How to connect a GE appliance to Home Assistant

- This tutorial will work through all the steps needed to connect a GE appliance locally to Home Assistant.
- It assumes you have got the Seeed adapter hardware from First Build, and that you are reasonably familiar with Home Assistant.

Program the adapter

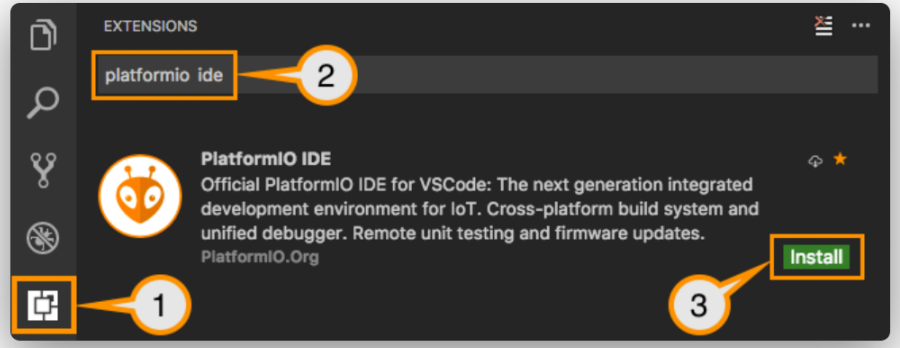
- This section sets up the adapter with information about your wifi, the Home Assistant MQTT broker, and what you want to call the appliance you are setting up.
- You will need the wifi SSID/password, the URL or IP address of the Home Assistant, and the MQTT broker password

Install or open PlatformIO

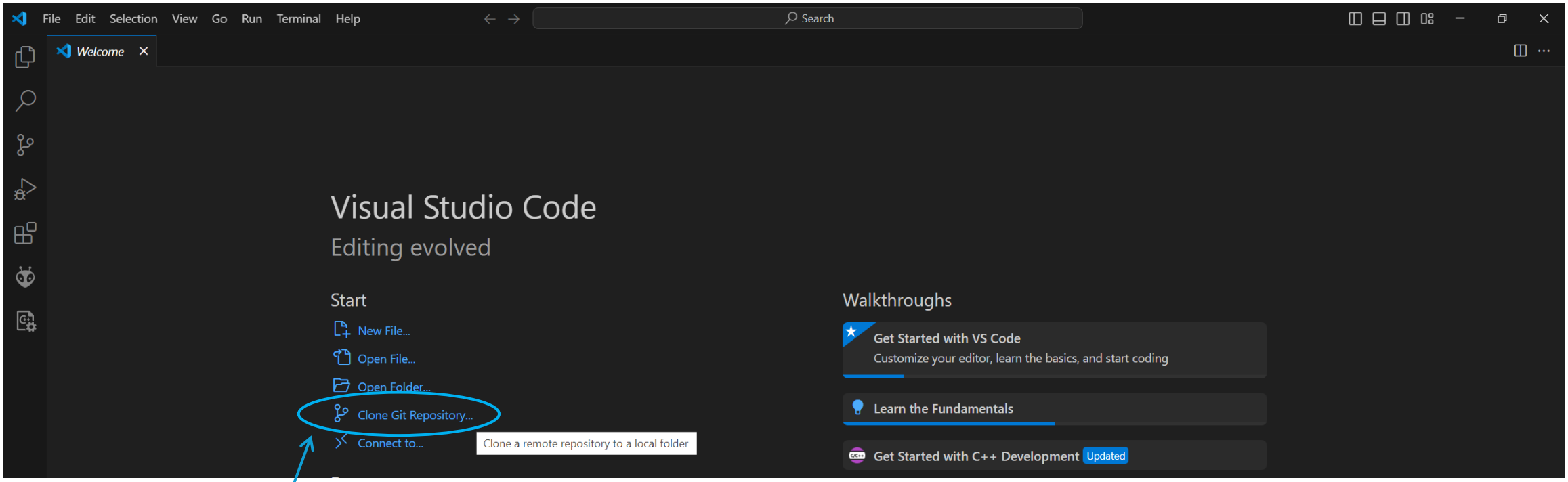
arted ▾ Solutions ▾ Registry Docs Community Support

Thank you for choosing PlatformIO IDE for VSCode

- Download and install official Microsoft's Visual Studio Code, PlatformIO IDE is built on top of it
- 1. **Open** VSCode Extension Manager
  2. **Search** for official PlatformIO IDE extension
  3. **Install** PlatformIO IDE.

