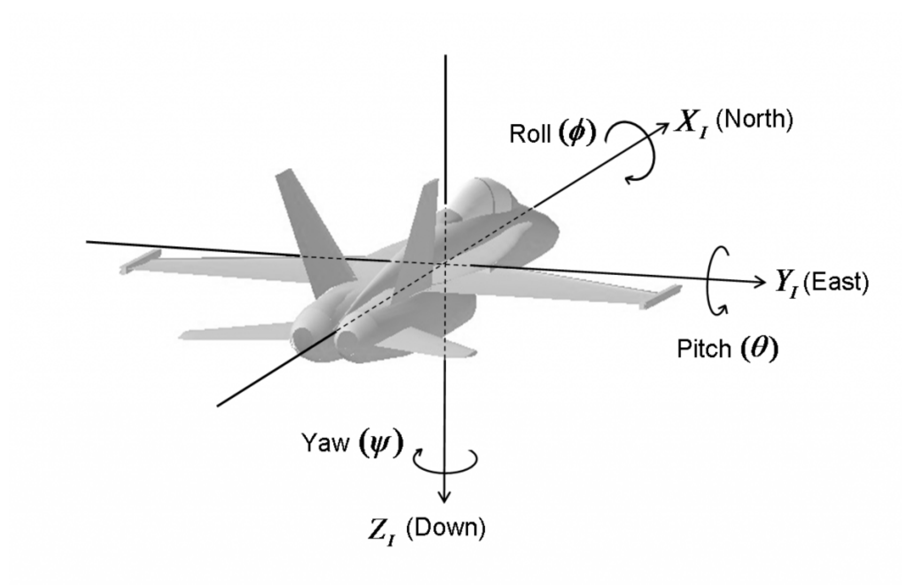
Make sure your reference is clear

OK



Not OK

Z is different



Now, lets get some data

sketch_feb18b | Arduino 1.6.0

File Edit Sketch Tools Help

sketch_feb18b

```
#include <Wire.h>        // load libraries
#include <L3G.h>
#include <LSM303.h>

L3G gyro;
LSM303 compass;

float A[4];              // declare variables A for accelerometer, G for gyro and M for magnetometer
float G[4];
float M[4];

void setup()
{
  Serial.begin(9600);    // start talking
  Wire.begin();
  gyro.init();           // start gyro
  gyro.enableDefault();
  compass.init();        // start accelerometer and magnetometer
  compass.enableDefault();
}

void loop()
{
  gyro.read();            // get data from sensors
  compass.read();

  A[1] = compass.a.x;     // record data
  A[2] = compass.a.y;
  A[3] = compass.a.z;
  G[1] = gyro.g.x;
  G[2] = gyro.g.y;
  G[3] = gyro.g.z;
  G[1] = gyro.g.x;
  G[2] = gyro.g.y;
  G[3] = gyro.g.z;

  Serial.print(A[1]); Serial.print(","); Serial.print(A[2]); Serial.print(","); Serial.print (A[3]); Serial.print(","); // print all data
  Serial.print(G[1]); Serial.print(","); Serial.print(G[2]); Serial.print(","); Serial.print (G[3]); Serial.print(",");
  Serial.print(M[1]); Serial.print(","); Serial.print(M[2]); Serial.print(","); Serial.print (M[3]); Serial.print(",");
  Serial.println();
  delay(20);
}
```

Done uploading.

```
Global variables use 526 bytes (25%) of dynamic memory, leaving 1,522 bytes for local variables. Maximum is 2,048 bytes.
```

28 Arduino Uno on COM6

Click upload and here it is:
raw data streaming to the serial port.

sketch_feb18b | Arduino 1.6.0

File Edit Sketch Tools Help

sketch_feb18b

```
#include <Wire.h>        // load libraries
#include <L3G.h>
#include <LSM303.h>

L3G gyro;
LSM303 compass;

float A[4];              // declare variables A for accelerometer, G for gyro and M for magnetomete
float G[4];
float M[4];

void setup()
{
  Serial.begin(9600);    // start talking
  Wire.begin();
  gyro.init();           // start gyro
  gyro.enableDefault();
  compass.init();        // start accelerometer and magnetometer
  compass.enableDefault();
}

void loop()
{
  gyro.read();            // get data from sensors
  compass.read();

  A[1] = compass.a.x;     // place holder for data conversion
  A[2] = compass.a.y;
  A[3] = compass.a.z;
  G[1] = gyro.g.x;
  G[2] = gyro.g.y;
  G[3] = gyro.g.z;
  G[1] = gyro.g.x;
  G[2] = gyro.g.y;
  G[3] = gyro.g.z;

  Serial.print(A[1]); Serial.print(","); Serial.print(A[2]); Serial.print(","); Serial.print (A[3]); Serial.print(","); // print all data
  Serial.print(G[1]); Serial.print(","); Serial.print(G[2]); Serial.print(","); Serial.print (G[3]); Serial.print(",");
  Serial.print(M[1]); Serial.print(","); Serial.print(M[2]); Serial.print(","); Serial.print (M[3]); Serial.print(",");
  Serial.println();
  delay(20);
}
```

COM6 (Arduino Uno)

Send

```
581.00,258.00,16804.00,91.00,-464.00,40.00,-32.00,146.00,-1544.00,
550.00,285.00,16823.00,81.00,-450.00,50.00,-33.00,148.00,-1542.00,
562.00,273.00,16805.00,83.00,-464.00,37.00,-33.00,148.00,-1542.00,
604.00,243.00,16826.00,88.00,-462.00,36.00,-33.00,148.00,-1542.00,
591.00,247.00,16791.00,81.00,-469.00,43.00,-30.00,142.00,-1537.00,
592.00,274.00,16795.00,75.00,-475.00,36.00,-30.00,142.00,-1537.00,
566.00,262.00,16775.00,84.00,-467.00,43.00,-29.00,150.00,-1552.00,
568.00,284.00,16824.00,73.00,-461.00,48.00,-29.00,150.00,-1552.00,
583.00,294.00,16816.00,75.00,-450.00,65.00,-35.00,143.00,-1533.00,
576.00,238.00,16787.00,77.00,-472.00,45.00,-35.00,143.00,-1533.00,
569.00,233.00,16828.00,79.00,-463.00,46.00,-26.00,146.00,-1544.00,
577.00,275.00,16808.00,94.00,-476.00,47.00,-26.00,146.00,-1544.00,
601.00,279.00,16779.00,93.00,-472.00,34.00,-26.00,146.00,-1544.00,
574.00,254.00,16822.00,78.00,-487.00,41.00,-34.00,147.00,-1553.00,
581.00,249.00,16771.00,88.00,-472.00,43.00,-34.00,147
```

Autoscroll | Newline | 9600 baud

Done uploading.

Global variables use 526 bytes (25%) of dynamic memory, leaving 1,522 bytes for local variables. Maximum is 2,048 bytes.

28 | Arduino Uno on COM6

Lets upload the raw data to Excel.
It is not necessary but so useful

First install PLX-DAQ add-in for Excel

Add a row variable

```
int row = 0;
```

Add these commands to setup() function

```
Serial.println("CLEARDATA");
Serial.println("LABEL,time,dt in ms,acce x,acce y,acce z,gyro x,gyro y,gyro z,mag x,mag y,mag z,roll,pitch,yaw");
```

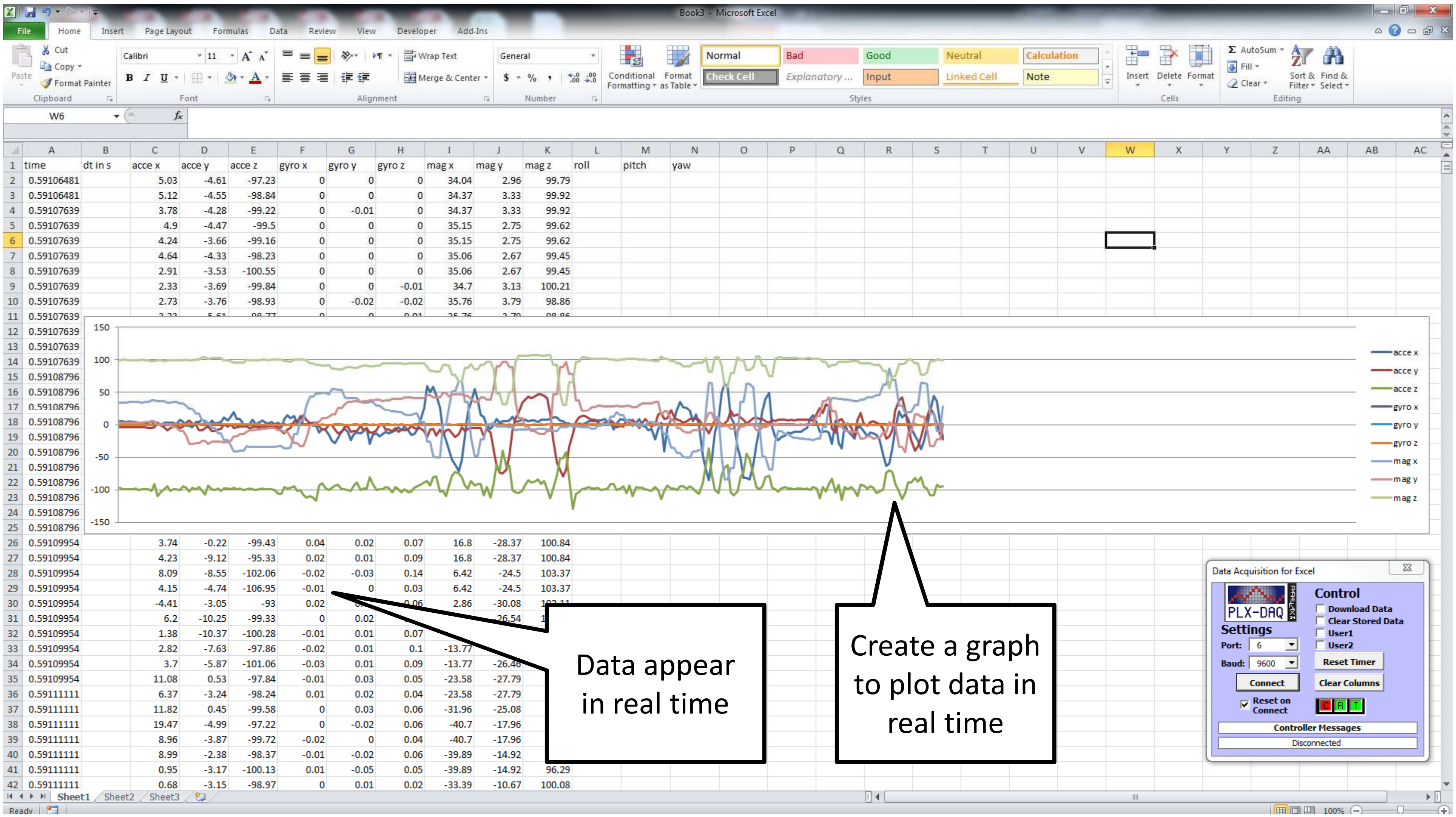
Replace print block in loop() function, by this one

```
Serial.print("DATA,TIME,");
Serial.print(",");  Serial.print(",");
Serial.print(A[1]); Serial.print(","); Serial.print(A[2]); Serial.print(","); Serial.print (A[3]); Serial.print(","); // print all data
Serial.print(G[1]); Serial.print(","); Serial.print(G[2]); Serial.print(","); Serial.print (G[3]); Serial.print(",");
Serial.print(M[1]); Serial.print(","); Serial.print(M[2]); Serial.print(","); Serial.print (M[3]); Serial.print(",");
Serial.println();

row++;
if (row > 500) {row=0; Serial.println("ROW,SET,2"); }
```

For a complete explanation, see: http://robottini.altervista.org/arduino-and-real-time-charts-in-excel

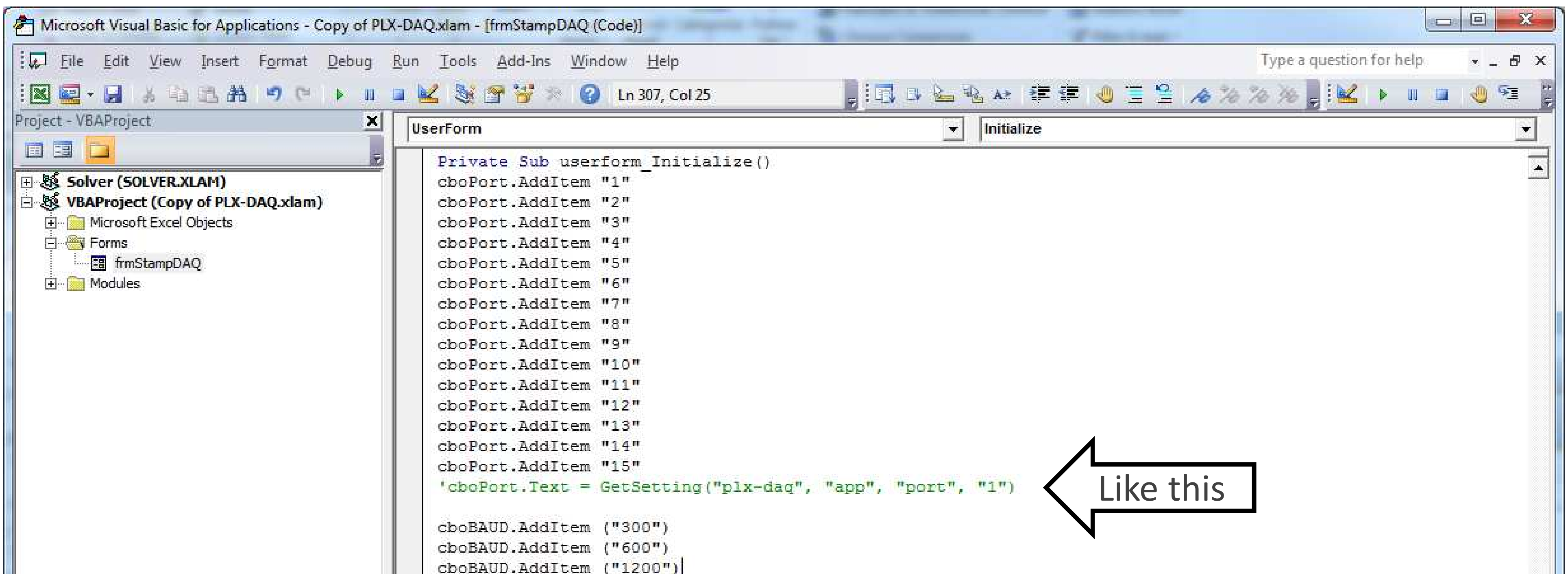
Open PLX add-in for Excel, choose the right serial port and click connect



Note: I had troubles with PLX add-in.

