Integrating Home Assistant with Adafruit IO

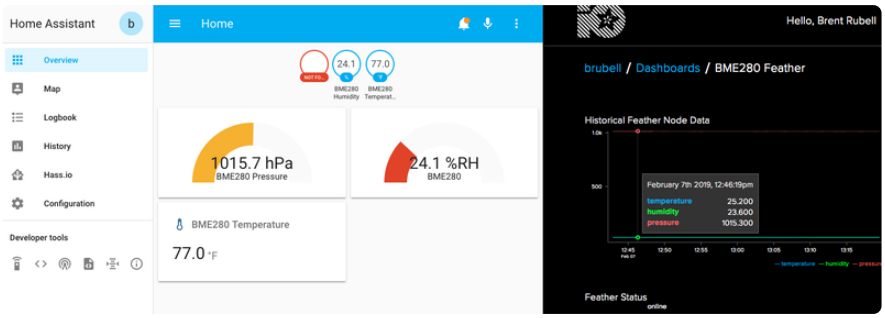
Created by Brent Rubell

https://learn.adafruit.com/integrating-adafruit-io-with-home-assistant

Last updated on 2024-03-08 03:18:48 PM EST

Table of Contents

Overview

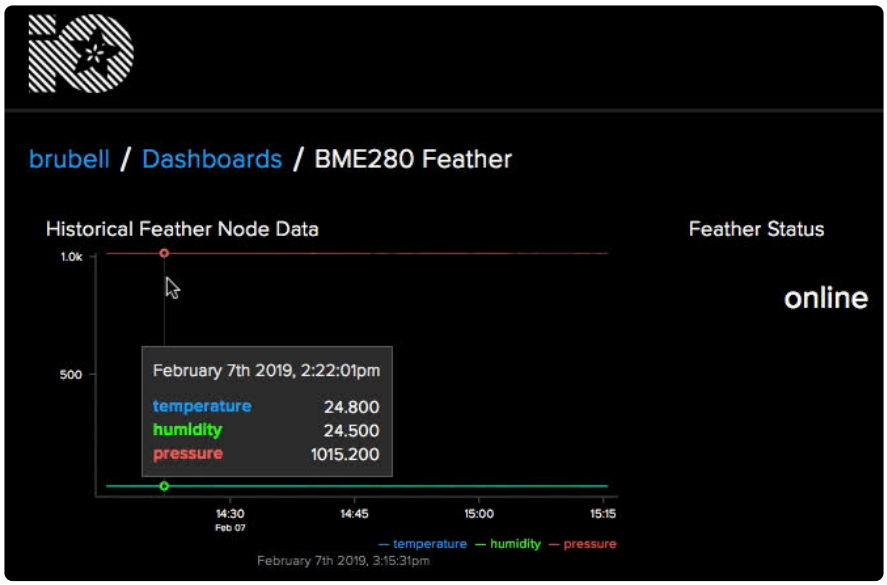


Home Assistant (https://adafru.it/DQw) is an open source home automation software which tracks the state of the smart-devices in your home so you don't have to. It easily integrates with most smart-devices (Google Cast, Philips Hue, Nest, Sonos, etc) and even interfaces with smart-assistants like Alexa or Google Assistant.

Adafruit sells a lot of different types of sensors. Following this guide will let you quickly connect them to Home Assistant using an Adafruit Feather.

We're going to set up an Adafruit Feather ESP8266 with a BME280 sensor to create a temperature, pressure, and humidity logging node. Then we'll install, set up, and configure Home Assistant on a Raspberry Pi.

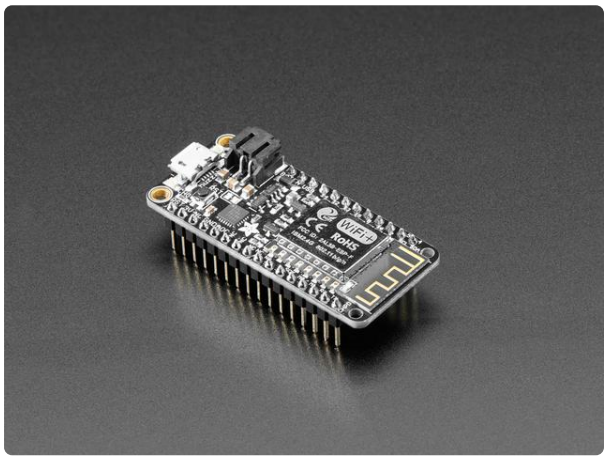
We'll configure the Feather to broadcast its sensor data to both Home Assistant and Adafruit IO (https://adafru.it/eIC) using ESPHomeYAML (https://adafru.it/DQx).



While Home Assistant (https://adafru.it/DQw) is fantastic at displaying data in real-time, you might want to manipulate and log the data. We'll be using Adafruit

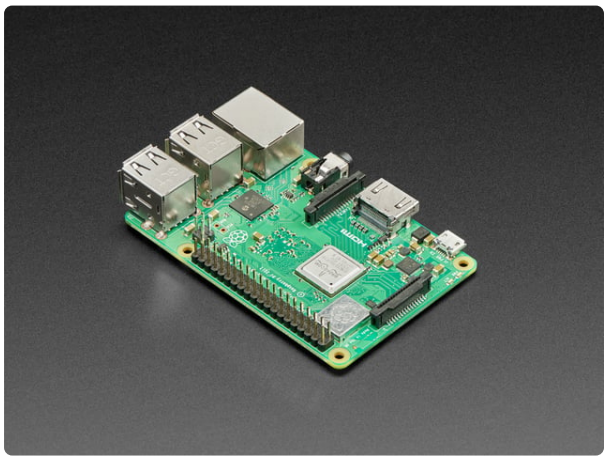
IO (https://adafru.it/eIC) to perform real-time and long-term logging for visualization of data, and export the data from sensor feeds.

Parts

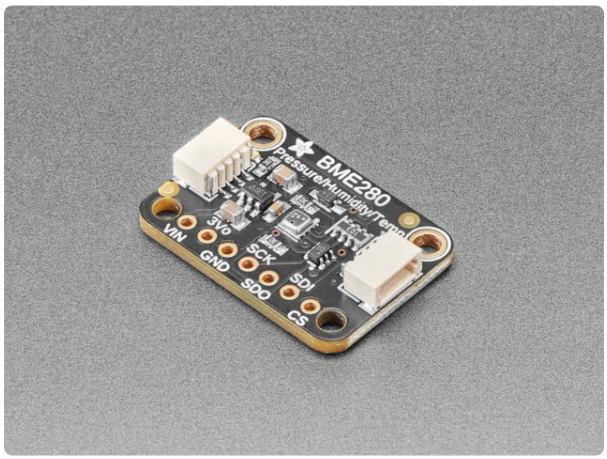


Assembled Adafruit Feather HUZZAH with ESP8266 With Headers
Feather is the flagship development board from Adafruit, and like its namesake, it is thin, light, and lets you fly! We designed Feather to be a new standard for portable...
https://www.adafruit.com/product/3046

Note: While you can run HassOS on a Pi Zero W, we do not suggest using one with this guide. The process of flashing firmware from a Pi Zero W is difficult. Compiling and uploading the Feather's firmware from a Pi Zero W takes significantly more time than a Pi 3B+.



