# AHRS for Adafruit's 9-DOF, 10-DOF, LSM9DS0 Breakouts

Created by Kevin Townsend

# Guide Contents

# Introduction

[AHRS](http://adafru.it/ddg) (http://adafru.it/ddg) is an acronym for **Attitude and Heading Reference System**, a system generally used for aircraft of any sort to determine *heading, pitch, roll, altitude* etc.

A basic IMU (Intertial Measurement Unit) generally provides raw sensor data, whereas an AHRS takes this data one step further, converting it into heading or direction in degrees, converting the raw altitude data into standard units like feet or meters, etc.

To help you get started designing your own AHRS system, or just tp help convert raw sensor data into useful numbers that you can relate to the real world, we've created a sample AHRS sketch for the [Adafruit 10-DOF IMU](http://adafru.it/1604) (http://adafru.it/1604), [Adafruit 9-DOF IMU](http://adafru.it/1714) (http://adafru.it/1714), and [Adafruit_LSM9DS0](http://adafru.it/dNS) (http://adafru.it/dNS) breakouts. The Adafruit 10-DOF breakout is required for a real AHRS system -- only the 10-DOF breakout incorporates a barometric pressure sensor capable of measuring altitude -- but we also reference the 9-DOF and LSM9DS0 breakout since this code can be used with either breakout for orientation calculations.
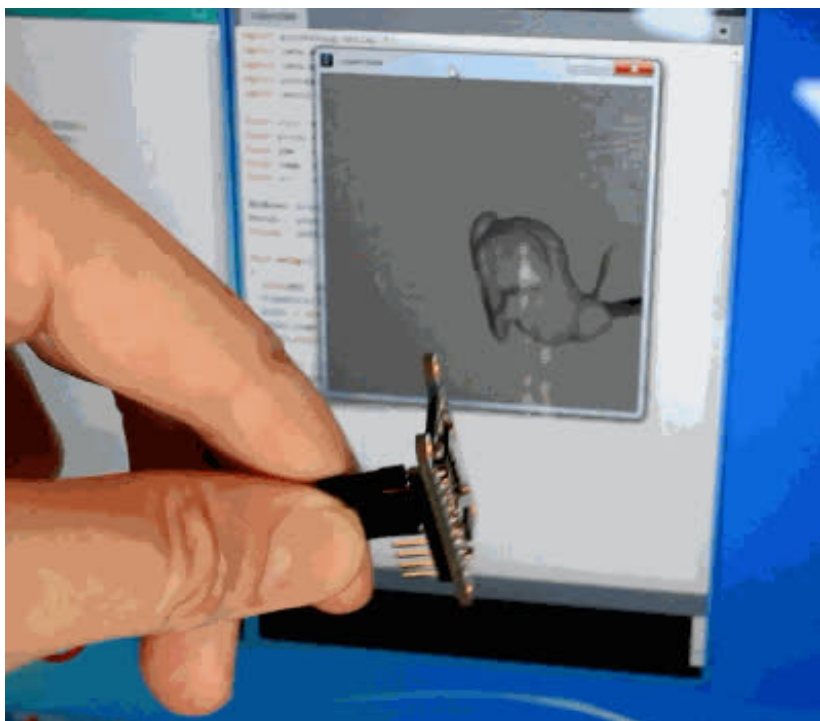


# Related Links

- [Adafruit's 10-DOF Breakout Learning Guide](http://adafru.it/ddh) (http://adafru.it/ddh)

- [Adafruit's 9-DOF Breakout Learning Guide](http://adafru.it/ddi) (http://adafru.it/ddi)
- [Adafruit's LSM9DS0 Breakout Learning Guide](http://adafru.it/dNS) (http://adafru.it/dNS)

# Installing the Software

The current example Arduino sketches are available on Github as part of the Adafruit AHRS libray (http://adafru.it/dNO).

# Downloading the AHRS Sample Code

To install the drivers and AHRS sketch for your breakout, you simply need to install the **Adafruit_AHRS** library and it's dependencies on your PC (the exact dependencies you choose will depend on which IMU board you are currently using).

We've put together a tutorial showing you everything you need to do to install dependent libraries if you're just getting started with your breakout. Follow the guide below to install the right libraries depending on which board you are using:

- Adafruit 10-DOF IMU Installation Guide (http://adafru.it/d8p)
- Adafruit 9-DOF IMU Installation Guide (http://adafru.it/ddj)
- Adafruit LSM9DS0 IMU Installation Guide (http://adafru.it/dNP)

In addition to the dependencies above, make sure to install the **Adafruit_AHRS** library. You can find a link to download the library below:
Download Adafruit_AHRS library
http://adafru.it/dNQ
If you aren't familiar with how to install an Arduino library, check out this guide on installing libraries (http://adafru.it/dNR).

# Loading the Sample Sketch

Once the repository has been added to your libraries folder ("/libraries/Adafruit_AHRS/", etc.) you should be able to access the AHRS examples from 'File > Examples > Adafruit_AHRS' in the Arduino IDE. There should be an **ahrs_9dof**, **ahrs_10dof**, and **ahrs_lsm9ds0** exampe.

Pick the appropriate example for your board, for example the **ahrs_10dof** example should be used with the **10-DOF** board:

File Edit Sketch Tools Help

ahrs

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_LSM303_U.h>
#include <Adafruit_BMP085_U.h>
#include <Adafruit_10DOF.h>

/* Assign a unique ID to the sensors */
Adafruit_10DOF                   dof   = Adafruit_10DOF();
Adafruit_LSM303_Accel_Unified accel = Adafruit_LSM303_Accel_Unified(30301);
Adafruit_LSM303_Mag_Unified    mag   = Adafruit_LSM303_Mag_Unified(30302);
Adafruit_BMP085_Unified         bmp   = Adafruit_BMP085_Unified(18001);

/* Update this with the correct SLP for accurate altitude measurements */
float seaLevelPressure = SENSORS_PRESSURE_SEALEVELHPA;

/****************************************************************************/
/*!
    @brief  Initialises all the sensors used by this example
*/
/****************************************************************************/
void initSensors()
{
  if(!accel.begin())
  {
    /* There was a problem detecting the LSM303 ... check your connections */
    Serial.println(F("Ooops, no LSM303 detected ... Check your wiring!"));
    while(1);
  }
  if(!mag.begin())
  {
```

Done uploading.