So if PLX add-in works you are lucky, if it does not work, do not panic: click *debug* or press *Alt+F11* to open VBA

Then find the line that bugs (it will be shown in yellow) and comment this line.



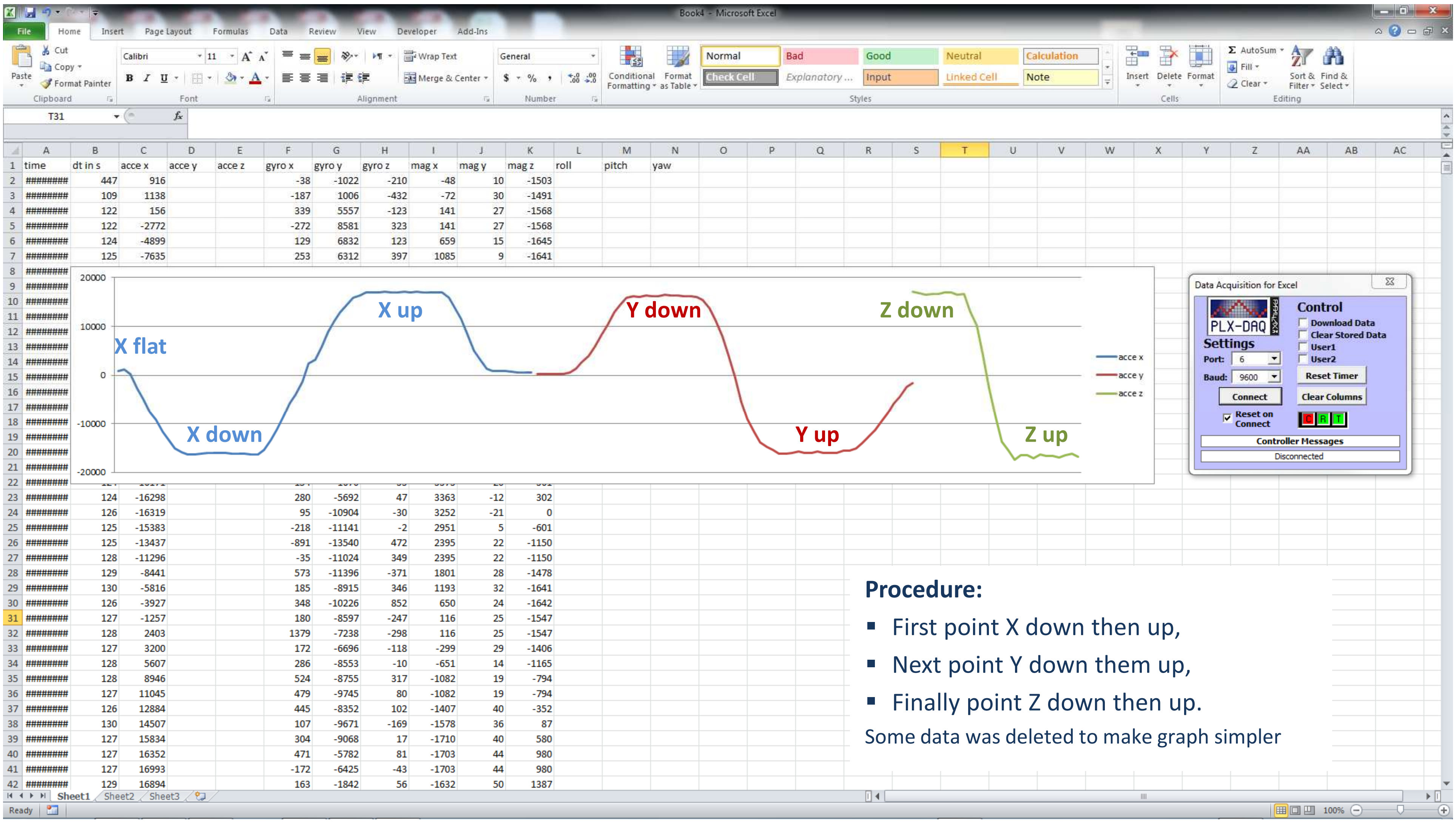
Now that you can plot the data in real time in Excel, we can start calibrating the accelerometer

The accelerometer measure the acceleration, however like any sensor it is not perfect and the accelerometer has an offset.

The best way to calibrate the offset of the accelerometer would be to go in space and measure the output of the sensor under no acceleration and no gravity, but this is not going to happen. So lets point the accelerometer down to measure gravity then point the sensor up to measure gravity again. The mean between the 2 values is the sensor’s offset.

Then repeat the experiment for all three axis.

Now that you can plot the data in real time in Excel, we can start calibrating the accelerometer



	time	dt in s	acce x	acce y	acce z	gyro x	gyro y	gyro z	mag x	mag y	mag z	roll	pitch	yaw
2	#########	447	916			-38	-1022	-210	-48	10	-1503			
3	#########	109	1138			-187	1006	-432	-72	30	-1491			
4	#########	122	156			339	5557	-123	141	27	-1568			
5	#########	122	-2772			-272	8581	323	141	27	-1568			
6	#########	124	-4899			129	6832	123	659	15	-1645			
7	#########	125	-7635			253	6312	397	1085	9	-1641			
23	#########	124	-16298			280	-5692	47	3363	-12	302			
24	#########	126	-16319			95	-10904	-30	3252	-21	0			
25	#########	125	-15383			-218	-11141	-2	2951	5	-601			
26	#########	125	-13437			-891	-13540	472	2395	22	-1150			
27	#########	128	-11296			-35	-11024	349	2395	22	-1150			
28	#########	129	-8441			573	-11396	-371	1801	28	-1478			
29	#########	130	-5816			185	-8915	346	1193	32	-1641			
30	#########	126	-3927			348	-10226	852	650	24	-1642			
31	#########	127	-1257			180	-8597	-247	116	25	-1547			
32	#########	128	2403			1379	-7238	-298	116	25	-1547			
33	#########	127	3200			172	-6696	-118	-299	29	-1406			
34	#########	128	5607			286	-8553	-10	-651	14	-1165			
35	#########	128	8946			524	-8755	317	-1082	19	-794			
36	#########	127	11045			479	-9745	80	-1082	19	-794			
37	#########	126	12884			445	-8352	102	-1407	40	-352			
38	#########	130	14507			107	-9671	-169	-1578	36	87			
39	#########	127	15834			304	-9068	17	-1710	40	580			
40	#########	127	16352			471	-5782	81	-1703	44	980			
41	#########	127	16993			-172	-6425	-43	-1703	44	980			
42	#########	129	16894			163	-1842	56	-1632	50	1387			

Procedure:

- First point X down then up,
- Next point Y down them up,
- Finally point Z down then up.

Some data was deleted to make graph simpler

Now that you can plot the data in real time in Excel, we can start calibrating the accelerometer

I added the calibration factors as definitions before the setup() function like this:

```
#define ACCEL_X_MIN ((float) -16340)  // Add Min an Max values from calibration
#define ACCEL_X_MAX ((float)  16975)
#define ACCEL_Y_MIN ((float) -15830)
#define ACCEL_Y_MAX ((float)  16380)
#define ACCEL_Z_MIN ((float) -16570)
#define ACCEL_Z_MAX ((float)  16910)
#define ACCEL_X_DIR ((int)    1    )  // If up and down are reversed then the direction of the sensor is negative
#define ACCEL_Y_DIR ((int)   -1    )
#define ACCEL_Z_DIR ((int)   -1    )
#define ACCEL_X_OFFSET ((ACCEL_X_MIN + ACCEL_X_MAX) / 2.0f)  // The Offset is the average of the Min and MAX values
#define ACCEL_Y_OFFSET ((ACCEL_Y_MIN + ACCEL_Y_MAX) / 2.0f)
#define ACCEL_Z_OFFSET ((ACCEL_Z_MIN + ACCEL_Z_MAX) / 2.0f)
#define ACCEL_X_SCALE  (100.0f / (ACCEL_X_MAX - ACCEL_X_OFFSET) ) // Scale all accelerometers between -100 and 100
#define ACCEL_Y_SCALE  (100.0f / (ACCEL_Y_MAX - ACCEL_Y_OFFSET) )
#define ACCEL_Z_SCALE  (100.0f / (ACCEL_Z_MAX - ACCEL_Z_OFFSET) )
```

Note: scaling all accelerometer's value between -100 and 100 is arbitrary, most people like to scale the accelerometer between -9.8 and 9.8 because gravity was used during calibration. In the end accelerometers data will be used to calculate angles so the scale factor does not change anything.

A good tutorial for calibration is found here: http://www.starlino.com/imu_guide.html

Now, process raw accelerometer data into calibrated accelerometer data

Replace this

```
A[1] = compass.a.x;     // place holder for data conversion
A[2] = compass.a.y;
A[3] = compass.a.z;
```

Into this

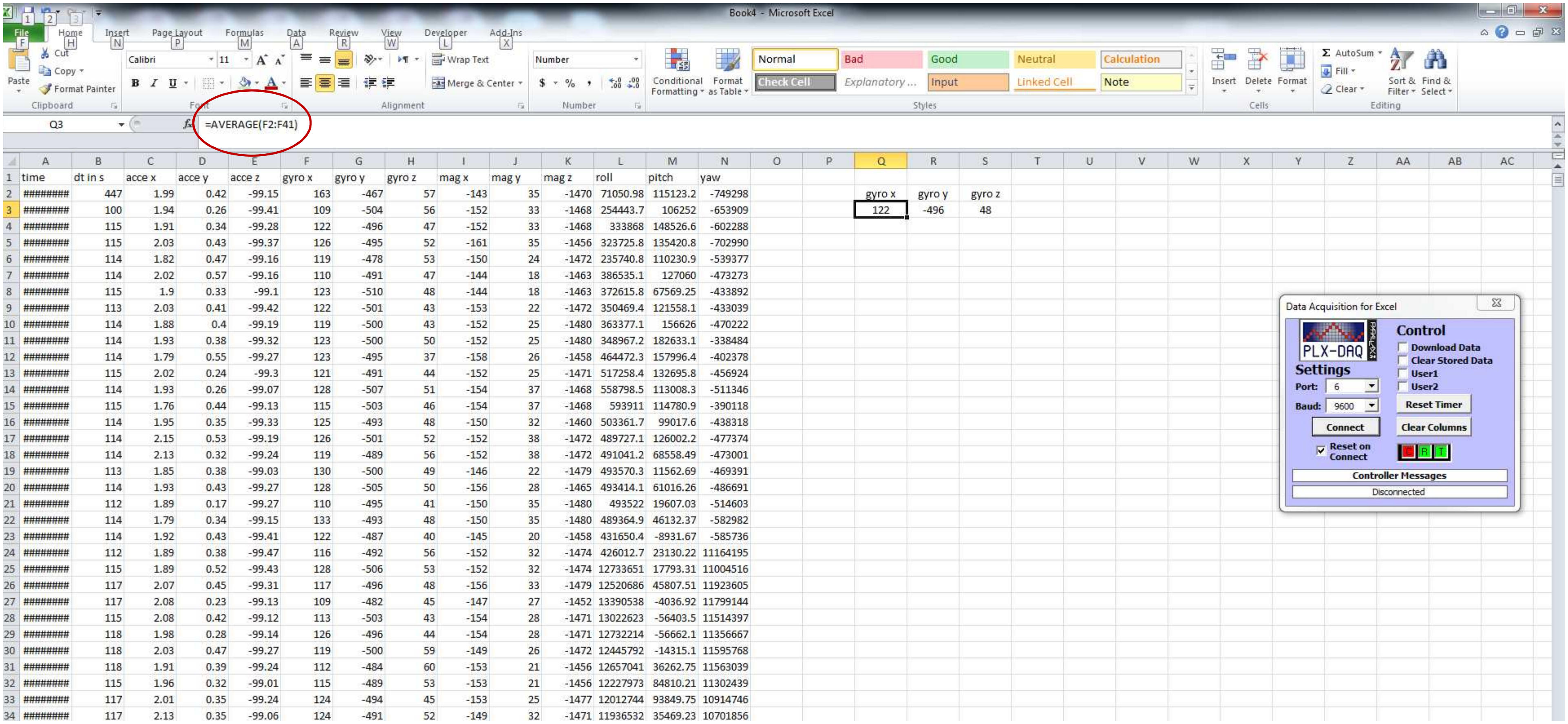
```
A[1] = (compass.a.x - ACCEL_X_OFFSET) * ACCEL_X_SCALE * ACCEL_X_DIR; // accelerometer' values are now between -100 and 100
A[2] = (compass.a.y - ACCEL_Y_OFFSET) * ACCEL_Y_SCALE * ACCEL_Y_DIR;
A[3] = (compass.a.z - ACCEL_Z_OFFSET) * ACCEL_Z_SCALE * ACCEL_Z_DIR;
```

Final results! Accelerometer values as a function of time

All values are fixed between -100 and 100 and value are negative when sensor is down and positive when sensor is up

Now let's calibrate the gyro's offset

Getting the offset of the gyro is easy, leave the sensor alone (not moving) and look at the gyro's values for x y and z.

