Advanced Setup

Covering more advanced setup topics as I learn them

- 433 MHz Devices
- Frigate NVR
- ESP Device Flashing Issues
- Tesla Fleet Setup

433 MHz Devices

Why 433 MHz?

While Wifi, Zigbee, Z-wave, and Matter/Thread are great, sometimes you just need a simple sensor that does one thing really cheaply. 433MHz devices (and others like 915MHz and 868MHz) are one-way sensors that can talk to Home Assistant to report things like temperature, water, moisture, open/close and much more.

The advantages of these devices are:

- They are cheap
- They are everywhere
- They have extremely long battery life
- They have decent range
- You might already have some
- The communication protocols are simple and reliable, no device pairing needed

The downsides are:

- You need a special receiver, called an SDR (Software Defined Radio)
- Sensors can *sometimes* require more configuration
- It can be difficult to determine which devices are supported
- Not everything listed as 433MHz can be used
- Setting up your SDR receiver takes more steps, but this guide will show you how

And as either a downside or an upside depending on your view is that you might learn a lot about your neighbors.

Equipment

SDR receiver

The most important thing you will need is a SDR receiver. This is a USB device with an antenna which can receive radio signals and pass them to your HA device.

I purchased the Nooelec NESDR Mini 2+ as an entry level device for under $40, but there are a range of devices of all prices out there. Just do a quick search to ensure it works with Home Assistant. If the Mini 2+ isn't on sale, the regular NESDR Mini 2 is great too. Any device using a

Realtek chip should work. Other companies make chips, but these are very affordable ones. You'll often see the term `rtl` referenced, which stands for *Realtek Limited* (Examples: rtl-srd, rtl-433).

Sensors

You're going to want something to sense. Here are some good suggestions to start with, but as you get comfortable search around for more device types, as you can do so much with SDR!

- **Temperature/Humidity:**
  - AcuRite
    - Get Sensor #2 without ABC since we don't need channel selection and it's cheaper
  - ecowitt
- **Water/Leak:**
  - Govee
    - I have not yet verified these actually use 433, they may have changed to LoRa
- **Motion Detection:**
  - Still investigating
- **Soil Moisture:**
  - Springfield 91746†
  - OPUS XT300†

Remember how I mentioned learning about your neighbors? Chances are that you and your neighbors already have some devices transmitting on 433MHz. Before I ever turned on my first sensor, my alarm system door sensors and my neighbors soil sensors both popped up in Home Assistant! Car TPMS (Tire Pressure Monitoring Systems) also show up, but because you likely don't want to have a dashboard filled with every car that drives by these are disabled by the default config below.

† These devices I have not tested myself, but my neighbors must be using them because I've picked up their signal and have been able to read them.

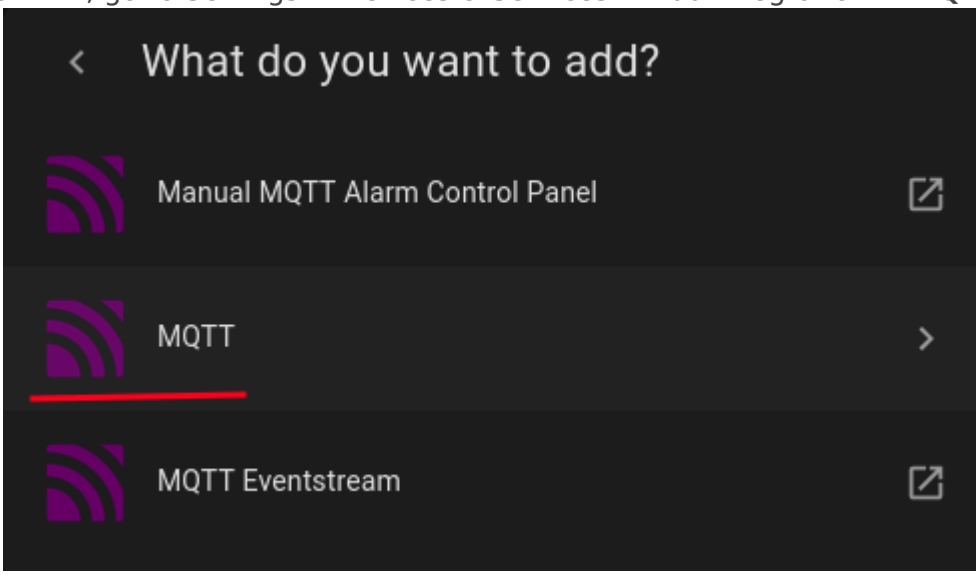
Setting Up Your SDR Receiver

You're receiver has arrived, let's set it up! If your sensor has arrived too, great! If it hasn't lets set the receiver up anyway and see what we find.