```
Binary sketch size: 14,410 bytes (of a 32,256 byte maximum)
Binary sketch size: 14,410 bytes (of a 32,256 byte maximum)
```

2                                                          Arduino Uno on COM12

If you compile the sketch and then program your Uno with the code, you should be able to open up the **Serial Monitor** (Tools > Serial Monitor), set the baud rate to 115200, and see the following output:

```
COM12                                                    [─] [□] [✕]

┌──────────────────────────────────────────────────────┐ ┌──────┐
│                                                        │ │ Send │
└──────────────────────────────────────────────────────┘ └──────┘

Adafruit 10 DOF Pitch/Roll/Heading Example              ▲

Orientation: 18.10 78.29 31.70
Alt: -45.80
Temp: 25.28
Orientation: 20.07 78.02 30.98
Alt: -46.40
Temp: 25.28
Orientation: 19.44 78.29 31.95
Alt: -45.89
Temp: 25.29                                             ▼

☐ Autoscroll              [No line ending ▾] [115200 baud ▾]
```
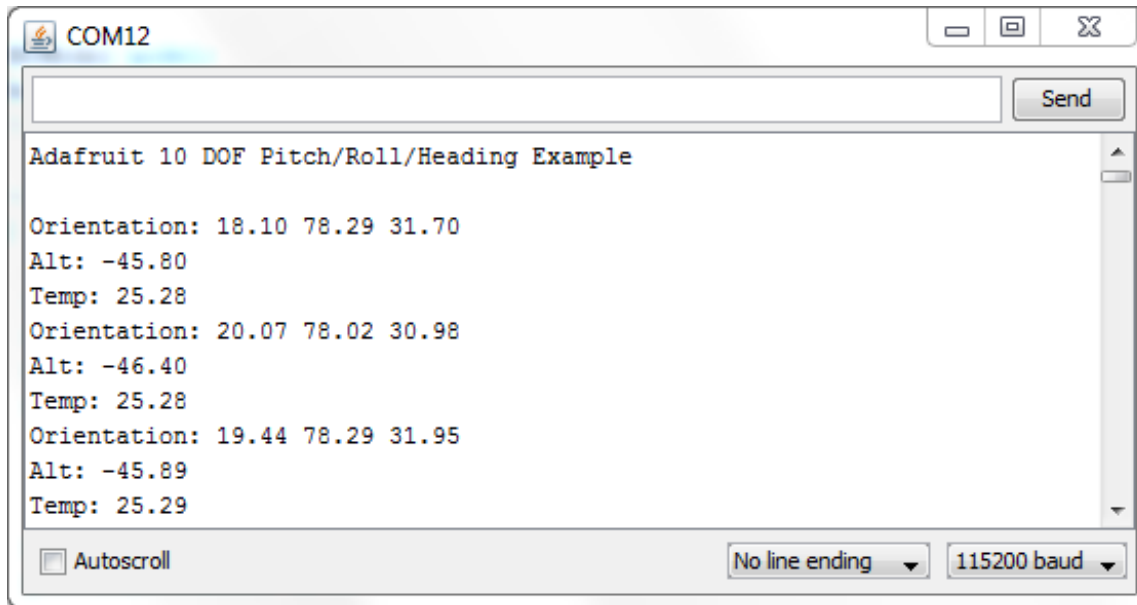
This raw data shows the main orientation data, consisting of **'roll'**, **'pitch'** and **'heading'** (or 'yaw) in degrees, followed by the current altitude and temperature if you are are using the 10-DOF breakout (the 9-DOF and LSM9DSO breakouts can't measure altitude).

Now close the Serial Monitor and read on to the next page to do something useful with this data!

# Using AHRS Data

This simple example uses uncalibrated data and doesn't integrate the on board gyroscope, but is fast and easy to use. For a more advanced example, see Sensor Fusion Algorithms later in this guide.

The AHRS example sketchs reads raw data from the board's accelerometer/magnetometer and converts the raw data into easy to understand **Euler angles**.

It does this with a bit of trigonometry (you remember high school math, right!?), but to save you from dusting off the textbooks or wading through endless application notes, we've wrapped up all of the calculation to convert raw accelerometer and magnetometer data to degrees in a convenient helper function:

bool getOrientation(sensors_vec_t *orientation)

The AHRS sketch creates an **Adafruit_Simple_AHRS** object which takes an accelerometer and magnetometer sensor as input to its constructor. You can actually pass any accelerometer or magnetometer object which supports the Adafruit unified sensor library (http://adafru.it/dGB) in to the AHRS algorithm, and the examples use the 9-DOF, 10-DOF, and LSM9DS0 sensors.

Once the simple AHRS object is created the**getOrientation** function is called to retrieve the current orientation. Internally the function will read raw data from the accelerometer and magnetomber, converts the data to Euler angles, and spits the three-dimensional orientation data out. If you're interested in the math, you can have a look at the source code on github (http://adafru.it/dNT), but for now we'll just treat it as a mathemagical black box:

# Euler Angles

Euler angles (http://adafru.it/ddl) describe orientation (in degrees) around a single reference point in three-dimensional space.

Various names are employed for the three angles, but the most common terminology with aircraft is Roll (x), Pitch (y) and Yaw (z).

The illustration below from the Wikipedia article on Euler angles should illustrate the concept clearly. You normally have both positive and negative angles (-180° to 180°) depending on the direction the airplane is tilted, with 0° in every direction corresponding to

the airplane being perfectly aligned with each axis:

Pitch Axis

Roll Axis

Yaw Axis
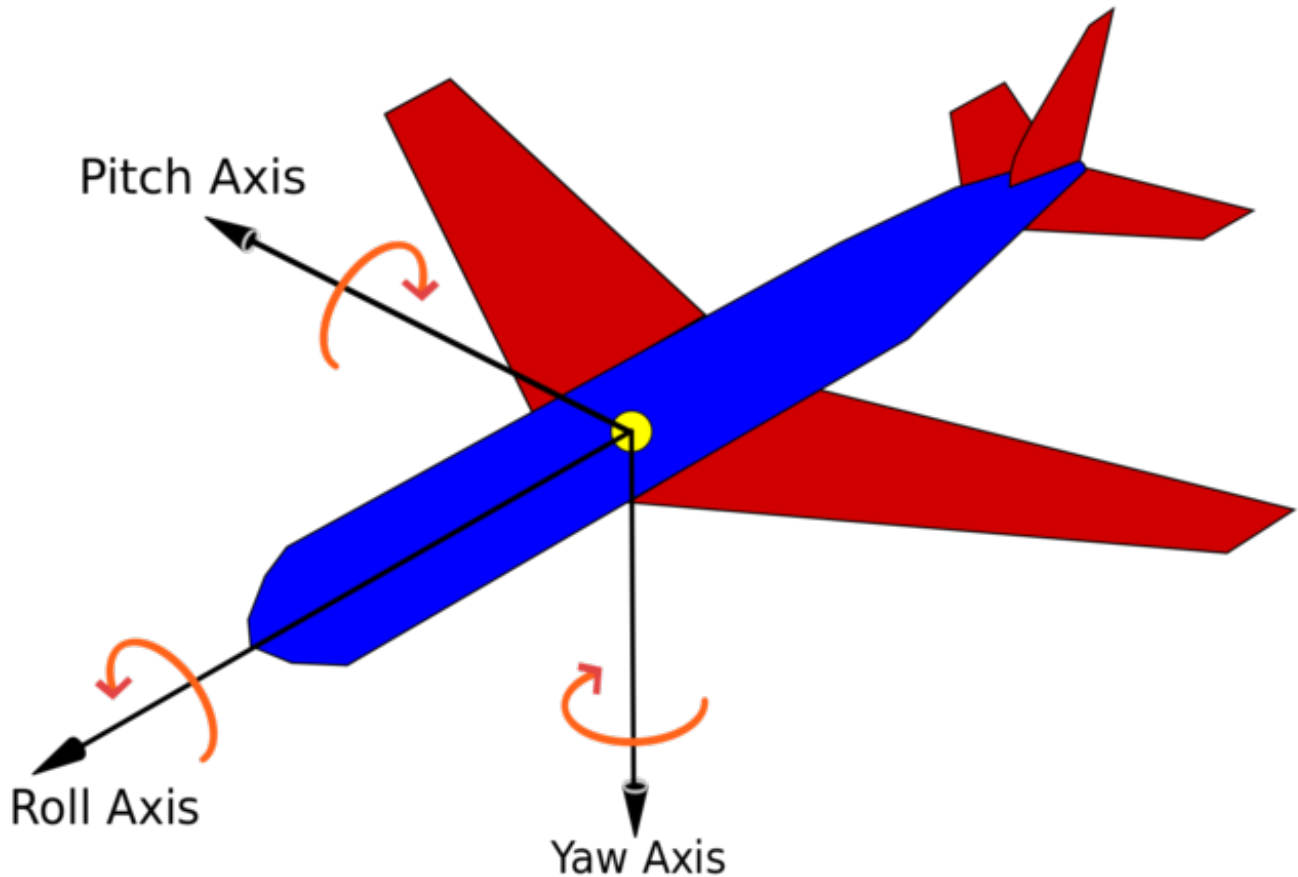
If you run the AHRS sketch, you should see something similar to the following in the Serial Monitor, where the three values after 'orientation' are the Euler angles:
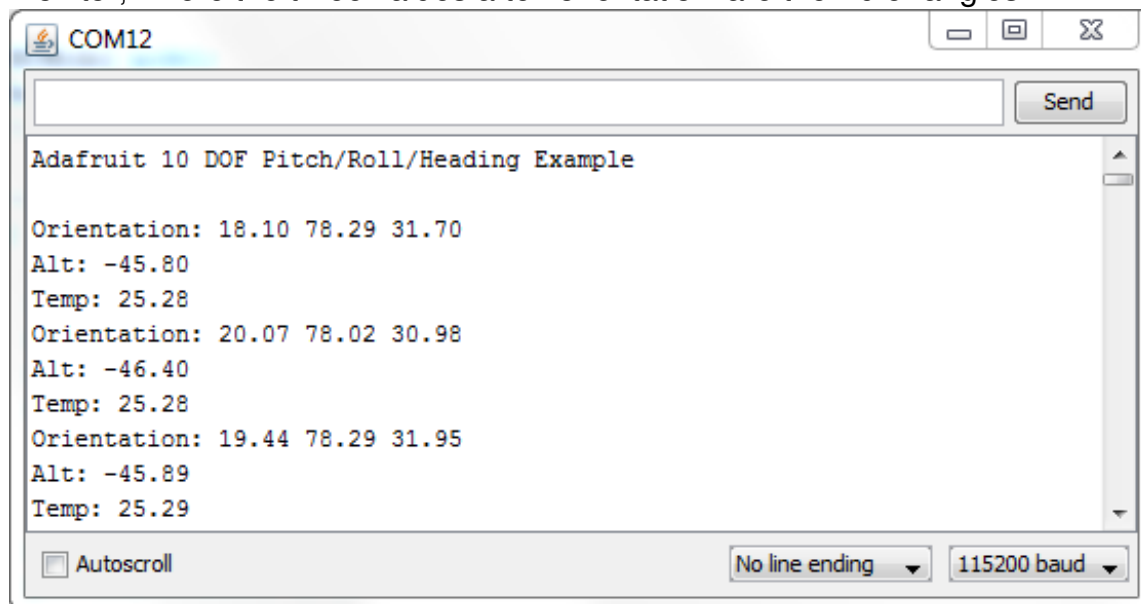
```
COM12                                                    — ▢ ✕