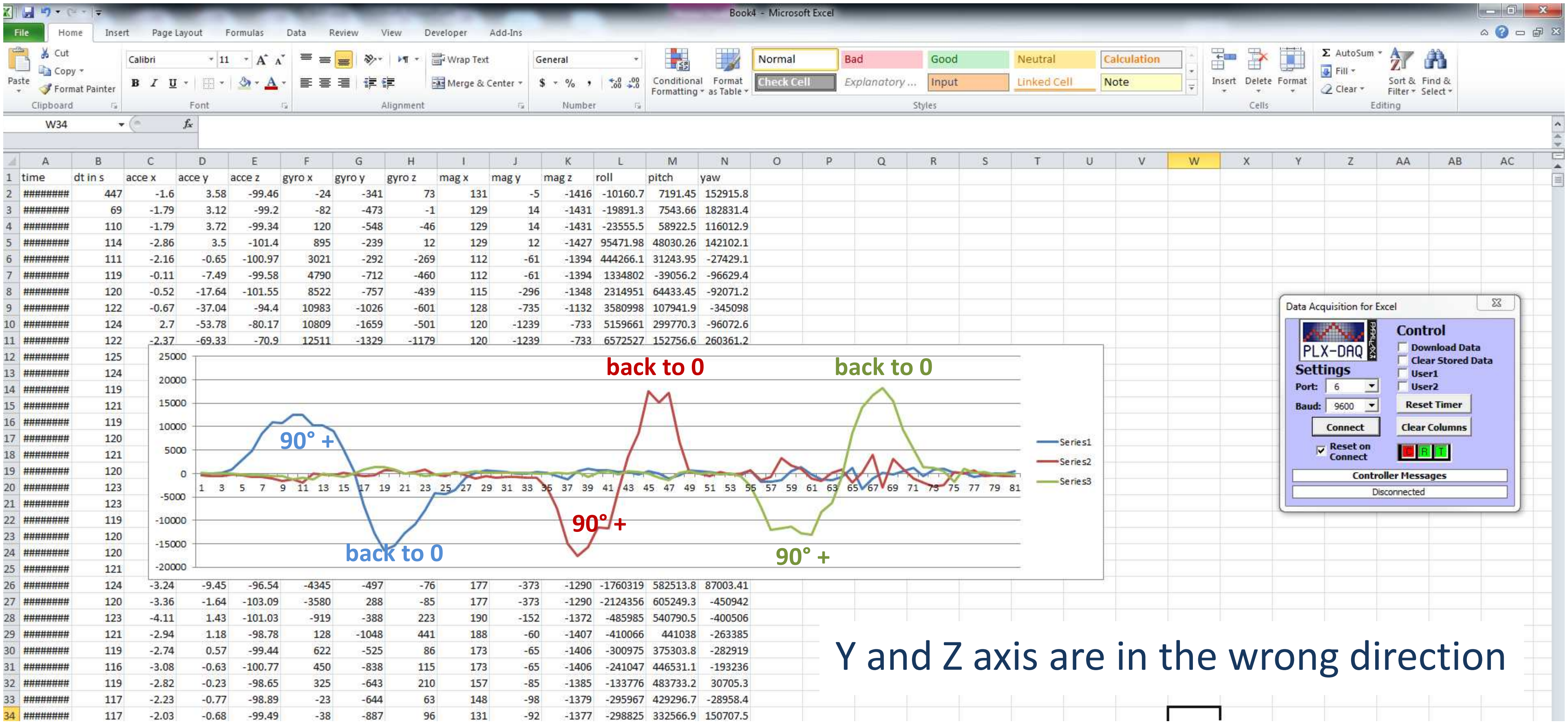


	gyro x	gyro y	gyro z
	122	-496	48

	time	dt in s	acce x	acce y	acce z	gyro x	gyro y	gyro z	mag x	mag y	mag z	roll	pitch	yaw
2	########	447	1.99	0.42	-99.15	163	-467	57	-143	35	-1470	71050.98	115123.2	-749298
3	########	100	1.94	0.26	-99.41	109	-504	56	-152	33	-1468	254443.7	106252	-653909
4	########	115	1.91	0.34	-99.28	122	-496	47	-152	33	-1468	333868	148526.6	-602288
5	########	115	2.03	0.43	-99.37	126	-495	52	-161	35	-1456	323725.8	135420.8	-702990
6	########	114	1.82	0.47	-99.16	119	-478	53	-150	24	-1472	235740.8	110230.9	-539377
7	########	114	2.02	0.57	-99.16	110	-491	47	-144	18	-1463	386535.1	127060	-473273
8	########	115	1.9	0.33	-99.1	123	-510	48	-144	18	-1463	372615.8	67569.25	-433892
9	########	113	2.03	0.41	-99.42	122	-501	43	-153	22	-1472	350469.4	121558.1	-433039
10	########	114	1.88	0.4	-99.19	119	-500	43	-152	25	-1480	363377.1	156626	-470222
11	########	114	1.93	0.38	-99.32	123	-500	50	-152	25	-1480	348967.2	182633.1	-338484
12	########	114	1.79	0.55	-99.27	123	-495	37	-158	26	-1458	464472.3	157996.4	-402378
13	########	115	2.02	0.24	-99.3	121	-491	44	-152	25	-1471	517258.4	132695.8	-456924
14	########	114	1.93	0.26	-99.07	128	-507	51	-154	37	-1468	558798.5	113008.3	-511346
15	########	115	1.76	0.44	-99.13	115	-503	46	-154	37	-1468	593911	114780.9	-390118
16	########	114	1.95	0.35	-99.33	125	-493	48	-150	32	-1460	503361.7	99017.6	-438318
17	########	114	2.15	0.53	-99.19	126	-501	52	-152	38	-1472	489727.1	126002.2	-477374
18	########	114	2.13	0.32	-99.24	119	-489	56	-152	38	-1472	491041.2	68558.49	-473001
19	########	113	1.85	0.38	-99.03	130	-500	49	-146	22	-1479	493570.3	11562.69	-469391
20	########	114	1.93	0.43	-99.27	128	-505	50	-156	28	-1465	493414.1	61016.26	-486691
21	########	112	1.89	0.17	-99.27	110	-495	41	-150	35	-1480	493522	19607.03	-514603
22	########	114	1.79	0.34	-99.15	133	-493	48	-150	35	-1480	489364.9	46132.37	-582982
23	########	114	1.92	0.43	-99.41	122	-487	40	-145	20	-1458	431650.4	-8931.67	-585736
24	########	112	1.89	0.38	-99.47	116	-492	56	-152	32	-1474	426012.7	23130.22	11164195
25	########	115	1.89	0.52	-99.43	128	-506	53	-152	32	-1474	12733651	17793.31	11004516
26	########	117	2.07	0.45	-99.31	117	-496	48	-156	33	-1479	12520686	45807.51	11923605
27	########	117	2.08	0.23	-99.13	109	-482	45	-147	27	-1452	13390538	-4036.92	11799144
28	########	115	2.08	0.42	-99.12	113	-503	43	-154	28	-1471	13022623	-56403.5	11514397
29	########	118	1.98	0.28	-99.14	126	-496	44	-154	28	-1471	12732214	-56662.1	11356667
30	########	118	2.03	0.47	-99.27	119	-500	59	-149	26	-1472	12445792	-14315.1	11595768
31	########	118	1.91	0.39	-99.24	112	-484	60	-153	21	-1456	12657041	36262.75	11563039
32	########	115	1.96	0.32	-99.01	115	-489	53	-153	21	-1456	12227973	84810.21	11302439
33	########	117	2.01	0.35	-99.24	124	-494	45	-153	25	-1477	12012744	93849.75	10914746
34	########	117	2.13	0.35	-99.06	124	-491	52	-149	32	-1471	11936532	35469.23	10701856

Now let's calibrate the gyro's direction

Let's check the direction of the sensor, by rolling 90° positive, pitching 90° positive and yawing 90° positive.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	time	dt in s	acce x	acce y	acce z	gyro x	gyro y	gyro z	mag x	mag y	mag z	roll	pitch	yaw
2	########	447	-1.6	3.58	-99.46	-24	-341	73	131	-5	-1416	-10160.7	7191.45	152915.8
3	########	69	-1.79	3.12	-99.2	-82	-473	-1	129	14	-1431	-19891.3	7543.66	182831.4
4	########	110	-1.79	3.72	-99.34	120	-548	-46	129	14	-1431	-23555.5	58922.5	116012.9
5	########	114	-2.86	3.5	-101.4	895	-239	12	129	12	-1427	95471.98	48030.26	142102.1
6	########	111	-2.16	-0.65	-100.97	3021	-292	-269	112	-61	-1394	444266.1	31243.95	-27429.1
7	########	119	-0.11	-7.49	-99.58	4790	-712	-460	112	-61	-1394	1334802	-39056.2	-96629.4
8	########	120	-0.52	-17.64	-101.55	8522	-757	-439	115	-296	-1348	2314951	64433.45	-92071.2
9	########	122	-0.67	-37.04	-94.4	10983	-1026	-601	128	-735	-1132	3580998	107941.9	-345098
10	########	124	2.7	-53.78	-80.17	10809	-1659	-501	120	-1239	-733	5159661	299770.3	-96072.6
11	########	122	-2.37	-69.33	-70.9	12511	-1329	-1179	120	-1239	-733	6572527	152756.6	260361.2
12	########	125												
13	########	124												
14	########	119												
15	########	121												
16	########	119												
17	########	120												
18	########	121												
19	########	120												
20	########	123												
21	########	123												
22	########	119												
23	########	120												
24	########	120												
25	########	121												
26	########	124	-3.24	-9.45	-96.54	-4345	-497	-76	177	-373	-1290	-1760319	582513.8	87003.41
27	########	120	-3.36	-1.64	-103.09	-3580	288	-85	177	-373	-1290	-2124356	605249.3	-450942
28	########	123	-4.11	1.43	-101.03	-919	-388	223	190	-152	-1372	-485985	540790.5	-400506
29	########	121	-2.94	1.18	-98.78	128	-1048	441	188	-60	-1407	-410066	441038	-263385
30	########	119	-2.74	0.57	-99.44	622	-525	86	173	-65	-1406	-300975	375303.8	-282919
31	########	116	-3.08	-0.63	-100.77	450	-838	115	173	-65	-1406	-241047	446531.1	-193236
32	########	119	-2.82	-0.23	-98.65	325	-643	210	157	-85	-1385	-133776	483733.2	30705.3
33	########	117	-2.23	-0.77	-98.89	-23	-644	63	148	-98	-1379	-295967	429296.7	-28958.4
34	########	117	-2.03	-0.68	-99.49	-38	-887	96	131	-92	-1377	-298825	332566.9	150707.5

Y and Z axis are in the wrong direction

We can start calibrating the gyro's data

Add offset and directions of the gyro as definitions

```
define GYRO_OFFSET_X ((float)   122.0 )
#define GYRO_OFFSET_Y ((float) -496.0 )
#define GYRO_OFFSET_Z ((float)   48.0 )

#define GYRO_X_DIR ((int)        1   )
#define GYRO_Y_DIR ((int)       -1   )
#define GYRO_Z_DIR ((int)       -1   )
```

