Bluetooth LE Sensor Nodes to Raspberry Pi WiFi Bridge

Created by John Park

https://learn.adafruit.com/bluetooth-le-broadcastnet-sensor-node-raspberry-pi-wifi-bridge

Last updated on 2023-08-29 04:25:19 PM EDT
# Table of Contents

## Overview
- Parts
- Optional Parts

## Prepare the CLUE
- Libraries
- Text Editor
- Code.py
- How It Works
- Measurements

## Prepare the Feather Sense
- Libraries
- Text Editor
- Code.py
- How It Works
- Measurements
- Other Boards

## Raspberry Pi Quick Setup
- SD Card Prep
- Power to the Pi
- Ping
- SSH
- Change the Password!
- Rename the Pi
- Update Application List
- Software Setup
- Make sure you’re using Python 3!
- Install Blinka

## Install Pi Bridge Software
- BLE Support
- Adafruit Blinka BLEio
- Bluetooth Group
- Install Python libraries
- Add Examples
- Adafruit IO
- Shhh Secrets
- Service Identity
- Launch BroadcastNet Bridge

## BroadcastNet on AIO
- Adafruit IO Getting Started
- Renaming AIO Feeds
- Dashboard Setup
- Blocks
- Measurement List
Overview

Use sensor-packed Bluefruit boards and a Raspberry Pi to build a network of sensors that send data to Adafruit IO!

This project takes advantage of CircuitPython running on the Feather Bluefruit Sense ( ) and CLUE boards () (any nRF52840 Adafruit board will work () and Python running on the Raspberry Pi.

The Pi receives advertised BLE sensor data from the Feather and CLUE and then sends that data over WiFi or Ethernet to your Adafruit IO feed.

It's a super easy way to make a wireless sensor node, using the built in wireless/networking capabilities of the Raspberry Pi!

Parts

You can use any of the Adafruit nRF52840 boards and sensors you like. We'll show how it works with the Feather Sense and CLUE boards, since they pack so many sensors right on board.
Adafruit Feather nRF52840 Sense
The Adafruit Feather Bluefruit Sense takes our popular Feather nRF52840 Express and adds a smorgasbord of sensors...
https://www.adafruit.com/product/4516

Adafruit CLUE - nRF52840 Express with Bluetooth LE
Do you feel like you just don't have a CLUE? Well, we can help with that - get a CLUE here at Adafruit by picking up this sensor-packed development board. We wanted to build some...
https://www.adafruit.com/product/4500

Raspberry Pi 4 Model B
The Raspberry Pi 4 Model B is the newest Raspberry Pi computer made, and the Pi Foundation knows you can always make a good thing better! And what could make the Pi 4 better...
https://www.adafruit.com/product/4297

Official Raspberry Pi Power Supply 5.1V 3A with USB C
The official Raspberry Pi USB-C power supply is here! And of course, we have 'em in classic Adafruit black! Superfast with just the right amount of cable length to get your Pi 4...
https://www.adafruit.com/product/4298
Raspberry Pi 3 - Model B+ - 1.4GHz Cortex-A53 with 1GB RAM
The Raspberry Pi 3 Model B is the most popular Raspberry Pi computer made, and the Pi Foundation knows you can always make a good thing better! And what could make the Pi 3...
https://www.adafruit.com/product/3775

USB cable - USB A to Micro-B
This here is your standard A to micro-B USB cable, for USB 1.1 or 2.0. Perfect for connecting a PC to your Metro, Feather, Raspberry Pi or other dev-board or...
https://www.adafruit.com/product/592

Raspberry Pi Zero W Basic Pack - Includes Pi Zero W
Remember those cereal commercials that would always say, "part of a complete breakfast"? Well the Pi Zero's a lot like that bowl of cereal - while it's...
https://www.adafruit.com/product/3409