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



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

Materials

You'll need the items below for this project. If you do not have them already, grab them from the Adafruit Store:

1 x Switching Power Supply with MicroUSB — https://www.adafruit.com/product/1995
5V 2.5A Switching Power Supply with 20AWG MicroUSB Cable

---

1 x Micro USB Cable — https://www.adafruit.com/product/592
USB cable - USB A to Micro-B - 3 foot long

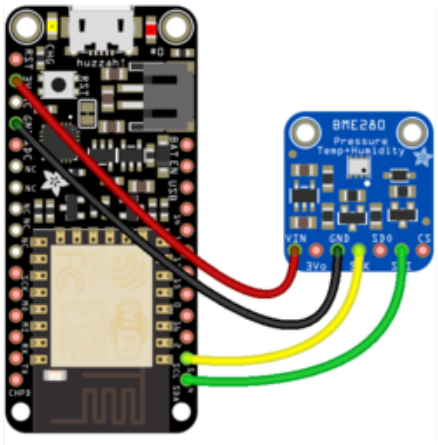
---

1 x Breadboard — https://www.adafruit.com/product/64
Half-size breadboard

---

1 x Wire Bundle — https://www.adafruit.com/product/153
Breadboarding wire bundle

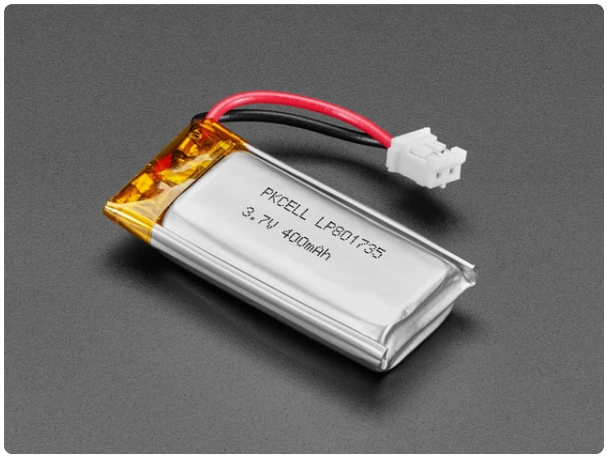
Feather Wiring



Make the following connections between the Feather Huzzah ESP8266 and the BME280:

Feather 3V to **BME280 VIN**
Feather GND to **BME280 GND**
Feather SCL to **BME280 SCK**
Feather SDA to **BME280 SDI**

You may also want to connect a LiPo battery to the JST connector on the Feather Huzzah for wire-free logging. This one is ideal for Feathers:



Lithium Ion Polymer Battery Ideal For Feathers - 3.7V 400mAh

Lithium-ion polymer (also known as 'lipo' or 'lipoly') batteries are thin, light, and powerful. The output ranges from 4.2V when completely charged to 3.7V. This...

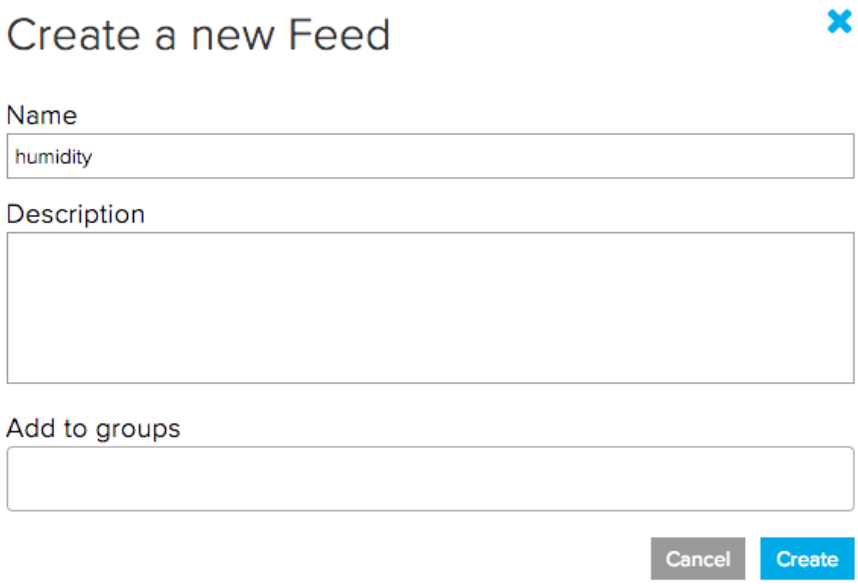
https://www.adafruit.com/product/3898

Adafruit IO Setup

If you do not already have an Adafruit IO account set up, head over to io.adafruit.com (https://adafru.it/fH9) to link your Adafruit.com account to Adafruit IO.

Navigate to the feeds page (https://adafru.it/mxC) on Adafruit IO. Then click **Actions -> Create New Feed**, and name this feed **humidity**.

- If you do not already know how to create a feed, head over to Adafruit IO Basics: Feeds (https://adafru.it/ioA).

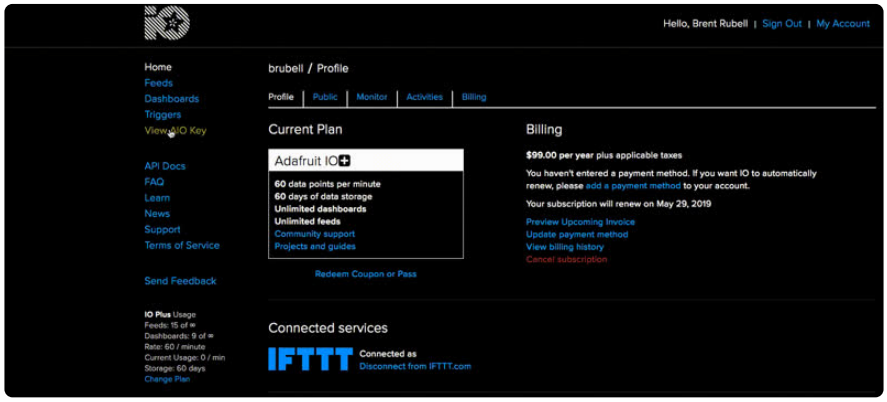


Then, **create a temperature and pressure feed** for the BME280's temperature and pressure sensors.

Also, create an additional feed named **status**. When your Feather turns on or off, it'll broadcast its connection status to this feed.

We're also going to need our Adafruit IO username and secret API key.

Navigate to your profile and click the **View AIO Key** button to retrieve them. Write them down in a safe place, we'll need them for later.



With Adafruit IO set up, we'll move on to installing HassOS

Installing HassOS on Raspberry Pi

We'll be installing Home Assistant locally on a Raspberry Pi using Hass.io (https://adafru.it/DQy), an All-in-One image which turns your Pi into a home assistant hub.

The process of installing and configuring HassOS takes a while. Luckily, you only have to do it once.

First, download the latest Hass.io image for your device from the home assistant downloads page.

Download the latest Hass.IO release

https://adafru.it/DQz

Then, **unzip the .img.gz** file. You should now have a **.img** file.

Burn the OS image to a fresh SD card (Home Assistant's website suggests a **32GB SD card or larger**).

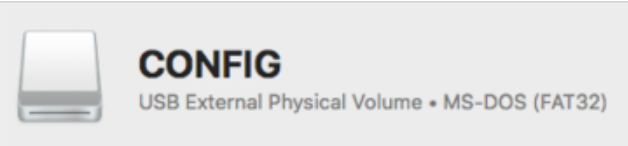
- If you do not know how to install an OS onto a SD card for the Raspberry Pi, follow this page and come back when you're done (https://adafru.it/DQA).

WiFi Configuration

If you want to connect your Pi over WiFi, you'll need to configure Hass first. HassOS uses NetworkManager (https://adafru.it/DQB) to control the host network and defaults to ethernet.

HassOS uses a **USB Drive** to load configuration files like WiFi onto the Pi. The capacity of the stick does not matter - we'll be adding one file to the drive.

We'll be overwriting the contents of the USB Drive, so back up the contents of the drive before you proceed further.



Plug the USB drive into your computer and reformat it with the following settings:

Rename the drive to **CONFIG**
Format the drive as **MS-DOS (FAT32)**

Once reformatted, **make a folder on the drive named network**.

Then, **create a new file called my-network inside the network folder we just created**.

Copy and paste the following into the file:

```
[connection]
id=hassos-network
uuid=72111c67-4a5d-4d5c-925e-f8ee26efb3c3
type=802-11-wireless

[802-11-wireless]
mode=infrastructure
ssid=MY_SSID
# Uncomment below if your SSID is not broadcasted
#hidden=true

[802-11-wireless-security]
auth-alg=open
key-mgmt=wpa-psk
psk=MY_WLAN_SECRET_KEY

[ipv4]
method=auto

[ipv6]
```

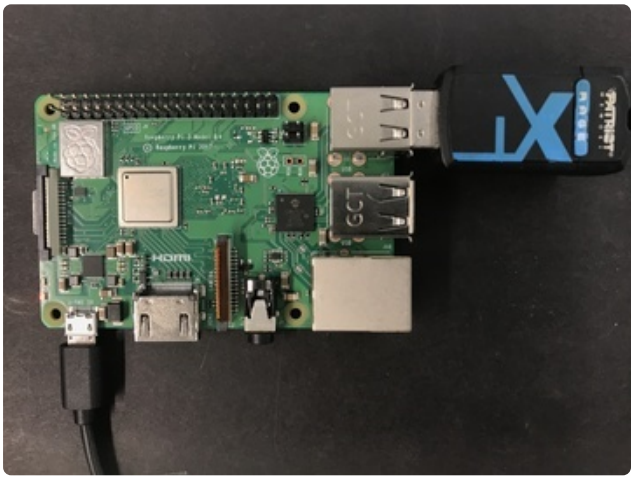
```
addr-gen-mode=stable-privacy
method=auto
```

In the configuration, **change `ssid` to your network's ssid** and **change `psk` to the password of your network.**

If you don't want the IP address of your Pi to change on every bootup - visit this page to generate a UUID (https://adafru.it/Izf) and **replace `uuid` in the configuration with the value from the website**.