Then, replace this block

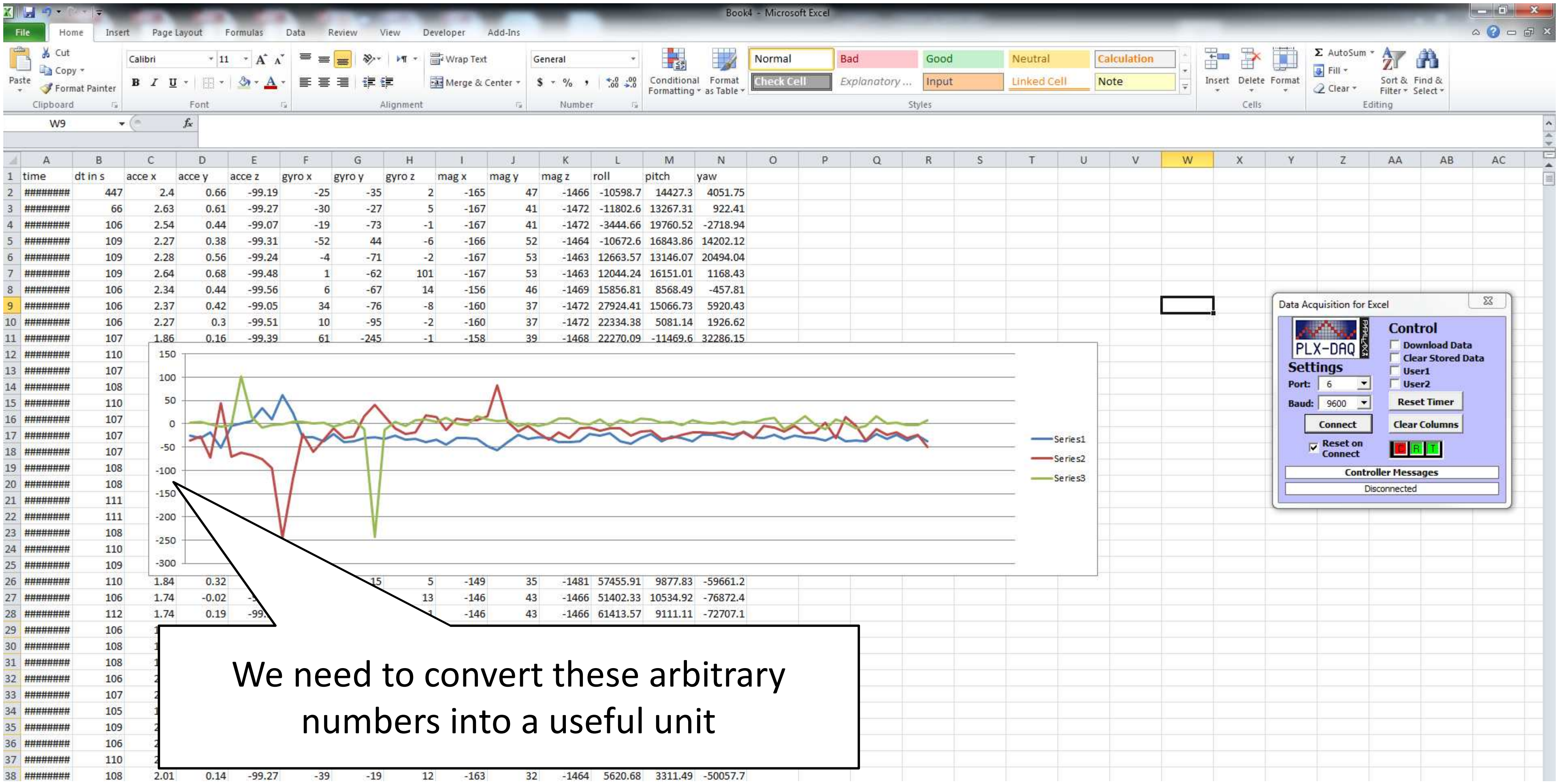
```
G[1] = gyro.g.x;
G[2] = gyro.g.y;
G[3] = gyro.g.z;
```

By this one

```
G[1] = (gyro.g.x - GYRO_OFFSET_X) * GYRO_X_DIR;  // This take into account the offset of the gyro and the direction
G[2] = (gyro.g.y - GYRO_OFFSET_Y) * GYRO_Y_DIR;  // scale factor is still missing
G[3] = (gyro.g.z - GYRO_OFFSET_Z) * GYRO_Z_DIR;
```

Now the gyro’s data is 0 when the gyro is not moving and direction correct, but data has no units

Let’s calibrate the gyro’s scale factor.



We need to find out the units of the gyro

The gyro provides a rate of change, not the change itself, so we need take time into account

First declare time variables

```
float dt;             // time between gyro readings in milliseconds
float t  = millis();  // time = now in milliseconds
```

Then right after reading the gyro data add

```
dt = millis() - t;
t  = millis();
```

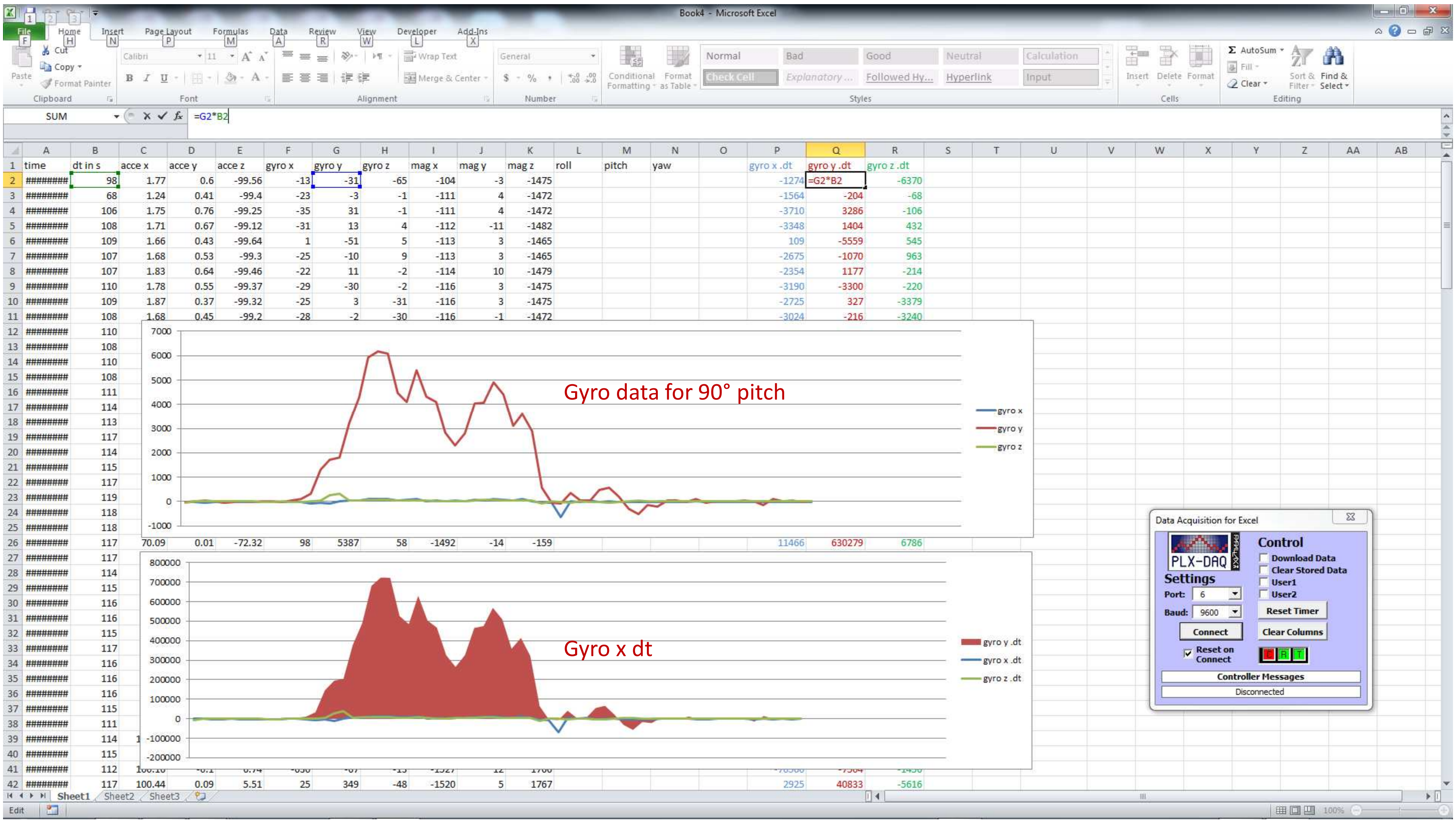
Add the time between readings into excel

```
Serial.print("DATA,TIME,");
Serial.print(dt);   Serial.print(",");  // dt is added here nothing else change
Serial.print(A[1]); Serial.print(","); Serial.print(A[2]); Serial.print(","); Serial.print (A[3]); Serial.print(",");
Serial.print(G[1]); Serial.print(","); Serial.print(G[2]); Serial.print(","); Serial.print (G[3]); Serial.print(",");
Serial.print(M[1]); Serial.print(","); Serial.print(M[2]); Serial.print(","); Serial.print (M[3]); Serial.print(",");
Serial.println();
```

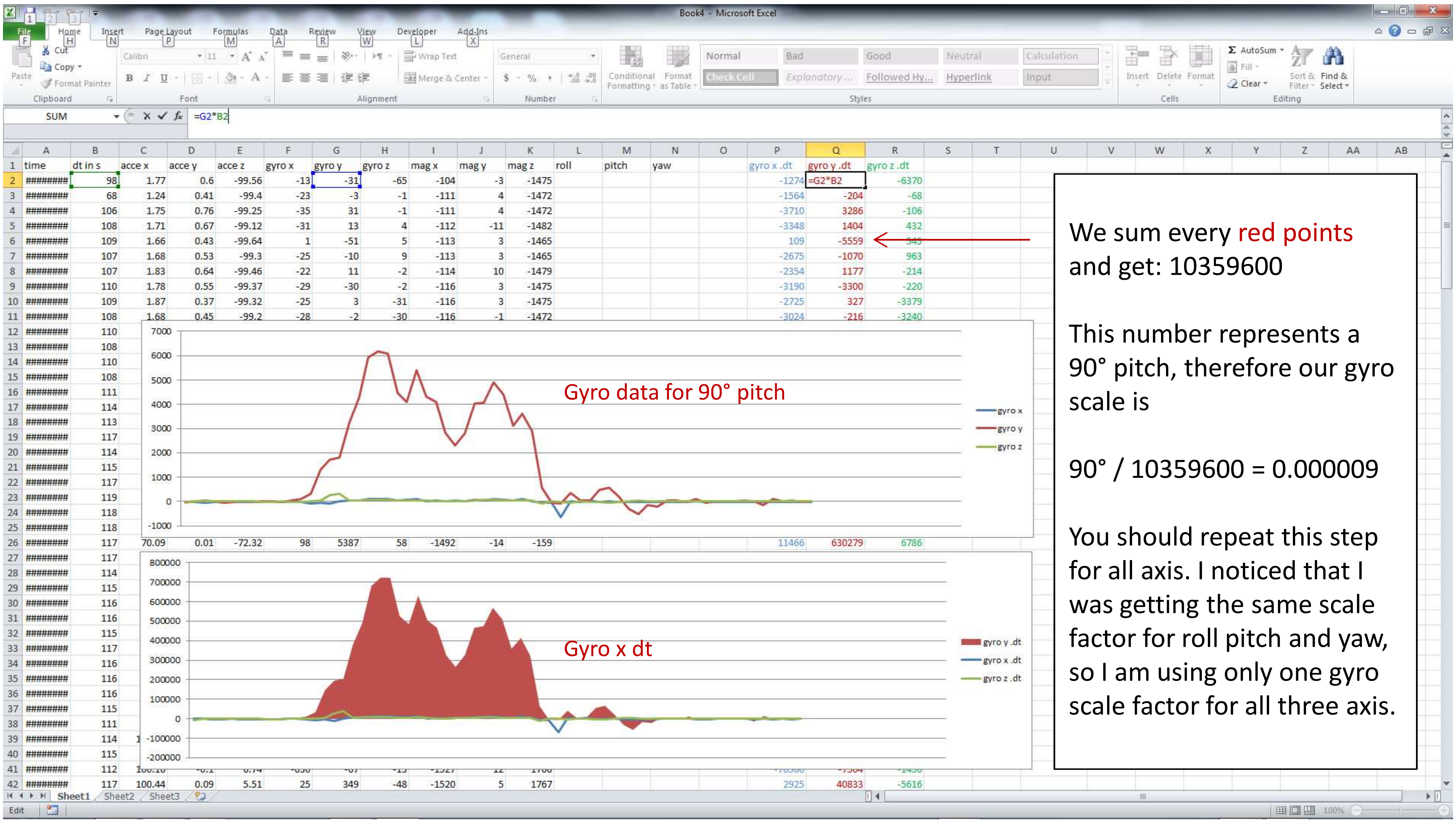
Now the gyro's data is 0 when the gyro is not moving and directions are correct.

Next, we need to convert the gyro arbitrary scale to something useful.
In order to do that we move the gyro by 90 degrees and record the data:

Add in Excel a column with the eq. Gyro x dt



Add in Excel a column with the eq. Gyro x dt



We sum every red points and get: 10359600

This number represents a 90° pitch, therefore our gyro scale is

90° / 10359600 = 0.000009

You should repeat this step for all axis. I noticed that I was getting the same scale factor for roll pitch and yaw, so I am using only one gyro scale factor for all three axis.