                                                         Send

Adafruit 10 DOF Pitch/Roll/Heading Example

Orientation: 18.10 78.29 31.70
Alt: -45.80
Temp: 25.28
Orientation: 20.07 78.02 30.98
Alt: -46.40
Temp: 25.28
Orientation: 19.44 78.29 31.95
Alt: -45.89
Temp: 25.29

☐ Autoscroll                    No line ending  ▾   115200 baud ▾
```

In this case, we can see that the roll is about 18°, the pitch is about 78° and the heading or yaw is about 32°, and the sketch will keep updating itself with the latest values at whatever

speed we've set in the example sketch.

# Visualizing Data

To help you visualize the data, we've put together a basic Processing sketch that loads a 3D model (in the .obj file format) and renders it using the data generated by the AHRS sketch on the Uno. The ahsr sketch on the uno published data over UART, which the Processing sketch reads in, rotating the 3D model based on the incoming orientation data.

# Requirements

- [Processing 2.x](http://adafru.it/ddm) (http://adafru.it/ddm)
- [Saito's OBJ Loader](http://adafru.it/ddn) (http://adafru.it/ddn) library for Processing ([installation tips here](http://adafru.it/ddn) (http://adafru.it/ddn))
- [G4P GUI library](http://adafru.it/dMO) (http://adafru.it/dMO) for Processing ([download the latest version here](http://adafru.it/dMP) (http://adafru.it/dMP) and copy the zip into the processing libraries folder like the tip for the OBJ loader library above mentions).

The OBJ library is required to load 3D models. It isn't strictly necessary and you could also render a boring cube in Processing, but why play with cubes when you have rabbits?!

# Opening the Processing Sketch

The processing sketch to render the 3D model is contained in the sample folder as the ahrs sketch for the Uno.

With Processing open, navigate to you Adafruit_AHRS library folder (ex.: **'libraries/Adafruit_AHRS'**), and open **'processing/bunnyrotate/bunnyrotate.pde'.** You should see something like this in Processing:

File Edit Sketch Tools Help

Java ▼

cuberotate ◉

```java
import processing.serial.*;
import java.awt.datatransfer.*;
import java.awt.Toolkit;
import processing.opengl.*;
import saito.objloader.*;
import g4p_controls.*;

float roll  = 0.0F;
float pitch = 0.0F;
float yaw   = 0.0F;
float temp  = 0.0F;
float alt   = 0.0F;

OBJModel model;

// Serial port state.
Serial      port;
String      buffer = "";
final String serialConfigFile = "serialconfig.txt";
boolean     printSerial = false;

// UI controls.
GPanel    configPanel;
GDropList serialList;
GLabel    serialLabel;
GCheckbox printSerialCheckbox;

void setup()
{
  size(400, 550, OPENGL);
  frameRate(30);
  model = new OBJModel(this);
  model.load("bunny.obj");
  model.scale(20);

