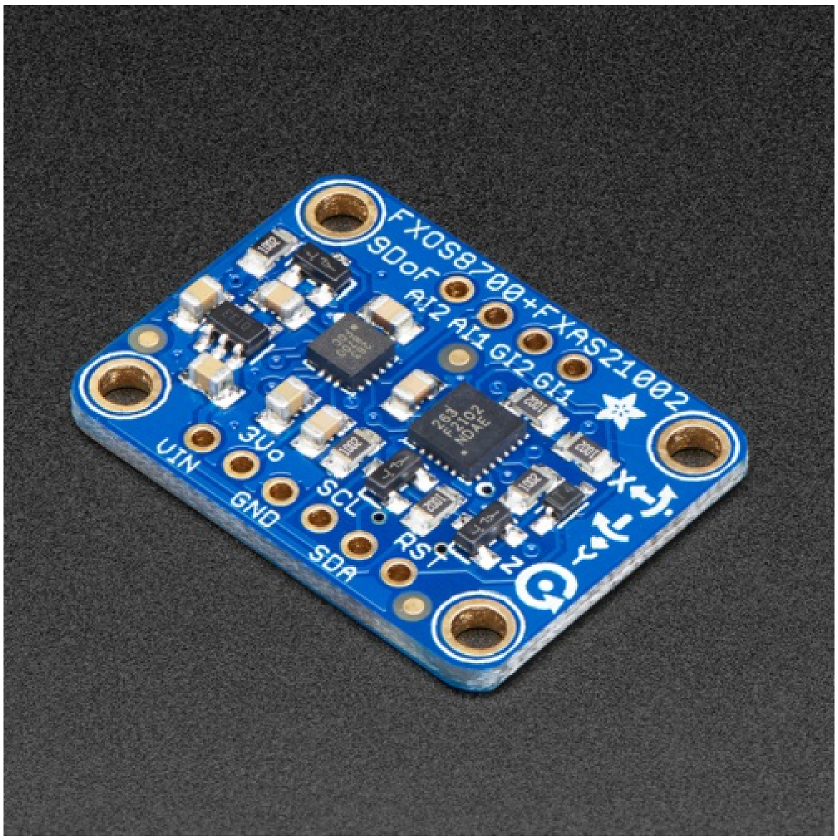
NXP Precision 9DoF Breakout

Created by Kevin Townsend

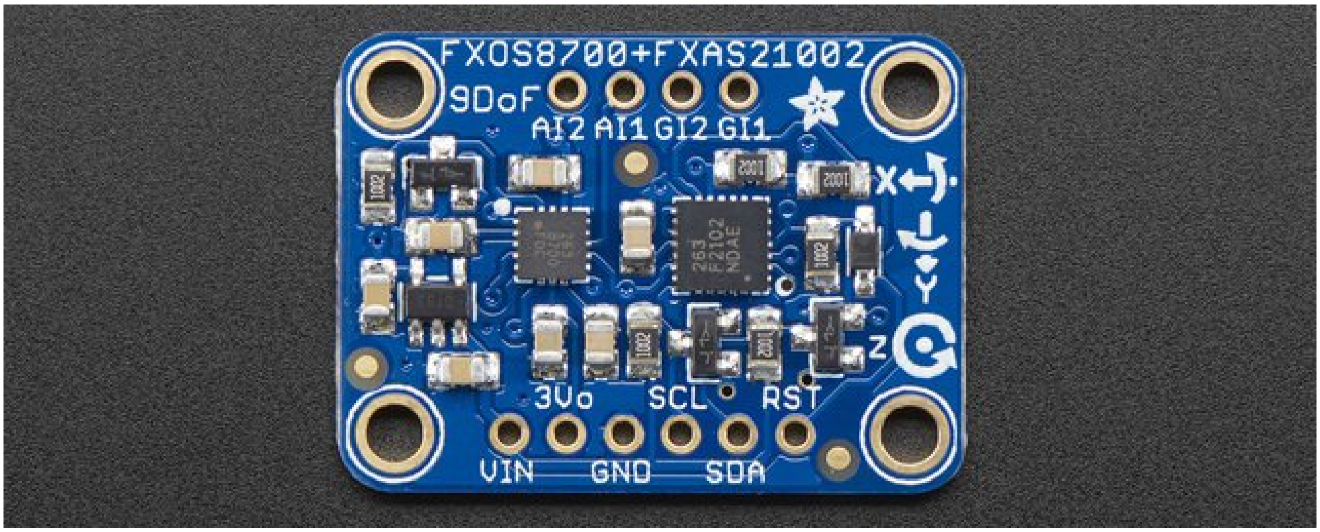


Last updated on 2017-07-04 02:54:00 AM UTC

Guide Contents

Overview

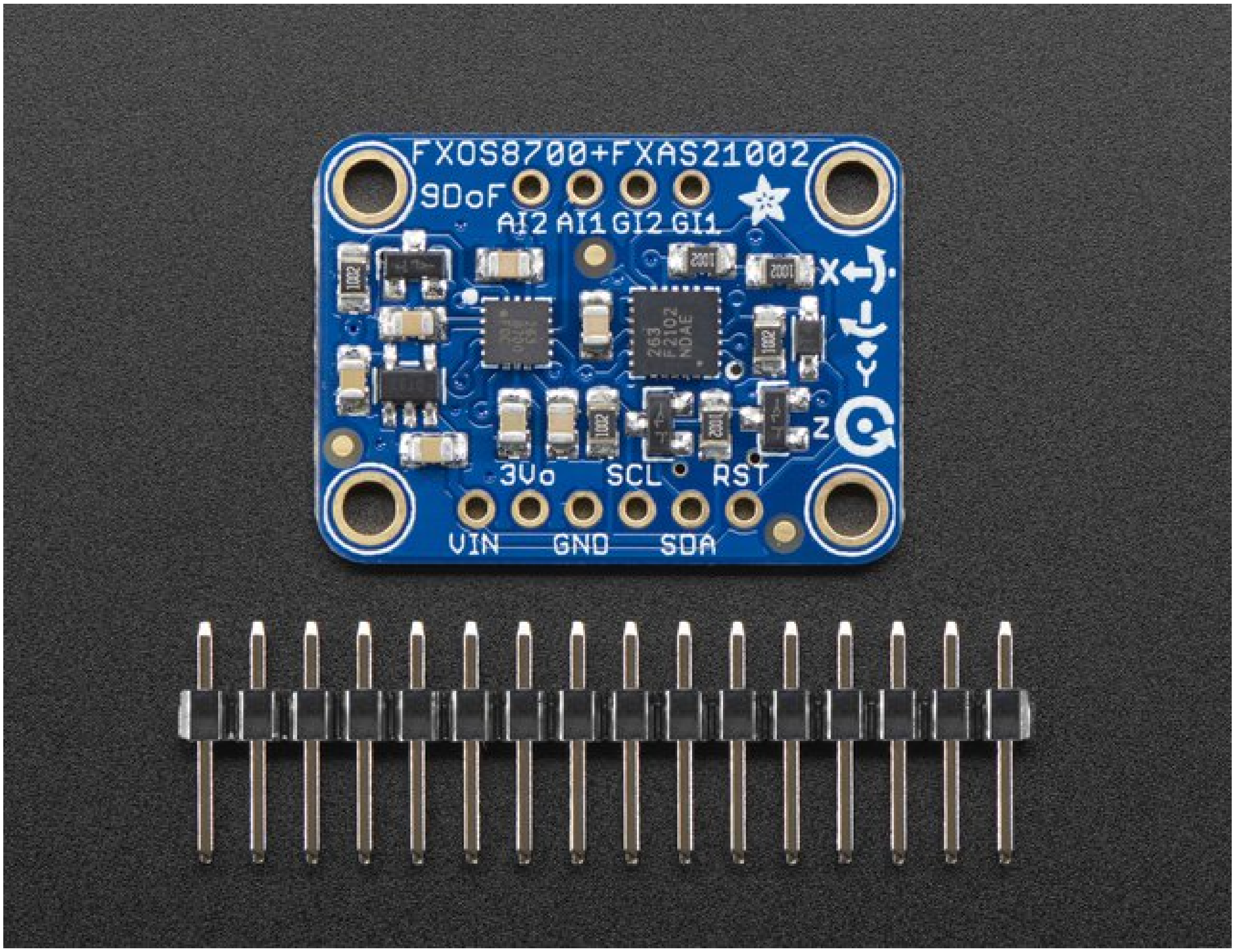
The NXP Precision 9DoF breakout combines two of the best motion sensors we've tested here at Adafruit: The **FXOS8700** 3-Axis accelerometer and magnetometer, and the **FXAS21002** 3-axis gyroscope.



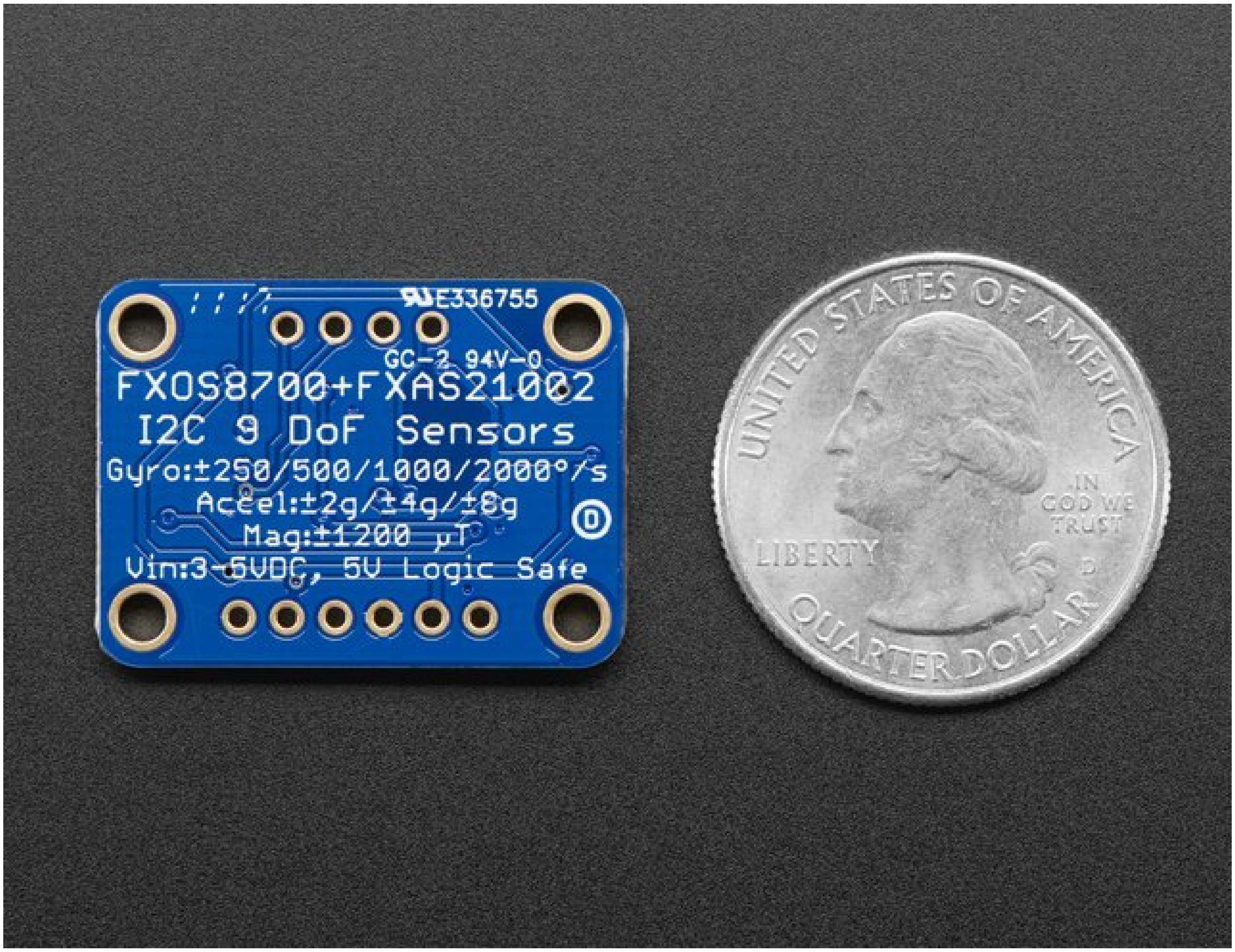
These two sensors combine to make a nice 9-DoF kit, that can be used for motion and orientation sensing. In particular, we think this sensor set is ideal for AHRS-based orientation calculations: the gyro stability performance is superior to the LSM9DS0 (http://adafru.it/vAu), LSM9DS1 (http://adafru.it/vAv), L3GD20H + LSM303, (http://adafru.it/dNY)MPU-9250, and even the BNO-055 (http://adafru.it/fE0) (see our Gyro comparison tutorial for more details (http://adafru.it/vAK))

Compared to the BNO055, this sensor will get you similar orientation performance but at a lower price because the calculations are done on your microcontroller, not in the sensor itself. The trade off is you will sacrifice about 15KB of Flash space, and computing cycles, to do the math 'in house'

To make it fast and easy for you to get started, we have a version of AHRS that we've adapted to work over USB or Bluetooth LE. Load the code onto your Arduino-compatible board and you will get orientation data in the form of Euler angles or quaternions! It will work on a ATmega328 but faster/larger chips such as M0 or ESP8266 will give you more breathing room.



