



BNO055 Absolute Orientation Sensor with Raspberry Pi & BeagleBone Black

Created by Tony DiCola



<https://learn.adafruit.com/bno055-absolute-orientation-sensor-with-raspberry-pi-and-beaglebone-black>

Last updated on 2022-12-01 02:34:41 PM EST

Table of Contents

Overview	3
Hardware	3
• Parts	
• Wiring	
• Raspberry Pi	
• BeagleBone Black	
Software	7
• Installation	
• Usage	
WebGL Example	12
• Dependencies	
• Start Server	
• Sensor Calibration	
• Usage	

Overview

Are you looking for an easy way to detect orientation, or how something is rotated? You could use a [9](http://adafru.it/1714) (<http://adafru.it/1714>) or [10-degree of freedom](#) () breakout that includes an accelerometer, gyroscope, magnetometer, and more but what do you do once you have all that sensor data? Unless you're an expert in Kalman filters and advanced math you'll need some way to fuse all the noisy raw sensor data into an accurate orientation reading. Luckily the BNO055 absolute orientation sensor can handle all the tricky sensor fusion for you using a little on-board ARM processor.

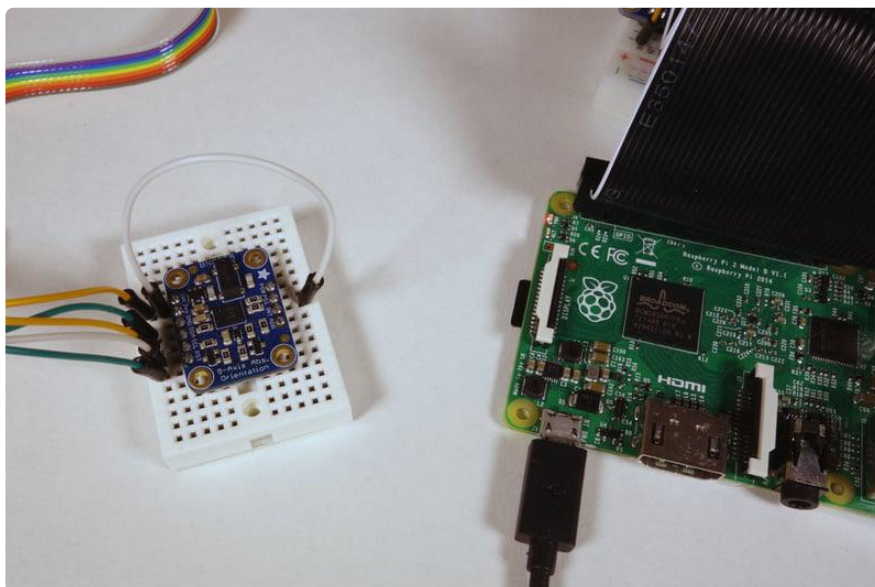
Using a Python module you can talk to the BNO055 sensor from your Raspberry Pi or BeagleBone Black and bring the magic of orientation sensing to your next project!

In this guide I'll show how to connect a BNO055 absolute orientation sensor to a Raspberry Pi or BeagleBone Black. I'll also show how to use a Python module to talk to the sensor and get orientation readings. Finally I'll walk through a demonstration of the sensor that uses WebGL to display and rotate a 3D model on a webpage based on the orientation of the BNO055 sensor.

Before you get started you'll want to read the [BNO055 guide](#) (), and make sure you're familiar with connecting to a Raspberry Pi or BeagleBone Black [using SSH](#) ().

When you're ready continue on to learn what parts you'll need and how to connect the BNO055 to your hardware.

Hardware



Parts

You'll need the following parts to build this project:

- [Adafruit BNO055 absolute orientation sensor breakout. \(\)](#)
- [Raspberry Pi \(\)](#) (any model like A, B, A+, B+, Pi 2, etc. will work) or [BeagleBone Black \(\)](#) (any revision).
- [Breadboard \(http://adafru.it/65\)](http://adafru.it/65) & [hookup wires \(\)](#).
- [Soldering tools. \(\)](#)

Follow the [BNO055 guide to assemble the breakout \(\)](#) by soldering headers to the board. If you're using the Raspberry Pi be sure to solder both rows of headers as you'll need to access the PS0 and PS1 pins.

Wiring

Follow the steps below based on the board you're using to connect the BNO055 to your hardware.