Optional Parts
If you want to use, say, a Feather nRF52840 plus some plug-in STEMMA QT breakouts, here are some great options:
Adafruit Feather nRF52840 Express
The Adafruit Feather nRF52840 Express is the new Feather family member with Bluetooth Low Energy and native USB support featuring the nRF52840! It's...
https://www.adafruit.com/product/4062

SparkFun Qwiic / Stemma QT
FeatherWing (Shield for Thing Plus)
The SparkFun Qwiic FeatherWing (Shield for Thing Plus) provides an easy-to-assemble way to add the
https://www.adafruit.com/product/4515

STEMMA QT / Qwiic JST SH 4-pin Cable - 100mm Long
This 4-wire cable is a little over 100mm / 4" long and fitted with JST-SH female 4-pin connectors on both ends. Compared with the chunkier JST-PH these are 1mm pitch instead of...
https://www.adafruit.com/product/4210

STEMMA QT / Qwiic JST SH 4-Pin Cable - 50mm Long
This 4-wire cable is 50mm / 1.9" long and fitted with JST SH female 4-pin connectors on both ends. Compared with the chunkier JST PH these are 1mm pitch instead of 2mm, but...
https://www.adafruit.com/product/4399
Adafruit DPS310 Precision Barometric Pressure / Altitude Sensor
How high are you right now? If you had a precision altitude sensor, you would know for sure! The DPS310 sensor from Infineon a high-precision barometric sensor, perfect for measuring...
https://www.adafruit.com/product/4494

Adafruit PCT2075 Temperature Sensor - STEMMA QT / Qwiic
The Adafruit PCT2075 Temperature Sensor is a 'code compatible' drop-in replacement for a very...
https://www.adafruit.com/product/4369

Adafruit VCNL4040 Proximity and Lux Sensor - STEMMA QT / Qwiic
The VCNL4040 is a handy two-in-one sensor, with a proximity sensor that works from 0 to 200mm (about 7.5 inches) and light sensor with range of 0.0125 to 6553 lux. We've all...
https://www.adafruit.com/product/4161

Adafruit APDS9960 Proximity, Light, RGB, and Gesture Sensor
This breakout is chock full o' sensors! Add basic gesture sensing, RGB color sensing, proximity sensing, or ambient light sensing to your project with the Adafruit...
https://www.adafruit.com/product/3595
Adafruit BMP280 I2C or SPI Barometric Pressure & Altitude Sensor
Bosch has stepped up their game with their new BMP280 sensor, an environmental sensor with temperature, barometric pressure that is the next generation upgrade to the...
https://www.adafruit.com/product/2651

Adafruit BME280 I2C or SPI Temperature Humidity Pressure Sensor
Bosch has stepped up their game with their new BME280 sensor, an environmental sensor with temperature, barometric pressure and humidity! This sensor is great for all sorts...
https://www.adafruit.com/product/2652

Prepare the CLUE

We’ll set up the CLUE board to act as a sensor for the BroadcastNet.
First, set up CircuitPython on the CLUE following the instructions on this page.

**Libraries**
Next, install the libraries needed. This guide page will show you where to download them.

You'll need the following libraries for this project:
- adafruit_apds9960
- adafruit_ble
- adafruit_ble_broadcastnet
- adafruit_bmp280.mpy
- adafruit_bus_device
- adafruit_clue.mpy
- adafruit_register
- adafruit_lis3mdl.mpy
- adafruit_lsm6ds
- adafruit_sht31d.mpy
- neopixel.mpy

**Text Editor**
Adafruit recommends using the Mu editor for using your CircuitPython code with the CLUE board. You can get more info in this guide.

Alternatively, you can use any text editor that saves files.