Each board comes with the two chips soldered onto a breakout with 4 mounting holes. While the chips support SPI, they don't tri-state the MISO pin, so we decided to go with plain I2C which works well and is supported by every modern microcontroller and computer chip set. There's a 3.3V regulator and level shifting on the I2C and Reset lines, so you can use the breakout safely with 3.3V or 5V power/logic. Each order comes with a fully assembled and tested breakout and a small strip of header. Some light soldering is required to attach the header if you want to use in a breadboard.



So what makes this so 'Precision'-y, eh?

Glad you asked! This particular sensor combination jumped out at us writing the Comparing Gyroscopes (http://adafru.it/vAw) learning guide since the FXAS21002 exhibited the lowest **zero-rate level** off any of the gyroscopes we've tested, with the the following documented levels (converted to degrees per second for convenience sake):

- At +/- 2000 dps **3.125 dps**
- At +/- 250 dps **0.3906 dps**

The zero-rate level is important in orientation since it represents the amount of angular velocity a gyroscope will report when the device is immobile. High zero-rate levels can cause all kinds of problems in orientation systems if the data isn't properly compensated out, and distinguishing zero-rate errors from actual angular velocity can be non-trivial. This is particularly important in sensor fusion algorithms where the gyroscope plays an important part in predicting orientation adjustments over time. A high zero-rate level will cause constant rotation even when the device is immobile!

By comparison, most other sensors tested have 10-20 times these zero-rate levels, which is why we consider this particular part very **precise**. There is little work to do out of the box to get useful, actionable data out of it. See the table of similar parts below to compare for yourself:

	+/- 250 dps	+/- 500 dps	+/- 2000 dps
L3GD20	10 dps	15 dps	75 dps
FXAS21002C	0.3906 dps	0.78125 dps	3.125 dps
LSM9DS0	10 dps	15 dps	25 dps
LSM9DS1	??? <= 30 dps	??? <= 30 dps	30 dps
MPU-9250	5 dps	???	???
BMI055	? 1 dps ?	???	???

Technical Details

The NXP Precision 9DoF board consists of two separate ICs, described in detail below:

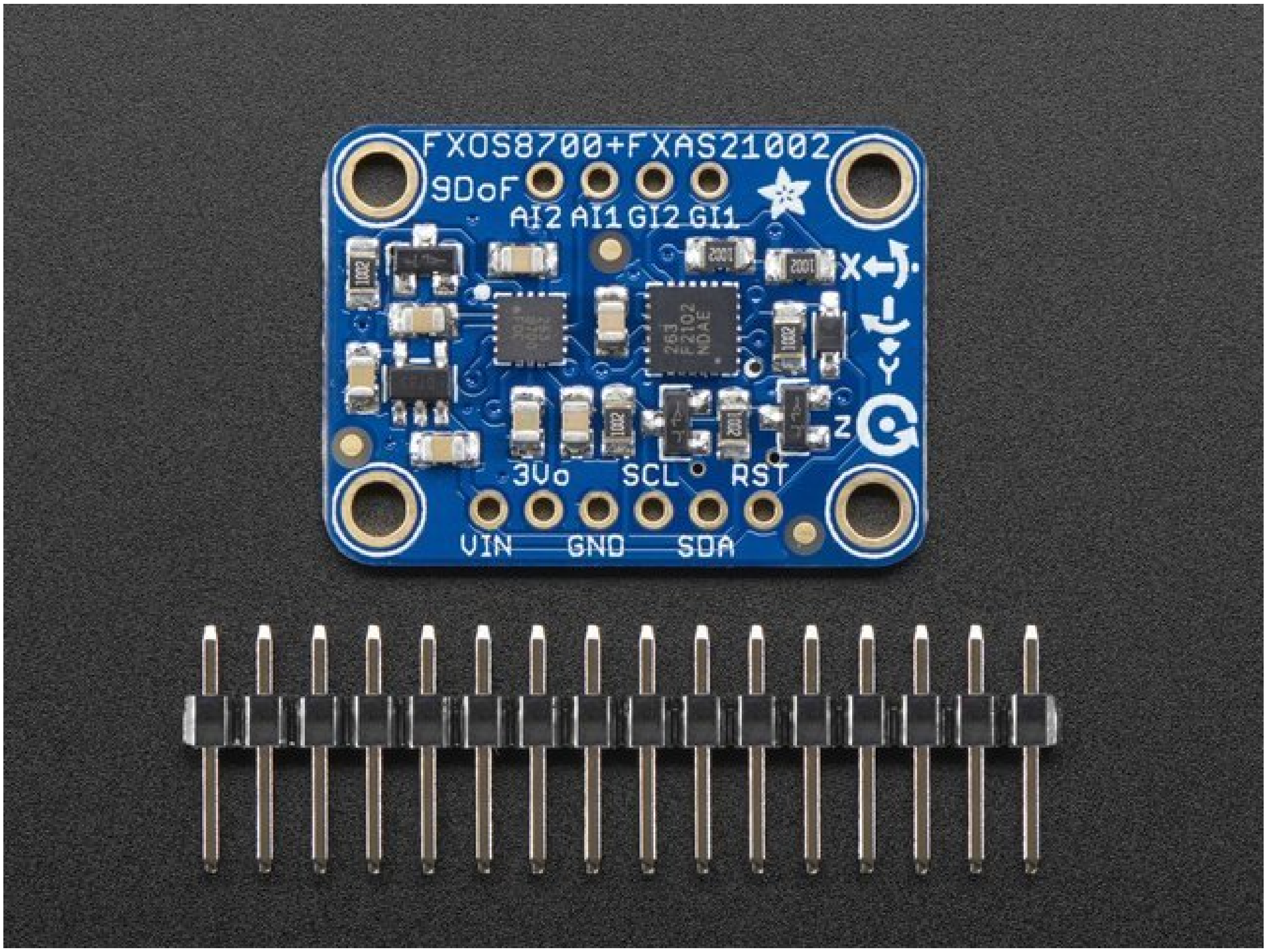
FXOS8700 3-Axis Accelerometer/Magnetometer

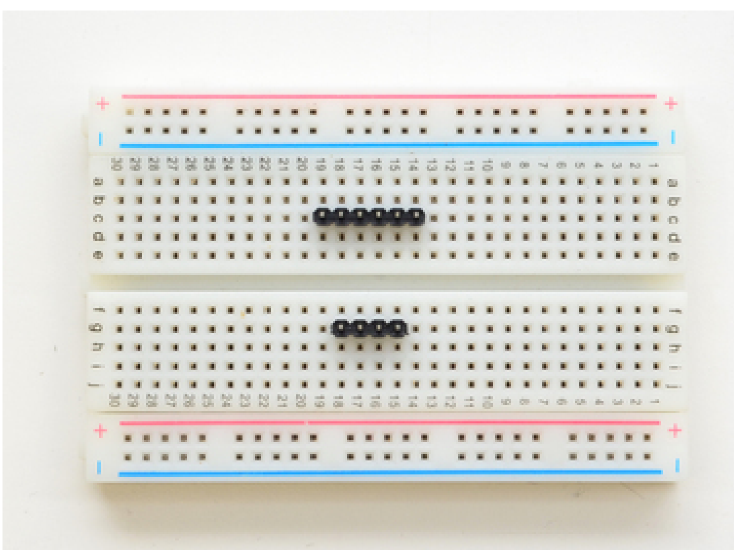
- 2-3.6V Supply
- ±2 g/±4 g/±8 g adjustable acceleration range
- ±1200 µT magnetic sensor range
- Output data rates (ODR) from 1.563 Hz to 800 Hz
- 14-bit ADC resolution for acceleration measurements
- 16-bit ADC resolution for magnetic measurements

FXAS21002 3-Axis Gyroscope

- 2-3.6V Supply
- ±250/500/1000/2000°/s configurable range
- Output Data Rates (ODR) from 12.5 to 800 Hz
- 16-bit digital output resolution
- 192 bytes FIFO buffer (32 X/Y/Z samples)

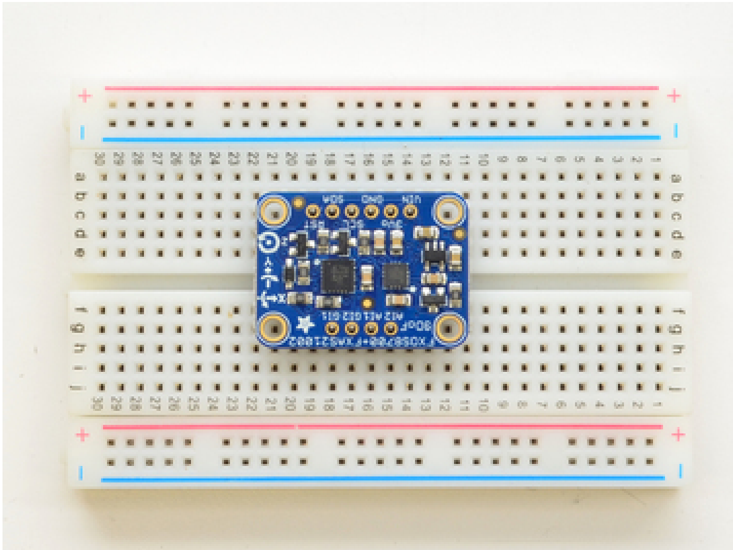
Assembly



- 

Prepare the header strip:

Cut the strip to length if necessary. It will be easier to solder if you insert it into a breadboard - **long pins down**

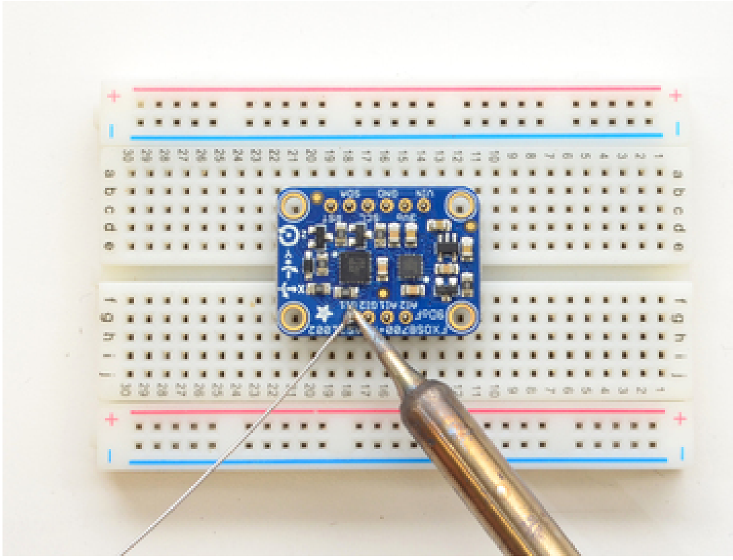
- 

Add the breakout board:

Place the breakout board over the pins so that the short pins poke through the breakout pads

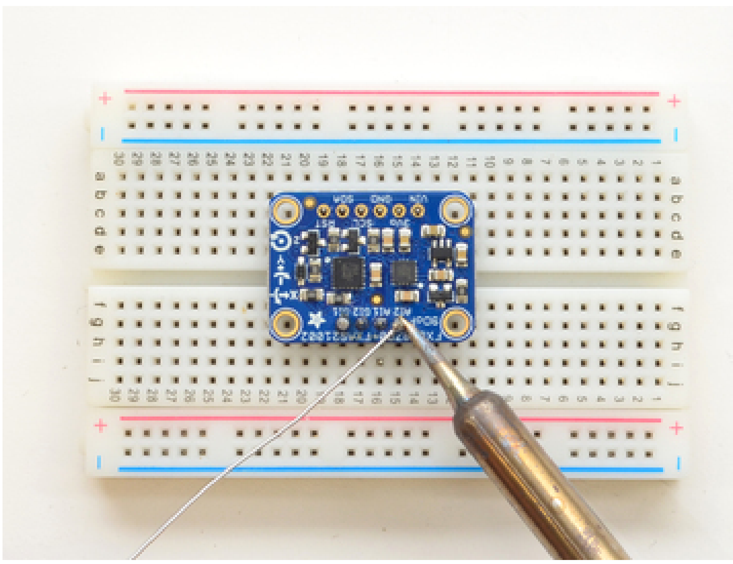
And Solder!