Final gyro's calibration

Add offset and directions of the gyro as definitions

```
define GYRO_OFFSET_X ((float)   122.0      )
#define GYRO_OFFSET_Y ((float) -496.0      )
#define GYRO_OFFSET_Z ((float)   48.0      )
#define GYRO_SCALE ((float)       0.0000090)
#define GYRO_X_DIR ((int)         1        )
#define GYRO_Y_DIR ((int)        -1        )
#define GYRO_Z_DIR ((int)        -1        )
```

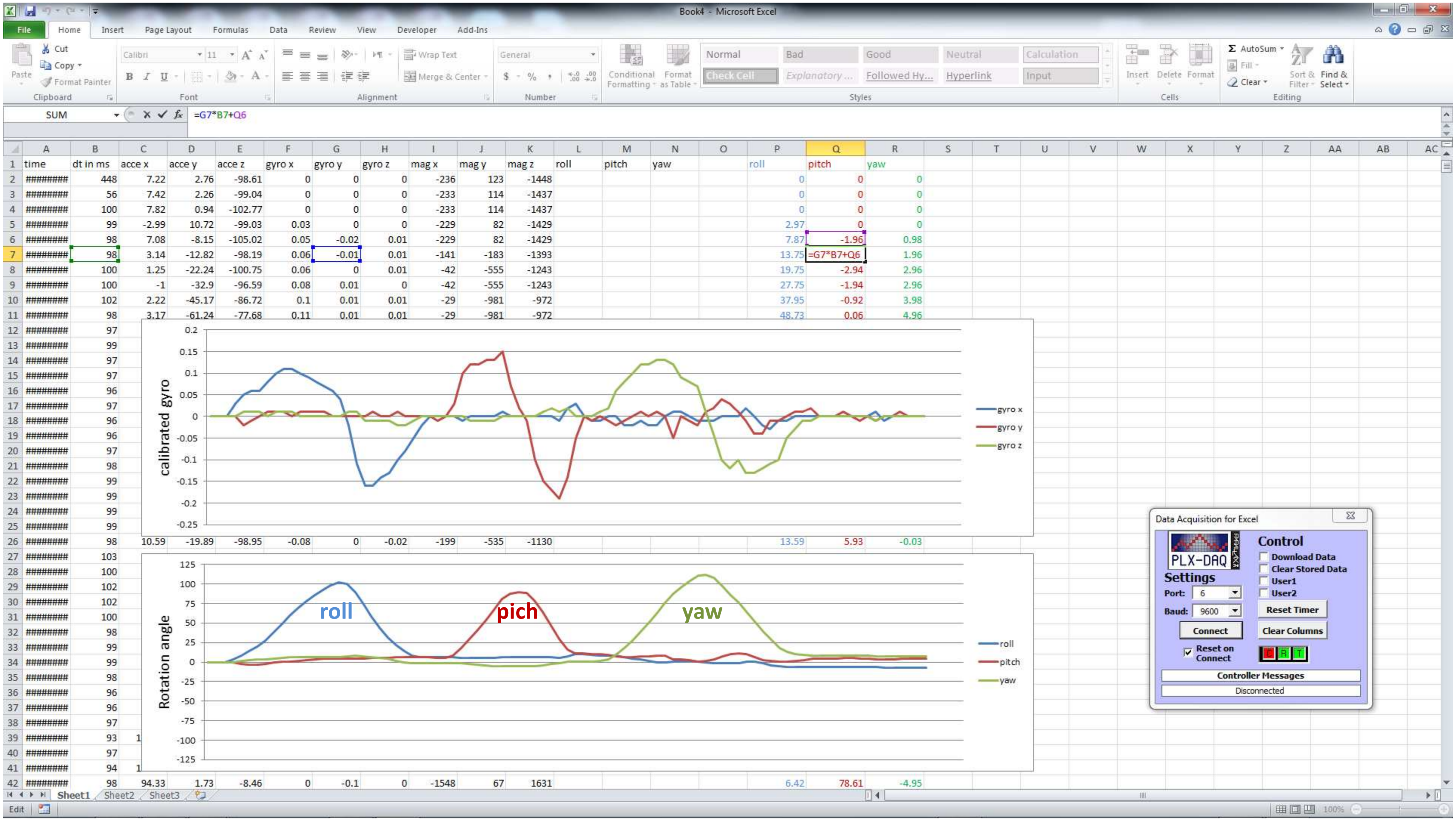
Then, replace this block

```
G[1] = (gyro.g.x - GYRO_OFFSET_X) * GYRO_X_DIR;  // This take into account the offset of the gyro and the direction
G[2] = (gyro.g.y - GYRO_OFFSET_Y) * GYRO_Y_DIR;
G[3] = (gyro.g.z - GYRO_OFFSET_Z) * GYRO_Z_DIR;
```

By this one

```
G[1] = (gyro.g.x - GYRO_OFFSET_X) * GYRO_SCALE * GYRO_X_DIR;  // now the gyro's data is fully calibrated [degrees / ms / gyro unit]
G[2] = (gyro.g.y - GYRO_OFFSET_Y) * GYRO_SCALE * GYRO_Y_DIR;
G[3] = (gyro.g.z - GYRO_OFFSET_Z) * GYRO_SCALE * GYRO_Z_DIR;
```

Gyro is now calibrated !



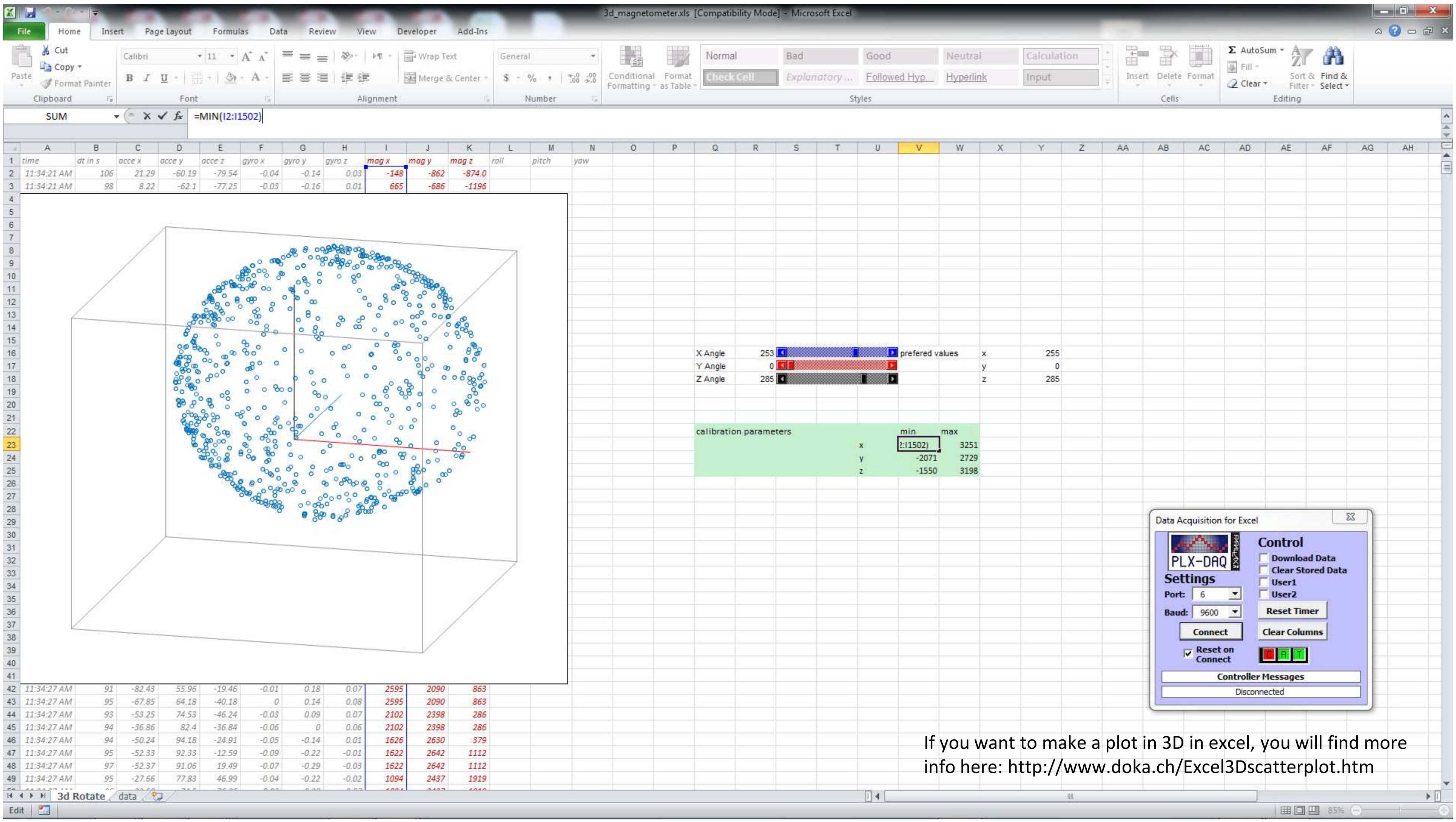
Calibration of the magnetometer can be tricky…

A good tutorial can be found here:
https://github.com/ptrbrtz/razor-9dof-ahrs/wiki/Tutorial

I used a simple calibration using min and max similar to the accelerometer's calibration with good results. The tricky part is that it is not easy to get the min and max values because you need to orientate the accelerometer exactly in the direction of the earth's magnetic field.

One solution can be to take lots and lots of points and hope to be lucky. That works, but I prefer to see the data in a graph in order to make sure I am at the right place. This was done in Excel (Matlab would be much easier).

Lets find magnetometer's mins and maxs



If you want to make a plot in 3D in excel, you will find more info here: http://www.doka.ch/Excel3Dscatterplot.htm

Magnetometer calibration

Visualisation of the data in 3D is not necessary, but useful to make sure nothing strange is happening (for example, magnetometer's data is messed up by your computer's screen magnetic field). Once min and max have been identified add calibration parameters as definitions

```
#define MAGN_X_MIN ((float) -1643)
#define MAGN_X_MAX ((float)  3251)
#define MAGN_Y_MIN ((float) -2071)
#define MAGN_Y_MAX ((float)  2729)
#define MAGN_Z_MIN ((float) -1550)
#define MAGN_Z_MAX ((float)  3198)
#define MAGN_X_OFFSET ((MAGN_X_MIN + MAGN_X_MAX) / 2.0f)
#define MAGN_Y_OFFSET ((MAGN_Y_MIN + MAGN_Y_MAX) / 2.0f)
#define MAGN_Z_OFFSET ((MAGN_Z_MIN + MAGN_Z_MAX) / 2.0f)
#define MAGN_X_SCALE (100.0f / (MAGN_X_MAX - MAGN_X_OFFSET)) // again magnetometer is calibrated to be set between -100 and 100
#define MAGN_Y_SCALE (100.0f / (MAGN_Y_MAX - MAGN_Y_OFFSET))
#define MAGN_Z_SCALE (100.0f / (MAGN_Z_MAX - MAGN_Z_OFFSET))
```

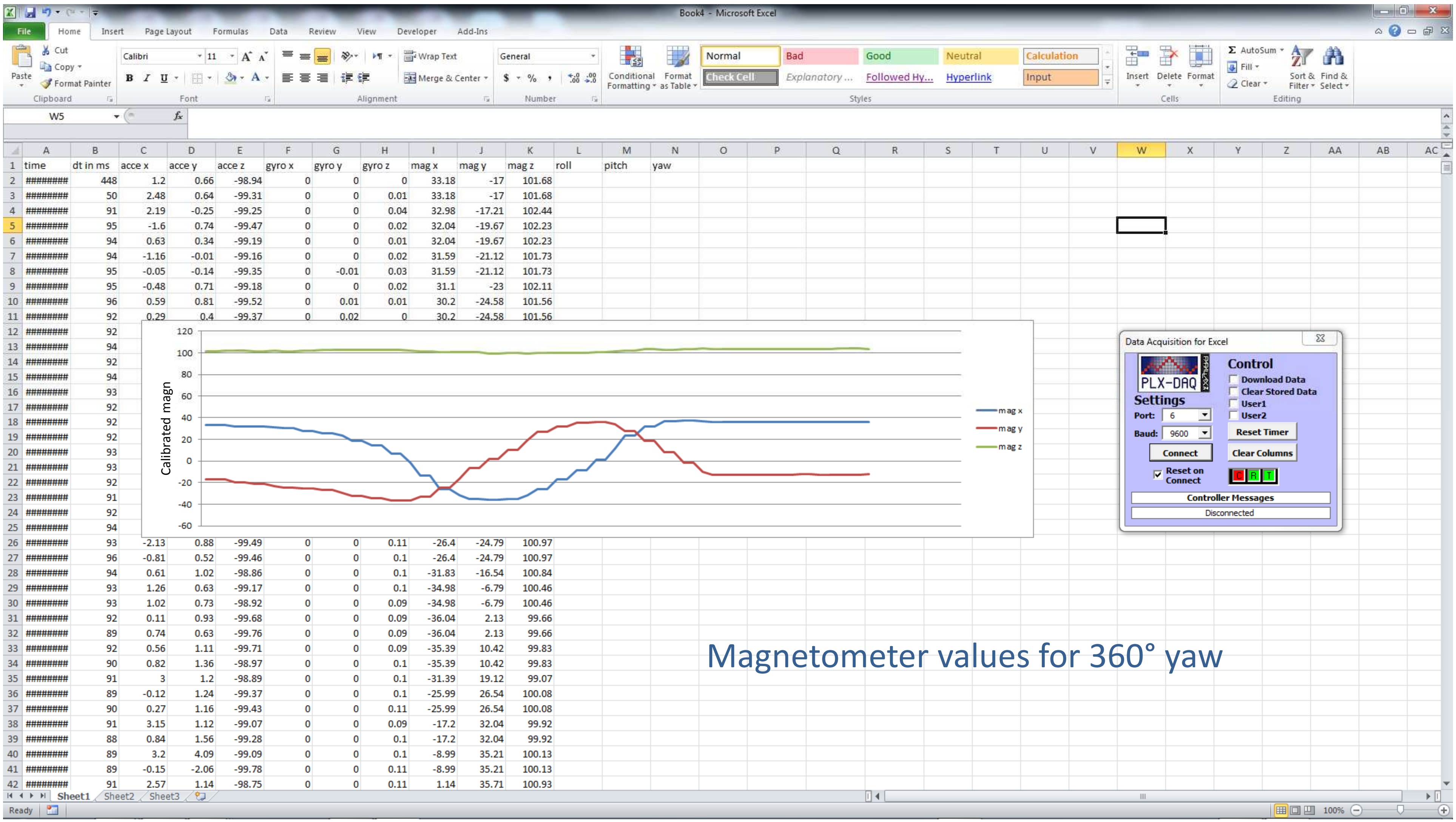
Change the magnetometer values from

```
M[1] = (compass.m.x);
M[2] = (compass.m.y);
M[3] = (compass.m.z);
```