**Code.py**
Copy the code shown below, paste it into Mu. Save the code from Mu to the CLUE's CIRCUITPY drive as code.py

```python
# SPDX-FileCopyrightText: 2020 John Park for Adafruit Industries
#
# SPDX-License-Identifier: MIT

"""This uses the CLUE as a Bluetooth LE sensor node.""

import time
from adafruit_clue import clue
import adafruit_ble_broadcastnet

print("This is BroadcastNet CLUE sensor:", adafruit_ble_broadcastnet.device_address)
```

©Adafruit Industries
while True:
    measurement = adafruit_ble_broadcastnet.AdafruitSensorMeasurement()
    measurement.temperature = clue.temperature
    measurement.pressure = clue.pressure
    measurement.relative_humidity = clue.humidity
    measurement.acceleration = clue.acceleration
    measurement.magnetic = clue.magnetic
    print(measurement)
    adafruit_ble_broadcastnet.broadcast(measurement)
    time.sleep(60)

How It Works

When this code runs on the CLUE board, it'll first import the `time`, `clue`, and `adafruit_ble_broadcastnet` libraries.

Then, it will print the unique ID for the board based on the board's BLE adapter's MAC address. This address will be used when messages are sent so there won't be any clashes when multiple microcontroller boards are in range of the BroadcastNet base station.

NOTE: this is not a secure connection, so think of this system as "broadcasting" in the truest sense of the word!

Measurements

Next, we will have the main loop of the program. In it, the board sends out an advertisement to alert the base station central device that it is there.

Next we collect the different sensor readings such as `measurement.temperature = clue.temperature`

We then print the combined measurement to the screen and broadcast it.

Finally, the code will sleep for a certain amount of time. Depending on your needs this can be as frequent as every two seconds if sending a single data point to Adafruit IO* (or every one second to AIO+) to many minutes or hours between sensor
measurement broadcasts. In this case, it is pausing for 60 seconds between broadcasts.

*The data limit on Adafruit IO is 30 data points per minute and 60/min on Adafruit IO+)*

```python
while True:
    measurement = adafruit_ble_broadcastnet.AdafruitSensorMeasurement()
    measurement.temperature = clue.temperature
    measurement.pressure = clue.pressure
    measurement.relative_humidity = clue.humidity
    measurement.acceleration = clue.acceleration
    measurement.magnetic = clue.magnetic
    print(measurement)
    adafruit_ble_broadcastnet.broadcast(measurement)
    time.sleep(60)
```

Once we set up the Raspberry Pi to act as a Broadcastnet base station, the CLUE sensor data will be able to make its way all the way to Adafruit IO!

---

### Prepare the Feather Sense

We'll set up the Feather Sense board to act as a sensor for the BroadcastNet.
First, set up CircuitPython on the Feather Sense following the instructions on this page. NOTE: This board is so new, the instructions are actually for the regular nRF52840 Feather, so when you go to the CircuitPython.org download page, be sure to use the download for the Sense board instead!

Libraries

Next, install the libraries needed. This guide page will show you where to download them.

You'll need the following libraries for this project:

- adafruit_apds9960
- adafruit_ble
- adafruit_ble_broadcastnet
- adafruit_bmp280
- adafruit_bus_device
- adafruit_register
- adafruit_lis3mdl
- adafruit_lsm6ds
- adafruit_sht31d
- neopixel.mpy

Text Editor

Adafruit recommends using the Mu editor for using your CircuitPython code with the Feather Sense board. You can get more info in this guide.

Alternatively, you can use any text editor that saves files.

Code.py

Copy the code shown below, paste it into Mu. Save the code from Mu to the Feather's CIRCUITPY drive as code.py

```python
# SPDX-FileCopyrightText: 2020 John Park for Adafruit Industries
#
# SPDX-License-Identifier: MIT

"""This uses the Feather Sense as a Bluetooth LE sensor node."""

import time
import adafruit_ble_broadcastnet
import board
```
import adafruit_lsm6ds  # accelerometer
import adafruit_sht31d   # humidity sensor
import adafruit_bmp280   # barometric sensor
import adafruit_lis3mdl   # magnetic sensor