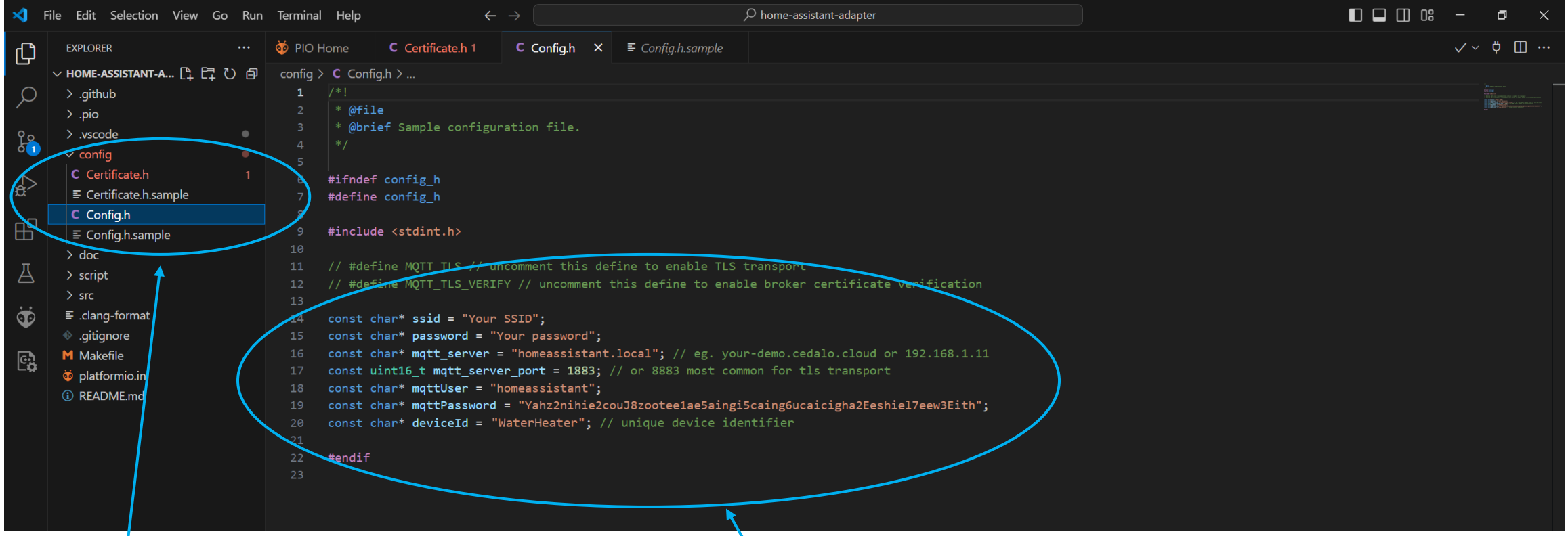


- Check Quick Start guide (highly recommended).



In a new window, select Clone Git Repository and clone the following repo:

https://github.com/geappliances/home-assistant-adapter

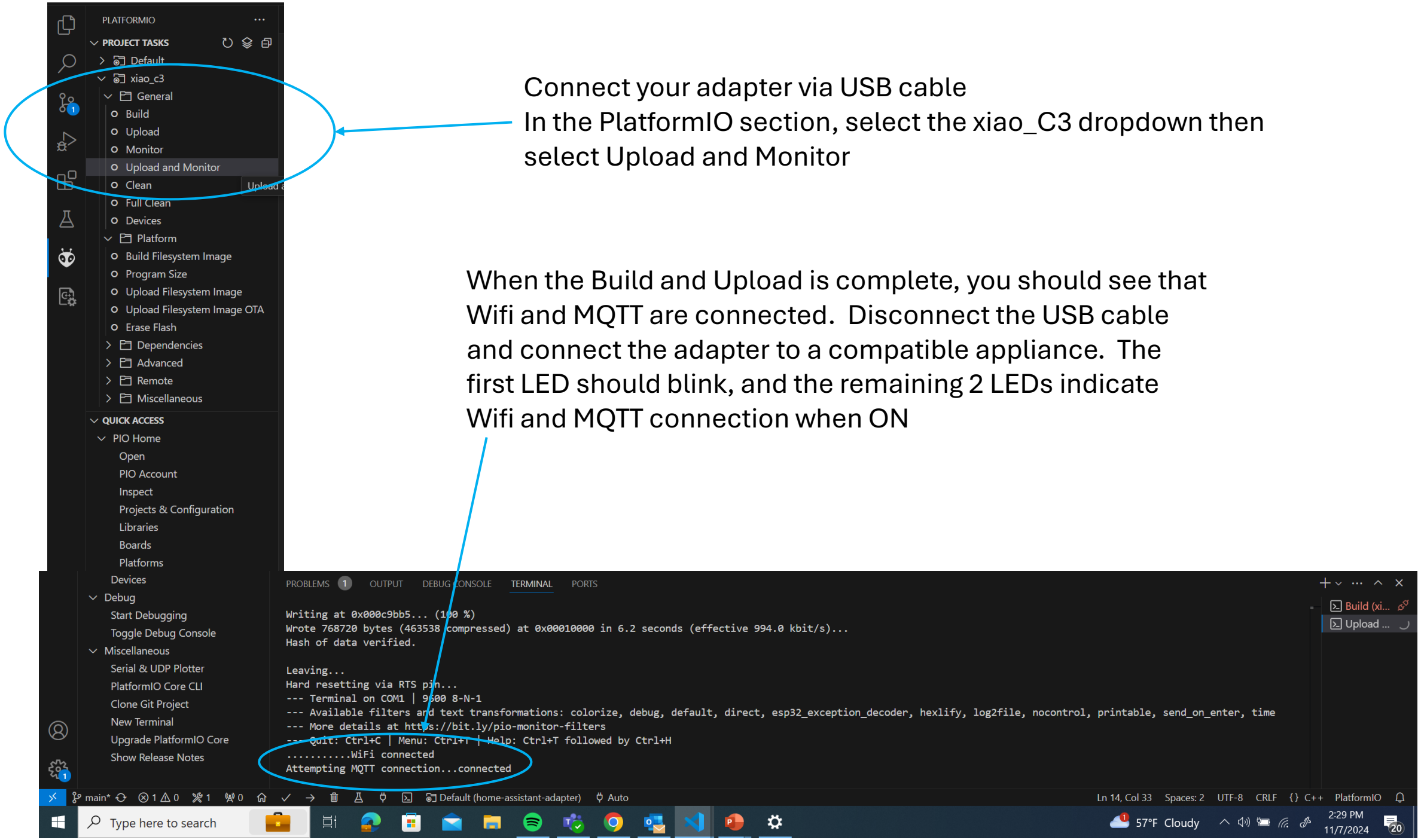


In the explorer, select the config dropdown

Edit Config.h.sample to add your Wifi credentials, Home Assistant user login info for mqtt and device name. Then save the file as Config.h (remove .sample)

Also select Certificate.h.sample and save as Certificate.h

The deviceId is important and case-sensitive – this is how the appliance will appear in MQTT



Connect your adapter via USB cable
In the PlatformIO section, select the xiao_C3 dropdown then select Upload and Monitor

When the Build and Upload is complete, you should see that Wifi and MQTT are connected. Disconnect the USB cable and connect the adapter to a compatible appliance. The first LED should blink, and the remaining 2 LEDs indicate Wifi and MQTT connection when ON