To

```
M[1] = (compass.m.x - MAGN_X_OFFSET) * MAGN_X_SCALE * ACCEL_X_DIR; // note the directions of the magnetometer are the same as the
M[2] = (compass.m.y - MAGN_Y_OFFSET) * MAGN_Y_SCALE * ACCEL_Y_DIR; // accelerometer in my IMU
M[3] = (compass.m.z - MAGN_Z_OFFSET) * MAGN_Z_SCALE * ACCEL_Z_DIR;
```

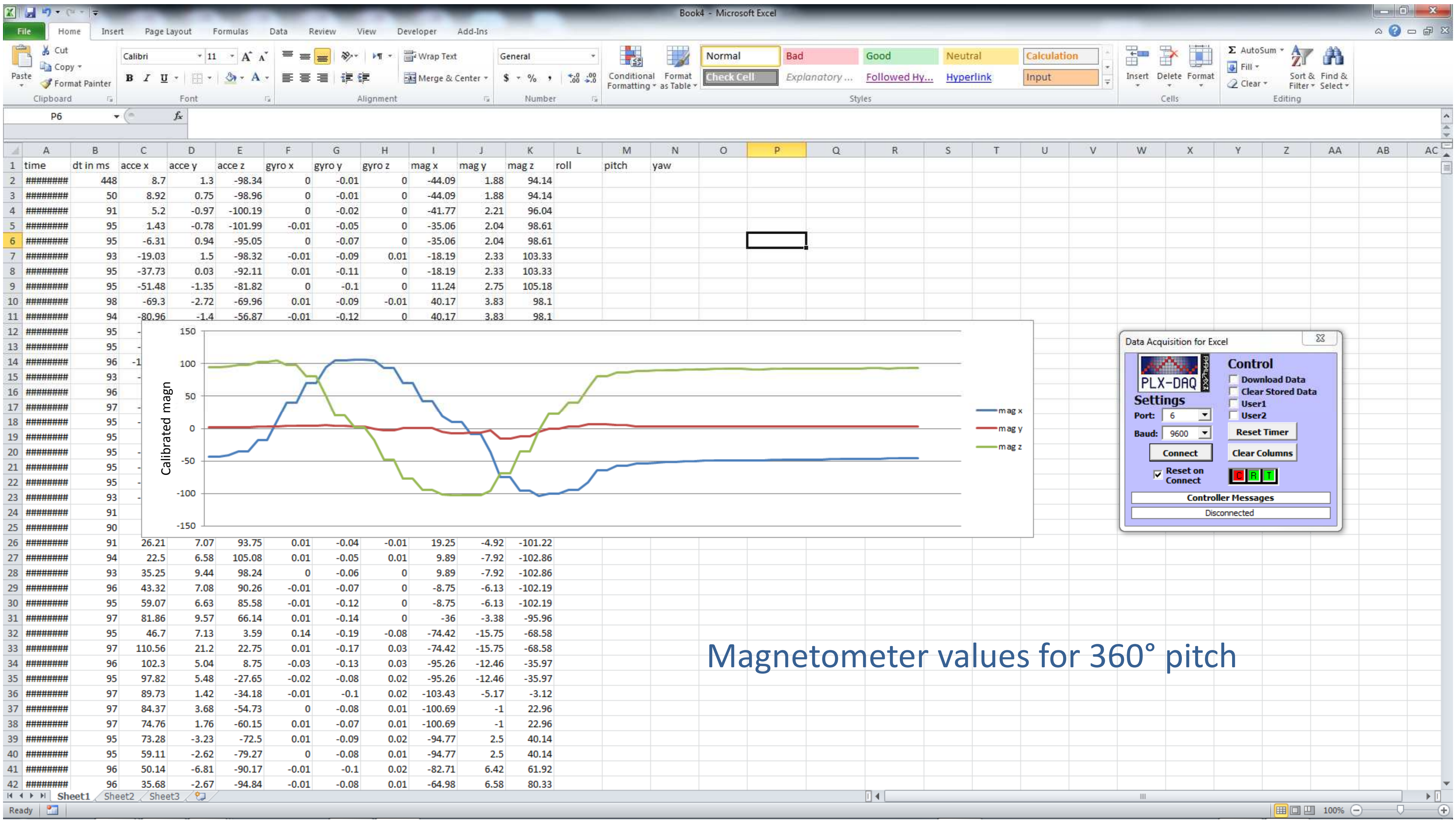
Magnetometer is now calibrated



	time	dt in ms	acce x	acce y	acce z	gyro x	gyro y	gyro z	mag x	mag y	mag z	roll	pitch	yaw
2	#########	448	1.2	0.66	-98.94	0	0	0	33.18	-17	101.68			
3	#########	50	2.48	0.64	-99.31	0	0	0.01	33.18	-17	101.68			
4	#########	91	2.19	-0.25	-99.25	0	0	0.04	32.98	-17.21	102.44			
5	#########	95	-1.6	0.74	-99.47	0	0	0.02	32.04	-19.67	102.23			
6	#########	94	0.63	0.34	-99.19	0	0	0.01	32.04	-19.67	102.23			
7	#########	94	-1.16	-0.01	-99.16	0	0	0.02	31.59	-21.12	101.73			
8	#########	95	-0.05	-0.14	-99.35	0	-0.01	0.03	31.59	-21.12	101.73			
9	#########	95	-0.48	0.71	-99.18	0	0	0.02	31.1	-23	102.11			
10	#########	96	0.59	0.81	-99.52	0	0.01	0.01	30.2	-24.58	101.56			
11	#########	92	0.29	0.4	-99.37	0	0.02	0	30.2	-24.58	101.56			
12	#########	92												
13	#########	94												
14	#########	92												
15	#########	94												
16	#########	93												
17	#########	92												
18	#########	92												
19	#########	92												
20	#########	93												
21	#########	93												
22	#########	92												
23	#########	91												
24	#########	92												
25	#########	94												
26	#########	93	-2.13	0.88	-99.49	0	0	0.11	-26.4	-24.79	100.97			
27	#########	96	-0.81	0.52	-99.46	0	0	0.1	-26.4	-24.79	100.97			
28	#########	94	0.61	1.02	-98.86	0	0	0.1	-31.83	-16.54	100.84			
29	#########	93	1.26	0.63	-99.17	0	0	0.1	-34.98	-6.79	100.46			
30	#########	93	1.02	0.73	-98.92	0	0	0.09	-34.98	-6.79	100.46			
31	#########	92	0.11	0.93	-99.68	0	0	0.09	-36.04	2.13	99.66			
32	#########	89	0.74	0.63	-99.76	0	0	0.09	-36.04	2.13	99.66			
33	#########	92	0.56	1.11	-99.71	0	0	0.09	-35.39	10.42	99.83			
34	#########	90	0.82	1.36	-98.97	0	0	0.1	-35.39	10.42	99.83			
35	#########	91	3	1.2	-98.89	0	0	0.1	-31.39	19.12	99.07			
36	#########	89	-0.12	1.24	-99.37	0	0	0.1	-25.99	26.54	100.08			
37	#########	90	0.27	1.16	-99.43	0	0	0.11	-25.99	26.54	100.08			
38	#########	91	3.15	1.12	-99.07	0	0	0.09	-17.2	32.04	99.92			
39	#########	88	0.84	1.56	-99.28	0	0	0.1	-17.2	32.04	99.92			
40	#########	89	3.2	4.09	-99.09	0	0	0.1	-8.99	35.21	100.13			
41	#########	89	-0.15	-2.06	-99.78	0	0	0.11	-8.99	35.21	100.13			
42	#########	91	2.57	1.14	-98.75	0	0	0.11	1.14	35.71	100.93			

Magnetometer values for 360° yaw

Magnetometer is now calibrated



	time	dt in ms	acce x	acce y	acce z	gyro x	gyro y	gyro z	mag x	mag y	mag z	roll	pitch	yaw
2	#########	448	8.7	1.3	-98.34	0	-0.01	0	-44.09	1.88	94.14			
3	#########	50	8.92	0.75	-98.96	0	-0.01	0	-44.09	1.88	94.14			
4	#########	91	5.2	-0.97	-100.19	0	-0.02	0	-41.77	2.21	96.04			
5	#########	95	1.43	-0.78	-101.99	-0.01	-0.05	0	-35.06	2.04	98.61			
6	#########	95	-6.31	0.94	-95.05	0	-0.07	0	-35.06	2.04	98.61			
7	#########	93	-19.03	1.5	-98.32	-0.01	-0.09	0.01	-18.19	2.33	103.33			
8	#########	95	-37.73	0.03	-92.11	0.01	-0.11	0	-18.19	2.33	103.33			
9	#########	95	-51.48	-1.35	-81.82	0	-0.1	0	11.24	2.75	105.18			
10	#########	98	-69.3	-2.72	-69.96	0.01	-0.09	-0.01	40.17	3.83	98.1			
11	#########	94	-80.96	-1.4	-56.87	-0.01	-0.12	0	40.17	3.83	98.1			
12	#########	95												
13	#########	95												
14	#########	96												
15	#########	93												
16	#########	96												
17	#########	97												
18	#########	95												
19	#########	95												
20	#########	95												
21	#########	95												
22	#########	95												
23	#########	93												
24	#########	91												
25	#########	90												
26	#########	91	26.21	7.07	93.75	0.01	-0.04	-0.01	19.25	-4.92	-101.22			
27	#########	94	22.5	6.58	105.08	0.01	-0.05	0.01	9.89	-7.92	-102.86			
28	#########	93	35.25	9.44	98.24	0	-0.06	0	9.89	-7.92	-102.86			
29	#########	96	43.32	7.08	90.26	-0.01	-0.07	0	-8.75	-6.13	-102.19			
30	#########	95	59.07	6.63	85.58	-0.01	-0.12	0	-8.75	-6.13	-102.19			
31	#########	97	81.86	9.57	66.14	0.01	-0.14	0	-36	-3.38	-95.96			
32	#########	95	46.7	7.13	3.59	0.14	-0.19	-0.08	-74.42	-15.75	-68.58			
33	#########	97	110.56	21.2	22.75	0.01	-0.17	0.03	-74.42	-15.75	-68.58			
34	#########	96	102.3	5.04	8.75	-0.03	-0.13	0.03	-95.26	-12.46	-35.97			
35	#########	95	97.82	5.48	-27.65	-0.02	-0.08	0.02	-95.26	-12.46	-35.97			
36	#########	97	89.73	1.42	-34.18	-0.01	-0.1	0.02	-103.43	-5.17	-3.12			
37	#########	97	84.37	3.68	-54.73	0	-0.08	0.01	-100.69	-1	22.96			
38	#########	97	74.76	1.76	-60.15	0.01	-0.07	0.01	-100.69	-1	22.96			
39	#########	95	73.28	-3.23	-72.5	0.01	-0.09	0.02	-94.77	2.5	40.14			
40	#########	95	59.11	-2.62	-79.27	0	-0.08	0.01	-94.77	2.5	40.14			
41	#########	96	50.14	-6.81	-90.17	-0.01	-0.1	0.02	-82.71	6.42	61.92			
42	#########	96	35.68	-2.67	-94.84	-0.01	-0.08	0.01	-64.98	6.58	80.33			

Magnetometer values for 360° pitch

All 9 sensors a now calibrated.

We need to use them to calculate roll pitch and yaw.
I am going to use a complementary filter.