Raspberry Pi

Normally the BNO055 is connected to a device using its I2C interface, however on the Raspberry Pi the BNO055's use of I2C clock stretching will cause problems with a [hardware I2C clock stretching bug in the Raspberry Pi \(\)](#). To work around this clock stretching issue you can instead connect to the BNO055 using its serial UART mode.

To use the UART mode of the BNO055 make sure you've soldered both the top and bottom row of headers to the breakout board (i.e. both the VIN, GND, SDA, SCL, etc and PS0, PS1, etc. rows of connections). You'll need to connect the PS1 pin from the top row to 3.3V power to put the BNO055 into UART mode. Once in UART mode the BNO055's SCL and SDA pins will become serial RX and TX pins.

When using a serial UART device with the Raspberry Pi you'll need to make sure you disable the kernel's use of the Pi's serial port. Normally when the Pi kernel boots up it will put a login terminal on the serial port, however if you connect a device like the BNO055 to this serial port it will get confused by the login terminal. Luckily it's easy to disable the kernel's use of the serial port by using the raspi-config tool.

To disable the kernel serial port connect to the Raspberry Pi in a terminal ([using SSH \(\)](#)) and launch the raspi-config tool by running:

```
sudo raspi-config
```

Navigate the menus to the Advanced Options -> A8 Serial option and when prompted if you want a login shell over the serial port select No. Then select the Finish menu option to exit raspi-config.

Note on Raspbian Jessie you might also need to execute the following to disable the login terminal service on the serial port:

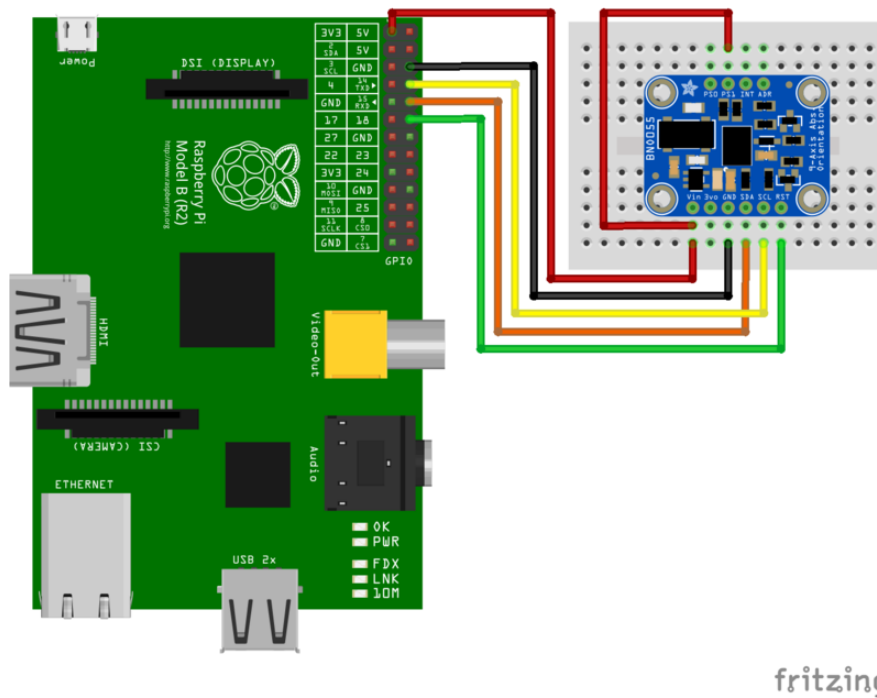
```
sudo systemctl disable serial-getty@ttyAMA0.service
```

Finally run the reboot command to reboot the Pi and make the change take effect:

```
sudo reboot
```

If you'd ever like to re-enable the kernel serial port in the future use the same raspi-config menu option to enable the login serial port and re-run the systemctl command but change disable to enable.

Once the Pi has rebooted and the serial port is disabled, follow the diagram below to wire up the BNO055 to the Raspberry Pi.