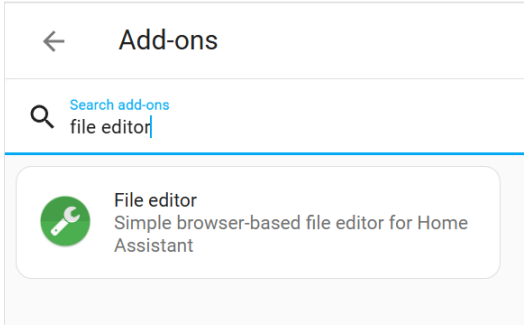
Set up Home Assistant add ons

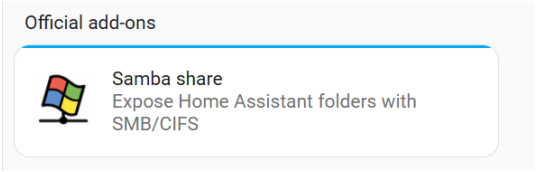
- This is where we set up the prerequisites:
  Mosquitto (MQTT broker) – this is how the adapter gets information to and from Home Assistant.
  Samba share – this allows you to access the Home Assistant file system over the network, and lets you create and edit files

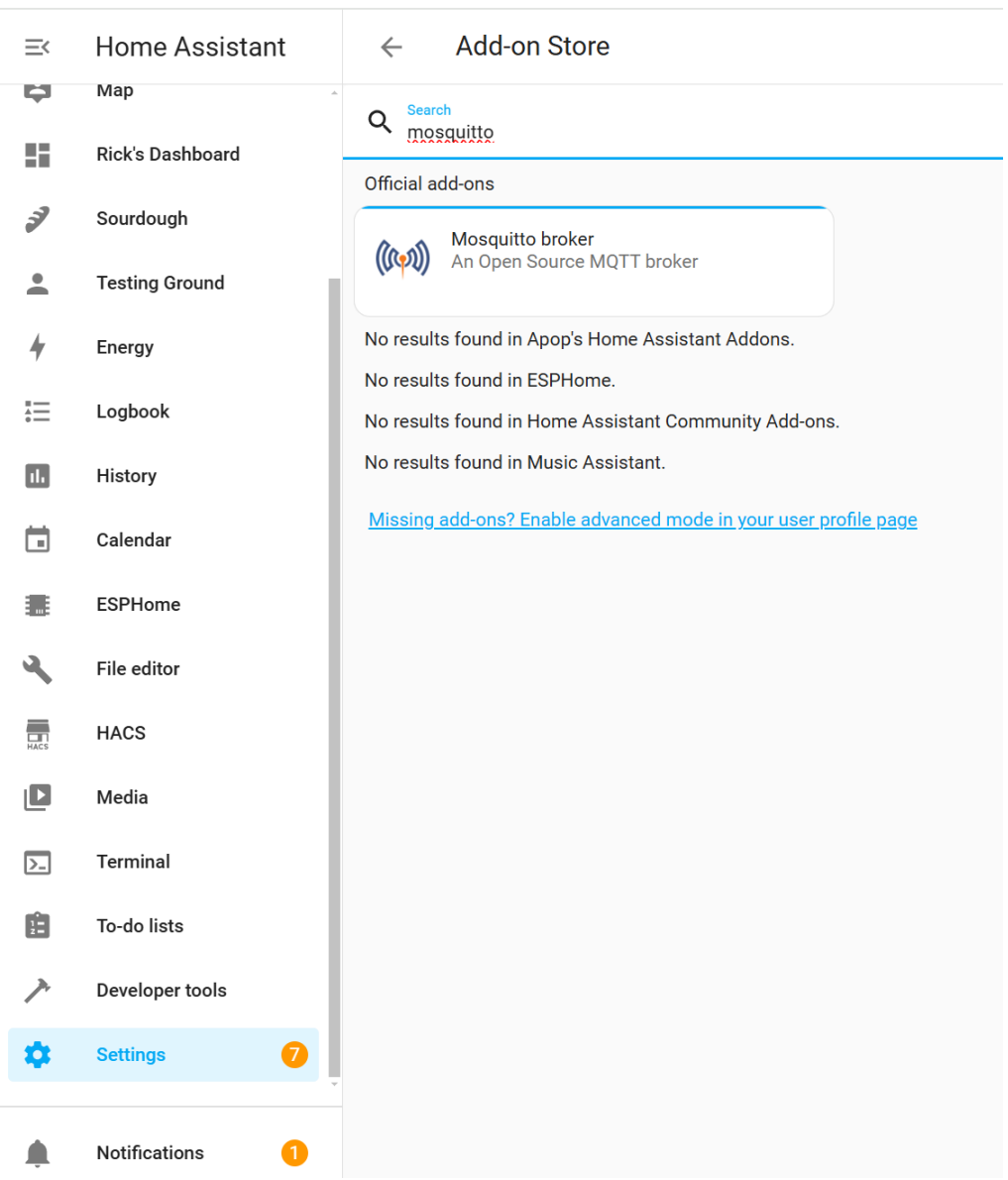
Within Home Assistant, install the Mosquitto MQTT broker by going to settings>Add-ons>ADD-ON STORE and searching for Mosquitto. Install and start the add-on.