  // Serial port setup.
  // Grab list of serial ports and choose one that was persisted earlier or default to the fi
  int selectedPort = 0;
  String[] availablePorts = Serial.list();
  if (availablePorts == null) {
```
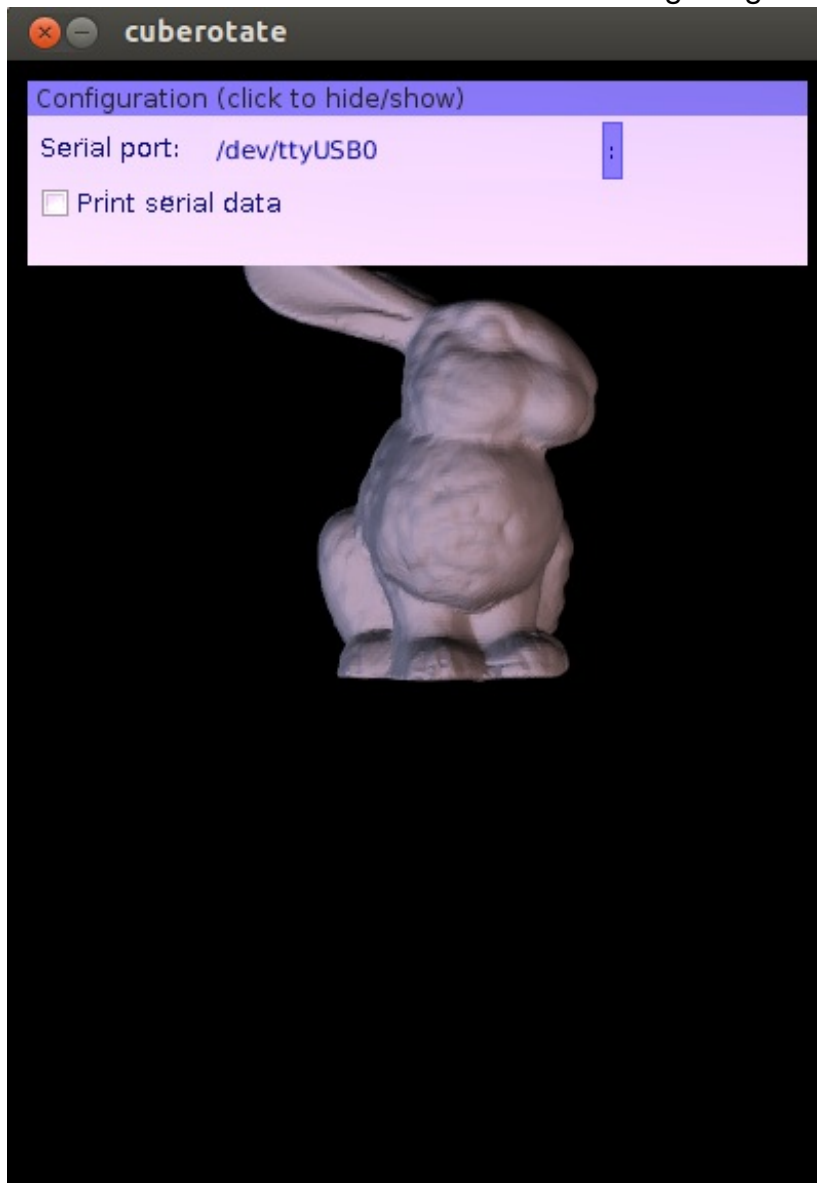
1

# Run the AHRS Sketch on the Uno

Make sure that the appropriate AHRS example sketch is running on the Uno (as described on the previous page), and that the Serial Monitor **is closed**.

With the sample sketch running on the Uno, click the triangular 'play' icon in Processing to start the sketch.

# Rabbit Disco!

You should see a rabbit similar to the following image:

Before the rabbit will rotate you will need to click the **:** to the right of the serial port name. This will open a list of available serial ports, and you will need to click the appropriate serial port that your Arduino uses (check the Arduino IDE to see the port name if you're unsure). The chosen serial port should be remembered if you later run the sketch again.

As you rotate your breakout board, the rabbit should rotate to reflect the movement of the breakout in 3D-space, as seen in the video below (**note that this video is using an older version of the cuberotate example and won't look exactly like what you see now**):

# A Note on Accuracy and Calibration

Note that the movement seen on the screen won't correlate exactly to the breakout since we are currently using **uncalibrated sensor data**, specifically uncalibrated magnetometer data which will throw the heading off. The magnetometer is required to generate 360° data, and if you want to improve the accuracy of your orientation data we have a tutorial on calibrating the LSM303 (http://adafru.it/ddo), but this particular example starts with raw data for simplicity sake.

# Magnetometer Calibration

For more accurate orientation output, the magnetometer needs to be adjusted to compensate for offset errors (meaning a shift in either direction on the X/Y/Z axis), as well as something called 'soft iron error', which causes what should be spherical output of the magnetometer to actually be an elongated 'pill' shape or something similar.

In order to compensate for these two problems and generate accurate magnetometer data, which is critical to getting high quality orientation results, you will need to calibrate your magnetometer in the final enclosure and setup in which the device will be used.