i2c = board.I2C()  # uses board.SCL and board.SDA
# i2c = board.STEMMA_I2C()  # For using the built-in STEMMA QT connector on a
# microcontroller

sense_accel = adafruit_lsm6ds.LSM6DS33(i2c)
sense_humid = adafruit_sht31d.SHT31D(i2c)
sense_barometric = adafruit_bmp280.Adafruit_BMP280_I2C(i2c)
sense_magnet = adafruit_lis3mdl.LIS3MDL(i2c)

print("This is BroadcastNet Feather Sense sensor:",
      adafruit_ble_broadcastnet.device_address)

while True:
    measurement = adafruit_ble_broadcastnet.AdafruitSensorMeasurement()

    measurement.temperature = sense_barometric.temperature
    measurement.pressure = sense_barometric.pressure
    measurement.relative_humidity = sense_humid.relative_humidity
    measurement.acceleration = sense_accel.acceleration
    measurement.magnetic = sense_magnet.magnetic

    # print(measurement)
    adafruit_ble_broadcastnet.broadcast(measurement)
    time.sleep(60)

How It Works

When this code runs on the Feather Sense board, it'll first import the time, board, and adafruit_ble_broadcastnet libraries, along with the library for each sensor in the board that we're using.

Next, I2C is instantiated on the board for communications with the sensors.

We create an object (with a nice name!) for each sensor as well, and instruct them to use I2C.

Then, it will print the unique ID for the board based on the board's BLE adapter MAC address. This address will be used when messages are sent, so there won't be any clashes when multiple microcontroller boards are in range of the BroadcastNet base station.

NOTE: this is not a secure connection, so think of this system as "broadcasting" in the truest sense of the word!
import adafruit_bmp280  # barometric sensor
import adafruit_lis3mdl  # magnetic sensor
i2c = board.I2C()

sense_accel = adafruit_lsm6ds.LSM6DS33(i2c)
sense_humid = adafruit_sht31d.SHT31D(i2c)
sense_barometric = adafruit_bmp280.Adafruit_BMP280_I2C(i2c)
sense_magnet = adafruit_lis3mdl.LIS3MDL(i2c)

print("This is BroadcastNet Feather Sense sensor:",
adafruit_ble_broadcastnet.device_address)

measurements

Next, we will have the main loop of the program. In it, the board sends out an
advertisement to alert the base station central device that it is there.

Next we collect the different sensor readings such as measurement.temperature =
sense_barometric.temperature (we're using the temperature sensor that is built
onto the barometric pressure sensor package).

We then broadcast the combined measurement over BLE.

Finally, the code will sleep for a certain amount of time. Depending on your needs,
this can be as frequent as every two seconds, if sending a single data point to
Adafruit IO* (or every one second to AIO+) to many minutes or hours between sensor
measurement broadcasts. In this case, it is pausing for 60 seconds between
broadcasts.

*The data limit on Adafruit IO is 30 data points per minute and 60/min on Adafruit IO+)

measurement = adafruit_ble_broadcastnet.AdafruitSensorMeasurement()

    measurement.temperature = sense_barometric.temperature
    measurement.pressure = sense_barometric.pressure
    measurement.relative_humidity = sense_humid.relative_humidity
    measurement.acceleration = sense_accel.acceleration
    measurement.magnetic = sense_magnet.magnetic

    # print(measurement)
    adafruit_ble_broadcastnet.broadcast(measurement)
    time.sleep(60)

Once we set up the Raspberry Pi to act as a BroadcastNet base station, the CLUE
sensor data will be able to make its way all the way to Adafruit IO!
Other Boards

If you choose to use a different Adafruit nRF52840 board and external sensors, the setup will be very nearly the same. Check the guide for the board and sensors for the particulars.