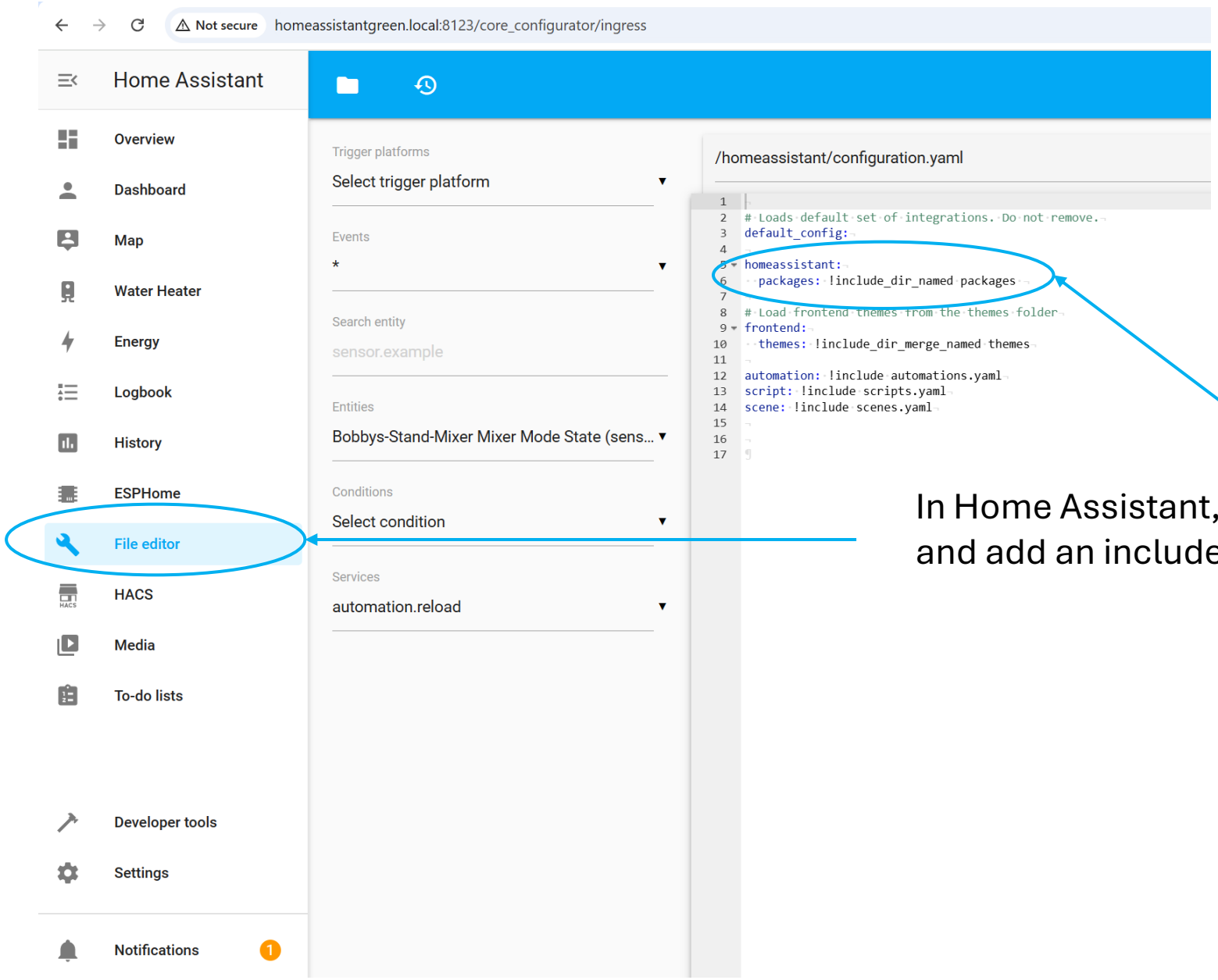
Install the File editor Add-on to edit configuration files