The calibration process should be done when the project is in it's final form and inside any enclosure being used, etc:

# PJRC MotionCal

There are a variety of calibration tools out there, butMotionCal (http://adafru.it/qUE) from always awesome PJRC (Teensy, etc.) is probably the easiest to use, and they provide pre-compiled binaries for all major operating systems.

Download the Motion Sensor Calibration tool for your OS from the following link and open it up on your system: http://www.pjrc.com/store/prop_shield.html (http://adafru.it/qUE)

By default, you should see something like this:
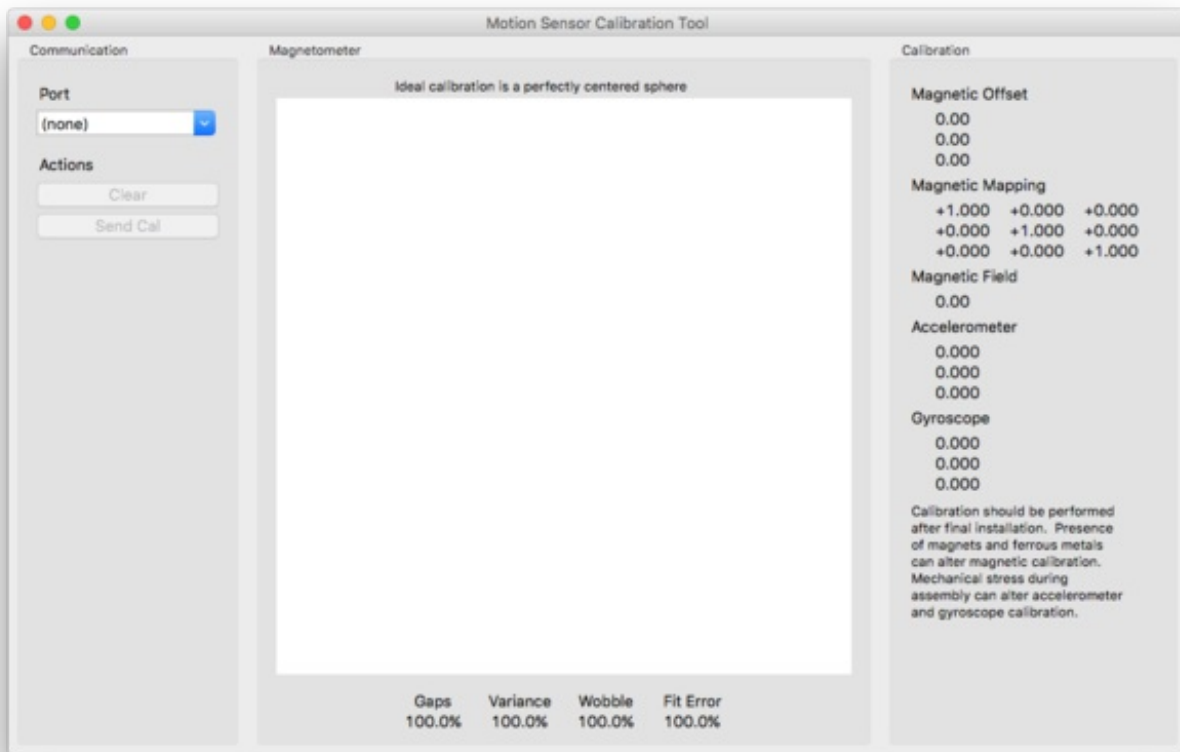
Next open up the ahrs_calibration (http://adafru.it/qUF) example from Adafruit_AHRS:

If you don't see this example, you may need to go into the Library Manager (Sketch > Include Library > Manage Libraries ...) and update your Adafruit_AHRS library to the latest version.
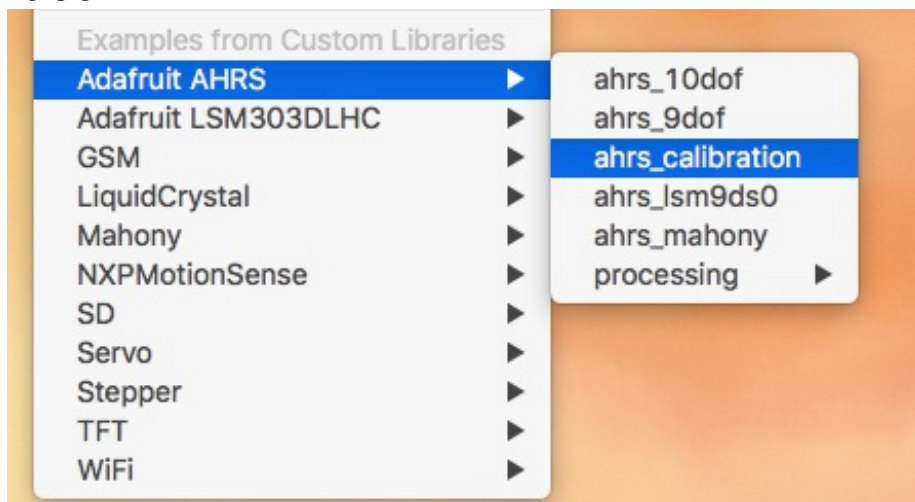


Be default this is setup for the sensors found on the Adafruit 9DOF and 10DOF breakouts, but can be adjusted to work with any accelerometer, magnetometer and gyroscope that is based on the Adafruit Unified Sensor system (http://adafru.it/dGB).

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_L3GD20_U.h>
#include <Adafruit_LSM303_U.h>

// Create sensor instances.
Adafruit_L3GD20_Unified        gyro(20);
Adafruit_LSM303_Accel_Unified accel(30301);
Adafruit_LSM303_Mag_Unified    mag(30302);

// This sketch can be used to output raw sensor data in a format that
// can be understoof by MotionCal from PJRC. Download the application
// from http://www.pjrc.com/store/prop_shield.html and make note of the
// magentic offsets after rotating the sensors sufficiently.
//
// You should end up with 3 offsets for X/Y/Z, which are displayed
// in the top-right corner of the application.
//
// Make sure you have the latest versions of the Adfruit_L3GD20_U and
// Adafruit_LSM303_U libraries when running this sketch since they
// use a new .raw field added in the latest versions (Oct 10 2016)

void setup()
{
  Serial.begin(115200);

  // Wait for the Serial Monitor to open (comment out to run without Serial Monitor)
  // while(!Serial);

  Serial.println(F("Adafruit 10 DOF Board AHRS Calibration Example")); Serial.println("");

  // Initialize the sensors.
  if(!gyro.begin())
  {
    /* There was a problem detecting the L3GD20 ... check your connections */
    Serial.println("Ooops, no L3GD20 detected ... Check your wiring!");
    while(1);
  }
```