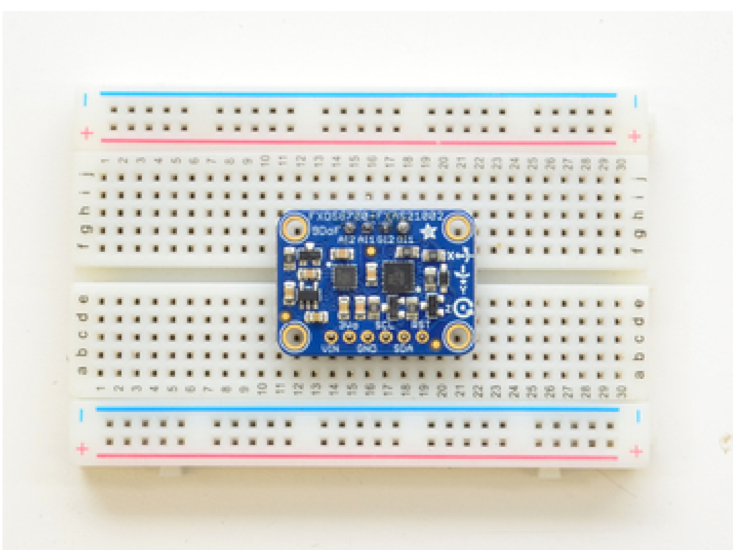
Be sure to solder all pins for reliable electrical contact.

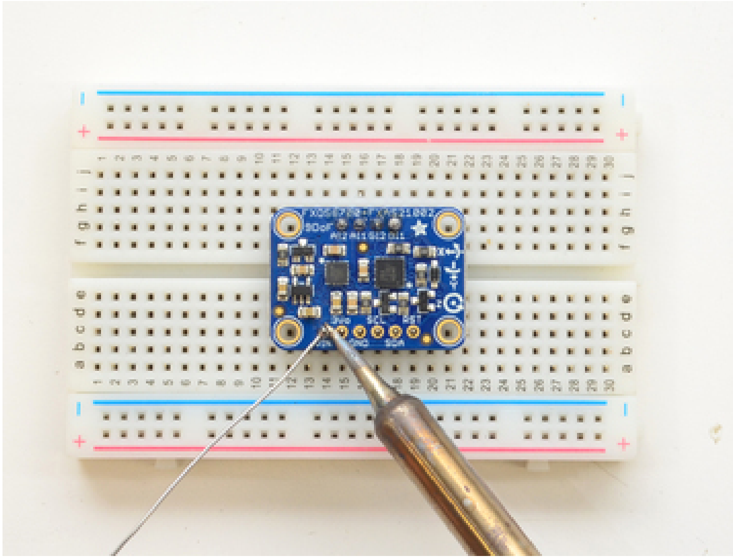
- 

Solder one side first

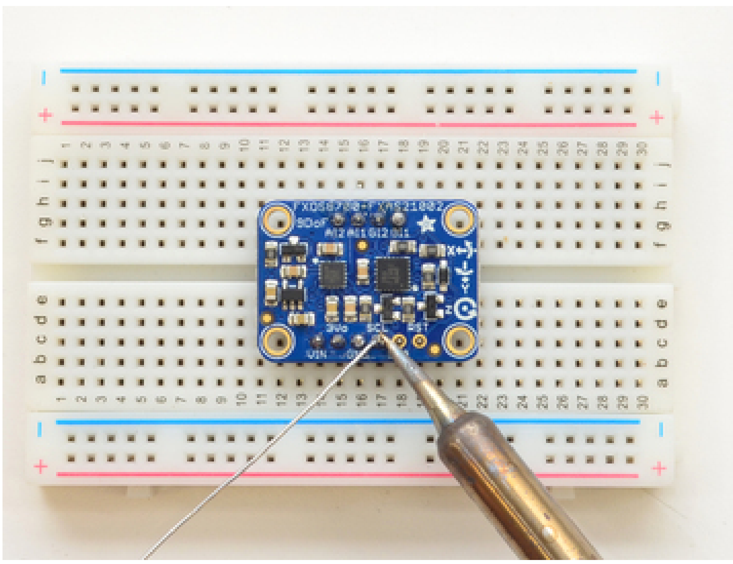
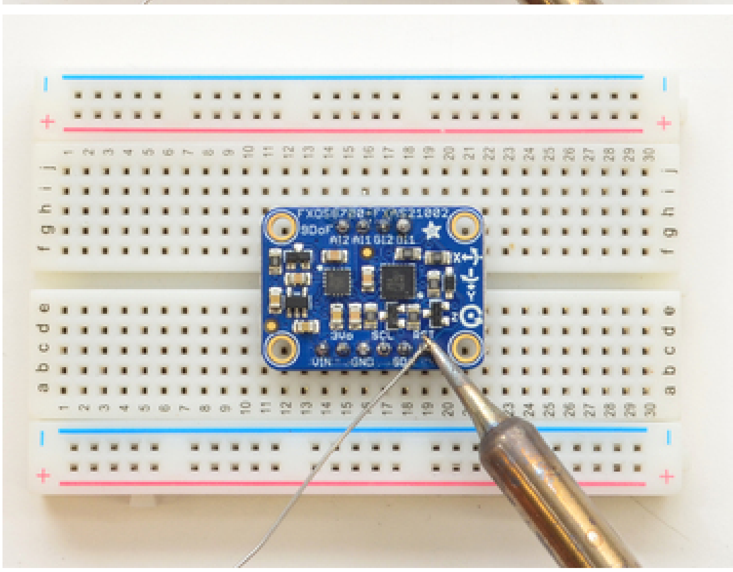
(For tips on soldering, be sure to check out our [Guide to Excellent Soldering](http://adafru.it/aTk) (http://adafru.it/aTk)).

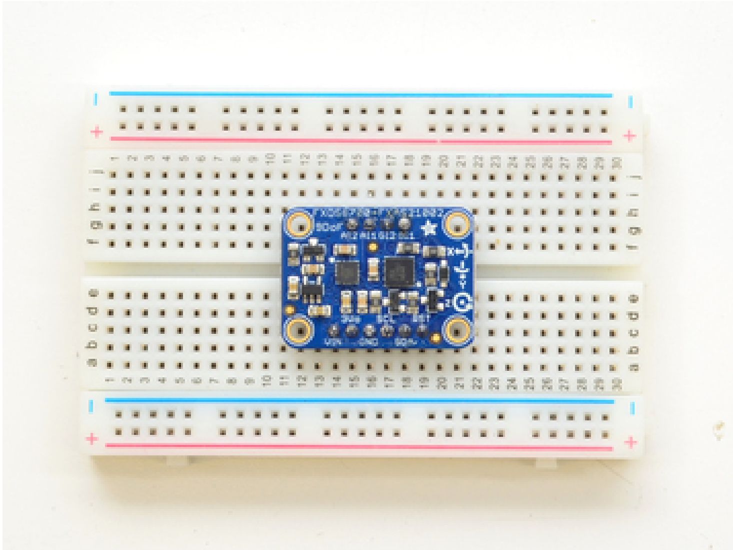
- 

- 

- 

Twist the board around and solder the other row!

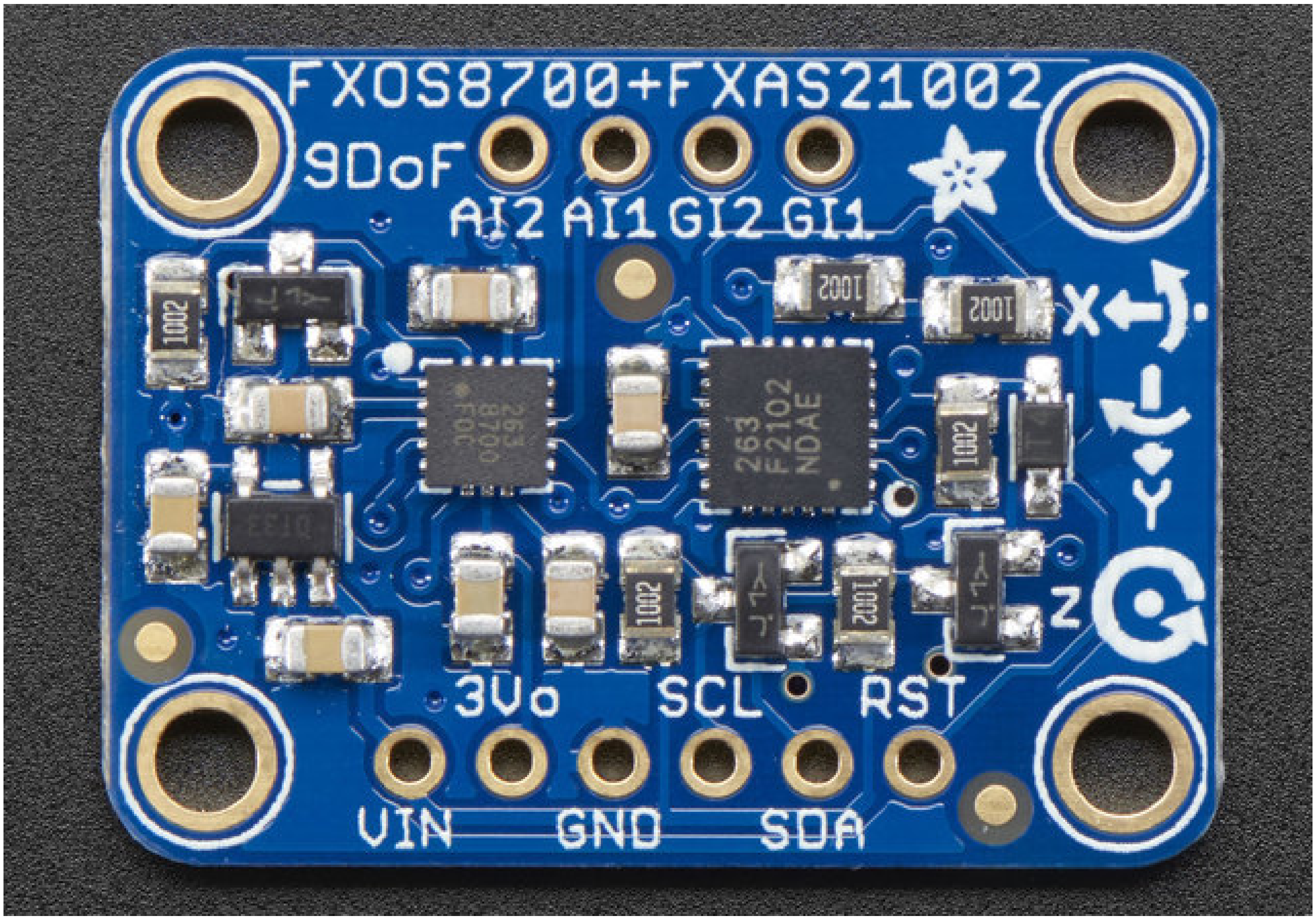
- 
- 

- 

You're done! Check your solder joints visually and continue onto the next steps

Pinout

The NXP Precision 9DoF breakout has the following pinout:



Power Pins

- **VIN** - 3.3-5V input, which feeds the on board 3.3V voltage regulator and optionally sets the signal levels for the I2C pins (SCL and SDA) if you are using a 5V system. On a **3.3V system** (any Adafruit Feather, for example), connect 3.3V to VIN for 3.3V logic throughout the system. On a **5.0V system**, connect VIN to 5V, and the signals will be shifted downward to 3.3V before reaching the NXP sensors (which are limited to 3.6V or less for the pins).
- **3Vo** - This is the output of the 3.3V linear regulator on the NXP Precision 9DoF Breakout. On a 5V system, you can use this as an additional 3.3V supply if you need some extra 3.3V power.
- **GND** - This should be connected to GND on your development board.