Also installing the Samba share add on is a convenient way to transfer yaml configuration files to your home assistant instance.







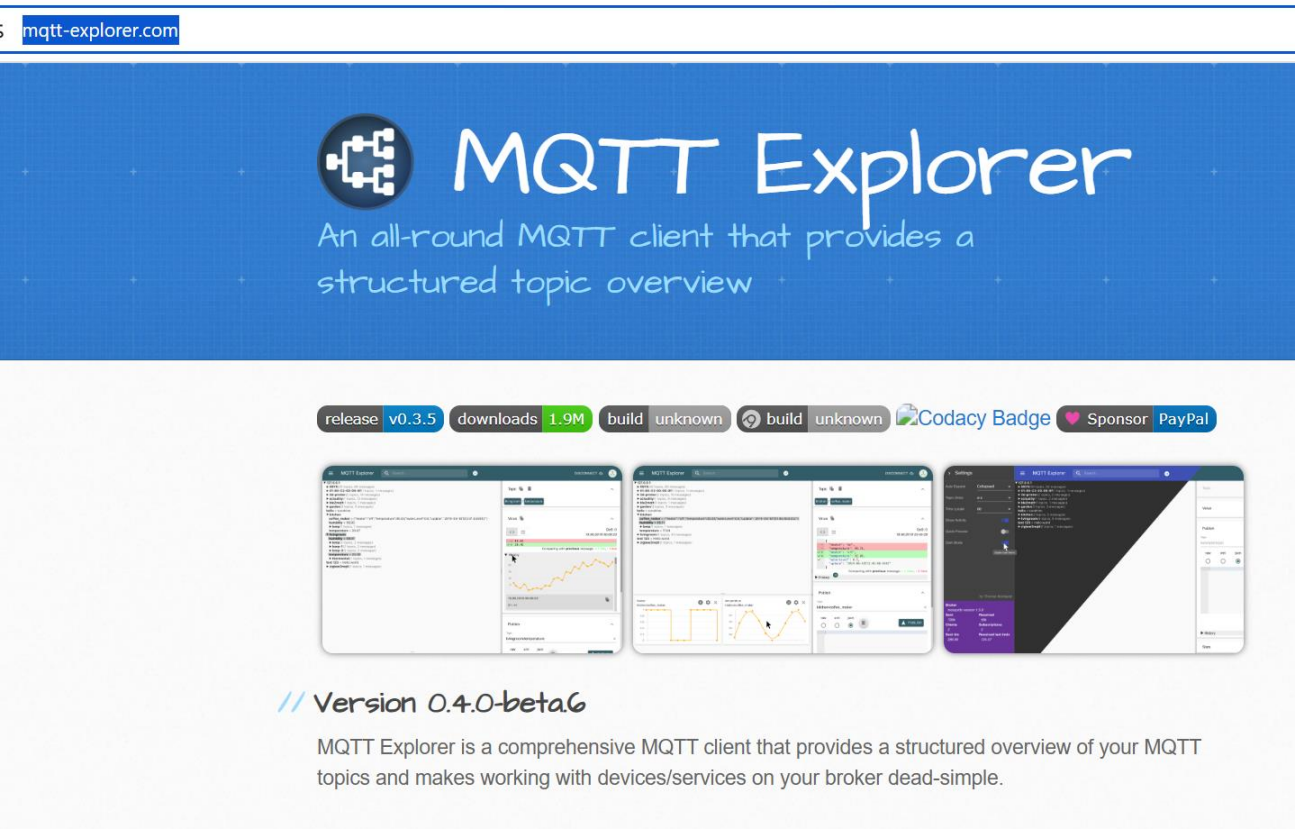


In Home Assistant, open configuration.yaml in the file editor and add an include for a packages directory.