Raspberry Pi Quick Setup

This quick-start guide will get you ready to connect and run all sorts of Adafruit projects on your Raspberry Pi computer

You will need:

- A modern Raspberry Pi computer (©Adafruit Industries), we recommend the 'full' size Pi 3 B+ and later, that come with Ethernet and WiFi built in
- Matching power supply (USB C type for Pi 4 and later (©Adafruit Industries), or Micro USB type for Pi 3 B+, Pi Zeros (©Adafruit Industries) or earlier)
- 8 GB or larger micro SD card (©Adafruit Industries) and USB reader/writer

SD Card Prep

First we need to perform the basic set up the Raspberry Pi. These are the first steps:

- download the latest Raspbian desktop operating system
- burn the OS to an SD card
- enable SSH
- add WiFi settings and credentials

For details on doing these things, follow the first four pages of this guide to get your Pi setup with Raspbian Lite (for 'headless' or remote operation) or NooBs (for graphical / desktop use), WiFi, and ssh enabled.

Once the SD card is prepared, come back here so we can continue.

Power to the Pi

Safely eject the SD card from your main computer. Then, with the Pi unplugged from power, insert the SD card into the Pi's SD card slot.

Then, plug in the power to the Pi. You should see some LEDs light up.

From here forward, we'll be connecting to the Pi remotely from our main computer. Make sure your computer is on the same WiFi network as the Pi.

Ping

After a minute or two, you can try and ping the Pi to see if it has connected to your network. Open a terminal on your computer, such as Terminal.app on a mac (Windows users will require some additional setup. Read here), and type this and press return:

```
ping -c 3 raspberrypi.local
```

You should see an output similar to this:

```
jep@machine ~ % ping -c 3 raspberrypi.local
PING raspberrypi.local (192.168.1.68): 56 data bytes
64 bytes from 192.168.1.68: icmp_seq=0 ttl=64 time=14.993 ms
64 bytes from 192.168.1.68: icmp_seq=1 ttl=64 time=20.565 ms
64 bytes from 192.168.1.68: icmp_seq=2 ttl=64 time=50.571 ms
--- raspberrypi.local ping statistics ---
3 packets transmitted, 3 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 14.993/28.710/50.571/15.625 ms
```

This tells us that the Pi is on the network, and has an address of 192.168.1.68.
SSH

We can now connect to it through the secure shell (SSH) so we can control it remotely.

Here's some great info on SSH () from the Raspberry Pi Foundation.

Type the following in your terminal:

```
ssh pi@raspberrypi.local
```

or

```
ssh pi@192.168.1.68
```

(or whatever address you saw returned earlier when you pinged it.)

You may see a message warning you that the authenticity of the host can't be established, Are you sure you want to continue connecting (yes/no)?

To this question you can reply by typing `yes` and pressing return. This will add the Pi to your list of known hosts and should not pop up again with this question.

You will now see the password prompt for the Pi:

```
pi@raspberrypi.local's password:
```

Go ahead and type in the default password, which is: `raspberry`

**Change the Password!**

You will now be logged into the Pi over ssh! There will be a warning about changing your password from the default and instructions on how to do so. Do it now! Type `passwd`, hit return and follow the instructions to create a new, secure, non-default password!

**Rename the Pi**

You may end up with multiple Pi computers on your network, which can get confusing if they are all named raspberrypi. To rename this one for this project, type in `sudo raspi-config` and press return.
Use the arrows and enter key on your keyboard to navigate. Pick 2 Network Options and press return on your keyboard.

Then, pick N1 Hostname.

Enter a new hostname -- I typed `raspberrypiBlinkaBridge`

Use the Tab key to get to the <Ok> button and press return.

Press tab twice to get to the <Finish> button and press return.

Now, we'll restart the Pi to make the changed host name show up on the network by typing `sudo reboot` and pressing return.

Note, this will close the SSH session. After the Pi restarts, wait about a minute, and then reconnect over SSH. This time, you'll use the new name: `ssh pi@raspberrypiBlinkaBridge`
Update Application List

Once logged into the Pi, we'll do some software updates to make sure we have the latest of everything.