# Generating Calibration Data

Next run the sketch, and if it isn't already open, open the MotionCal app.

Select the appropriate serial port for your board, and once select start rotating the board around gently in figure eight type movements.

As you move the device, points should start to appear in a point cloud.

You need to keep moving the device in as many orientations as possible until the sphere is mostly complete and the **Gaps** percentage on the bottom of the app is reasonably low.

A good example of what you should aim for is shown below:
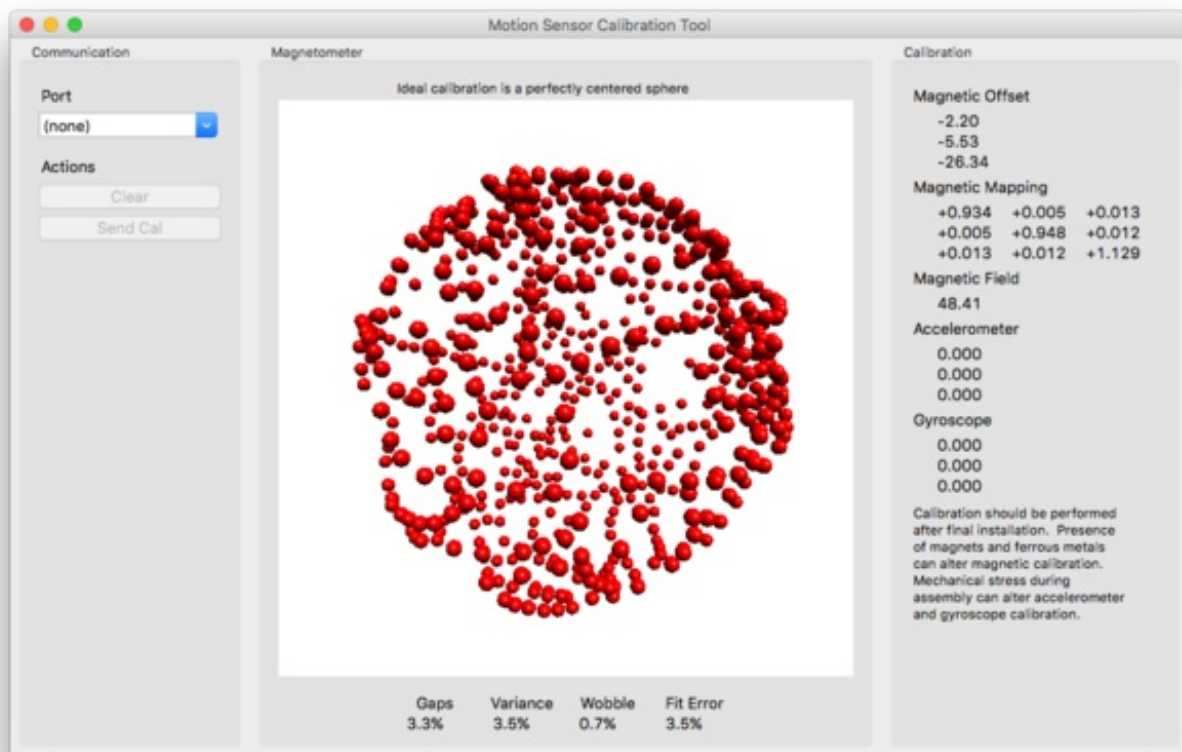
# Note Offsets and Magnetic Mapping Values