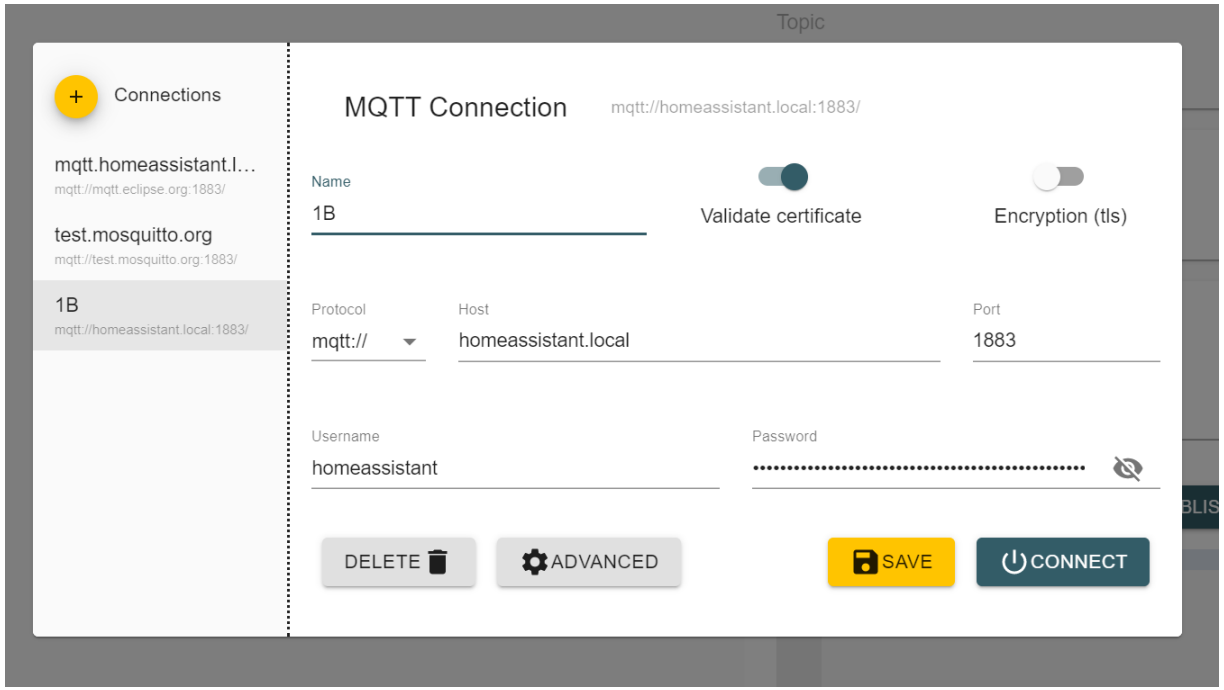
Nice to have tools

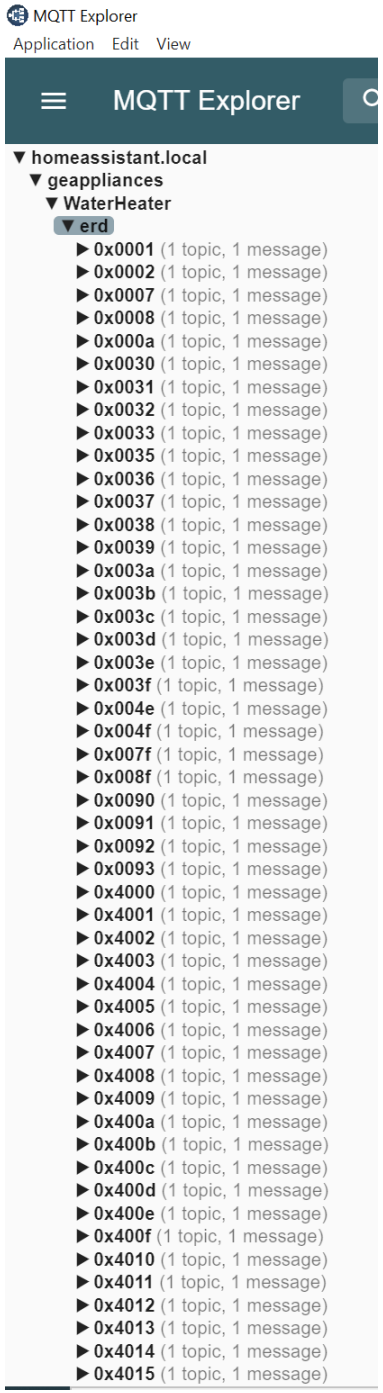
- While not essential, MQTT explorer running on a computer is VERY helpful for checking what is going on and making sure everything is working