We really really recommend the lastest Raspbian only. If you have an older Raspbian install, run "sudo apt-get update" and "sudo apt-get upgrade" to get the latest OS!

Type:

```
sudo apt-get update
```

And then press return. This will run for a little bit as it reads the available package lists, and then finish, reporting that it is done.

You can now apply any of these changes by typing:

```
sudo apt-get upgrade
```
Type \texttt{Y} and press enter to continue.

Software Setup

Now that the Pi is updated and renamed on the network, we'll make sure we have Python 3 setup, as Python 2 is no longer used or supported.

\texttt{pip3}, is the software package installer we'll use. Let's upgrade it to the latest version with this command:

```
    sudo pip3 install --upgrade setuptools
```

If above doesn't work try

```
    sudo apt-get install python3-pip
```

Once that has finished you'll be returned to the prompt.

Make sure you're using Python 3!

The default Python on your computer may not be Python 3. Python 2 is officially discontinued and all our libraries are Python 3 only.

We'll be using \texttt{python3} and \texttt{pip3} in our commands, use those versions of Python and pip to make sure you're using 3 and not 2.
Install Blinka

Blinka is the software that allows us to run CircuitPython and its libraries on Linux. You can [read more about it here](#).

To install blinka, type:

`sudo pip3 install adafruit-blinka`

That's it! You can reboot again if you like.

---

Install Pi Bridge Software

### BLE Support

Next we'll set up BLE support with the BlueZ Bluetooth protocol stack. Type this and hit return:

```
sudo apt-get install bluez-hcidump
```

```
pi@raspberrypiBlinkaBridge:~ $ sudo apt-get install bluez-hcidump
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  bluez-hcidump
0 upgraded, 1 newly installed, 0 to remove and 4 not upgraded.
Need to get 146 kB of archives.
After this operation, 414 kB of additional disk space will be used.
  5.50-1+rpt1 [146 kB]
Fetched 146 kB in 1s (154 kB/s)
Selecting previously unselected package bluez-hcidump.
(Reading database ... 93517 files and directories currently installed.)
Preparing to unpack .../bluez-hcidump_5.50-1+rpt1_armhf.deb ...
Unpacking bluez-hcidump (5.50-1+rpt1) ...
Setting up bluez-hcidump (5.50-1+rpt1) ...
Processing triggers for man-db (2.8.5-2) ...
```

---

**Adafruit Blinka BLEio**

"Old MacDonald had a snake, BLEeeee-iiiiii-BLEeeeeiiiiii-oooooo!"

Next, we'll install the Adafruit Blinka bleio library, which is required for CircuitPython BLE to work properly.

In the ssh terminal type:
pip3 install adafruit-blinka-bleio

Bluetooth Group

In order to have the Pi scan for BLE devices we need to increase the permissions granted to the pi user.

To add the user to the bluetooth group do:

```
sudo usermod -a -G bluetooth pi
```

(Or, whatever your username is if not the default.)

Now, the pi user won't actually have these new permissions until the next time we log in, so type `exit` and then re-login to the ssh with

```
ssh pi@raspberrypiblinkabridge.local
```

Enter your password when prompted, then you can check the groups the pi user is in by typing:

```
groups
```

```
pi@raspberrypiBlinkaBridge:~ $ groups
pi adm dialout cdrom sudo audio video plugdev games users input netdev bluetooth gpio i2c spi
```

Now, we'll set permissions so that we can do comprehensive BLE scanning.