Now that you have determined the values to compensate for magnetometer offset and soft-iron error, make a note of the values in the top-right hand corner of the app, which in this case is:

## Magnetic Offset

-2.20, -5.53, -26.34

## Magnetic Mapping

0.934, 0.005, 0.013
0.005, 0.948, 0.012
0.013, 0.012, 1.129

## Magnetic Field

The magnetic field value is not currently used but should be noted anyway.

# That's It!

Congratulations, you have everything you need to generate reasonably accurate ouput on the magnetometer taking into account any local interference in your environment or caused by the board itself.

You'll need these values in the next step, which is feeding your compensated sensor output into a sensor fusion algorithm, but you can close the MotionCal app for now until you need to calibrate the device again, such as if you move it to a different setup or environment.

# Sensor Fusion Algorithms

There are a variety of sensor fusion algorithms out there, but the two most common in small embedded systems are the Mahony and Madgwick filters.

Mahony is more appropriate for very small processors, whereas Madgwick can be more accurate with 9DOF systems at the cost of requiring extra processing power (it isn't appropriate for 6DOF systems where no magnetometer is present, for example).

We will use the Mahony fusion algorithm in this example since it is the most relevant to a wide variety of devices, but it's easy to switch to Madgwick if you prefer to test another algorithm.

This page and code is still a work in progress and subject to change in the future!

# MahonyAHRS and MadgwickAHRS Libraries

For best results, use the fork of MahonyAHRS and MadgwickAHRS from PJRC (based on the original Arduino libraries) available here:

- https://github.com/PaulStoffregen/MahonyAHRS (http://adafru.it/rUA)
- https://github.com/PaulStoffregen/MadgwickAHRS (http://adafru.it/rUB)

You will need to download these as a zip file and then install them in your libraries folder, or git clone them in the Arduino libraries folder (ex. 'git clone https://github.com/PaulStoffregen/MahonyAHRS.git').

For help installing libraries, see our Arduino Libraries Learning Guide (http://adafru.it/dNR).

# Enter Calibration Data into ahrs_mahony

To get started, open the **ahrs_mahony example** (http://adafru.it/rUC) from the Adafruit_AHRS folder.

You will need to enter the magnetometer calibration values calculated earlier in this guide in the appropriate field, as shown in the screenshot below:

```
ahrs_mahony

#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_L3GD20_U.h>
#include <Adafruit_LSM303_U.h>
#include <MahonyAHRS.h>
#include <MadgwickAHRS.h>

// Note: This sketch requires the MahonyAHRS sketch from
// https://github.com/PaulStoffregen/MahonyAHRS, or the
// MadgwickAHRS sketch from https://github.com/PaulStoffregen/MadgwickAHRS

// Note: This sketch is a WORK IN PROGRESS

// Create sensor instances.
Adafruit_L3GD20_Unified       gyro(20);
Adafruit_LSM303_Accel_Unified accel(30301);
Adafruit_LSM303_Mag_Unified   mag(30302);

// Mag calibration values are calculated via ahrs_calibration.
// These values must be determined for each baord/environment.
// See the image in this sketch folder for the values used
// below.

// Offsets applied to raw x/y/z values
float mag_offsets[3]            = { -2.20F, -5.53F, -26.34F };

// Soft iron error compensation matrix
float mag_softiron_matrix[3][3] = { { 0.934, 0.005, 0.013 },
                                    { 0.005, 0.948, 0.012 },
                                    { 0.013, 0.012, 1.129 } };

float mag_field_strength        = 48.41F;

// Mahony is lighter weight as a filter and should be used
// on slower systems
Mahony filter;
//Madgwick filter;

void setup()
{
  Serial.begin(115200);

  // Wait for the Serial Monitor to open (comment out to run without Serial Monitor)
  // while(!Serial);
```

These values are based on the following calibration data:

Communication

Magnetometer

Calibration

Port

(none)

Actions

Clear

Send Cal

Ideal calibration is a perfectly centered sphere

Magnetic Offset
-2.20
-5.53
-26.34
Magnetic Mapping
+0.934   +0.005   +0.013
+0.005   +0.948   +0.012
+0.013   +0.012   +1.129
Magnetic Field
48.41
Accelerometer
0.000
0.000
0.000
Gyroscope
0.000
0.000
0.000

Calibration should be performed after final installation. Presence of magnets and ferrous metals can alter magnetic calibration. Mechanical stress during assembly can alter accelerometer and gyroscope calibration.

| Gaps | Variance | Wobble | Fit Error |
|------|----------|--------|-----------|
| 3.3% | 3.5% | 0.7% | 3.5% |

# Run the Sketch

Next, compiled and run your sketch.