Save the file to the USB drive and eject the drive from your computer. Safely eject the SD card from your computer at this time, too.

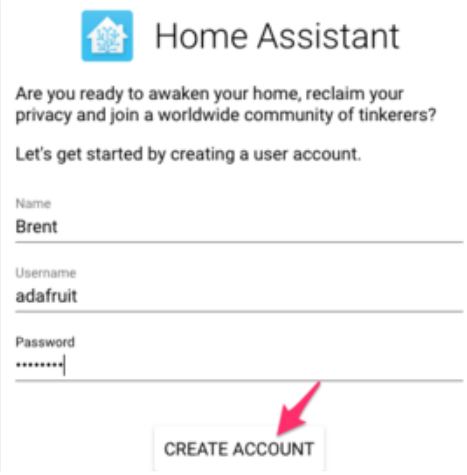
HassOS Setup



Make sure the Pi is unplugged from power.

Insert the SD card with HassOS into the Pi. Then, **plug the USB drive into one of the Pi 3B+'s USB ports.**

Plug the Pi in to the power supply to power it on.

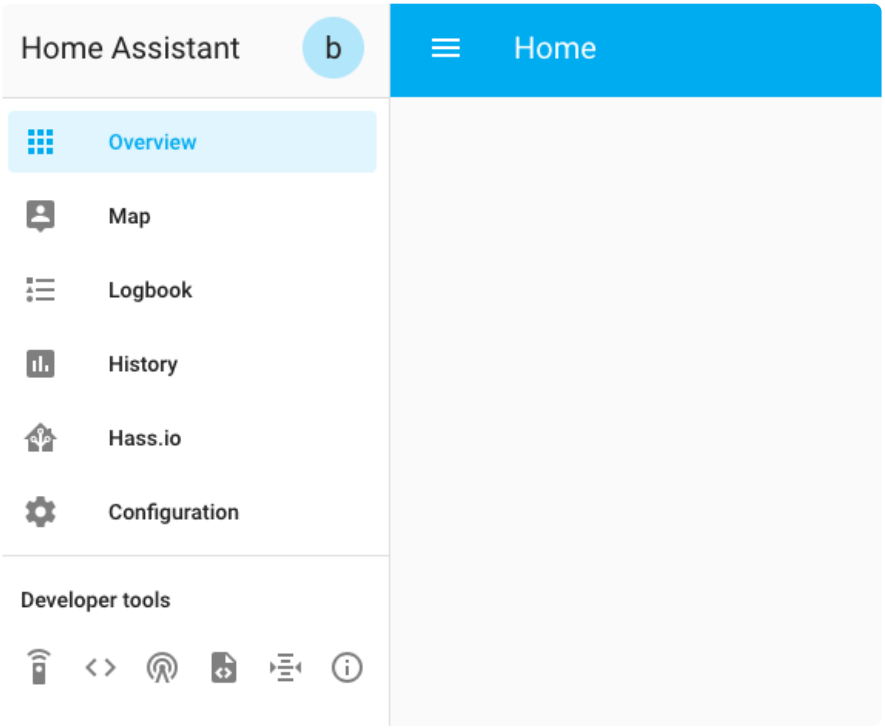


Home Assistant will take from 15-20 minutes to load up on first boot. Grab a coffee and browse adafruit.com (https://adafru.it/dAR) for awhile.

Once some time has passed, navigate to hassio.local:8123 (https://adafru.it/DQD).

You'll eventually be greeted with a welcome screen. Set up a new user account (this is a local account on the Raspberry Pi) and **click Create Account**.

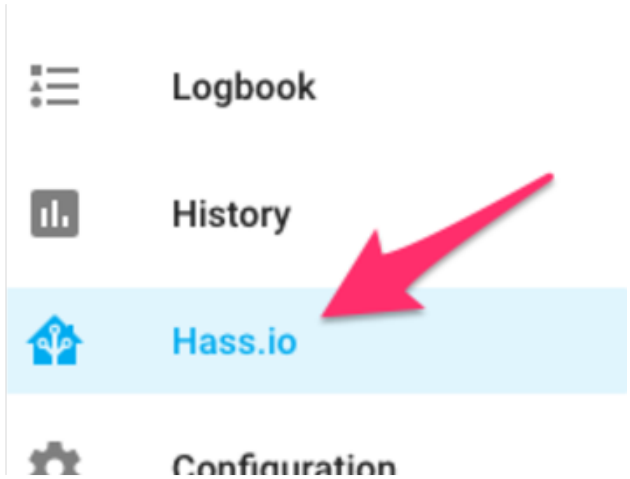
After setup, you'll be directed to the Home Assistant login page. Remove the USB drive from the Raspberry Pi.



Next, we'll configure Home Assistant to be used with the Feather.

Configuring HassOS

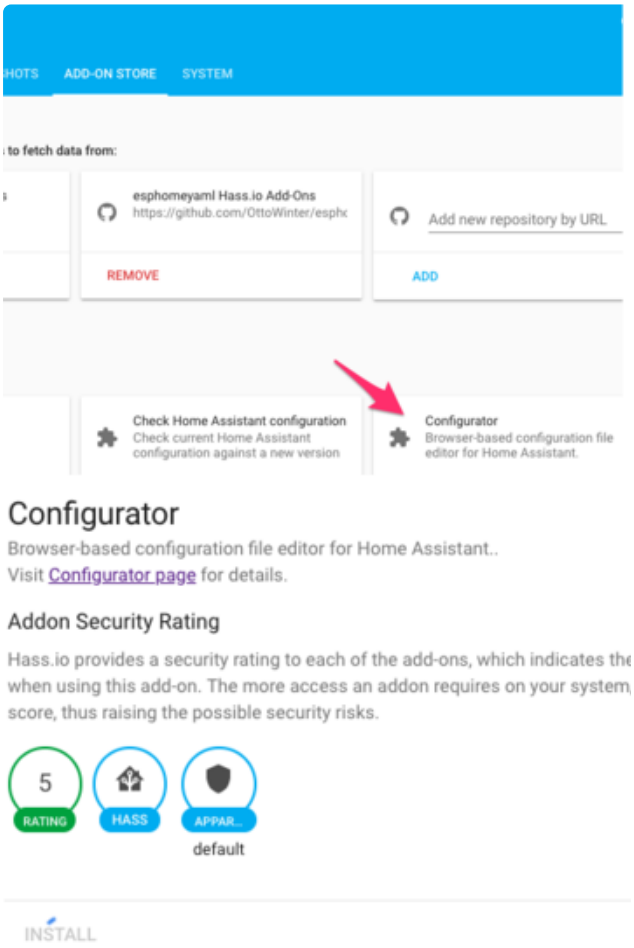
HassOS requires a bit of configuration before it's ready to be used with the Feather.



From the Home Assistant sidebar, **click on Hass.io**.

Home Assistant uses a YAML file named `configuration.yaml` (https://adafru.it/DQE) for each component used by Home Assistant. Previously, editing this file involved setting up SSH or SAMBA sharing on the Raspberry Pi.

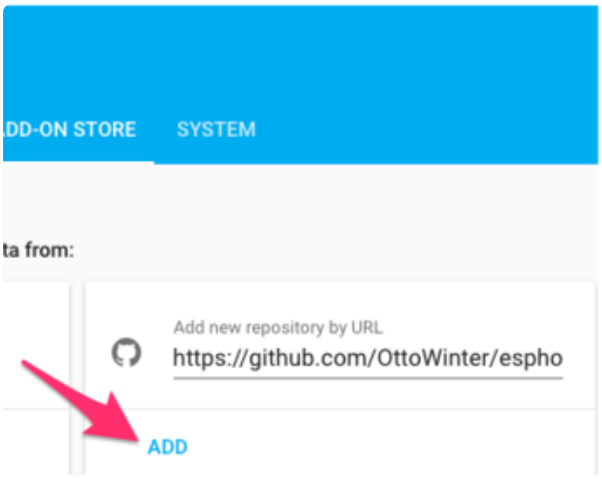
We're going to use an add-on which lets us edit this file right from our browser.