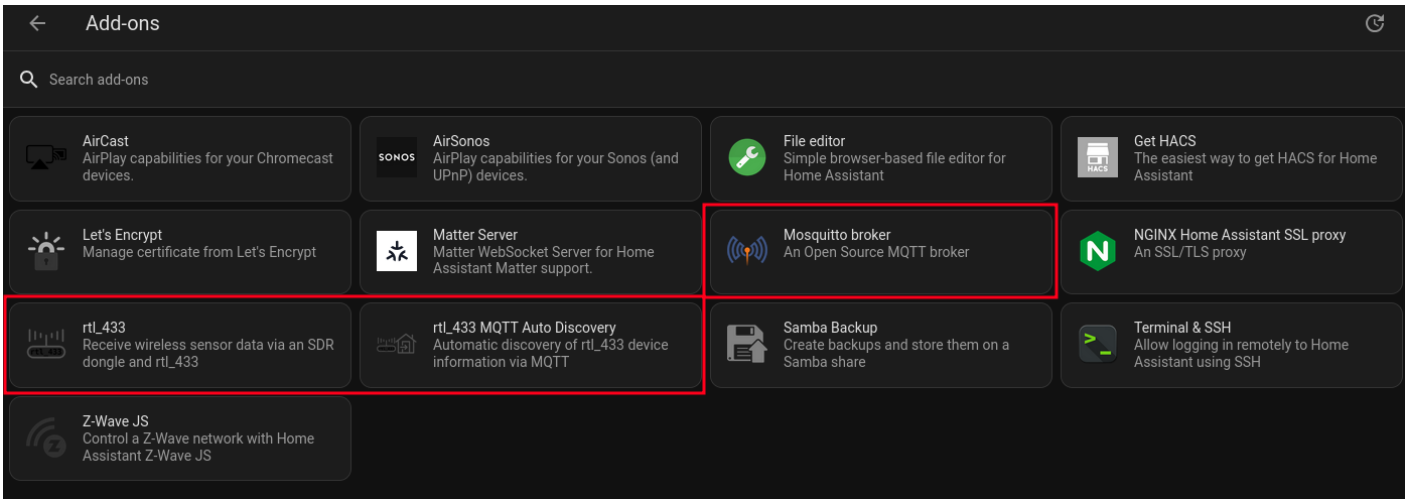
> **Note:** If you want to play with the SDR receiver on your linux desktop/laptop first, see my guide here: https://www.wswapps.com/books/debian-ubuntu-pop-os/page/install-configure-

> sdr-receiver

Add-on Installation

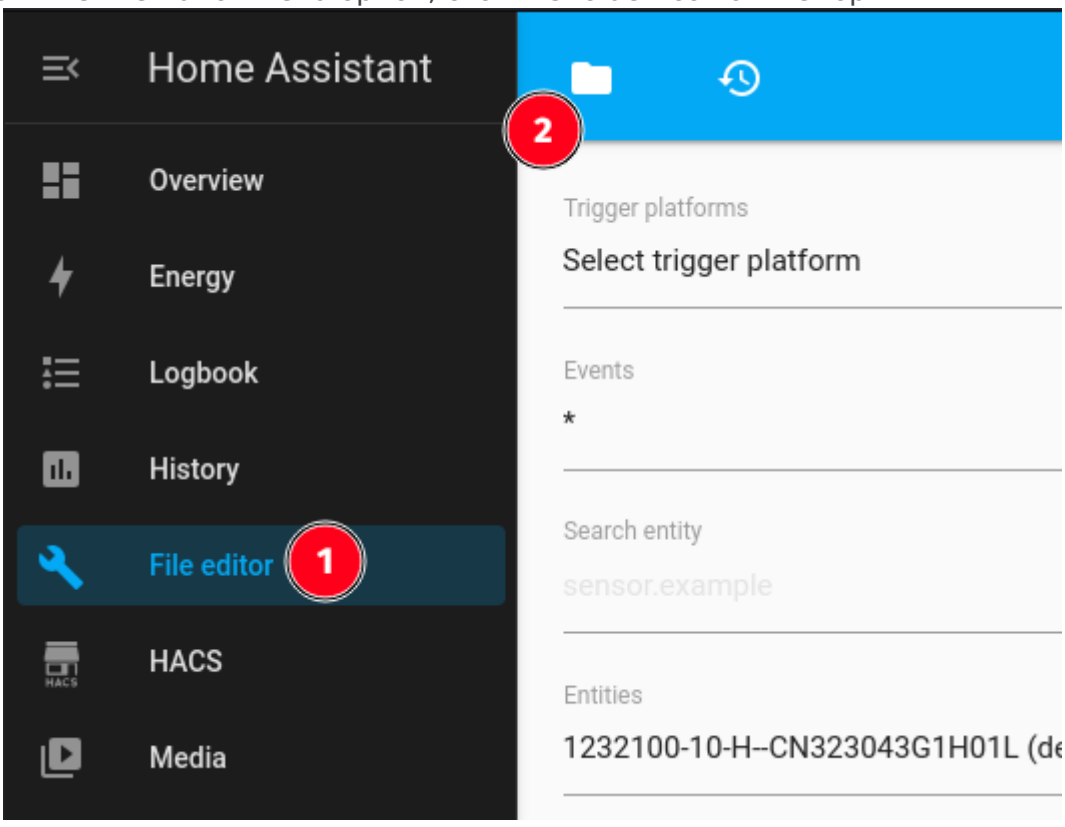
- From HA, go to Settings > Devices & Services > Add Integration > **MQTT**
  - 