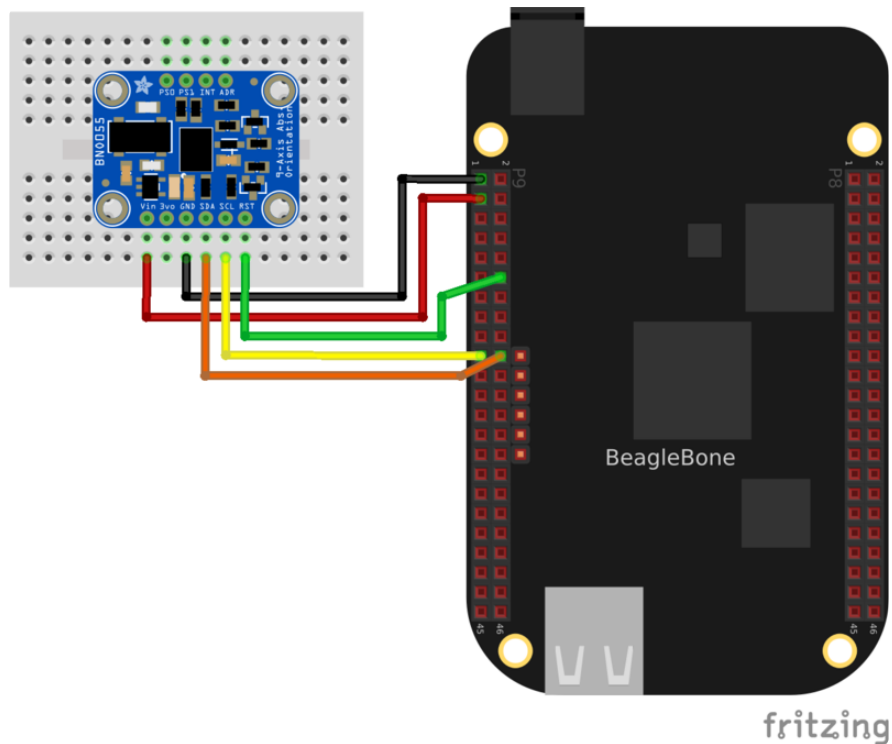


- Connect BNO055 Vin to Raspberry Pi 3.3V power.
- Connect BNO055 GND to Raspberry Pi ground.
- Connect BNO055 SDA (now UART TX) to Raspberry Pi RXD pin.
- Connect BNO055 SCL (now UART RX) to Raspberry Pi TXD pin.
- Connect BNO055 PS1 to BNO055 Vin / Raspberry Pi 3.3V power.

BeagleBone Black

On a BeagleBone Black you can use the BNO055's I2C communication mode as the hardware fully supports I2C clock stretching. Connect your BeagleBone Black to the BNO055 as follows.

If you aren't familiar with how pins are numbered on the BeagleBone Black be sure to [read this guide \(\)](#) first.



- Connect BNO055 Vin to BeagleBone Black 3.3V power pin P9_3.
- Connect BNO055 GND to BeagleBone Black ground pin P9_1.
- Connect BNO055 SDA to BeagleBone Black I2C2_SDA pin P9_20.
- Connect BNO055 SCL to BeagleBone Black I2C2_SCL pin P9_19.

Continue on to learn how to install the BNO055 Python module and use the sensor.

Software

Installation

To install the software for this project you'll need to make sure your Raspberry Pi is running the latest [Raspbian](#) () operating system, or your BeagleBone Black is running the latest [Debian](#) () operating system. Make sure your board is connected to the internet through a wireless or wired network connection too.

Connect to your board in a command line terminal and run the following commands to install the necessary dependencies:

```
sudo apt-get update
sudo apt-get install -y build-essential python-dev python-smbus python-pip git
```

Next run the following commands to download and install the latest version of the [BN0055 Python module code from GitHub \(\)](#):

```
cd ~
git clone https://github.com/adafruit/Adafruit_Python_BNO055.git
cd Adafruit_Python_BNO055
sudo python setup.py install
```

If you see the installation fail with an error message go back and check the dependencies above were installed and try again. Also make sure your board is connected to the internet as the installation will download and install some required Python modules.

Once the module is installed you're ready to start using it. See the section below for details on running the examples and using the BNO055 module.

Usage

To learn how to use the BNO055 Python module I'll walk through running the included `simpletest.py` example below. Before you get started make sure you've wired up the sensor and installed the library code following the steps above.

Also note if you're using the sensor on a Raspberry Pi or other device in UART mode be sure you've [followed the steps to disable the kernel from accessing the serial port \(\)](#) at the same time as the BNO055 sensor!