From the Hass.io screen, **click Add-on Store**. Click **Configurator**.

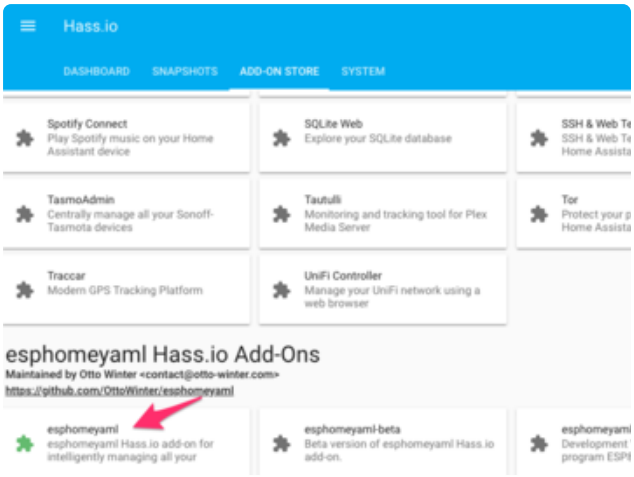
On the configurator page, click **Install.**

The installation takes a while depending on your internet speed. Once the loading circle over installs disappears, the add-on is installed.

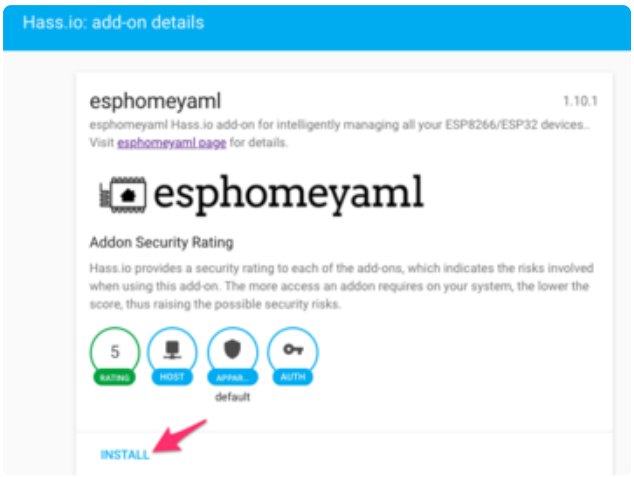


Next up, we need to install ESPHomeYAML. Since we're using a community project, ESPHomeYAML, we'll want to add the ESPHome repository to Hass.io.

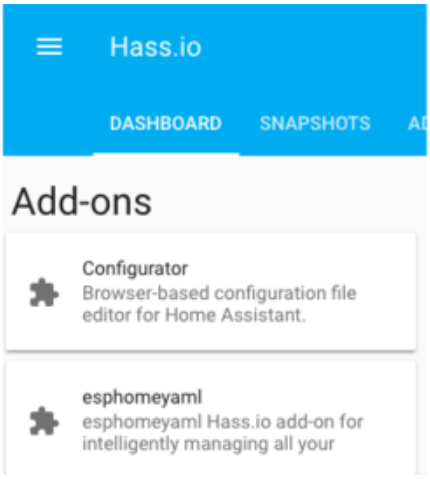
Under **Repositories -> Add new repository by URL, add the ESPHome repository**: https://github.com/OttoWinter/esphomeyaml (https://adafru.it/DQF)



After the repository has been added, **scroll down** the page and click on **esphomeyaml**.



From the esphomeyaml add-on page, **click Install**



Once installed, esphomeyaml should appear on the Hass.io dashboard page along with the configurator we installed earlier.

Configuring Add-ons

While the Configurator and esphomeyaml add-ons are installed, they need some configuration. We'll start with the Configurator add on.

From the Dashboard, **click the Configurator add-on**. You'll be presented with its description, some settings (such as when it updates, when it starts) and the configuration. We'll want to change the password from null to a value enclosed by quotes.

For example, **change password from**:

```
"password": null,
```

to

```
"password": "feather",
```

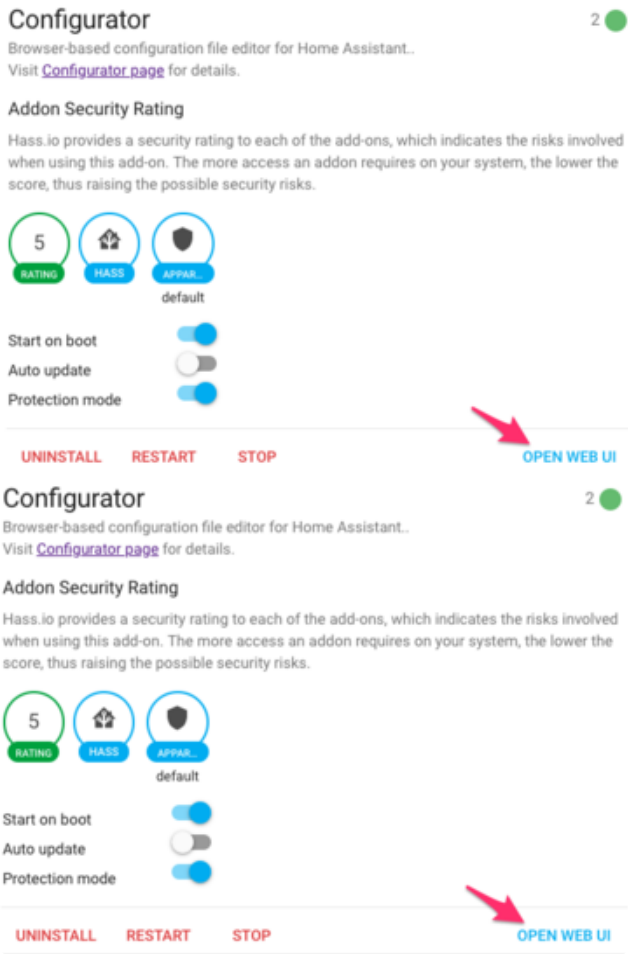
Then, **click save**.

Next, we'll need to add our IP address to the `allowed_networks` section of the configuration file.

Under the two preconfigured IP addresses, add your computer's IP under `allowed_networks` enclosed in quotations.

Don't know your IP address? You can find it on whatsmyip.org (https://adafru.it/DQG). Then, **Click Save**

After you've configured both the IP and the password, **click Start.** A button to **Open Web UI** will appear when the Configurator successfully starts.



Click **Open Web UI**.

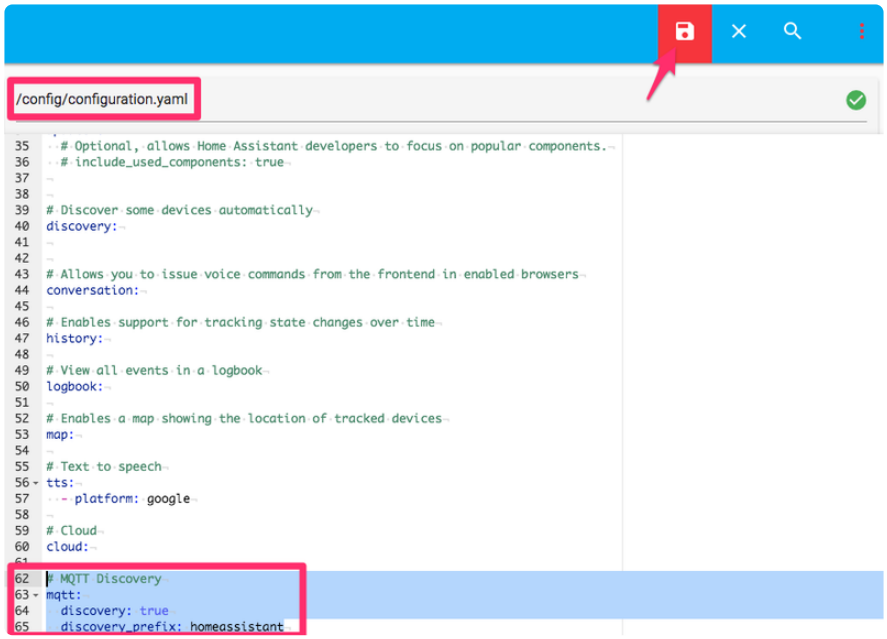
From the new tab, **enter the username and password we specified in the Config.**

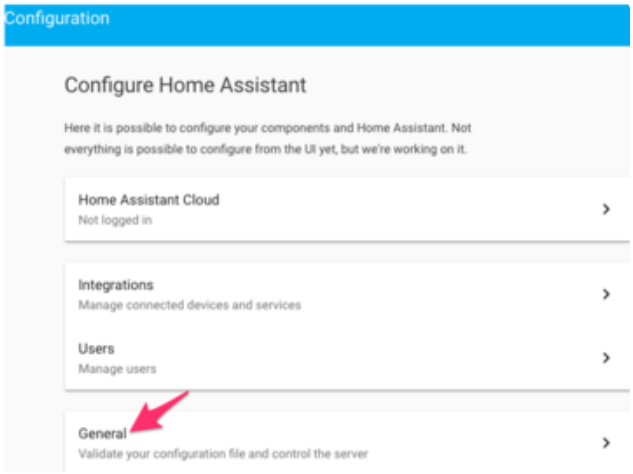
In the Configurator editor, make sure the file you are editing is `/config/configuration.yaml`.