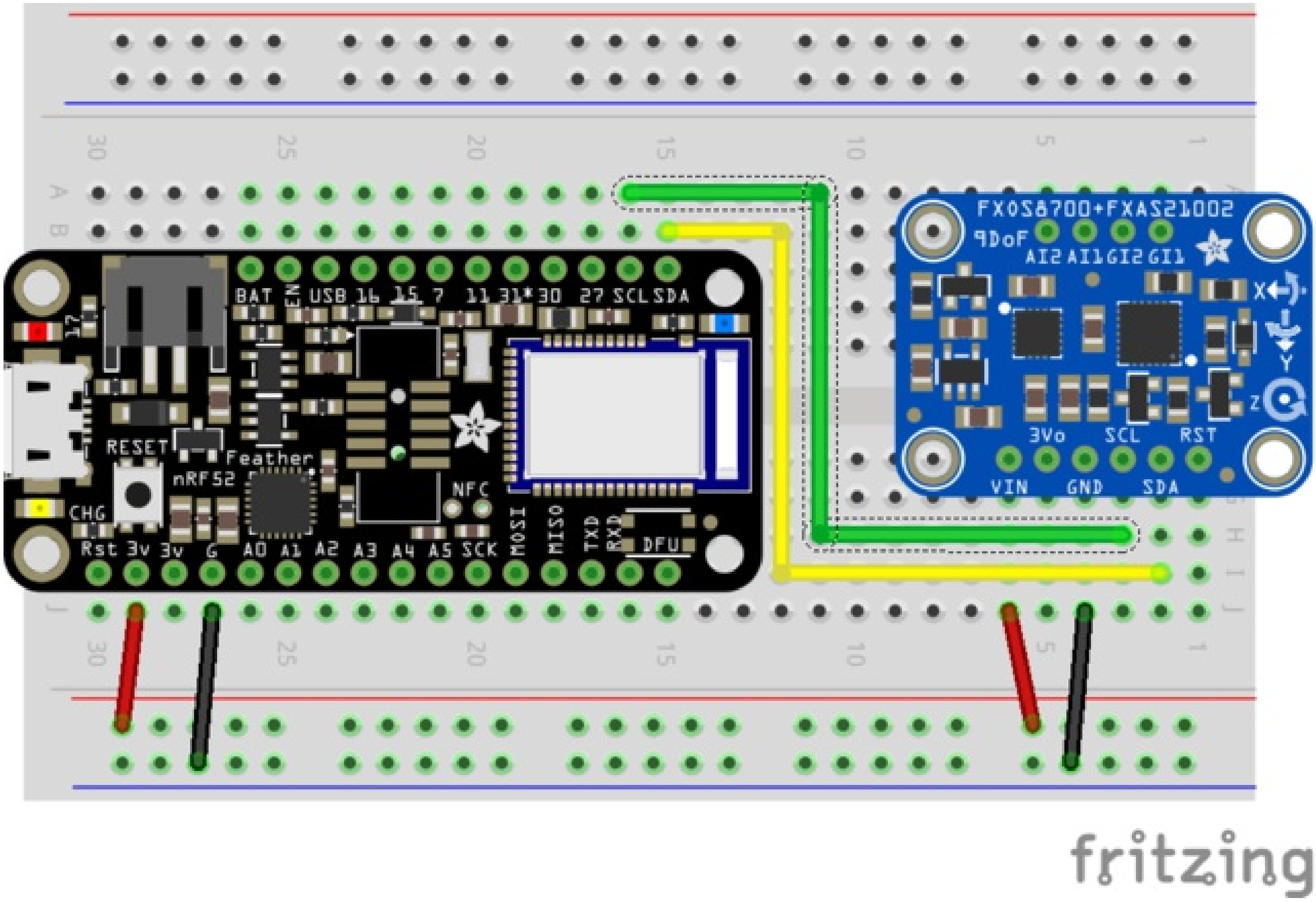
Digital Pins

- **SCL** - I2C, Connect this to SCL on your development board. This pin is level-shifted and 3-5V logic safe.
- **SDA** - I2C, Connect this to SDA on your development board. This pin is level-shifted and 3-5V logic safe.
- **RST** - Optionally connect this to RST on your development board (depending on the logic level used), or to a GPIO pin if you wish to manually reset the sensors on the breakout. This pin isn't required in most circumstances, but can be useful to recover from error conditions on long running systems where the sensors might have entered an unknown config state. This pin is level-shifted and 3-5V logic safe.
- **AI1, AI2** - This two pins allow interrupts *from* the Accelerometer/Magnetometer (see the datasheet for details). These are not level shifted but since they are outputs only, you can use with 3 or 5V logic systems.
- **GI1, G12** - These two pins allow interrupts from the Gyroscope (see the datasheet for details). These are not level shifted but since they are outputs only, you can use with 3 or 5V logic systems.

Breadboard Connection

Since 9DoF sensors are usually used for orientation and detecting movement, you'll normally want to securely connect the breakout to something before using it.

The pinout below shows how you can connect the NXP Precision 9DoF Breakout to any Adafruit Feather development board. The image below uses the Bluefruit nRF52 Feather (http://adafru.it/vAx), which is a great MCU to combine with the NXP Precision 9DoF since the ARM Cortex M4F has a lot of processing power, and Bluetooth Low Energy makes it easy to get the orientation data onto your phone or computer without any cables getting in the way!



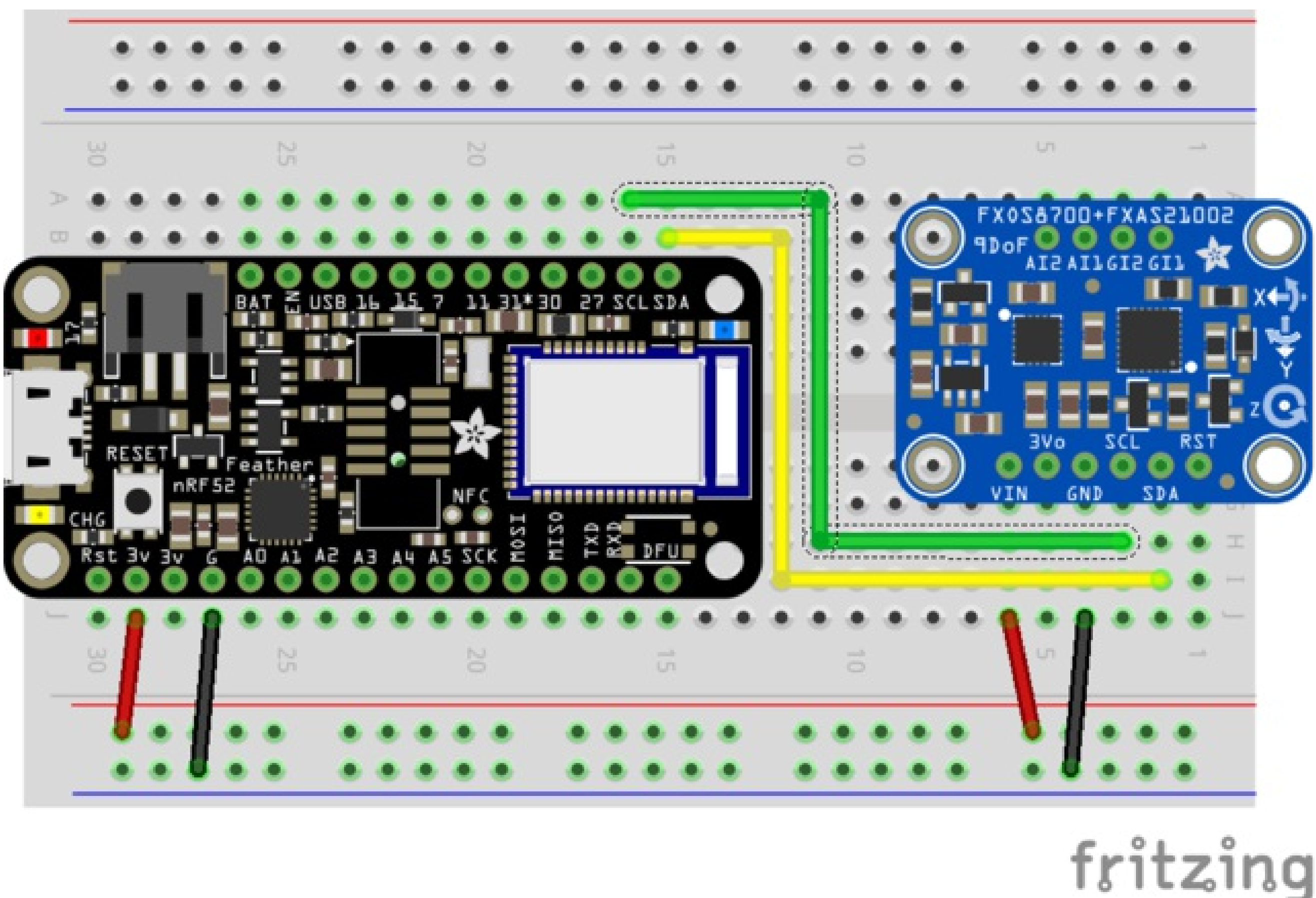
FXOS8700+FXAS21002
9DoF
AI2 AI1 GI2 GI1
3Vo SCL RST
VIN GND SDA
BAT EN USB 16 15 7 11 31 30 27 SCL SDA
RESET
Feather
nRF52
NFC
CHG
Rst 3v 3v G A0 A1 A2 A3 A4 A5 SCK MOSI MISO TXD RXD DFU
fritzing

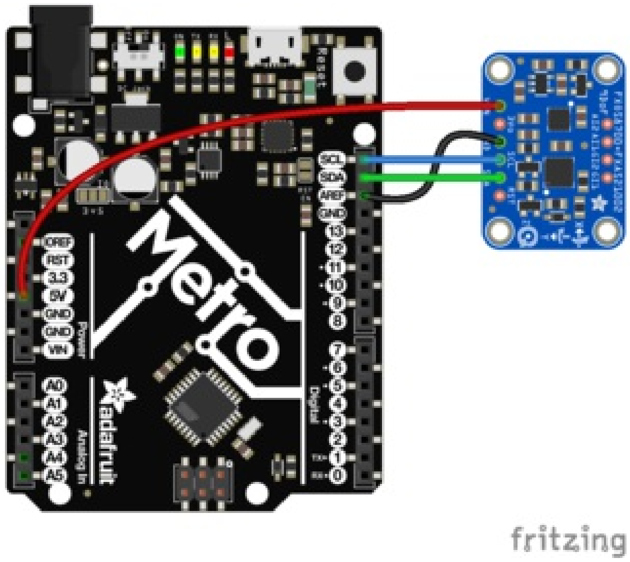
Wiring & Test

Breadboard Connection

Since 9DoF sensors are usually used for orientation and detecting movement, you'll normally want to securely connect the breakout to something before using it.

The pinout below shows how you can connect the NXP Precision 9DoF Breakout to any Adafruit Feather development board. The image below uses the Bluefruit nRF52 Feather (http://adafru.it/vAx), which is a great MCU to combine with the NXP Precision 9DoF since the ARM Cortex M4F has a lot of processing power, and Bluetooth Low Energy makes it easy to get the orientation data onto your phone or computer without any cables getting in the way!



- 

But you can also use with an Arduino-compatible. Just make sure you connect **Vin** to 3-5V, **GND** to ground, and **SCL + SDA** to your microcontroller's I2C pin

Required Arduino Libraries

The following libraries can all be installed from the Arduino Library Manager:

- Adafruit_FXOS8700 (http://adafru.it/vAz)
- Adafruit_FXAS21002C (http://adafru.it/vAA)
- Adafruit_Sensor (http://adafru.it/aZm)

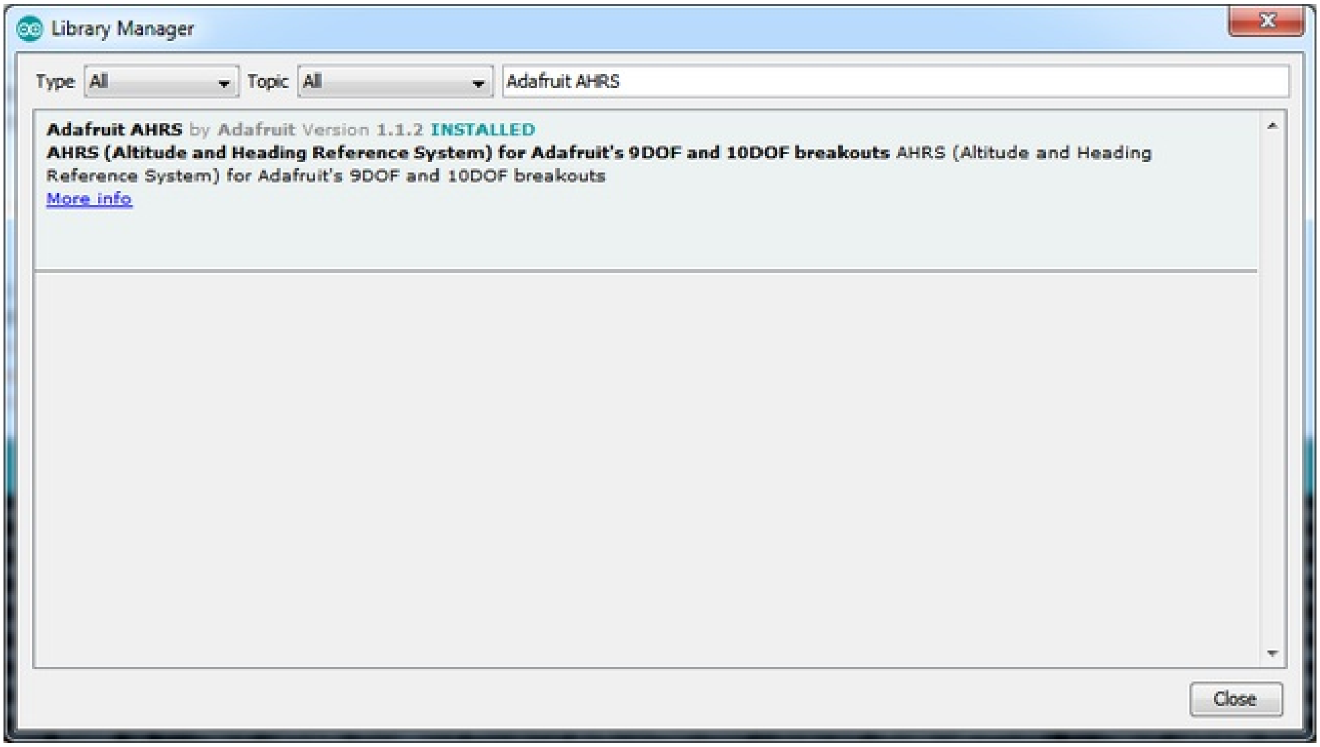
Optionally, if you wish to generate orientation data from the raw sensor outputs (recommended!), you will also need the following library:

- Adafruit_AHRS (http://adafru.it/dNO)

Installing the Libraries

The libraries mentioned above are already available in the Arduino Library Manager, and should be installed there to facilitate version tracking and easy software updates.