Connect to the board in a command terminal and navigate to the library examples folder by running the following command:

```
cd ~/Adafruit_Python_BNO055/examples
```

Before you run the example you might need to change the code to initialize the BNO055 sensor depending on how it's connected to your board. Open the file using the nano text editor by running:

```
nano simpletest.py
```

Then navigate down to this section of the example code:

```
# Create and configure the BNO sensor connection. Make sure only ONE of the
# below 'bno = ...' lines is uncommented:
# Raspberry Pi configuration with serial UART and RST connected to GPIO 18:
```



```
 bno = BN0055.BN0055(serial_port='/dev/ttyAMA0', rst=18)
# BeagleBone Black configuration with default I2C connection (SCL=P9_19, SDA=P9_20),
# and RST connected to pin P9_12:
#bno = BN0055.BN0055(rst='P9_12')
```

Like the comments mention you'll want to leave only one of the `bno = ...` lines uncommented depending on how it's connected to your board. For a Raspberry Pi that's using the serial port and GPIO 18 as the reset pin leave this line uncommented:

```
 bno = BN0055.BN0055(serial_port='/dev/ttyAMA0', rst=18)
```

If you're using a different serial port or GPIO for the reset line then adjust the `serial_port` and `rst` parameters of the initializer appropriately.

However for a BeagleBone Black that's using the hardware I2C bus and pin P9_12 as the reset pin leave only this line uncommented:

```
 bno = BN0055.BN0055(rst='P9_12')
```

If you're using a different I2C bus you can specify it by adding a `bus=` parameter to the call, like `bus=0` for using the `/dev/i2c0` bus. You can also change the reset pin to any free digital GPIO (just be careful that the chosen pin is available as a GPIO with the current [device tree overlays](#) ())--if in doubt stick with the default P9_12 pin).

For either the Raspberry Pi or BeagleBone Black If you're not using the reset line with the sensor you can remove the `rst=...` part of the initializer call. In this mode a software reset command will be sent to reset the board instead of using the hardwired reset pin.

Once you've finished modifying the file save it and exit nano by pressing Ctrl-S, enter, and then Ctrl-X.

Now run the code as a root user with `sudo` by executing:

```
sudo python simpletest.py
```

If everything is working correctly you should see something like the following after a few seconds:

```
System status: 5
Self test result (0x0F is normal): 0x0F
Software version: 776
Bootloader version: 21
Accelerometer ID: 0xFB
```

```
Magnetometer ID:    0x32
Gyroscope ID:      0x0F

Reading BNO055 data, press Ctrl-C to quit...
Heading=0.00 Roll=0.00 Pitch=0.00   Sys_cal=0 Gyro_cal=0 Accel_cal=0 Mag_cal=0
Heading=0.00 Roll=-0.69 Pitch=0.81  Sys_cal=0 Gyro_cal=3 Accel_cal=0 Mag_cal=0
Heading=0.00 Roll=-0.69 Pitch=0.81  Sys_cal=0 Gyro_cal=3 Accel_cal=0 Mag_cal=0
Heading=0.00 Roll=-0.69 Pitch=0.81  Sys_cal=0 Gyro_cal=3 Accel_cal=0 Mag_cal=0
Heading=0.00 Roll=-0.69 Pitch=0.81  Sys_cal=0 Gyro_cal=3 Accel_cal=0 Mag_cal=0
Heading=0.00 Roll=-0.69 Pitch=0.81  Sys_cal=0 Gyro_cal=3 Accel_cal=0 Mag_cal=0
```

If you see an error message carefully check that the BNO055 is connected to the hardware as shown on the previous page. Make sure the software and its dependencies are installed as shown above too.

If you receive errors on a Raspberry Pi in UART mode make sure you've disabled the kernel's use of the serial port as mentioned above and on the previous page. Also don't try to use the BNO055 in I2C mode with the Raspberry Pi as the Pi's I2C hardware has buggy behavior with I2C clock stretching that the BNO055 sensor uses.

If you receive errors on a BeagleBone Black in I2C mode make sure you don't have any [device tree overlays](#) () loaded that might have changed the behavior of the I2C pins. The stock BeagleBone Black device tree configuration should expose the I2C and GPIO pins that are used in this guide.

Once the example code is running you can see it start out by printing diagnostic details about the BNO055 sensor. You can learn more about the sensor status and other values in [its datasheet](#) (), but a status of 5 means the fusion algorithm is running and a self test result of 0x0F means all of the sensors are working as expected.

Every second the orientation data for the sensor is printed as [Euler angles](#) () that represent the heading, roll, and pitch of the sensor in degrees.