Then, add an entry in the configuration.yaml file to enable MQTT Discovery (https://adafru.it/DQH):

```
# MQTT Discovery
mqtt:
  discovery: true
  discovery_prefix: homeassistant
```

and **Click Save.**





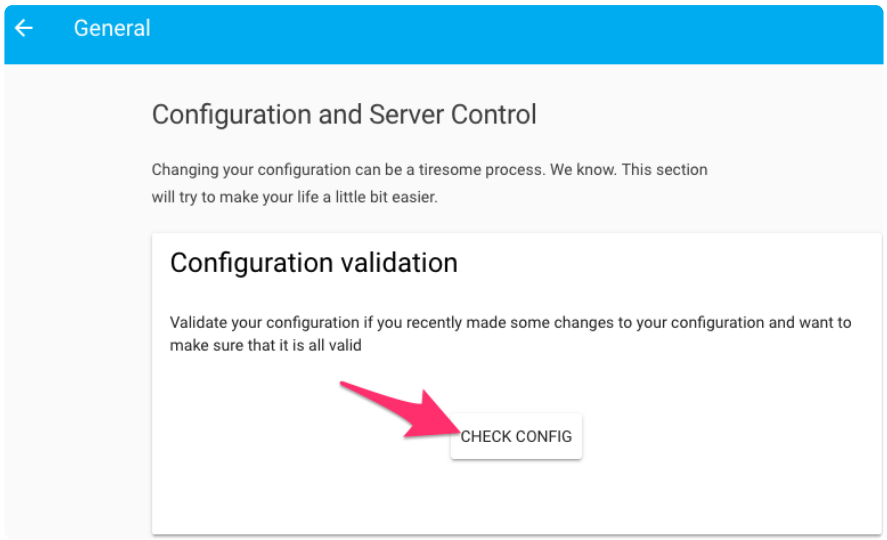
After saving, **navigate back to the Home assistant dashboard**.

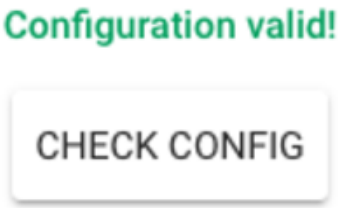
From the sidebar, **click Configuration**.

From Configuration, **click General**.

Since we changed the `configuration.yaml`, we'll want to validate the file to make sure nothing goes wrong when we restart Home Assistant.

Click Check Config

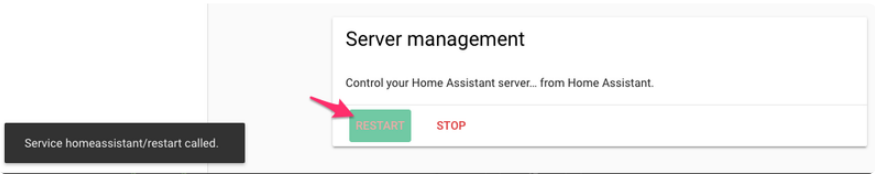




If everything checks out, a message will pop up displaying **Configuration valid!**

After checking the configuration, we'll want to restart the server. We don't need to SSH into the Raspberry Pi and issue a restart command - Home Assistant lets us do this from within the Configuration page.

From the Configuration page, scroll down to **Server Management** and **click Restart.**



If you refresh the page, you'll encounter a Connection lost. Reconnecting... message. Since we restarted the server, it'll take a few minutes to get itself back online.

After a few minutes, **refresh the page.**

Once we're connected back to HassOS, we'll proceed to configuring the Feather with ESPHomeYAML.

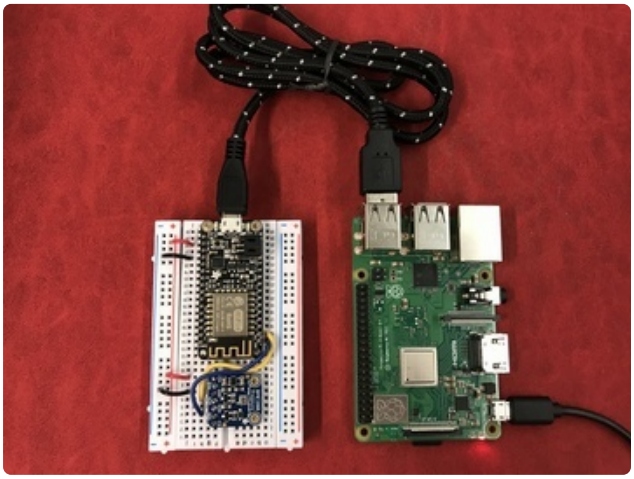
ESPHome Setup

Flashing firmware with Raspberry Pi

ESPHomeYAML (https://adafru.it/DQx) is a tool which creates custom firmware for ESP8266/ESP32 boards and sensors from a Yet-Another-Markup-Language (YAML) file (https://adafru.it/DQX). We're going to be using the official Home Assistant add-on for ESPHomeYAML to help generate firmware for the Feather. Then, we'll upload the firmware directly to the board.

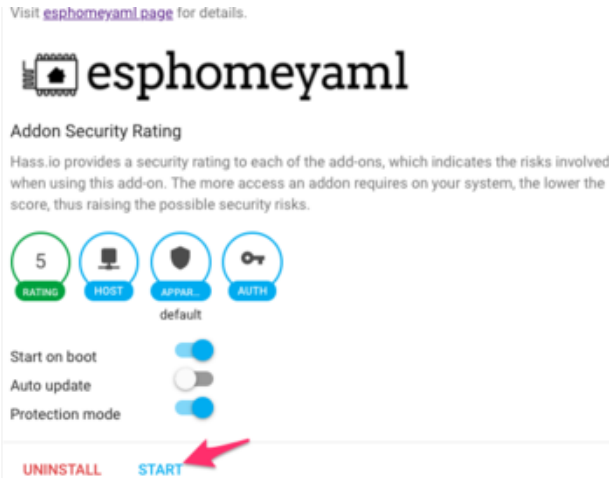
Note: If you are using any single-board computer slower than a Pi 3, such as a Pi Zero, expect to wait a significant amount of time while the firmware compiles and uploads to the Feather Huzzah.

Plug the Feather into a USB Cable, and then into one of the Raspberry Pi 3B+'s USB ports.



Since we'll be compiling and uploading the firmware directly from the Pi to the Feather, we'll connect the two before we begin.

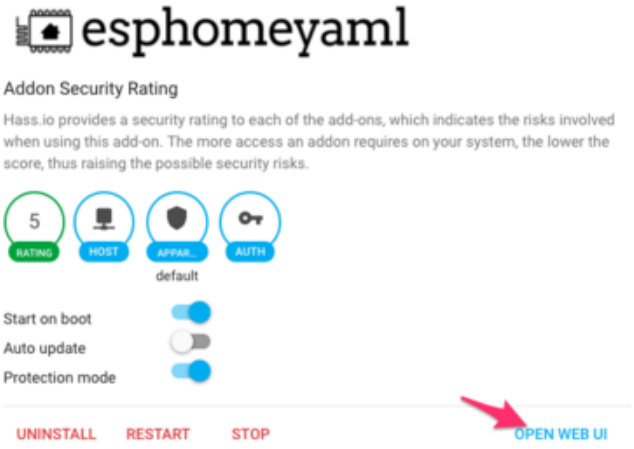
Plug the Feather into a Micro-USB Cable, and then into one of the Raspberry Pi 3B+'s USB ports.