Open the Serial Monitor, and you should see a **millisecond timestamp**, followed by the output of the sensor fusion algorithm, which will give you Euler Angles for **Heading**, **Pitch** and **Roll** in that order:

```
/dev/cu.usbmodem1D11 (Adafruit WICED Feather)                    Send

68308 - Orientation: 252.28 -20.13 -6.54
68328 - Orientation: 252.30 -20.14 -6.55
68348 - Orientation: 252.33 -20.13 -6.55
68368 - Orientation: 252.36 -20.13 -6.55
68388 - Orientation: 252.38 -20.12 -6.55
68408 - Orientation: 252.41 -20.12 -6.55
68428 - Orientation: 252.44 -20.12 -6.55
68448 - Orientation: 252.47 -20.12 -6.55
68468 - Orientation: 252.49 -20.12 -6.55
68488 - Orientation: 252.52 -20.13 -6.56
68508 - Orientation: 252.55 -20.13 -6.56
68528 - Orientation: 252.58 -20.13 -6.56
68548 - Orientation: 252.61 -20.13 -6.57
68568 - Orientation: 252.64 -20.13 -6.57
68588 - Orientation: 252.66 -20.13 -6.57
68608 - Orientation: 252.69 -20.12 -6.57
68628 - Orientation: 252.71 -20.11 -6.56
68648 - Orientation: 252.74 -20.10 -6.56
68668 - Orientation: 252.77 -20.10 -6.57
68688 - Orientation: 252.80 -20.10 -6.57
68708 - Orientation: 252.83 -20.10 -6.58
68728 - Orientation: 252.86 -20.11 -6.58
68748 - Orientation: 252.88 -20.11 -6.59
68768 - Orientation: 252.91 -20.11 -6.59
68788 - Orientation: 252.94 -20.12 -6.60
68808 - Orientation: 252.97 -20.12 -6.60
68828 - Orientation: 253.00 -20.11 -6.59
68848 - Orientation: 253.02 -20.11 -6.59
68868 - Orientation: 253.04 -20.11 -6.59
68888 - Orientation: 253.07 -20.11 -6.60

☐ Autoscroll                        No line ending ◇    115200 baud ◇
```

# Tuning the Filter

One particularity of fusion algorithms (and most DSP algorithms) is that they are sensitive to timing. Make a note of the millisecond timestamp before the individual samples, can count approximately how many samples per second your device is outputting.

You need to tell the fusion algorithm how many samples per second we are generating to more accurately estimate positions in 3D space based on the accel, gyro and mag data.

In the filters **setup()** function, find the following lines and adjust this value to match the number of samples per second your system is outputting:

```
// Filter expects 50 samples per second
filter.begin(50);
```

Run the filter again and you should get slightly more accurate results.

# Switching to Madgwick

To switch to the Madgwick filter, simply comment the appropriate class instance at the top of the sketch and run the code again. The two filters have identical APIs and are easily interchangeable.

Change the code from this ...
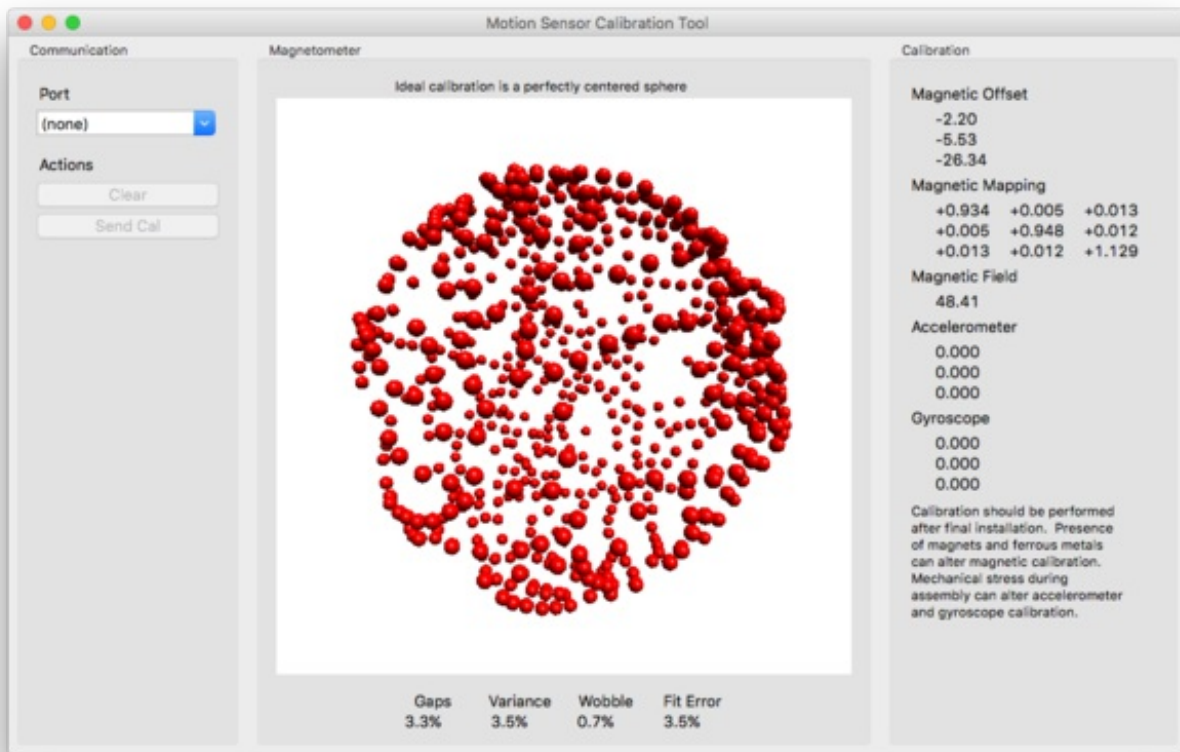
```
// Mahony is lighter weight as a filter and should be used
// on slower systems
Mahony filter;
//Madgwick filter;
```

... to this:

```
//Mahony filter;
Madgwick filter;
```

# A Note on Orientation Values

Please note that at present all orientation values are **relative** and not **absolute**. What this means is that the orientation values will be relative to the starting position of the device, not absolute magnetic north or south, etc.

Before starting, put your device in a known, repeatable position if you require reasonably repeatable, consistent output.