More info about it and why we would use a complementary filter found here:
https://b94be14129454da9cf7f056f5f8b89a9b17da0be.googledrive.com/host/0B0ZbiLZrqVa6Y2d3UjFVWDhNZms/filter.pdf

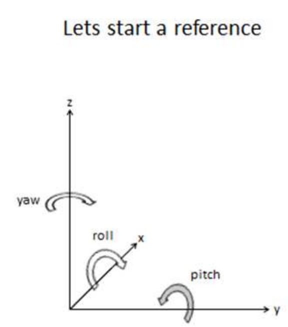
AHRS

AHRS

Step 1 Calculate roll pitch yaw using accelerometer and magnetometer

More definitions

- roll is defined in the [-180, 180] range and pitch is defined in the [-90, 90] range, therefore you can not use same equation to calculate both roll and pitch as it is shown in many tutorials



few equations

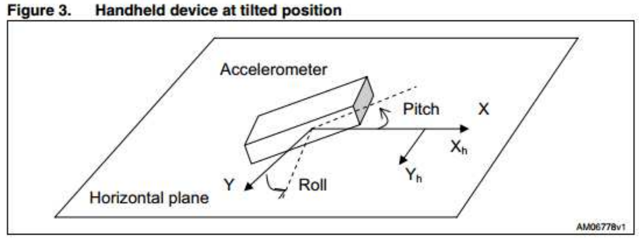
a = accelerometer g = gyro m = magnetometer

$$roll_a = \text{atan2}(a_y, a_z)$$

$$pitch_a = \text{atan}\left(\frac{a_x}{\sqrt{a_y^2 + a_z^2}}\right)$$

$$yaw_m = atan2(Y_h, X_h\ )$$

Figure 3. Handheld device at tilted position



Equation 2

$$X_h = X_M \cos Pitch + Z_M \sin Pitch$$
$$Y_h = X_M \sin Roll \sin Pitch + Y_M \cos Roll - Z_M \sin Roll \cos Pitch$$

Where, X_M, Y_M, and Z_M are magnetic sensor measurements.

https://www.pololu.com/file/download/...?file_id=0J434

First calculate roll pitch and yaw

Add 2 definitions

```
#define TO_RAD(x) (x * 0.01745329252)  // *pi/180
#define TO_DEG(x) (x * 57.2957795131)  // *180/pi
```

Then after data conversion in loop() function add

```
roll_A  = TO_DEG( atan2(A[2], A[3]));

pitch_A = TO_DEG( atan(A[1]/sqrt(A[2]*A[2]+A[3]*A[3])));

Xh      = M[1] * cos(TO_RAD(pitch)) + M[3] * sin(TO_RAD(pitch));
Yh      = M[1] * sin(TO_RAD(roll)) * sin(TO_RAD(pitch)) + M[2] * cos(TO_RAD(roll)) - M[3] * sin(TO_RAD(roll)) * cos(TO_RAD(pitch));
yaw_M   = TO_DEG(atan2(Yh,Xh));
```

This is a good beginning but these equations assumes that the IMU is flat (roll = pitch = 0) and pointing in the right direction (yaw = 0)

These equations take into account that the IMU might not be exactly flat and pointing in the right direction:

```
roll_A  = TO_DEG( atan2(A[2], A[3])) - roll_init ;

pitch_A = TO_DEG( atan(A[1]/sqrt(A[2]*A[2]+A[3]*A[3])) ) - pitch_init ;

Xh      = M[1] * cos(TO_RAD(pitch)) + M[3] * sin(TO_RAD(pitch));
Yh      = M[1] * sin(TO_RAD(roll)) * sin(TO_RAD(pitch)) + M[2] * cos(TO_RAD(roll)) - M[3] * sin(TO_RAD(roll)) * cos(TO_RAD(pitch));
yaw_M   = TO_DEG(atan2(Yh,Xh)) - Heading ;

// roll_init, pitch_init and Heading are the original roll pitch and yaw when the code started
```

But the equations gives results outside of the [-180, 180] or [-90, 90] ranges

So I created a function to set all angles in the right ranges:

```
float Correction (int angle, float aa)  // corect angles between -90 to 90 or -180 to 180
{
  if       (aa >  angle) { aa -= angle*2; }
  else if (aa < -angle) { aa += angle*2; }
  else                  { aa = aa;   }
  return aa;
}
```

The Final roll pitch yaw corected equations are:

```
roll_A  = Correction( 180 , TO_DEG( atan2(A[2], A[3])) - roll_init );

pitch_A = Correction( 90 , TO_DEG( atan(A[1]/sqrt(A[2]*A[2]+A[3]*A[3])) ) - pitch_init );

Xh      = M[1] * cos(TO_RAD(pitch)) + M[3] * sin(TO_RAD(pitch));
Yh      = M[1] * sin(TO_RAD(roll)) * sin(TO_RAD(pitch)) + M[2] * cos(TO_RAD(roll)) - M[3] * sin(TO_RAD(roll)) * cos(TO_RAD(pitch));
yaw_M   = Correction( 180 , TO_DEG(atan2(Yh,Xh)) - Heading );
```

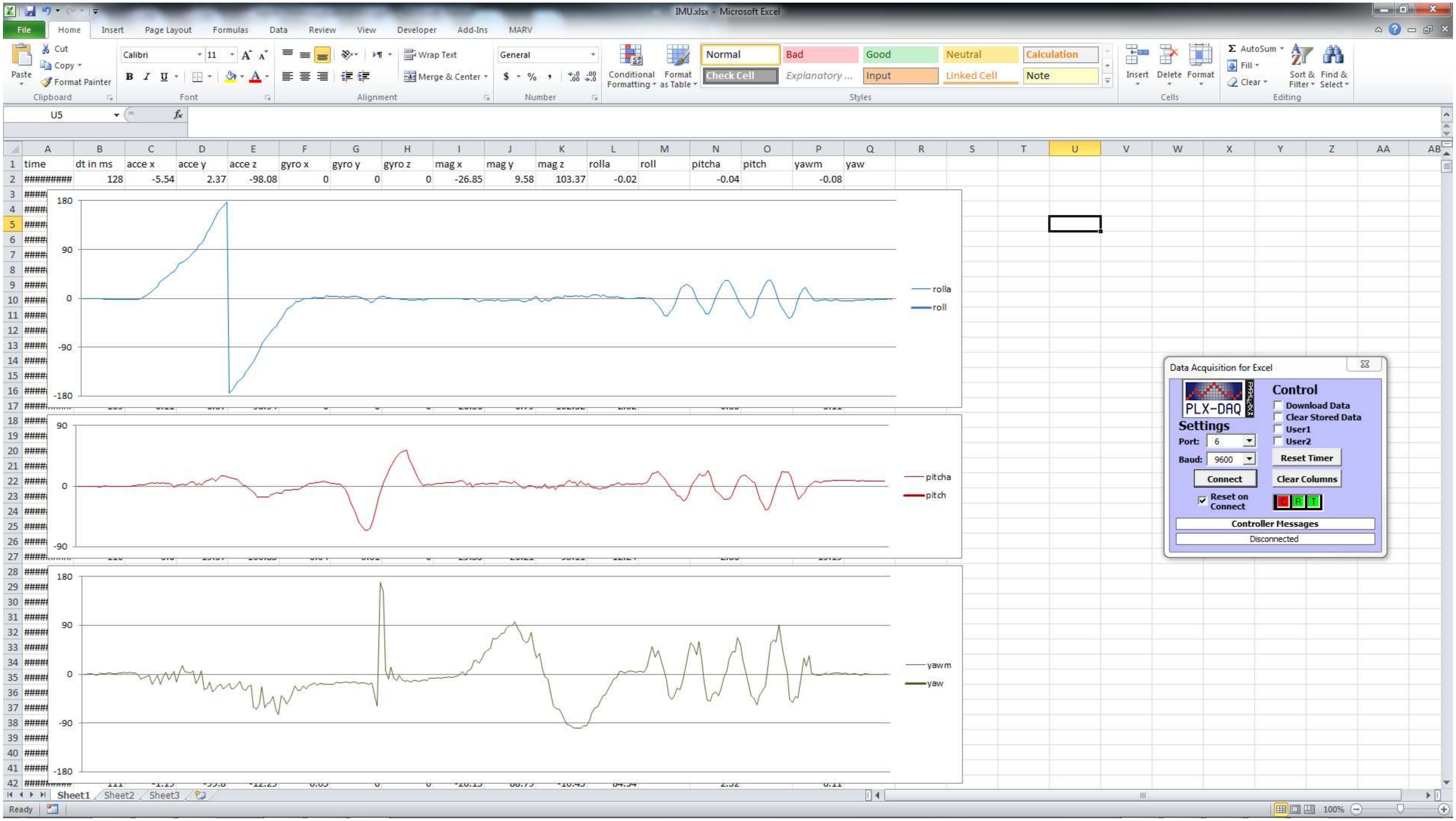
Add the data into Excel

Just to check that everything is OK

Replace print block by this one:

```
Serial.print("DATA,TIME,");
Serial.print(dt);      Serial.print(",");
Serial.print(A[1]);    Serial.print(","); Serial.print(A[2]); Serial.print(","); Serial.print (A[3]); Serial.print(",");
Serial.print(G[1]);    Serial.print(","); Serial.print(G[2]); Serial.print(","); Serial.print (G[3]); Serial.print(",");
Serial.print(M[1]);    Serial.print(","); Serial.print(M[2]); Serial.print(","); Serial.print (M[3]); Serial.print(",");
Serial.print(roll_A);  Serial.print(","); // roll calculated using accelerometer’s data
Serial.print(roll);    Serial.print(","); // final roll, will be used in step 2, not used for now
Serial.print(pitch_A); Serial.print(","); // pitch calculated using accelerometer’s data
Serial.print(pitch);   Serial.print(","); // final pitch, will be used in step 2 , not used for now
Serial.print(yaw_M);   Serial.print(","); // yaw calculated using magnetometer’s data
Serial.print(yaw);     Serial.print(","); // final yaw, will be used in step 2 , not used for now
Serial.println();
```

Add the data into Excel



AHRS

Step 2 Calculate roll pitch yaw using gyro

Using gyro

If IMU is flat it is simple

```
roll_G  = roll +  dt * G[1] ;
pitch_G = pitch + dt * G[2] ;
yaw_G   = yaw +   dt * G[3] ;
```

If IMU is not flat it is not simple, we must correct the angular rates of the gyro to Euler's angular rates first

The resulting transformation matrix for converting body-frame angular rates to Euler angular rates is given by

$$D(\phi,\theta,\psi) = \begin{pmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi)/\cos(\theta) & \cos(\phi)/\cos(\theta) \end{pmatrix}.$$

Source: http://www.chrobotics.com/library/understanding-euler-angles

Final gyro's equations:

```
roll_G  = roll + dt * (G[1]+sin(TO_RAD(roll))*tan(TO_RAD(pitch))*G[2]+cos(TO_RAD(roll))*tan(TO_RAD(pitch))*G[3]);
pitch_G = pitch + dt * (cos(TO_RAD(roll))*G[2]-sin(TO_RAD(roll))*G[3]);
yaw_G   = yaw + dt * (sin(TO_RAD(roll))/cos(TO_RAD(pitch))*G[2]+cos(TO_RAD(roll))/cos(TO_RAD(pitch))*G[3] );
```

AHRS

Step 3 Combine step 1 and 2

Complementary filter

A Closer Look at the Angle Complementary Filter

```
angle = (0.98)*(angle + gyro*dt) + (0.02)*(x_acc);
```

Integration.

Low-pass portion acting on the accelerometer.

Something resembling a high-pass filter on the integrated gyro angle estimate. It will have approximately the same time constant as the low-pass filter.

If this filter were running in a loop that executes 100 times per second, the time constant for both the low-pass and the high-pass filter would be:

$$\tau = \frac{a \cdot dt}{1-a} = \frac{0.98 \cdot 0.01\,\mathrm{sec}}{0.02} = 0.49\,\mathrm{sec}$$

This defines where the boundary between trusting the gyroscope and trusting the accelerometer is. For time periods shorter than half a second, the gyroscope integration takes precedence and the noisy horizontal accelerations are filtered out. For time periods longer than half a second, the accelerometer average is given more weighting than the gyroscope, which may have drifted by this point.

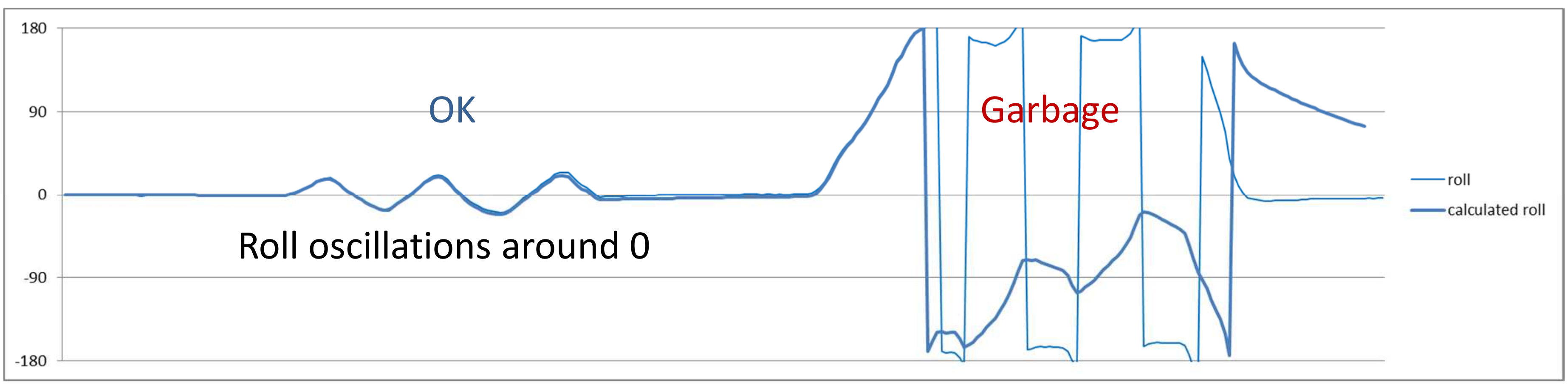
Source: https://b94be14129454da9cf7f056f5f8b89a9b17da0be.googledrive.com/host/0B0ZbiLZrqVa6Y2d3UjFVWDhNZms/filter.pdf