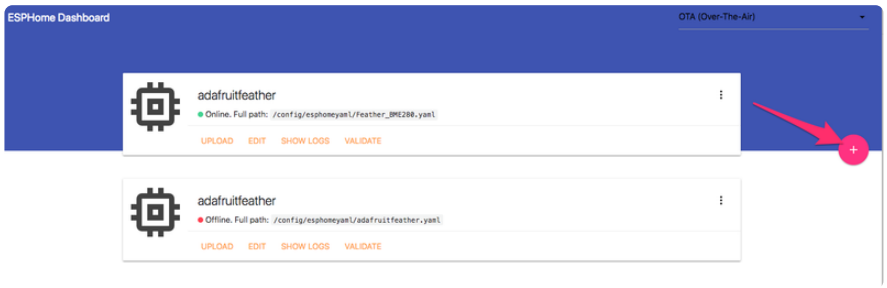
From the esphomeyaml add on page, **click Start**.

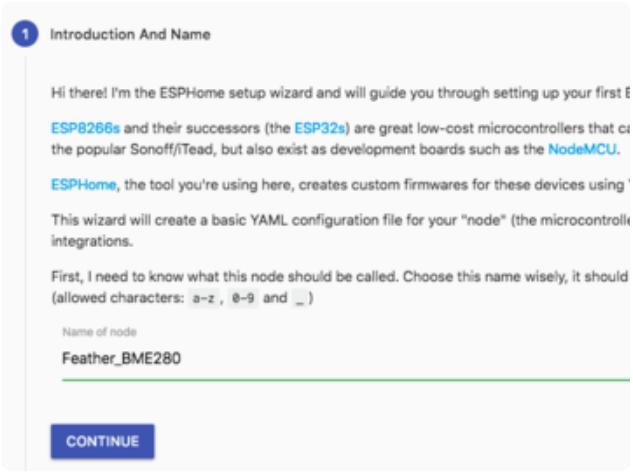
Once ESPHomeYAML is finished installing, a new button will appear Open Web UI.

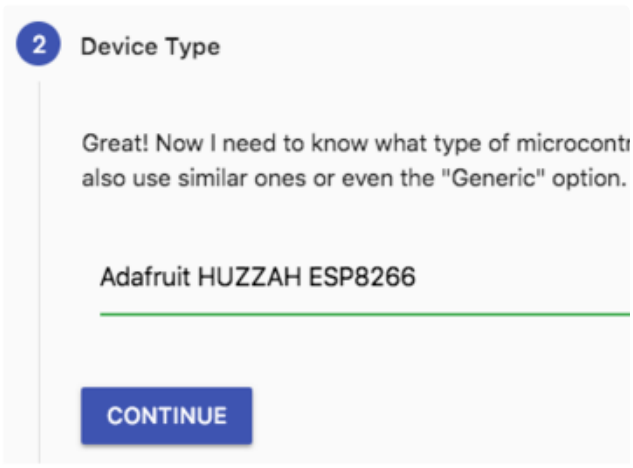
Click Open Web UI

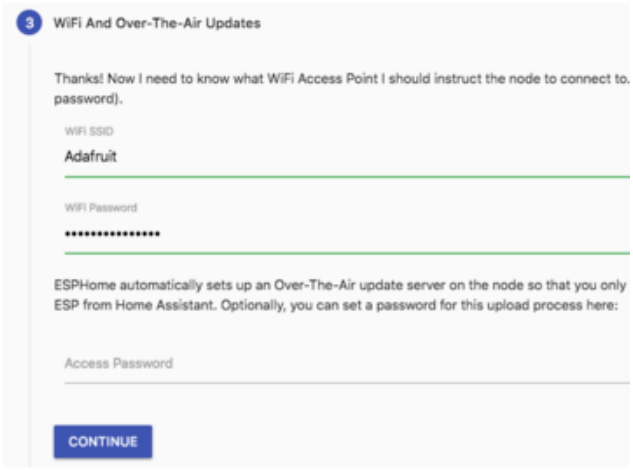


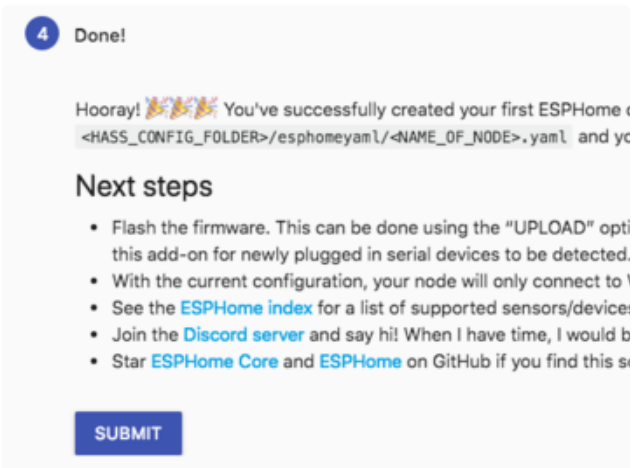
From the ESPHomeYAML dashboard, **click the Plus Icon to launch the ESPHome Setup Wizard.**









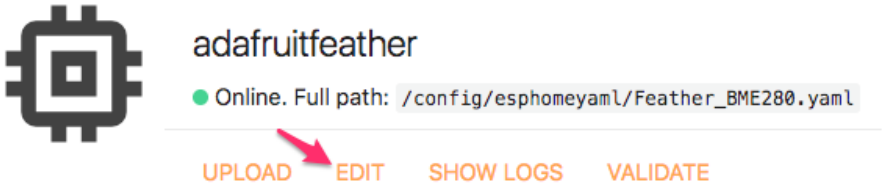


The ESPHomeYAML wizard will walk you through setting up a ESP-powered device using ESPHome.

First, name your device. If you're hooking up a lot of smart-devices to Home Assistant - make it as descriptive as possible.
Then, select **Adafruit HUZZAH ESP8266** as the Device Type
Enter the WiFi SSID and password for your network.
You can also **configure a password for** performing **Over-the-Air (OTA) updates** at this step. This'll allow you to connect to the Feather Huzzah from Home Assistant without connecting a USB cable, and flash firmware to it.
You're done with setup! **Click Submit**

After the setup wizard completes, a new node will appear on the ESPHome dashboard. We're not ready to upload the settings yet.

Click Edit



The `feather_bme280.yaml` file below is based off of the template provided in Boris Hajduk's guide `Using adafruit.io MQTT with esphomeyaml` (https://adafru.it/DQV).

Copy and paste the file below into the ESPHome Dashboard file editor:

```
# ESPHomeYAML Configuration
# for Home Assistant and Adafruit IO

substitutions:
  device_name: feather_bme280
  io_username: YOUR_ADAFRUIT_IO_USERNAME
  io_key: YOUR_ADAFRUIT_IO_KEY
  update_interval: 30s

esphomeyaml:
  name: ${device_name}
  platform: ESP8266
  board: huzzah

# WiFi Configuration
wifi:
  ssid: 'YOUR_SSID'
  password: 'YOUR_SSID_PASSWORD'
  # Uncomment this for Static IP Configuration
  #manual_ip:
    # Set this to the IP of the ESP
    #static_ip: 10.0.0.0
    # Set this to the IP address of the router. Often ends with .1
    #gateway: 10.0.0.1
    # The subnet of the network. 255.255.255.0 works for most home networks.
    #subnet: 255.255.255.0

# Adafruit IO MQTT Setup
mqtt:
  broker: 'io.adafruit.com'
  username: ${io_username}
  password: ${io_key}
  topic_prefix: '${io_username}/feeds'
  birth_message:
    topic: ${io_username}/feeds/status
    payload: Online
  will_message:
    topic: ${io_username}/feeds/status
    payload: Offline
  log_topic:

# Enable Home Assistant API
api:
```

```
# Enable OTA Access
ota:

# Enable verbose logging over serial
logger:

# Create BME Sensor on I2C
i2c:
  sda: SDA
  scl: SCL
  scan: False
binary_sensor:
    - platform: status
      name: "BME280 Status"
      id: status
      internal: True
sensor:
  - platform: bme280
    address: 0x77
    temperature:
      name: "BME280 Temperature"
      id: temperature
      state_topic: ${io_username}/feeds/temperature
      discovery: False
      filters: []
    pressure:
      name: "BME280 Pressure"
      id: pressure
      state_topic: ${io_username}/feeds/pressure
      discovery: False
      filters: []
    humidity:
      name: "BME280 Humidity"
      id: humidity
      state_topic: ${io_username}/feeds/humidity
      discovery: False
      filters: []
    update_interval: ${update_interval}
```

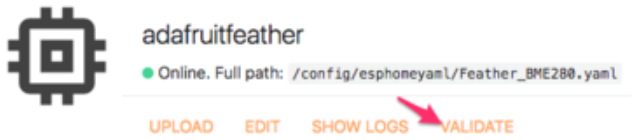
Before uploading the file, we'll need to **make the following adjustments to the YAML file** under `substitutions`:

- **Change** `device_name` to reflect the name of the device you created in the ESPHome Wizard.
- **Set** `io_username` to your Adafruit IO Username
- **Set** `io_key` to your Adafruit IO Key
- **Change** `update_interval` to the frequency (in seconds) which you'd like the feather to read the sensor and send its data to Home Assistant and Adafruit IO.