In addition to orientation the calibration level of each sensor is printed each second. Calibrating the BNO055 sensor is very important to ensure you get good orientation readings. You can see the system (the fusion algorithm) and each sensor has a separate calibration level. A level of 0 is uncalibrated and 3 is fully calibrated (with 1 and 2 being levels of partial calibration). You'll learn more about [how to calibrate the sensor](#) () on the next page.

To learn how to use the library I'll walk through the code for it in a little more detail. Stop running the `simpletest.py` example (by pressing Ctrl-C) and then open the file in a text editor (like nano). You can see the file starts by importing some required dependencies and the `Adafruit_BNO055` module:

```
import logging
import sys
import time

from Adafruit_BNO055 import BNO055
```

Next the code creates and configures a BNO055 sensor instance. You've already seen this part of the code described above when preparing to run the example.

After creating and configuring the BNO055 sensor the code optionally turns on Python's logging module output (if a -v parameter is passed when running the script--this is useful to see the raw commands sent and received with the BNO055 sensor), and then initializes the BNO055 sensor.

```
# Enable verbose debug logging if -v is passed as a parameter.
if len(sys.argv) == 2 and sys.argv[1].lower() == '-v':
    logging.basicConfig(level=logging.DEBUG)

# Initialize the BNO055 and stop if something went wrong.
if not bno.begin():
    raise RuntimeError('Failed to initialize BNO055! Is the sensor connected?')
```

It's very important to initialize the BNO055 sensor by calling the begin() function. This function will return true if it succeeds in initializing the sensor and false if it fails for some reason. The function might also throw an exception that provides more details about why the sensor failed to initialize.

Next you can see how status and diagnostic information is retrieved using the get_system_status() and get_revision() functions.

```
# Print system status and self test result.
status, self_test, error = bno.get_system_status()
print('System status: {0}'.format(status))
print('Self test result (0x0F is normal): 0x{0:02X}'.format(self_test))
# Print out an error if systemstatus is in error mode.
if status == 0x01:
    print('System error: {0}'.format(error))
    print('See datasheet section 4.3.59 for the meaning.')

# Print BNO055 software revision and other diagnostic data.
sw, bl, accel, mag, gyro = bno.get_revision()
print('Software version: {0}'.format(sw))
print('Bootloader version: {0}'.format(bl))
print('Accelerometer ID: 0x{0:02X}'.format(accel))
print('Magnetometer ID: 0x{0:02X}'.format(mag))
print('Gyroscope ID: 0x{0:02X}\n'.format(gyro))
```

Now the main loop of the program runs. You can see the read_euler() function returns a tuple of heading, roll, and pitch data. In addition the get_calibration_status() function is called to retrieve the calibration level for each part of the sensors.

```
print('Reading BNO055 data, press Ctrl-C to quit...')
while True:
    # Read the Euler angles for heading, roll, pitch (all in degrees).
    heading, roll, pitch = bno.read_euler()
    # Read the calibration status, 0=uncalibrated and 3=fully calibrated.
    sys, gyro, accel, mag = bno.get_calibration_status()
    # Print everything out.
    print('Heading={0:0.2F} Roll={1:0.2F} Pitch={2:0.2F}\tSys_cal={3} Gyro_cal={4}
    Accel_cal={5} Mag_cal={6}'.format(
        heading, roll, pitch, sys, gyro, accel, mag))
```

That's really all you need to use to initialize and get orientation from the BNO055 sensor! However the code continues with commented examples of how to read other data from the sensor, including:

- Orientation as a quaternion (a useful orientation representation that fixes some problems with Euler angles, see [more information from Wikipedia \(\)](#)).
- Temperature of the sensor (in degrees Celsius).
- Raw magnetometer reading for the X, Y, Z axis (in micro-Teslas).
- Raw gyroscope reading for the X, Y, Z axis (in degrees per second).
- Raw accelerometer reading for the X, Y, Z axis (in meters per second squared).
- Linear acceleration reading for the X, Y, Z axis (in meters per second squared). This is the acceleration from movement of the object, not from the downward pull of gravity.
- Gravity acceleration reading for the X, Y, Z axis (in meters per second squared). This is just the force of gravity on the object.

In addition you can view a description of the BNO055 library functions using the pydoc tool:

```
pydoc Adafruit_BNO055.BNO055
```

If you run into problems with the library or would like to contribute then check out the [library's home on GitHub \(\)](#) for issues and sending pull requests.