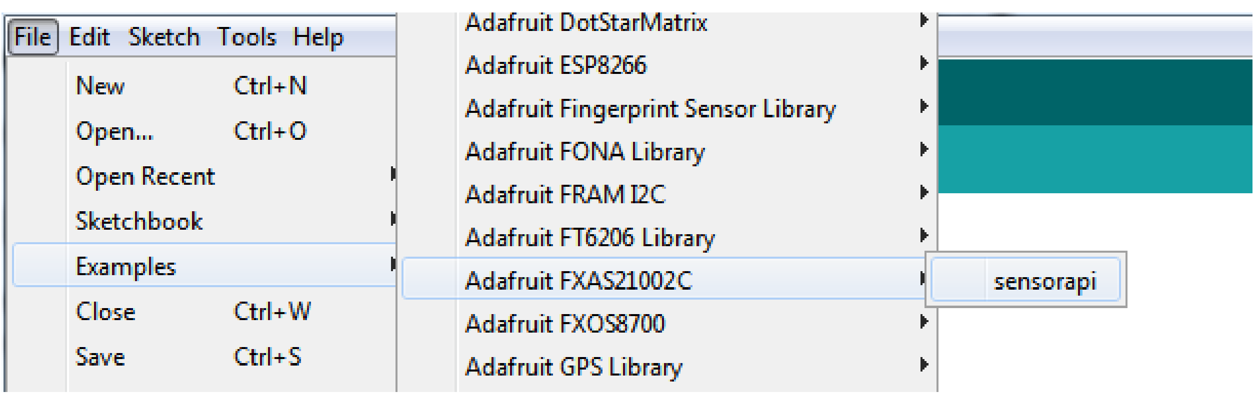
Open up the **Library Manager...** through the menu **Sketch->Include Library->Library Manager...**. Then type in "Adafruit AHRS", "Adafruit Sensor", etc., to locate and install the libraries



Once you are done installing all 4 libraries, quit and re-start the Arduino IDE.

Testing the Sensors and Library Installation

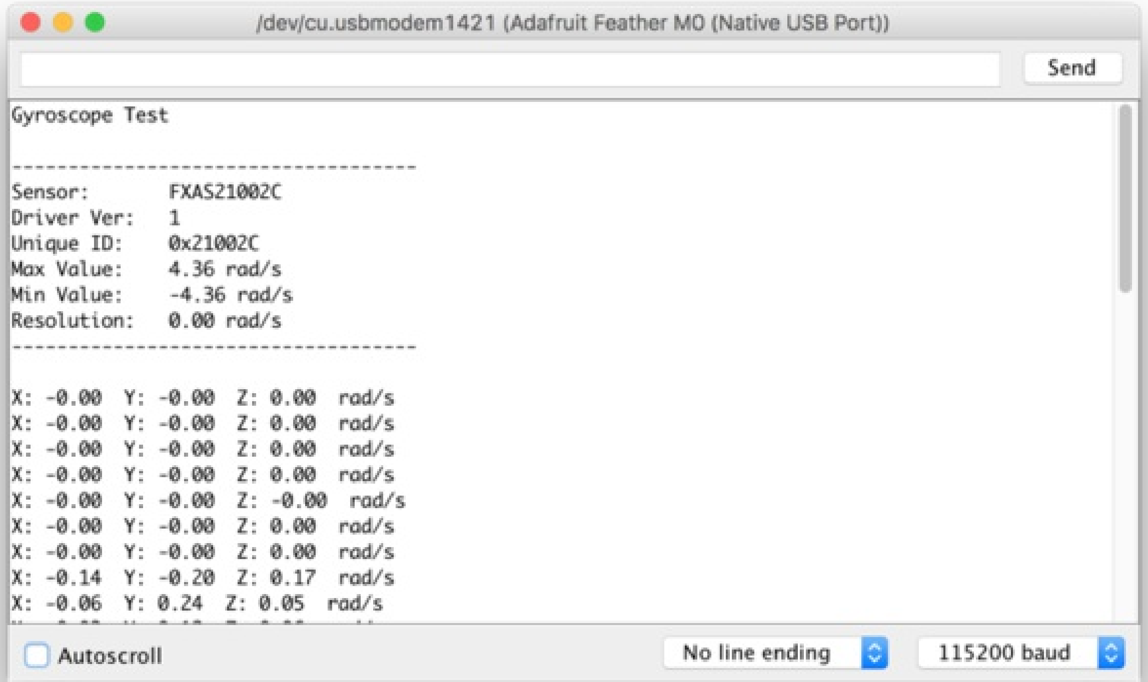
The **Adafruit_FXOS8700** and **Adafruit_FXAS21001C** repositories both contain a single example called **sensorapi** which demonstrates how to get raw sensor using the Unified Sensor Library (Adafruit_Sensor):



Both of these examples require Adafruit_Sensor to be installed on your system! If you're getting compilation errors, make sure you have it installed!

Load either of these examples, flash the sketch to your board, and then open the**Serial**

Monitor and if everything is connected correctly you should see something resembling the following output (for the Gyroscope in this example):



Try moving around the board, spinning it or tilting it, to see the data change with motion!

If you see the sensor data shown above, everything is properly setup and connected, you can continue onto the next steps

Calibration (USB)

Calibration is an important part of any orientation system. Any magnetometer has two sources of error called Soft Iron Error and Hard Iron Error.

Soft Iron Error is generally caused by local magnetic interference from specific metals, causing a deviation in magnitude or direction. Soft iron errors will cause what should be 'spherical' data to be deformed into a pill shape. Correcting for soft-iron error is the process of transforming the pill shaped output back into a spherical form using matrix multiplication with the soft-iron coefficients against the raw x/y/z data.

Hard Iron Error is a caused by local magnetic fields, which 'add' to the existing magnetic field, causing a specific change in position relative to the ideal, centered spherical output you would see in a perfect environment with a perfect sensor. This is a static set of values for x/y/z that are applied to 'recenter' the magnetometer output.

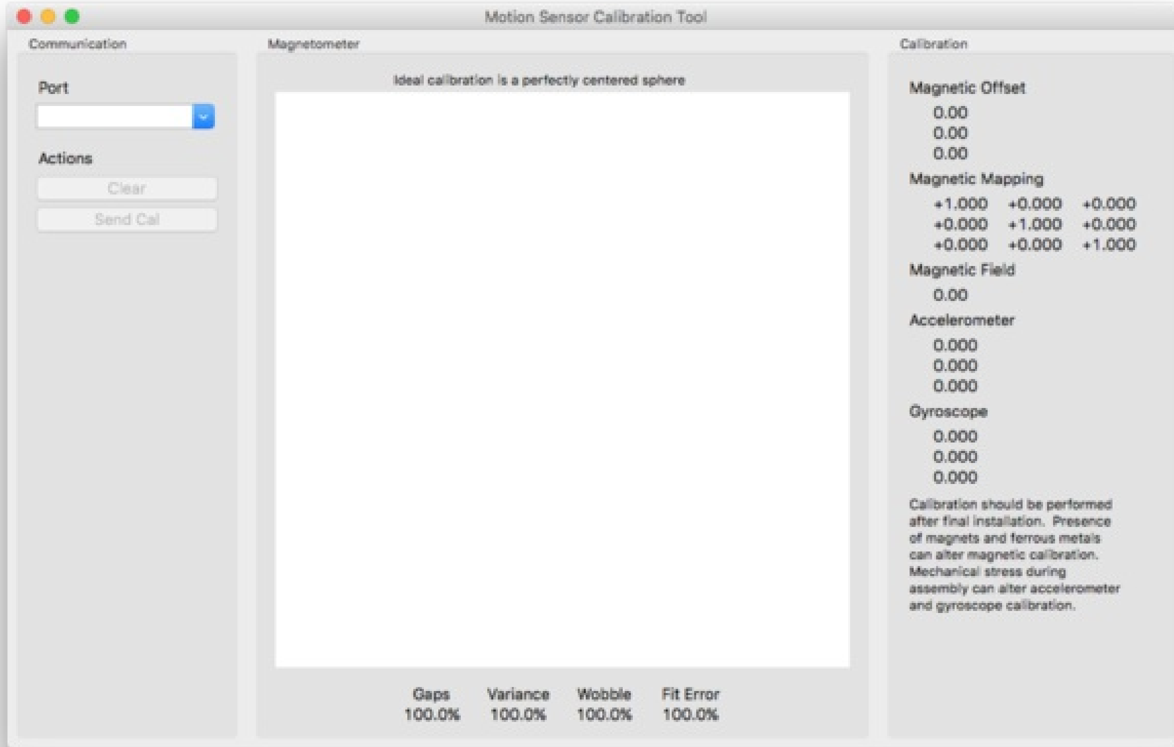
Calibration should always be done in the final environment and project setup (enclosure, etc.) where the sensor will be used, since much of the calibration process involves compensating out local environmental variables!

Generating Calibration Data

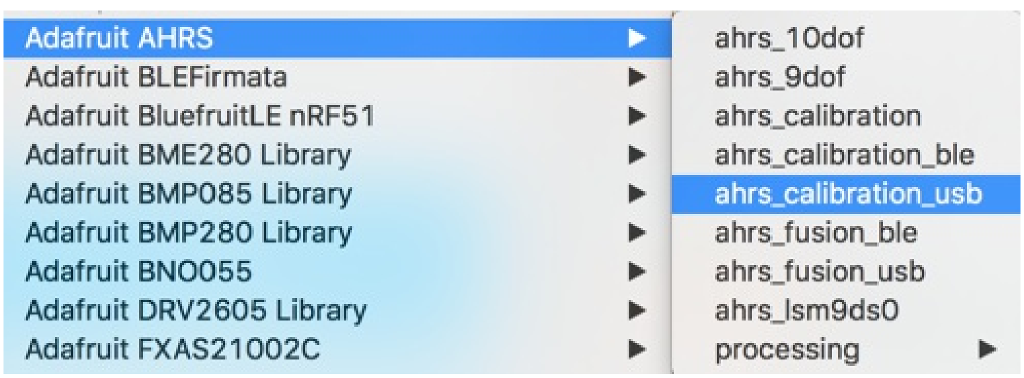
To determine the soft iron and hard iron error coefficients for your specific sensors and environment using a USB cable, you can make use of the MotionCal app from PJRC (http://adafru.it/vAH).

Download and run the appropriate binary for your operating system, and place it somewhere convenient.