We'll also need to configure the WiFi network:

- **Set** `ssid` to your network's ssid
- **Set** `password` to your network's password

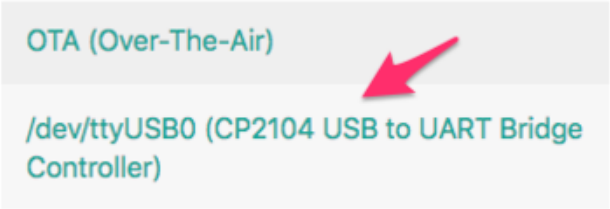
After configuration of the YAML file is complete, **click save**.



After uploading, we'll need to check the YAML configuration. From the ESPHome Dashboard, **click Validate**

If everything is OK, the dialog will display `INFO Configuration is Valid!`

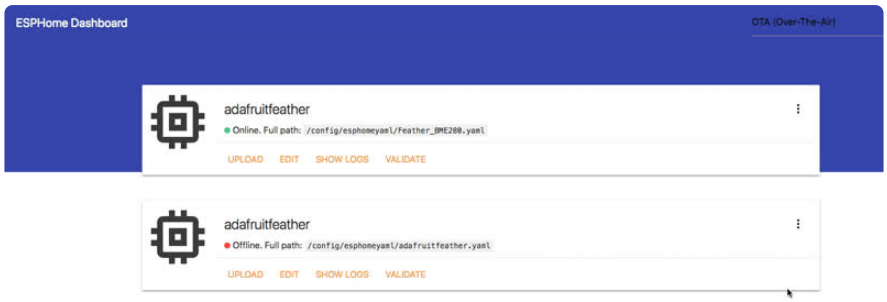
Finally, we'll move onto uploading the firmware to the ESP8266.

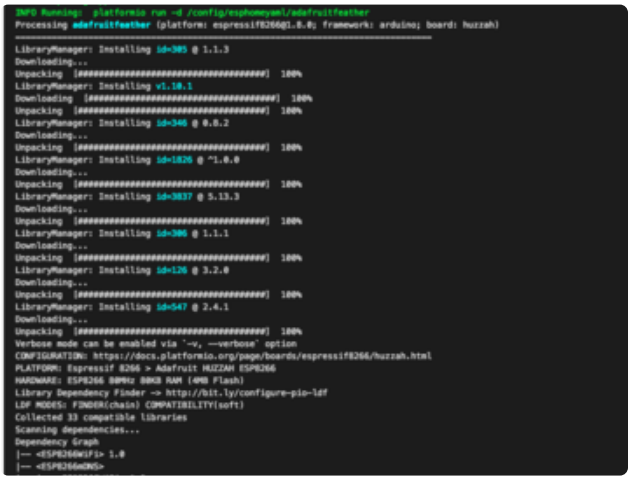


On the top right corner of the ESPHome Dashboard, make sure you have selected the **USB to UART Bridge Converter** and not OTA (over-the-air).

The ESPHome Add-On is not capable of discovering new USB ports after the Add-On has started. If you started the application before plugging in the Feather, restart the Add-On before proceeding with this step.

Under the settings for the Feather, **click UPLOAD**.





This will compile the firmware for the Feather and upload it over USB.

Since we're compiling firmware from the Raspberry Pi, this process will take a while to complete.

If everything compiles and uploads successfully, the Feather on the ESPHomeYAML dashboard will display as **Online**.

Flashing firmware using the Command Line

You can also flash firmware from the command line by following the instructions on the esphomeyaml Getting Started documentation (https://adafru.it/DQI).

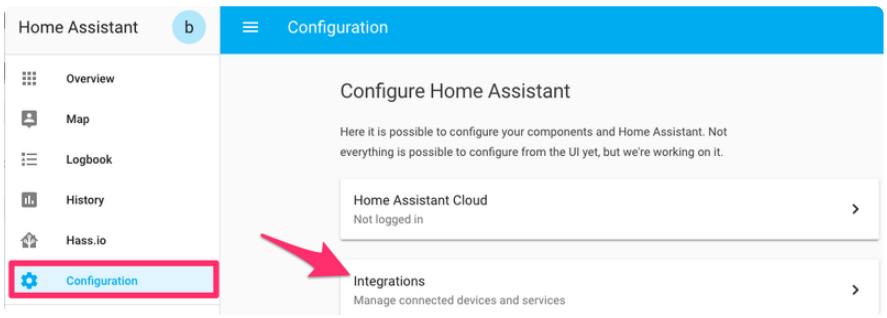
Note: To use the PlatformIO Python dependency required by esphomeyaml CLI - you'll need to install Python 2.7.9. If you don't want to mess with multiple installations of Python on your computer - you can install this version through PyEnv (https://adafru.it/DQJ).

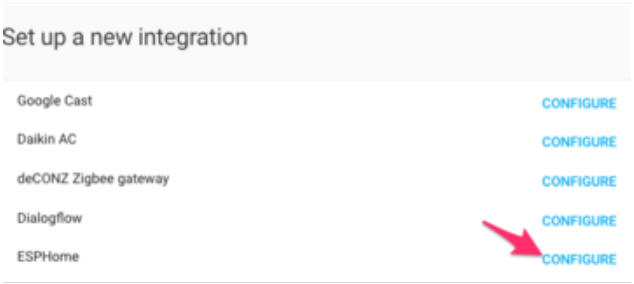
Usage

Viewing data with Home Assistant

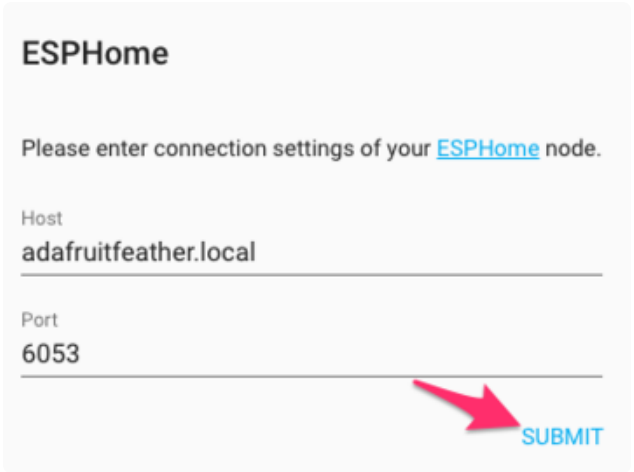
From the Home Assistant sidebar, **click Configuration**

Then, **click Integrations.**





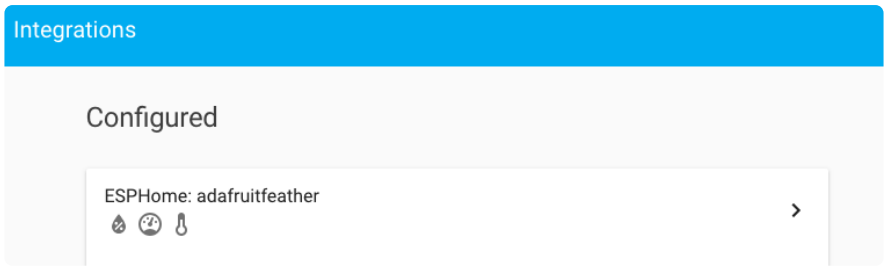
Scroll down to Set up a new integration, find ESPHome and **click Configure**



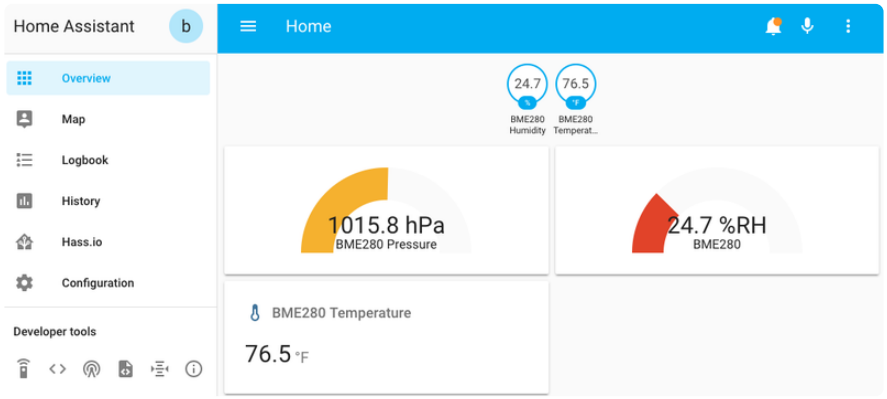
Set the Host to the `device_name` you configured during the YAML configuration and **set the Port to 6053** (default for ESPHomeYAML).