Type the following lines into the ssh terminal, pressing enter after each one:

```
sudo chown :bluetooth /usr/bin/hcidump /usr/bin/hcitool
sudo chmod o-x /usr/bin/hcidump /usr/bin/hcitool
sudo setcap 'cap_net_raw,cap_net_admin+eip' /usr/bin/hcitool
sudo setcap 'cap_net_raw,cap_net_admin+eip' /usr/bin/hcidump
```

Install Python libraries

Now you're ready to install the Python support library for the BLE BroadcastNet bridge. Run the following command to install it:

```
pip3 install adafruit-circuitpython-ble-broadcastnet
```
Add Examples

Let's add some example code to try by cloning a Git repository locally.

Type `cd` to go to your home directory on the Pi.

Note, you can type `pwd` (print working directory) to see where you are at any time:

```
pi@raspberrypiBlinkaBridge:~ $ pwd
/home/pi
```

Here we will clone the example directory. Type:

```
git clone https://github.com/adafruit/Adafruit_CircuitPython_BLE_BroadcastNet.git
```

Once it finishes copying the files, you can have a look. Go to the directory by typing `cd` Adafruit_CircuitPython_BLE_BroadcastNet (Hint, you can press Tab to complete that long directory name after you type a couple of the letters).

Check out the contents by typing `ls -l`

```
pi@raspberrypiBlinkaBridge:~/Adafruit_CircuitPython_BLE_BroadcastNet $ ls -l
adafruit_ble_broadcastnet.py  CODE_OF_CONDUCT.md
 docs                    LICENSE
 examples                README.rst
 requirements.txt        setup.py
```

The adafruit_ble_broadcastnet.py file is the Python library we'll be using when we run the code. The code we want to run on the Pi is inside the /examples directory. Let's go there by typing:

```
cd ./examples/
```

ble_broadcastnet_blinka_bridge.py is the one we'll run on the Pi (after setting up the secrets.py file below), while the others are examples we can run on different microcontrollers such as the CLUE, Circuit Playground Bluefruit, Feather nRF52840, Feather Sense, and ItsyBitsy nRF52840.

- ble_broadcastnet_battery_level_neopixel.py
- ble_broadcastnet_battery_level.py
In order to use the Adafruit IO Internet of Things service (which is absolutely free to use) you'll need to log in with your Adafruit account. If you don't already have an Adafruit login, create one here.

If you haven't used Adafruit IO before, check out this guide for more info.

Once you have logged into your account, there are two pieces of information you'll need to place in your secrets.py file: Adafruit IO username, and Adafruit IO key. Head to io.adafruit.com and simply click the View AIO Key link on the left hand side of the Adafruit IO page to get this information.

Shhh Secrets

The ble_broadcastnet_blinka_bridge.py program will need to authenticate your Adafruit IO credentials in order to run. So, we'll create a file called secrets.py to contain this info.

Inside is a Python dictionary named secrets with a line for each entry. Each entry has an entry name (say 'aio_username') and then a colon to separate it from the entry key 'daphne_von_person'

```python
secrets = {
'aio_username' : '_your_aio_username_',
'aio_key' : '_your_big_huge_super_long_aio_key_
}
```

Don't share your secrets.py - keep that out of GitHub, Discord or other project-sharing sites.

Use a text editor to create and save this file into the /Adafruit_CircuitPython_BLE_Broadcast/examples directory. Since you are ssh'd into the Pi, the easiest way to do this is with the nano text editor. Type:

```
nano secrets.py
```
In the nano editor that launches, copy and paste the secrets text from above, replacing the username and key with your own, then save & exit.

Your /examples directory should now look like this:

- ble_broadcastnet_battery_level_neopixel.py
- ble_broadcastnet_battery_level.py
- ble_broadcastnet_blinka_bridge.py
- ble_broadcastnet_cpb.py
- ble.broadcastnet_expo_backoff.py
- ble_broadcastnet_multisensor.py
- ble_broadcastnet_simplestest.py
- secrets.py

Service Identity

Next, we'll install the service_identity module, which is used behind the scenes for certificate verification. Type this into the ssh terminal and press return:

```bash
pip3 install service_identity
```

Launch BroadcastNet Bridge

We're now ready to run the broadcast bridge software on the Pi! When this runs it will look for BLE sensors, collect their message data, and then send it to your Adafruit IO account as a uniquely named feed.