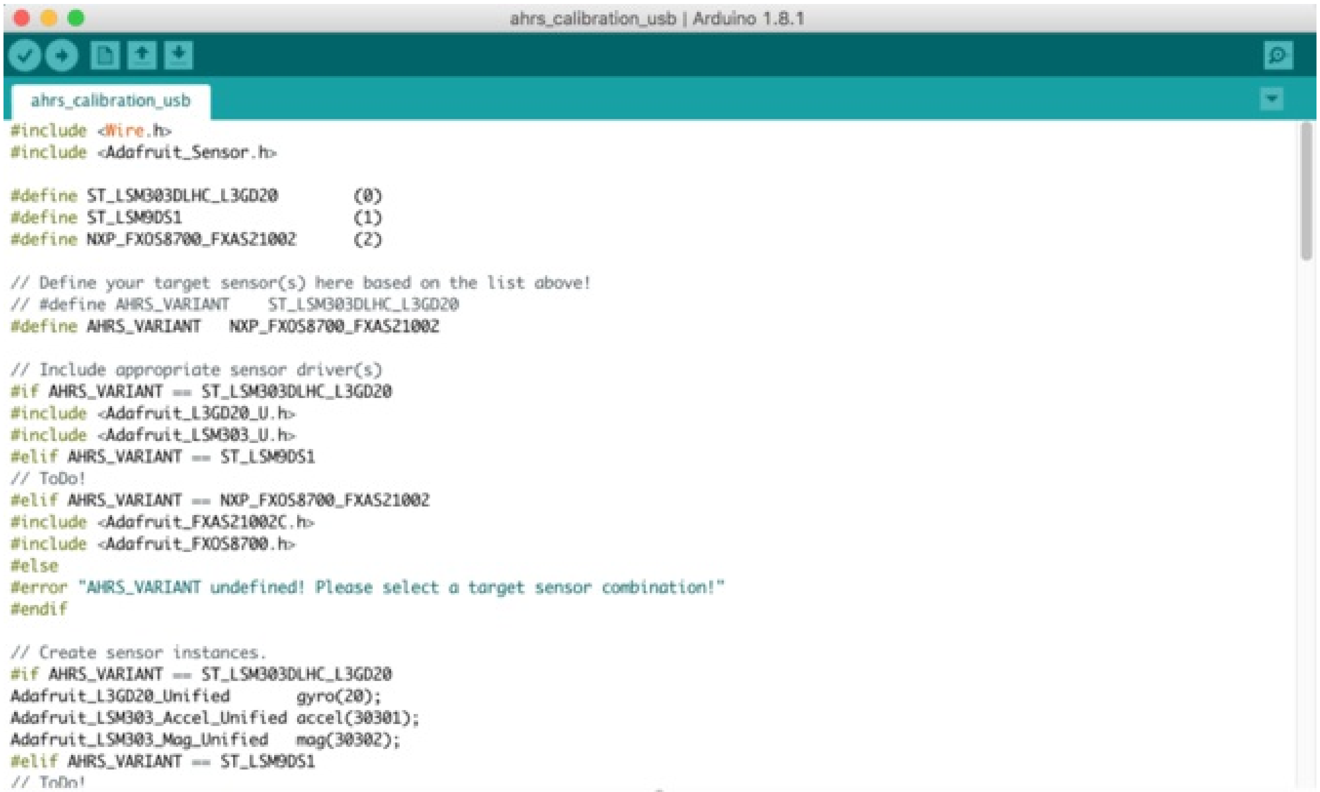
When you run the app, by default you will see something like this:



Next, with your development board and the NXP Precision 9DoF connected, load the **ahrs_calibration_usb** sketch from **Adafruit_AHRS** (source code on Github (http://adafru.it/vAI)):



You should see something resembling the following code:

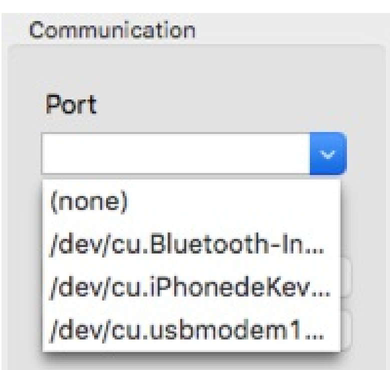


Make sure that you have the correct sensor target selected via the **AHRS_VARIANT** macro, which should be set as shown in the code below:

```
// Define your target sensor(s) here based on the list above!
// #define AHRS_VARIANT   ST_LSM303DLHC_L3GD20
#define AHRS_VARIANT   NXP_FXOS8700_FXAS21002
```

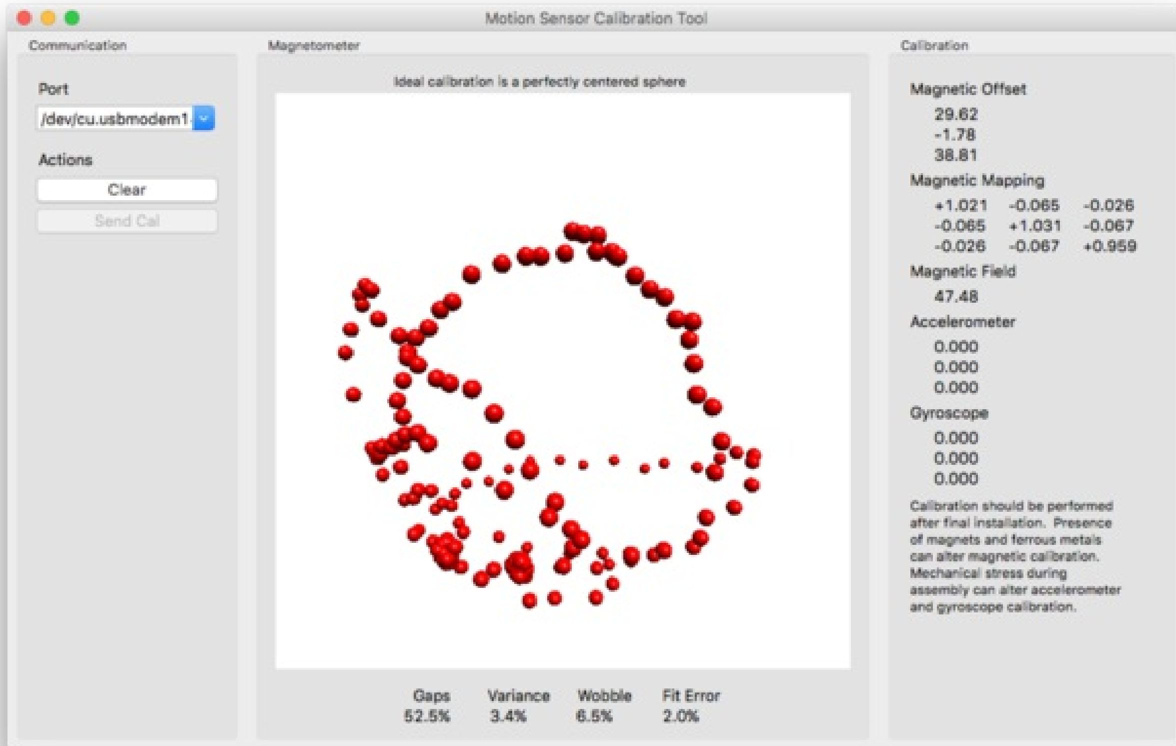
Build and flash the sketch, and then **without opening the Serial Monitor** go back to the MotionCal app.

In MotionCal select the correct serial port from the drop-down list in the top-left-hand corner:

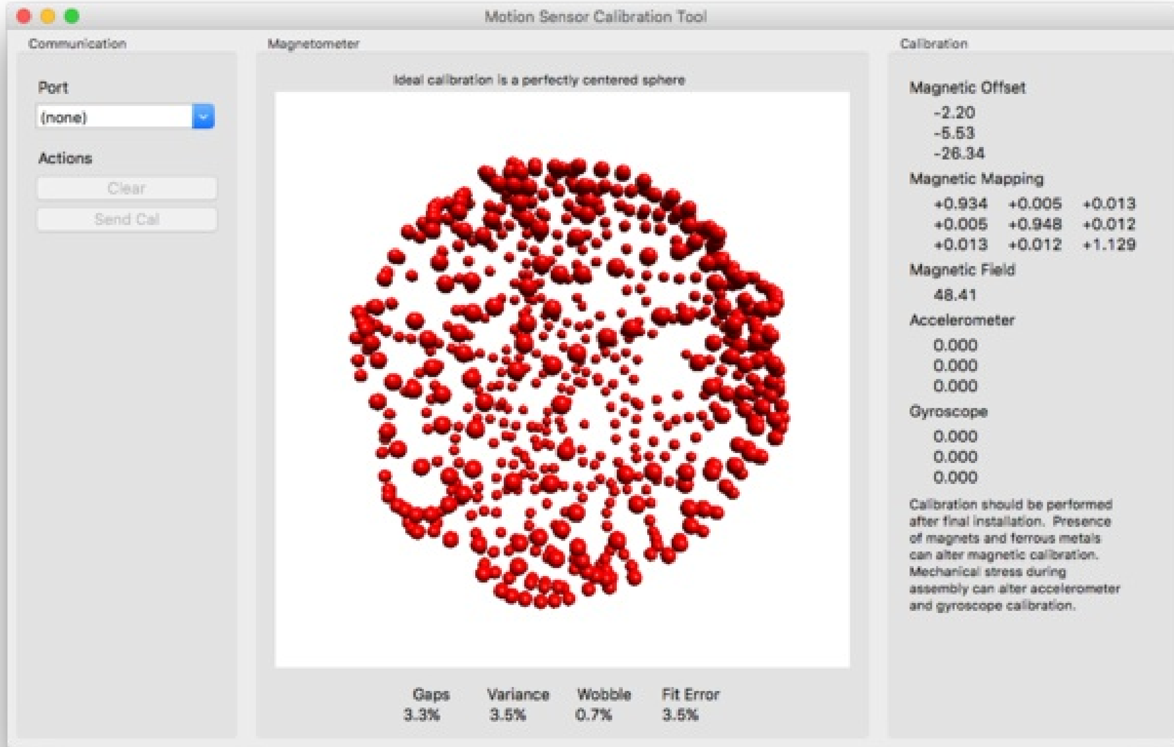


Once you are connected, start rotating the orientation sensors, and red dots will start to show up on the screen, and the soft iron error (Magnetic Mapping) and hard iron error

(Magnetic Offset) will start to populate themselves:



Once you have a reasonably complete sphere (check the error percentages on the bottom of the screen!), make note of the Magnetic field values on the right-hand side:



Make Note of the Compensation Coefficients

The values generated above represent the coefficients that you will need to apply when running the **ahrs_fusion_usb** sketch (code on Github (http://adafru.it/vAJ)).

Note these values down (save a screenshot, for example!), and then plug them into **ahrs_fusion_usb** using the following setup (the values below are taken from the screenshot above):

```
// Offsets applied to raw x/y/z mag 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;

// Offsets applied to compensate for gyro zero-drift error for x/y/z
```

```
float gyro_zero_offsets[3]      = { 0.0F, 0.0F, 0.0F };
```

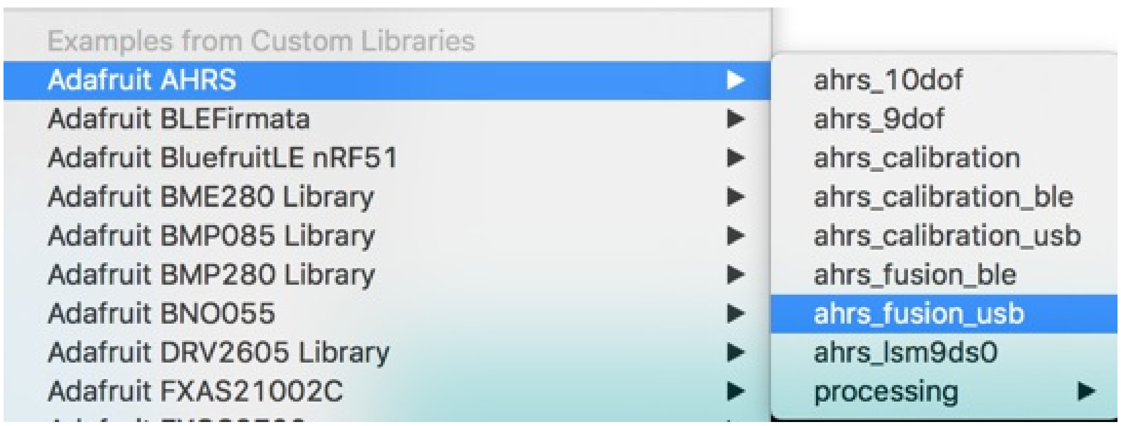
Orientation Test (USB)

The Adafruit_AHRS (http://adafru.it/dNO) repository contains everything you need to run a sensor fusion algorithm and get orientation data out of the NXP Precision 9DoF Breakout.

Generating Orientation Data

Before you can view or work with orientation, you first need to run the raw sensor output through something called a **sensor fusion algorithm**. As the name implies, this takes the output from a variety of sensors, and merges the results into orientation output, typically in Euler angles or Quaternions.