Home Assistant should attempt to connect to the Feather.

After connecting, the feather with BME280 will appear under **Configured** with three icons for the three types of data the BME280 sensor produces - humidity, pressure, and temperature.



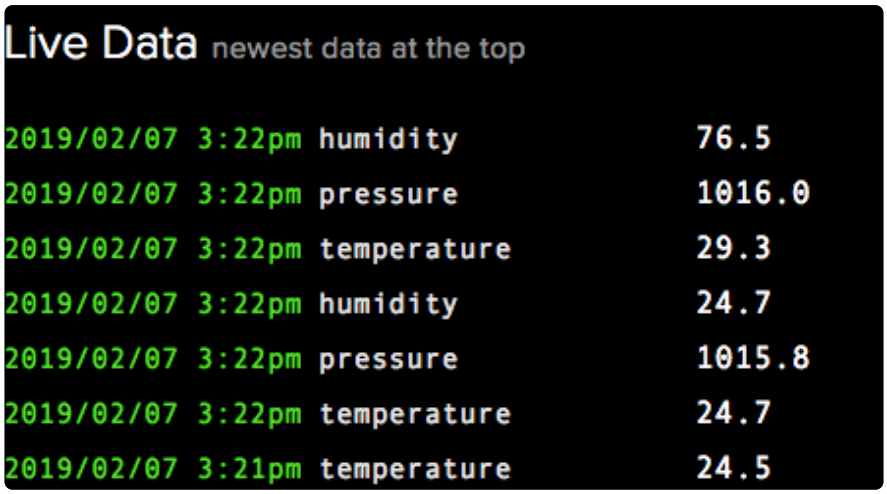
Navigating to the **Home** page should bring up the dashboard with the BME280 temperature, pressure, and humidity displayed.



Logging Sensor Data with Adafruit IO

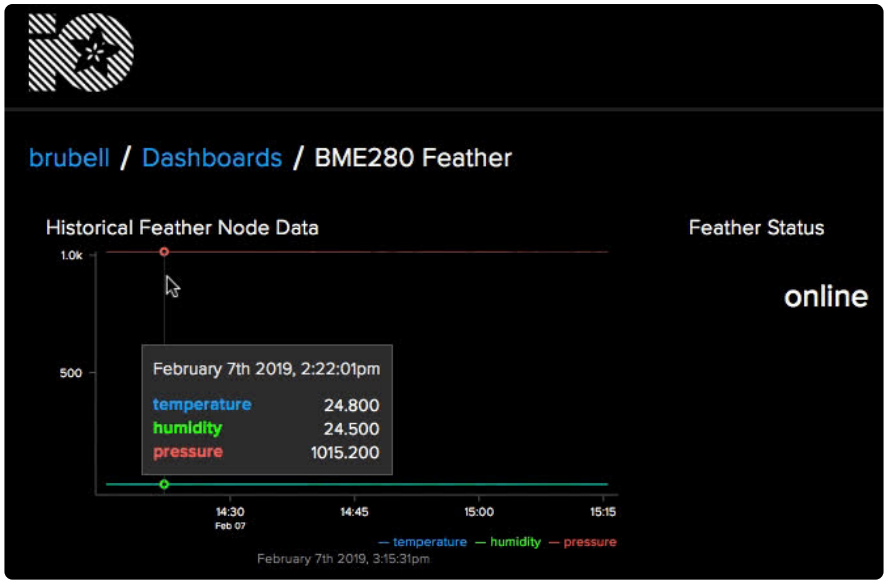
Data from the Feather Huzzah sensor is also sent to Adafruit IO every `updated_interval` seconds.

You can view this data in real-time from the Adafruit IO monitor page:

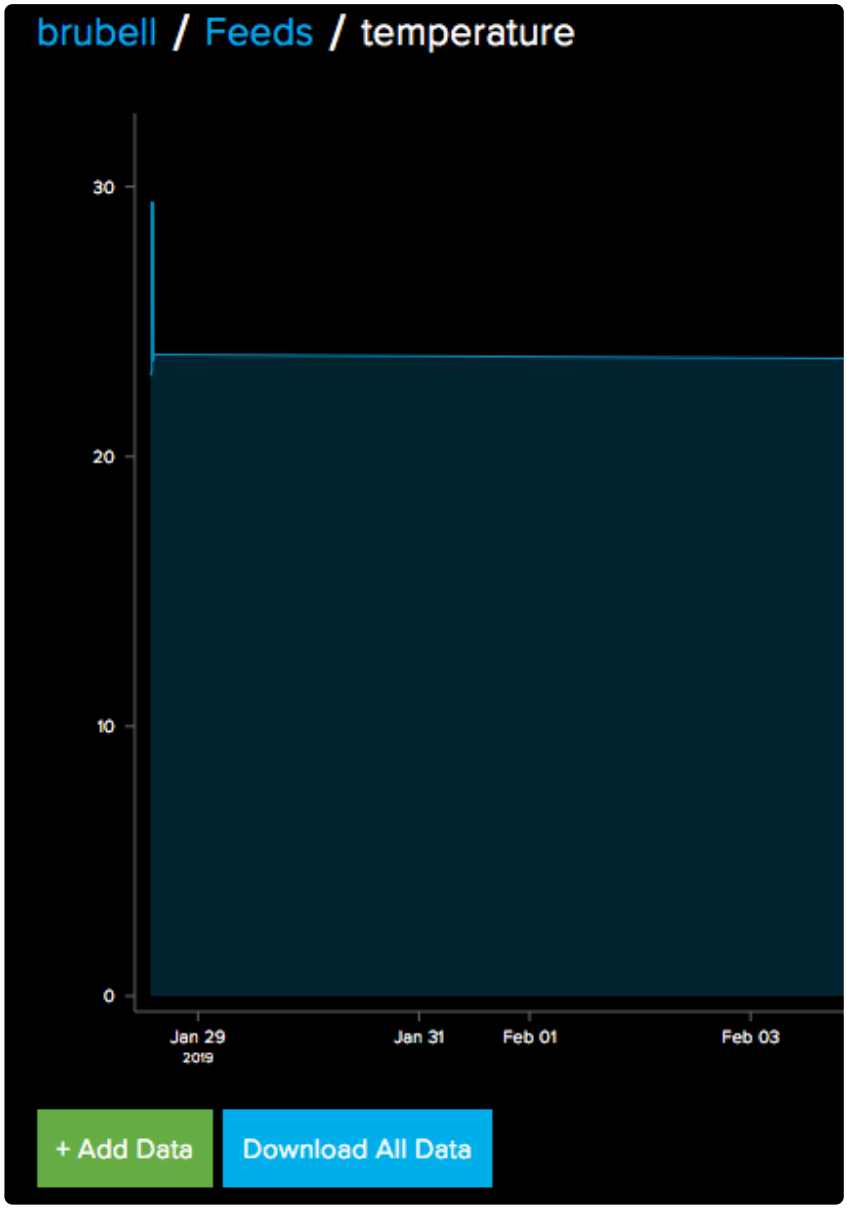


You can also create an Adafruit IO Dashboard with line graphs to visualize the data from the sensor over multiple weeks.

- For more information about working with dashboards in Adafruit IO, visit the Adafruit IO Basics: Dashboards (https://adafru.it/f5m)guide. (https://adafru.it/f5m)



Exporting Adafruit IO Data



If you would like to download all of the stored data from a feed (the BME280's temperature data, for example), you can do so by following this guide (https://adafru.it/DQW).