Continue to the next page to learn how to run a more complex BNO055 example that rotates a 3D model on a webpage using the sensor.

WebGL Example

Included with the BNO055 library is an example of how to send orientation readings to a webpage and use it to rotate a 3D model. Follow the steps below to setup and run this example.

Dependencies

First you need to install the [flask Python web framework \(\)](#). Connect to your board in a command terminal and run the following command (assuming you've already followed the previous steps to [install the BNO055 library and its dependencies \(\)](#)):

```
sudo pip install flask
```

You will also need to be using a web browser that [supports WebGL \(\)](#) on your computer or laptop. I recommend and have tested the code for this project with the latest version of [Chrome \(\)](#).

Start Server

Next navigate to the webgl_demo example folder by running:

```
cd ~/Adafruit_Python_BNO055/examples/webgl_demo
```

You will need to edit the server.py file and modify the bno setup lines near the top [just like you did for the simplestest.py example on the previous page \(\)](#). Be sure to leave only one bno = ... line uncommented depending on how the BNO055 is connected to your hardware.

Now run the server.py web server by executing:

```
sudo python server.py
```

You should see text like the following after the server starts running:

```
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
* Restarting with stat
```

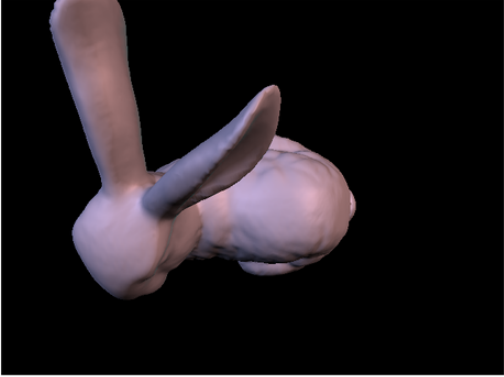
If you see an error message carefully check you've [installed the BNO055 library and the simplestest.py example works \(\)](#). Also ensure the [flask web framework \(\)](#) has been installed and try again.

Now open a web browser on your computer and navigate to your board's IP address or hostname on port 5000. For example on a Raspberry Pi [http://raspberrypi:5000/ \(\)](http://raspberrypi:5000/) might work, or on a BeagleBone Black [http://beaglebone:5000/ \(\)](http://beaglebone:5000/) is the URL to try. If

neither URL works you'll need to [look up the IP address of your device \(\)](#) and then access it on port 5000. For example if your board has the IP address 192.168.1.5 you would access [http://192.168.1.5:5000/ \(\)](http://192.168.1.5:5000/).

Once the page loads you should see something like the following:

Adafruit BNO055 Absolute Orientation Sensor Demo



<p>Orientation (degrees): Heading = 359.9375 Roll = -0.6875 Pitch = -89.25</p>	<p>Calibration: (0=uncalibrated, 3=fully calibrated) System = 0 Gyro = 3 Accelerometer = 0 Magnetometer = 0</p>	<p>Actions:</p> <p>Model: <input type="text" value="Bunny"/></p> <p><input type="button" value="Straighten"/></p> <p><input type="button" value="Save Calibration"/></p> <p><input type="button" value="Load Calibration"/></p>
--	---	---

If you move the BNO055 sensor you should see the 3D model move too. However when the demo first runs the sensor will be uncalibrated and likely not providing good orientation data. Follow the next section to learn how to calibrate the sensor.

Sensor Calibration

The video below shows an overview of calibrating the BNO055 sensor. Following the video is a more detailed description of the calibration process.

To use the BNO055 sensor you'll need to make sure it's calibrated every time the sensor is powered on or reset. Luckily the BNO055 takes care of most calibration for you, but you will need to move the sensor in certain ways to complete the calibration.

Section 3.10 of the [datasheet \(\)](#) has all the details on calibration, but in general you can follow the steps below to calibrate each sensor.

First notice the bottom middle column of the web page shows the current calibration status of the BNO055 sensor. There are 4 parts of the sensor that are individually calibrated, the system (or fusion algorithm), gyroscope, accelerometer, and magnetometer. Each component has a calibration level from 0 to 3 where 0 is uncalibrated and 3 is fully calibrated. Ideally you want all 4 components to be at least

a calibration level of 3 to get the best orientation data. However you should still get reasonable results if a few of the sensors and the system are calibrated to level 2 or 3.