To run it, type this in the ssh shell and press return:

```bash
python3 ble_broadcastnet_blinka_bridge.py
```

If your Feather Sense and/or CLUE boards you set up earlier are powered on and within range, you will start to see the measurement values being received successfully:

```
bridge-dca63202680a-sensor-fdc492775544 ["key": 'missed-message-count', 'value': 0], "key": 'acceleration-0-x', 'value': -5.6099700927734375}, {
  'key': 'acceleration-0-y', 'value': 2.15712833404541}, {
  'key': 'acceleration-0-z', 'value': 7.956132888793945}, {
  'key': 'pressure-0', 'value': 992.184814453125}]
Done logging measurement to IO. Took 0.628571397000087 seconds
```

```
bridge-dca63202680a-sensor-fdc492775544 ["key": 'missed-message-count', 'value': 2], "key": 'acceleration-0-x', 'value': -5.6506500244140625}, {
  'key': 'acceleration-0-y', 'value': 2.1284141540527344}, {
  'key': 'acceleration-0-z',
```
If you open your Adafruit IO Feeds page, you'll see this feed there.

Let's have a look at how to work with your feed data.

BroadcastNet on AIO

Adafruit IO Getting Started

If you're new to Adafruit IO, this guide page () has lots of great info and links to check out.
Renaming AIO Feeds

When you head to your Adafruit IO Feeds page, you'll see the Raspberry Pi bridge name and sensor name pairings of any sensor boards you're running. Here, I have two:
Since these names are derived from the MAC addresses of the devices themselves, they are good unique IDs, but not very friendly looking! Let's change them.

First, look at the CLUE screen and make note of the sensor ID. In my case it reports **caa19e07860b**.

Then, click on the feed group of the same name. You'll see a settings gear icon in the Group Info box. Click that.

In the Edit Group popup, give the Name field a more descriptive name. Be sure to leave the Key value as it is or things will break!

Repeat this process for any other boards you're using.
Dashboard Setup

Set up a new dashboard -- I named mine BroadcastNet Dashboard.

Hit Create to make the new dashboard, which we can then fill with feed data blocks.
Blocks

Now we'll add feed data blocks to the Dashboard.

Press the + icon to add new blocks.

Here I've added a graph widget with the two temperatures from the two different boards.
You can continue to add elements to the dashboard depending on what you'd like to see!

Measurement List

The BroadcastNet library has many different measurements it can read with the `measurement` command. You can check out the documentation here.

For convenience, this is the full list:

- **TVOC**
  
  Total Volatile Organic Compounds as a float in parts per billion.

- **acceleration**
  
  Acceleration as (x, y, z) tuple of floats in meters per second per second.

- **battery_voltage**
  
  Battery voltage in millivolts. Saves two bytes over voltage and is more readable in bare packets.

- **color**
  
  Color as RGB integer.
Current as a float in milliamps.

**distance**

Distance as a float in centimeters.

**duty_cycle**

16-bit PWM duty cycle. Independent of frequency.

**eCO2**

Equivalent CO2 as a float in parts per million.

**frequency**

As integer Hertz

**gyro**

Gyro motion as (x, y, z) tuple of floats in radians per second.

**light**

Brightness as a float without units.

**lux**

Brightness as a float in SI lux.

**magnetic**

Magnetism as (x, y, z) tuple of floats in micro-Tesla.

**orientation**

Absolution orientation as (x, y, z) tuple of floats in degrees.

**pressure**

Pressure as a float in hectopascals.
relative_humidity

Relative humidity as a float percentage.

temperature

Temperature as a float in degrees centigrade.

value

16-bit unit-less value. Used for analog values and for booleans.

voltage

Voltage as a float in Volts.

weight

Weight as a float in grams.