To generate orientation data, load the **ahs_fusion_usb** sketch (code on Github (http://adafru.it/vAI)) in the Arduino IDE:

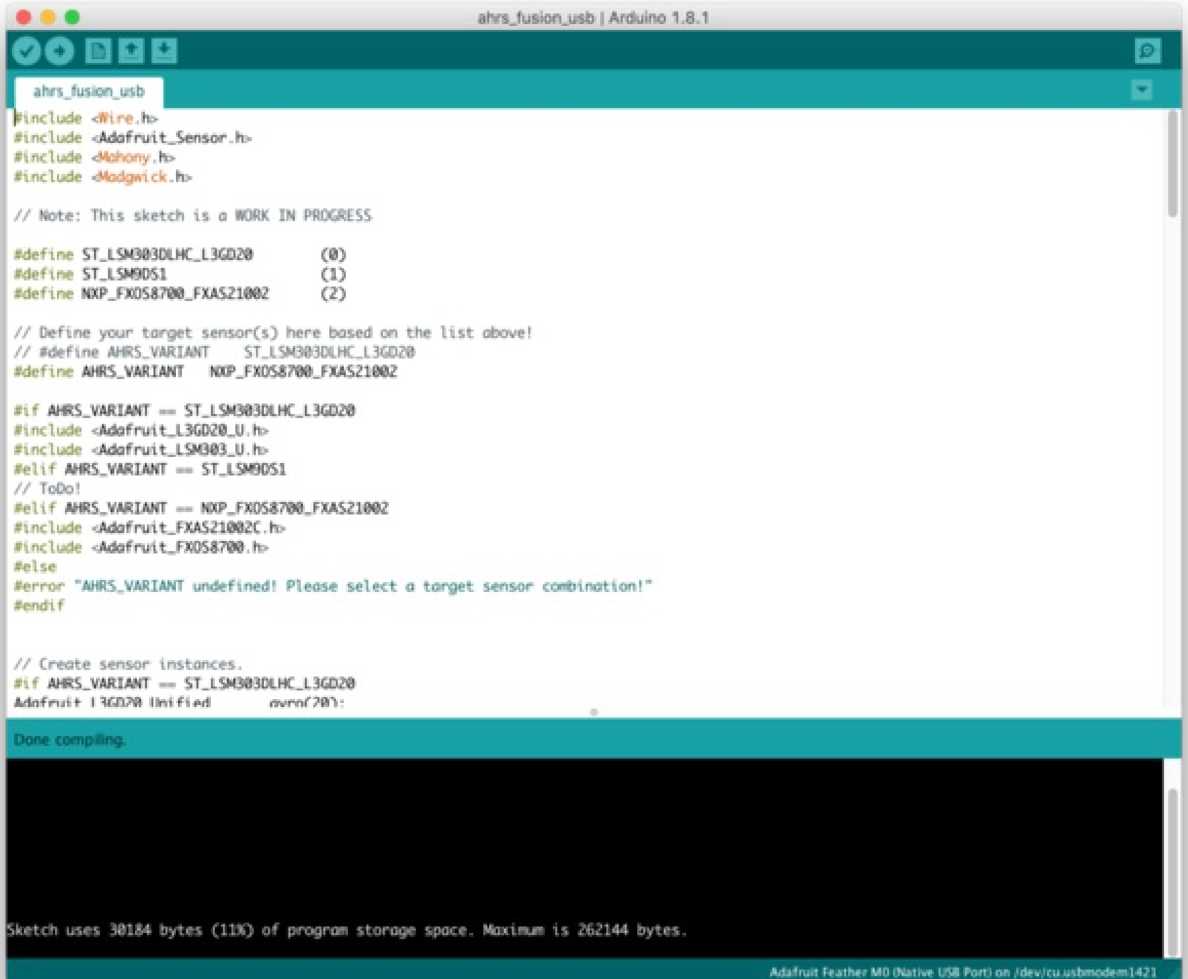


With this sketch loaded, make sure that you are targeting the right set of sensors, since this sketch can be used with a variety of different sensor models.

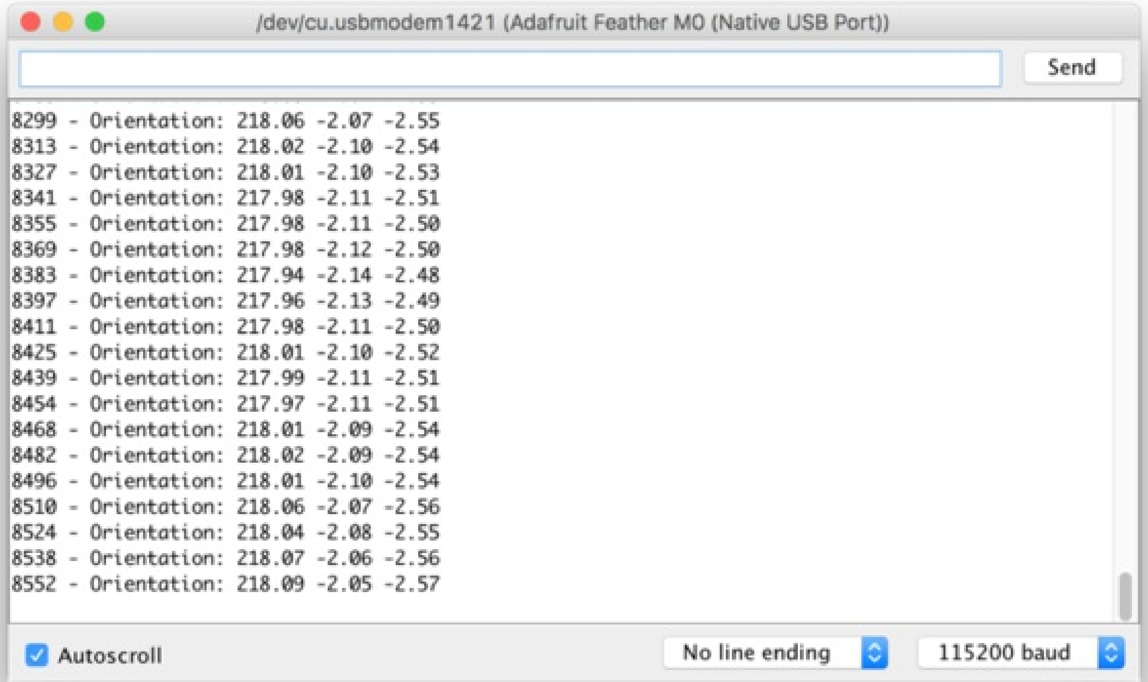
Make sure that you have selected **NXP_FXOS8700_FXAS21002** as the **AHRS_VARIANT** macro:

```
// Define your target sensor(s) here based on the list above!
// #define AHRS_VARIANT    ST_LSM303DLHC_L3GD20
#define AHRS_VARIANT   NXP_FXOS8700_FXAS21002
```

Compiling your sketch should produce output similar to this, though the final file size will vary depending on the platform you are compiled against:



You can test the output of this sketch by opening the Serial Monitor, where you should see some orientation output in **Euler Angles** by default:



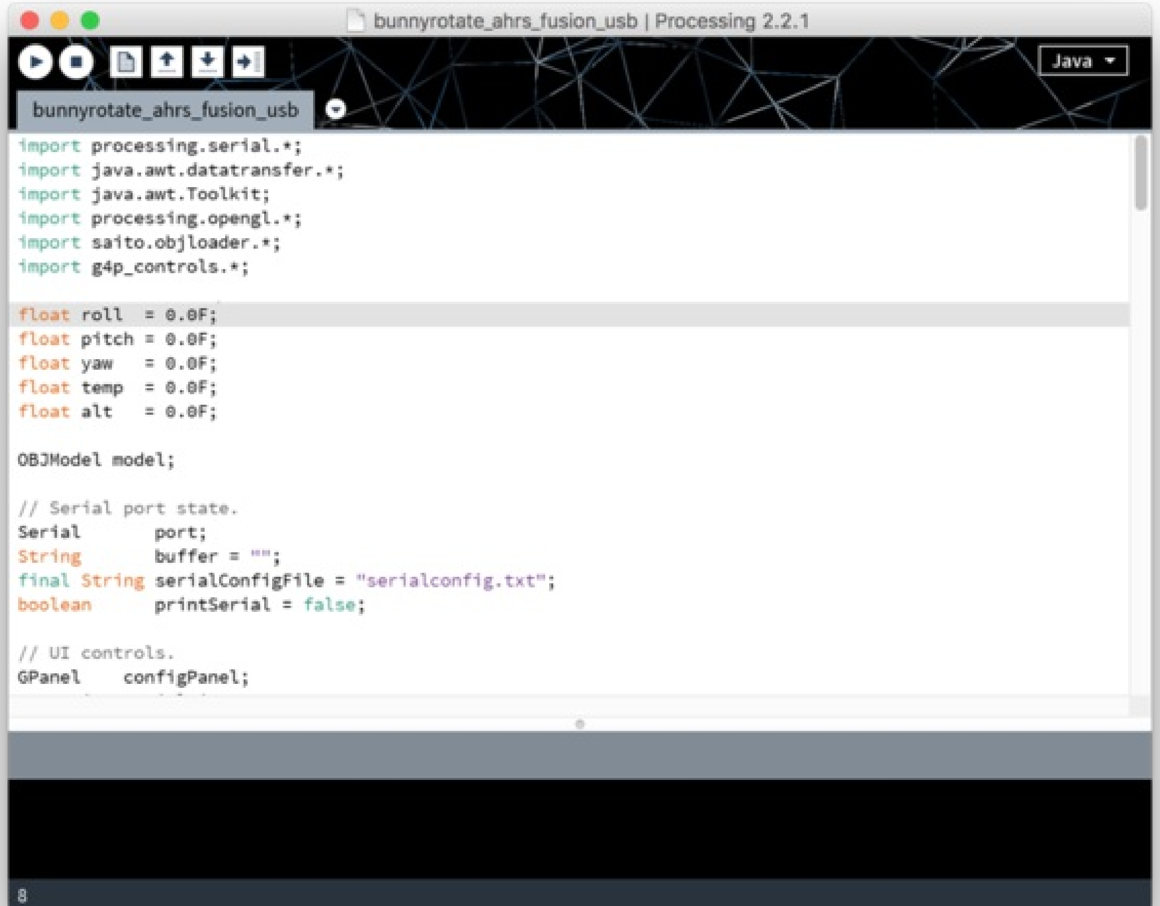
Visualizing the Orientation Data

Next, you can visualize the orientation data if you want using an optional Processing (http://adafru.it/ddm) sketch called **bunnyrotate_ahrs_fusion_usb.pde** (source on Github (http://adafru.it/vCq)).

You can find the Processing sketch in ~/Documents/Arduino/libraries/Adafruit_AHRS/processing or you can click here to download the entire zip and extract the subdirectory (http://adafru.it/dNQ)

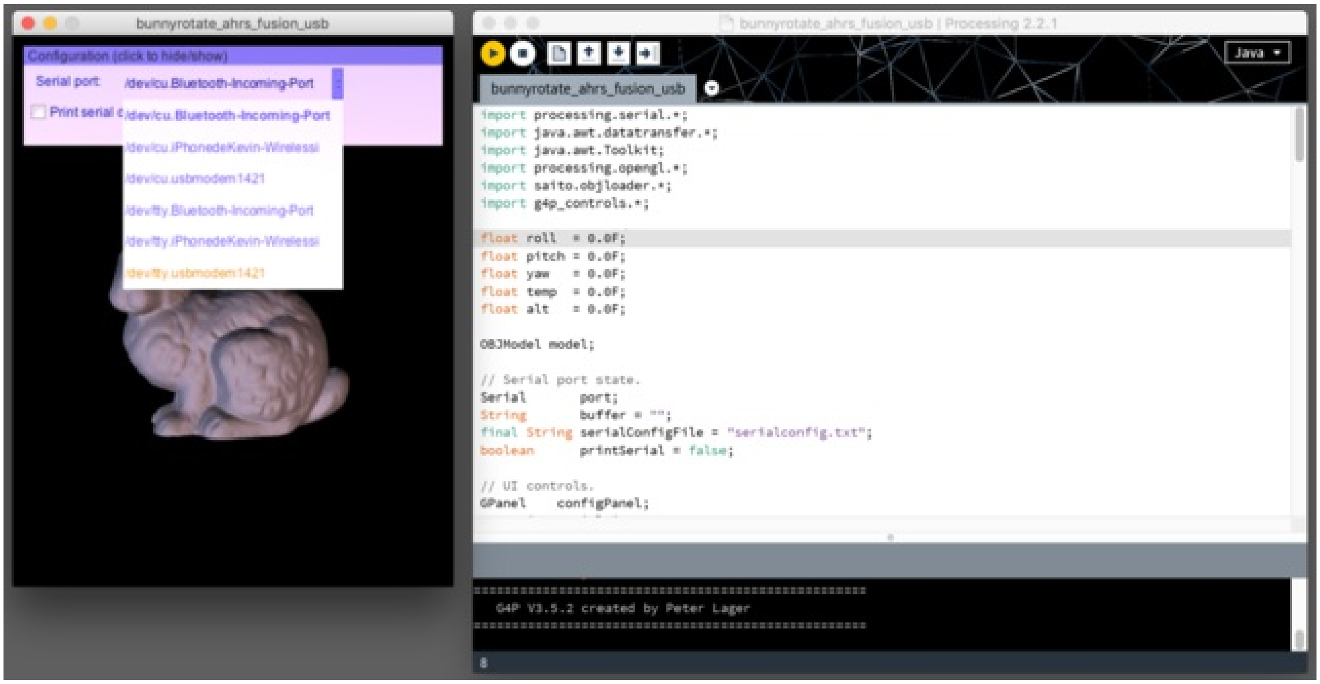
With the **ahrs_fusion_usb** sketch running in the background, as describe in the section above, open the **bunnyrotate_ahrs_fusion_usb** sketch in Processing, and run it.