Complementary filter

If roll pitch and yaw are small, then it is easy to code:

```
roll    = 0.02 * roll_A  + 0.98 * roll_G  ;
pitch   = 0.02 * pitch_A + 0.98 * pitch_G ;
yaw     = 0.02 * yaw_M   + 0.98 * yaw_G   ;
```

But this does not work when angles are large



Roll oscillations around 180
IMU is flipped

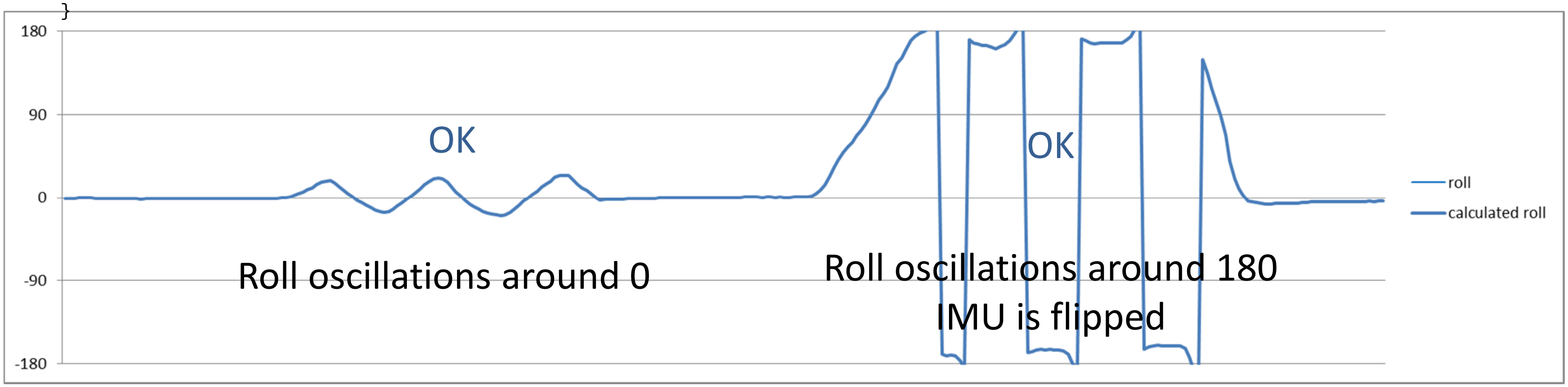
Complementary filter

If roll pitch and yaw are large we have to take into account that the arithmetic mean is not the same as the mean used on angles

```
roll    = CompFilter( 180, roll_A, roll_G );
pitch   = CompFilter( 90, pitch_A, pitch_G );
yaw     = CompFilter( 180, yaw_M, yaw_G );
```

I created a function that convert the angles before calculating the mean:

```
float CompFilter (int angle, float acce, float gyro)
{
  if(abs(acce - gyro) > angle)  {
    if(gyro < -angle) gyro += angle*2;
    if(gyro >  angle) gyro -= angle*2;
  }
  return 0.02 * acce + 0.98 * gyro;
}
```



Complementary filter

The `CompFilter` function could have been avoided using the mean of circular quantities like this

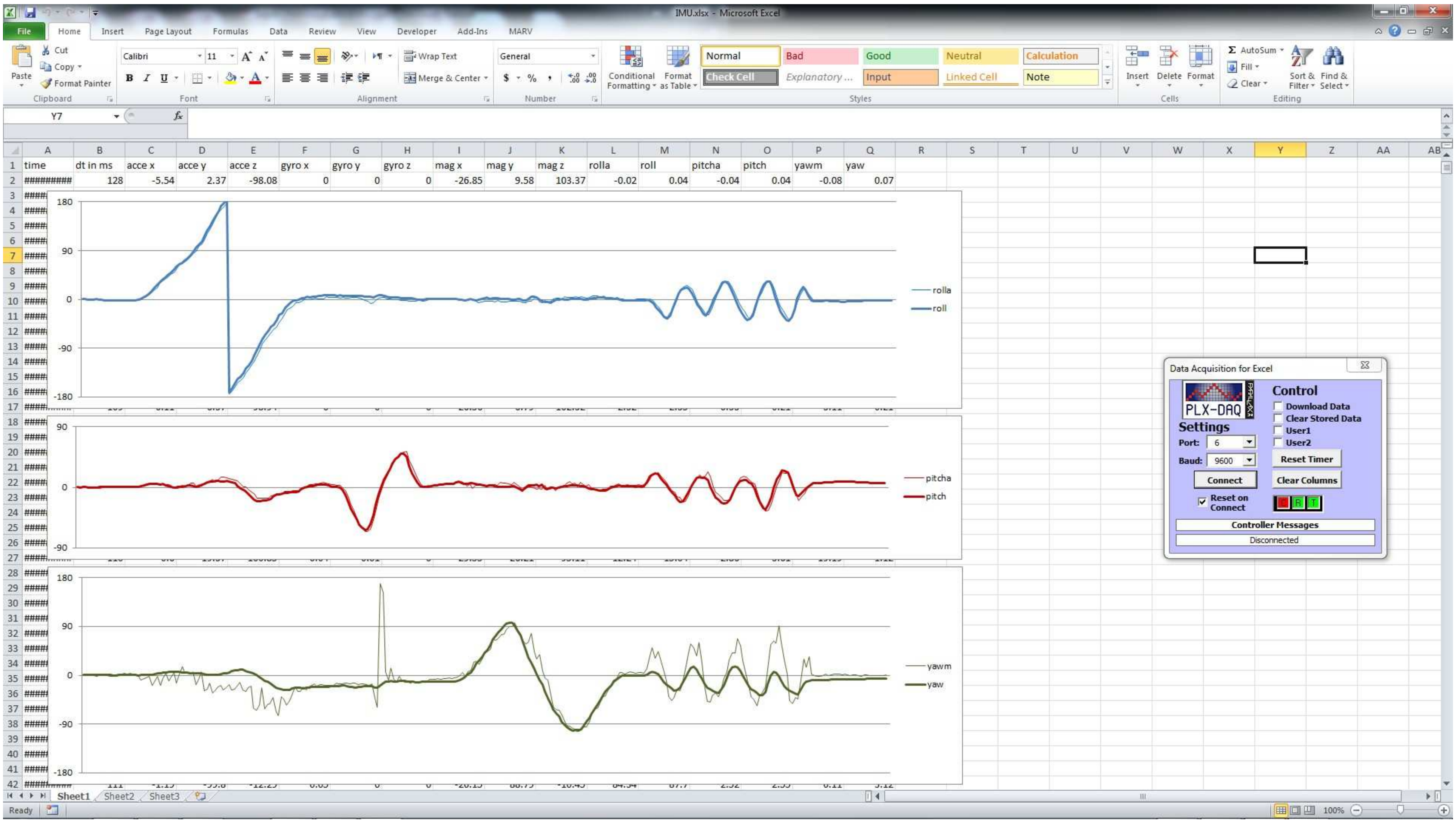
```
roll    = atan2(0.02*sin(roll_A) +0.98*sin(roll_G)  , 0.02*cos(roll_A) +0.98*cos(roll_G)) ;
pitch   = atan2(0.02*sin(pitch_A)+0.98*sin(pitch_G) , 0.02*cos(pitch_A)+0.98*cos(pitch_G));
yaw     = atan2(0.02*sin(yaw_M)  +0.98*sin(yaw_G)   , 0.02*cos(yaw_M)  +0.98*cos(yaw_G))  ;
```

This code is more elegant, does not require the `CompFilter` function but it is slightly slower.

More info here: http://en.wikipedia.org/wiki/Mean_of_circular_quantities

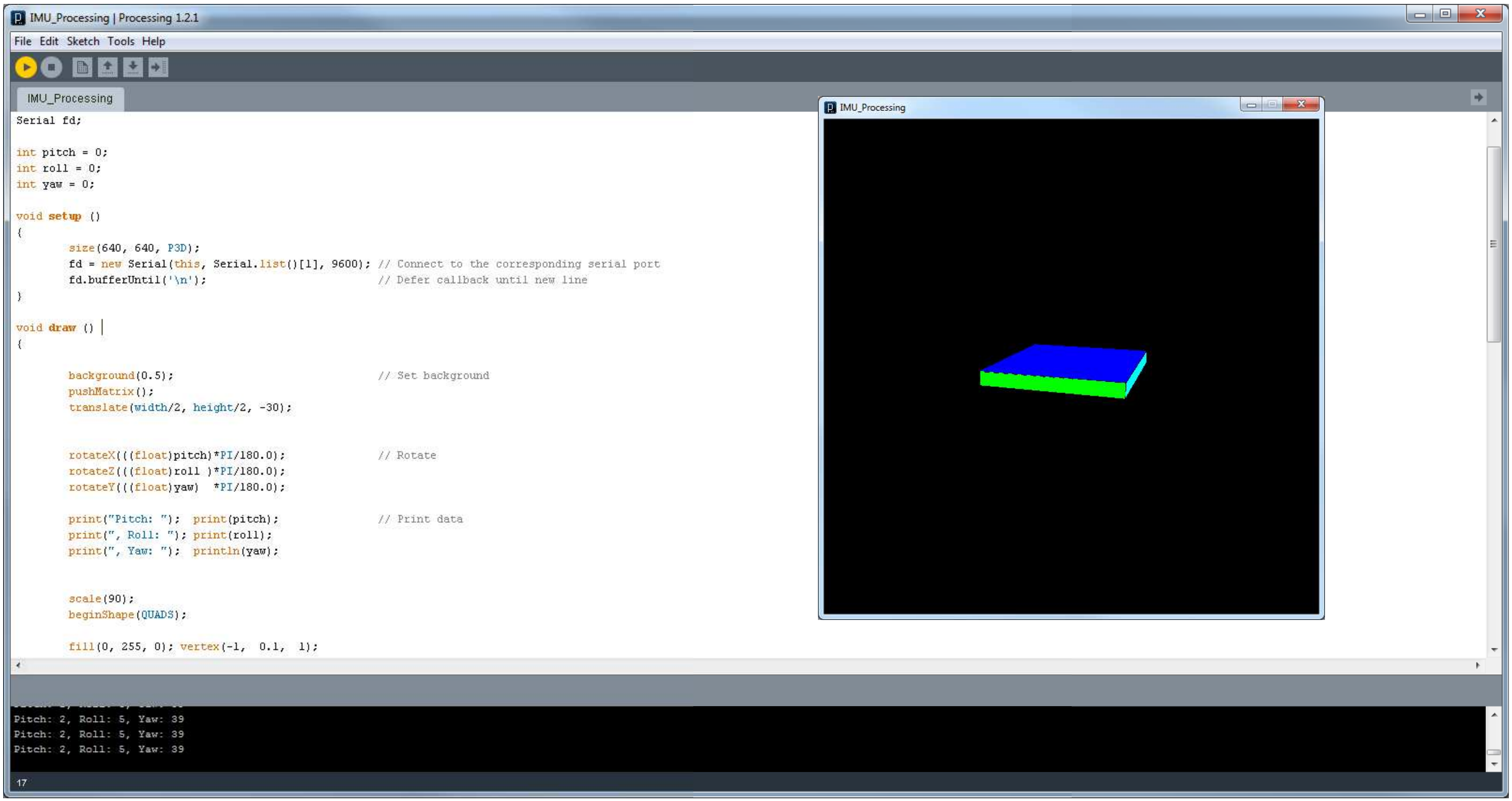
IMU and AHRS are working

Data visualization Excel



Data visualization Processing

Looking at the data in Excel is good for calibration, a processing sketch is better for visualization



Final note

Simpler algorithm

You might not need a full AHRS algorithm, for example in a balancing robot or hovering quadcopter where the IMU is flat

$$\sin(x) = x$$

if x is small, typically $x < 20°$

This means that you do not have to use any trigonometry, this lead to much faster algorithms

Also, you do not have to convert the gyro’s body frame to Euler’s angles, because they are almost the same. This lead to even faster algorithms

Example balancing robot

Declarations and Setup function are the same as previously

```
void loop() {

  // GET DATA ----------------------------------------------------------------------
  compass.read();
  gyro.read();
  dt = millis()-t;
  t  = millis();

  // CALIBRATE DATA ----------------------------------------------------------------
  A[2] = (compass.a.y - ACCEL_Y_OFFSET) * ACCEL_Y_SCALE * ACCEL_Y_DIR; // Gather only necessary data
  A[3] = (compass.a.z - ACCEL_Z_OFFSET) * ACCEL_Z_SCALE * ACCEL_Z_DIR;
  G[1] = (gyro.g.x    - GYRO_OFFSET_X)  * GYRO_SCALE    * GYRO_X_DIR ;

  // CALCULATE ROLL ONLY -----------------------------------------------------------
  roll_A  = TO_DEG(A[2]/A[3]) - roll_init;    // no trigonometry here, accurate for angles bellow 30 degrees
  roll_G  = roll + dt * G[1];                 // no body frame to Euler conversion here
  roll    = 0.04 * roll_A + 0.96 * roll_G;    // simple complementary filter here

  // PID ---------------------------------------------------------------------------
  error          =  Setpoint - roll;
  integral       =  integral + error * dt;
  derivative     =  (error - previous_error) / dt;
  previous_error =  error;
  Output         =  Kp*error + Ki*integral + Kd*derivative;

  // MOTORS ------------------------------------------------------------------------
    motorGo(0, Output);  // function that turn on the motor’s to a specific speed
    motorGo(1, Output);
}
```

The simplifications make the code short and simple

GL HF