- Gyroscope

- The gyroscope is the easiest to calibrate and will most likely be fully calibrated by the time you load the web page. To calibrate the gyroscope place the sensor down on a table and let it sit motionless for a few seconds.

- Magnetometer

- The magnetometer is slightly more complicated to calibrate. You need to pick up the BNO055 sensor and move it through a figure 8 or infinity pattern continuously until the magnetometer calibrates. In most cases the sensor will calibrate after about a dozen movements through the figure 8 pattern. Be careful as any large metal objects near the sensor could change or slow the calibration.

- Accelerometer

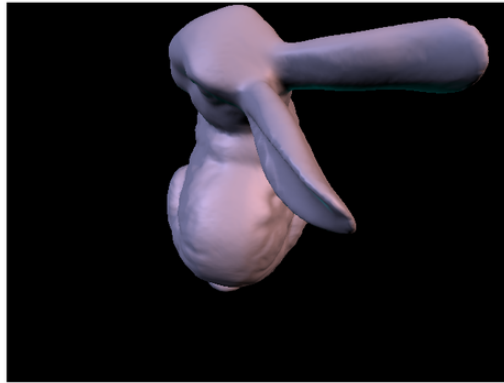
- The accelerometer calibration requires holding the sensor in about 6 different positions for a few seconds. Think of a cube and the 6 faces on it and try to slowly move the sensor between each face, holding it there for a few seconds. If the accelerometer is calibrating after about 3-4 faces you'll see its level jump from 0 to 1 and then up to 3 after moving to more faces.
- Another good way I've found to calibrate the accelerometer is to rotate the board along an axis and hold it for a few seconds at each 45 degree angle. You'll know the calibration is working when you see the calibration level go from 0 to 1 after holding at a few different 45 degree angles.

- System

- The system, or fusion algorithm, will calibrate once all the sensors have started to calibrate. You'll likely see the system calibration increase as each sensor finishes calibration. Once all the sensors have calibrated let the sensor sit for a few moments to finish calibrating the system.

Once the board is calibrated you should see each calibration level at a level of 3 like below:

Adafruit BNO055 Absolute Orientation Sensor Demo



Orientation (degrees):

Heading = 99.5625
Roll = -0.4375
Pitch = -89.375

Calibration:

(0=uncalibrated, 3=fully calibrated)
System = 3
Gyro = 3
Accelerometer = 3
Magnetometer = 3

Actions:

Model:

Bunny

Straighten

Save Calibration

Load Calibration

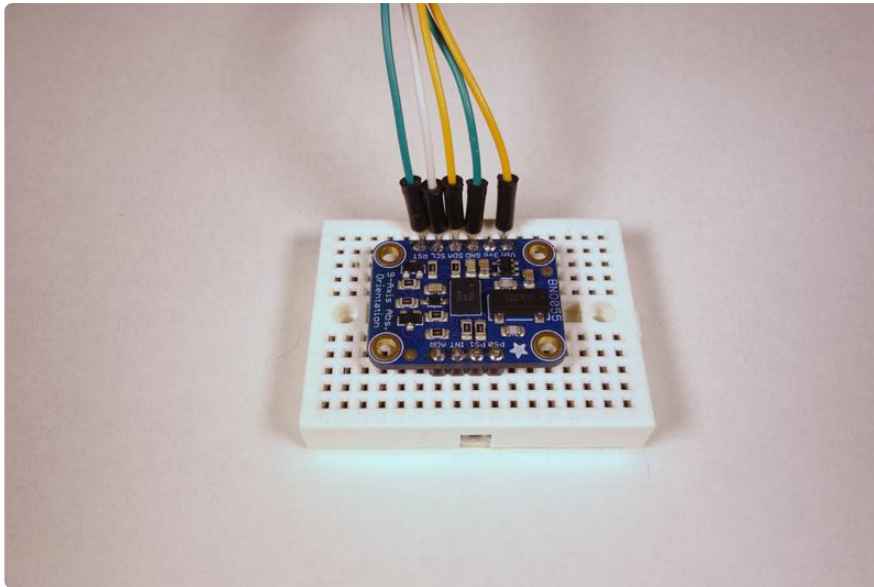
To save time in the future you can click the Save Calibration button on the right and the calibration data will be written to a calibration.json file. When you restart the server in the future press Load Calibration to load the file and its calibration. You might still need to calibrate the magnetometer again after loading calibration, but the accelerometer and system calibration generally happen much quicker from a loaded configuration.