Make sure that the Arduino Serial Monitor is closed before running the Processing app, or the serial port that provides the orientation data won't be available to Processing!

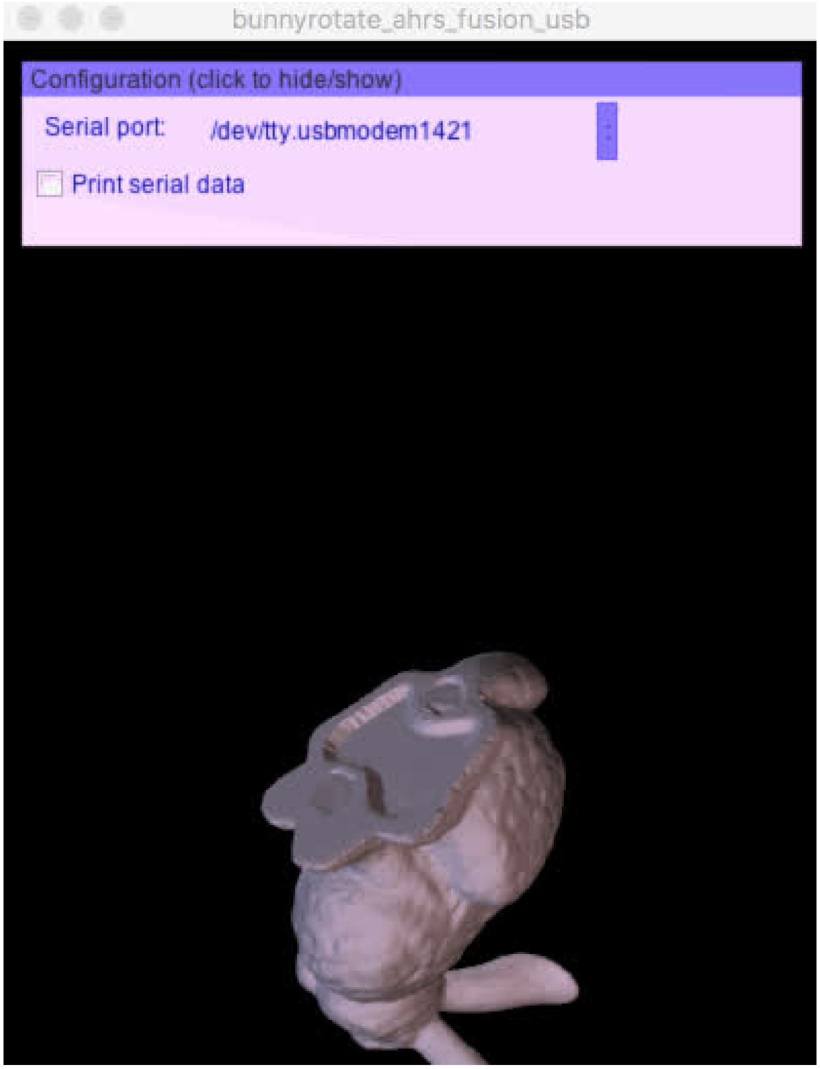


When you run the sketch, you will see a window pop up with a drop down box on the top section where you can select a serial port.

Select the serial port generated by your development board, and the Processing sketch should start capturing incoming aw orientation data. As the orientation data changes, the bunny on the screen will rotate along with the NXP Precision 9DoF Breakout:



As you rotate the board, the bunny should follow the movement similar to the .gif below:


bunnyrotate_ahrs_fusion_usb
Configuration (click to hide/show)
Serial port:
/dev/tty.usbmodem1421
Print serial data

Downloads

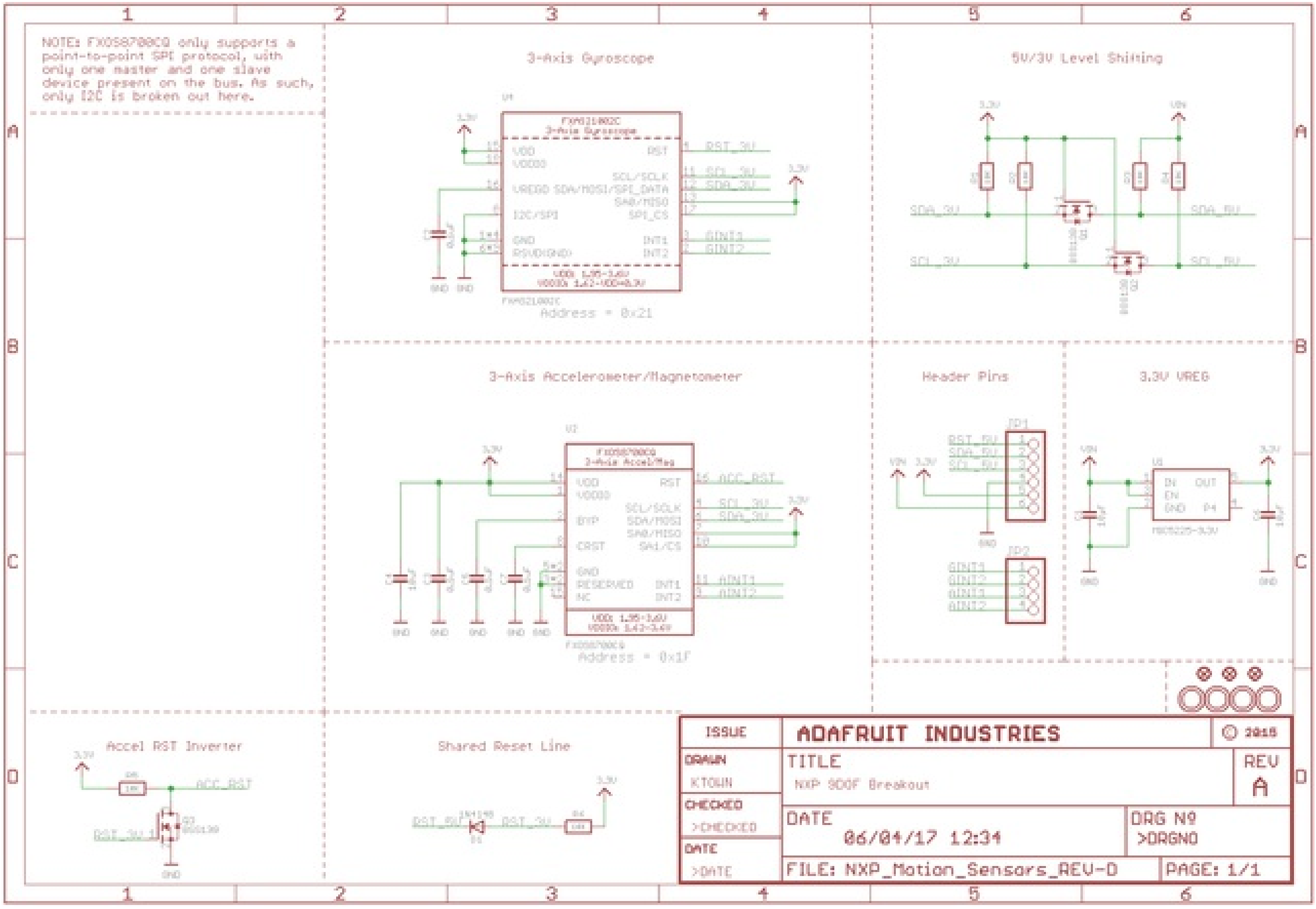
Files

- FXAS21002 Datasheet (http://adafru.it/vAL)
- FXOS8700CQ Datasheet (http://adafru.it/xsc)
- EagleCAD PCB files on GitHub (http://adafru.it/vAN)
- Fritzing object in Adafruit Fritzing library (http://adafru.it/aP3)

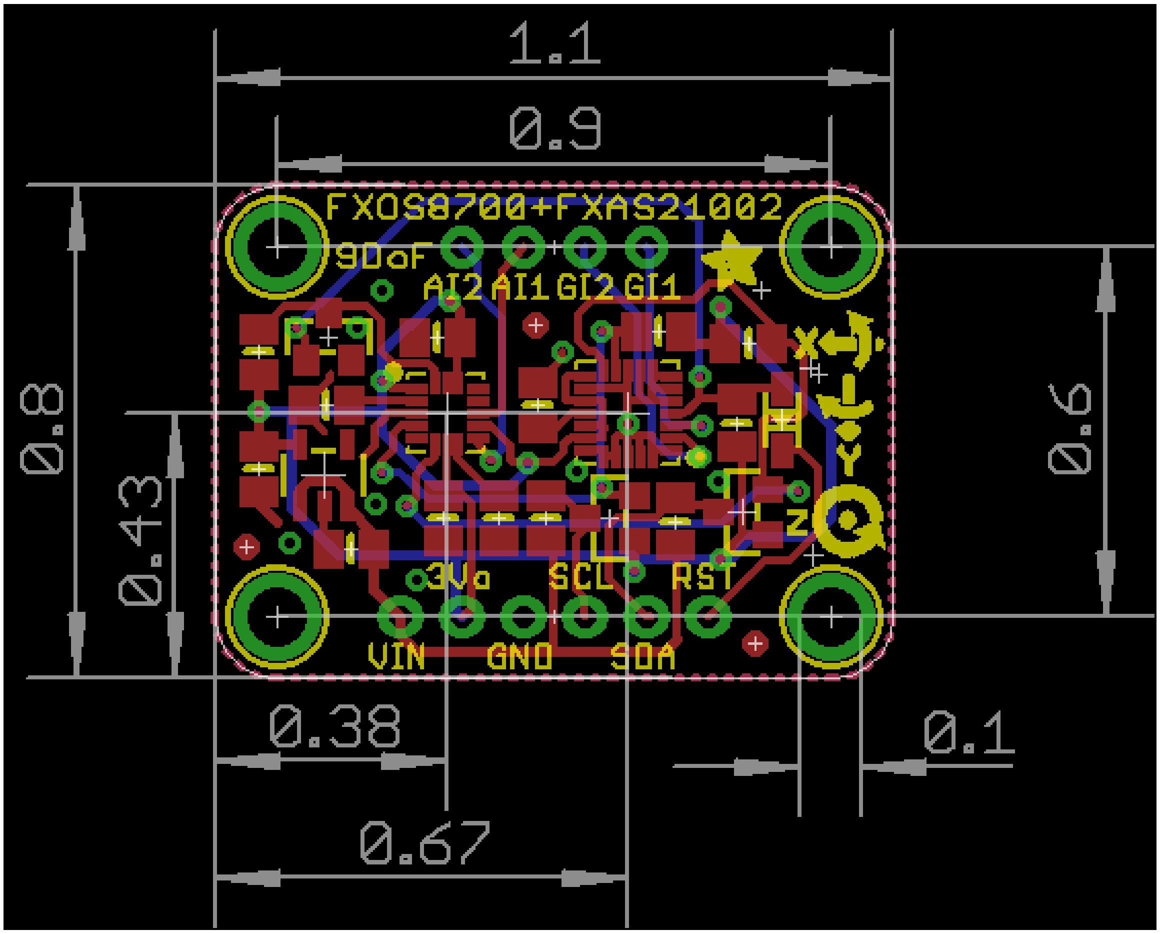
Arduino Libraries (Github)

- Adafruit_FSOX8700 (http://adafru.it/vAz)
- Adafruit_FXAS21002C (http://adafru.it/vAA)
- Adafruit_AHRS (http://adafru.it/dNO)

Schematic



Board Dimensions


1.1
0.9
0.8
0.43
0.6
FXOS8700+FXAS21002
9DoF
AI2 AI1 GI2 GI1
3Vo SCL RST
VIN GND SDA
0.38
0.67
0.1