Install MQTT explorer: https://mqtt-explorer.com/



Connect to your Home Assistant MQTT server, the Username and Password should be the credentials for a Home Assistant user (configured in Settings>People within Home Assistant)





In MQTT under geappliances and the device ID that you specified in config.h, you should see a list of ERDs. You may need to power cycle the appliance for the list to be fully populated.

The next step is to generate a yaml configuration file that will define what ERDs appear as entities in Home Assistant. ERD documentation can be found here: https://github.com/geappliances/public-appliance-api-documentation/

Details on configuration of MQTT components via Yaml in Home Assistant can be found here: https://www.home-assistant.io/integrations/mqtt/

Some example yaml files can be found in the packages directory of the github repo. To utilize the example files, transfer them to your Home Assistant instance (using Samba or other means) to /config/packages/

Let's walk through a specific example of creating Yaml to expose a specific ERD.....

Setting up Home Assistant

- This is where we add the configuration files that will make the data and controls for the appliance available inside Home Assistant

We might as well start at the beginning. ERD 0x001 is defined as follows in the ERD documentation:


```
 "name": "Model Number",
   "id": "0x0001",
   "operations": ["read", "publish", "subscribe"],
   "description": "The identifier used to specify what group an appliance belongs to. Unused bytes should be set to 0x00.",
   "updateClass": {
     "type": "legacy"
   },
   "data": [{
     "name": "Model Number",
     "type": "string",
     "offset": 0,
     "size": 32
```


This ERD reports the model number of the appliance. We can configure a MQTT sensor in Home Assistant to access this information. Details can be found here: https://www.home-assistant.io/integrations/sensor.mqtt/

Here is Yaml for this ERD. The unique_id is a way to identify the sensor. State_topic lets Home Assistant know where to find the sensor data, this can be found in MQTT explorer, notice the deviceId we specified in config.h is included in the state_topic. The value_template section converts the hex data of the ERD to a readable string. The device section defines the MQTT device in Home Assistant under which this sensor will appear. You can either create a yaml file in the packages directory using the file editor within Home Assistant, or create the file on another device and transfer via Samba. Once the file is in place you can use Developer Tools>Check Configuration from Home Assistant in order to verify you haven't generated an error that would prevent Home Assistant from starting, and then ALL YAML CONFIGURATION from the Yaml reloading section to load the Yaml. After the restart, under Settings>Devices and Services>MQTT you should see the devices and entities that you created.

```
# - ========================================
# - MQTT
# - ========================================
mqtt:
 # - ========================================
 # - MQTT Sensors
 # - ========================================
 sensor:
  - name: Model Number
   unique_id: Waterheater_modelnumber
   state_topic: "geappliances/WaterHeater/erd/0x0001/value"
   value_template: >
    {%- set line = value %}
    {%- set chars = " !'#$%&\'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_` abcdefghijklmnopqrstuvwxyz{|}~" %}
    {%- set n = 2 %}
    {%- set ns = namespace(value='') %}
    {%- for i in range(0, line | length, n) %}
    {%- set c = chars[line[i:i+n] | int('', 16) - 32 ] %}
    {%- if c is defined %}
    {%- set ns.value = ns.value ~ c %}
    {%- endif %}
    {%- endfor %}
    {{ ns.value[0:12] }}
   device:
    name: "WaterHeater"
    identifiers: “Heat Pump Water Heater"
    manufacturer: "GE Appliances"
```