Note that each time the sensor is powered on or reset (like when the server is run again) it will need to be calibrated again for best results! In your own scripts that use the BNO055 library after calibrating you can call the `get_calibration()` function and save the returned list of data (it will return 22 integers), then reload it later using the `set_calibration()` function of the library.

Usage

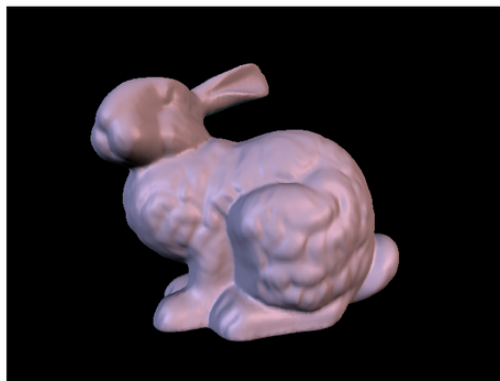
After calibrating the sensor you can move the board around and see the 3D model rotate. However you might notice the orientation and rotation of the 3D model doesn't exactly match that of the BNO055 sensor. This is because the axes of the BNO sensor need to be matched to the axes of the 3D model so that turning the sensor left/right turns the model left/right, etc.

You can line up the axes of the sensor and 3D model by using the Straighten button. First you'll need to place the BNO sensor in a very specific orientation. Place the sensor flat in front of you and with the row of SDA, SCL, etc. pins facing away from you like shown below:



Then click the Straighten button and you should see the 3D model snap into its normal position:

Adafruit BNO055 Absolute Orientation Sensor Demo



Orientation (degrees):

Heading = 99.5625
Roll = -0.4375
Pitch = -89.375

Calibration:

(0=uncalibrated, 3=fully calibrated)
System = 3
Gyro = 3
Accelerometer = 3
Magnetometer = 3

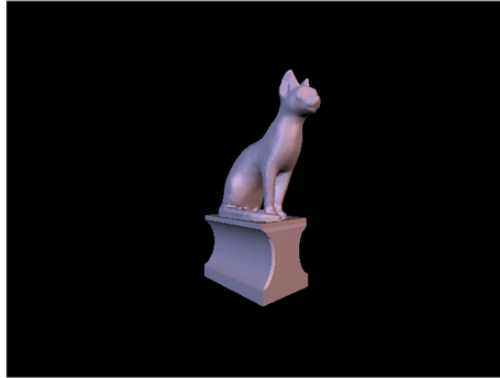
Actions:

Model:
Bunny

Now move the BNO055 sensor around and you should see your movements exactly matched by the 3D model!

You can also change the 3D model by clicking the Model drop down on the right and changing to a different model, like a cat statue:

Adafruit BNO055 Absolute Orientation Sensor Demo



Orientation (degrees):

Heading = 39.8125

Roll = -0.375

Pitch = -89.625

Calibration:

(0=uncalibrated, 3=fully calibrated)

System = 3

Gyro = 3

Accelerometer = 0

Magnetometer = 3

Actions:

Model:

Cat Statue

Straighten

Save Calibration

Load Calibration

That's all there is to using the BNO055 WebGL demo!

To stop the server go back to the terminal where it was started and press Ctrl-C. You might also need to run the following command to kill any Python process that remains running (sometimes if the browser is still running it can keep a zombie flask process running):

```
sudo pkill -9 python
```

Describing how all of the WebGL code works is a little too complex for this guide, however the high level components of the example are:

- [flask web service framework \(\)](#): This is a great, simple web framework that is used by server.py to serve the main index.html page and expose a few web service endpoints to read BNO sensor data and save/load calibration data.
- [HTML5 server sent events \(\)](#): This is how data is sent from the server to the webpage. With SSE a connection is kept open and data is pushed to the client web page. BNO sensor readings are taken and sent over SSE where they're use to update the orientation of the model. [This page \(\)](#) has a little more info on how to use HTML5 SSE with the flask framework (although it uses a more complex multiprocessing framework called [gevent \(\)](#) that isn't necessary for simple apps like this demo).
- [Three.js \(\)](#): This is the JavaScript library that handles all the 3D model rendering.
- [Bootstrap \(\)](#) & [jQuery \(\)](#): These are a couple other JavaScript libraries that are used for the layout and some core functionality of the page.

That's all there is to using the BNO055 WebGL demo. Enjoy using the BNO055 absolute orientation sensor in your own projects!