  - All your future devices will show up under this MQTT service
- Next, go to Settings > Add-ons > Add-on Store. Select and install: **Mosquito broker**
- Start Mosquito Broker
  - No configuration is needed
  - This add-on is necessary to take in the detected sensors/messages and publish them to Home Assistant as devices and sensors
- Also from the Add-on store, select the 3-Dot Menu > Repositories and Add the following: `https://github.com/pbkhrv/rtl_433-hass-addons`
  - This new repository contains the repos we need to use the SDR receiver
- Install both of the following Add-ons now and **start** them:
  - **rtl_433**
  - **rtl_433 MQTT Auto Discovery**
    - You can remove/disable this later if you prefer, but it will make getting started a lot easier. This add-on will auto-detect nearby devices and add them to HA MQTT automatically

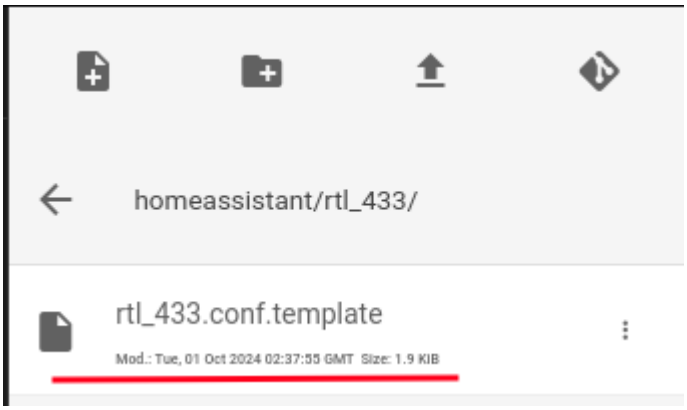


Add-On Configuration

We're almost done. We now need to create a configuration file for rtl_433. To do that you need a way to upload or edit files in Home Assistant. If you have a favorite way, use that. For this demo I will use the **File Editor** which you can install from the Add-on section. make sure to enable "Show in sidebar" for the File Editor after installing and starting it.

- From the File Editor menu option, click the folder icon at the top
  - 

- Click on the **rtl_433** folder and then click on the file named: **rtl_433.conf.template**

- 

- The default config file will be displayed. We will just make a few changes and I'll post the full file below in case that's easier:
  - Remove the `#` before `output kv`
    - This will enable you to see pretty logs on the rtl_433 Add-on of anything it detects.
  - Add the following new lines (anywhere, but I recommend just after the output kv line):
    - `frequency 433.92M`
      `convert si`
      - Those lines set the frequency to 433Mhz devices and ensure that measurement units are standardized. Optional but recommended.
- The rest of the file can remain as-is. It configures passing the data to MQTT and disables any Tire Pressure sensors that are detected.
- Click the Save icon at the top tos ave your changes.
- The full file at the end should look like this:

rtl_433.conf.template

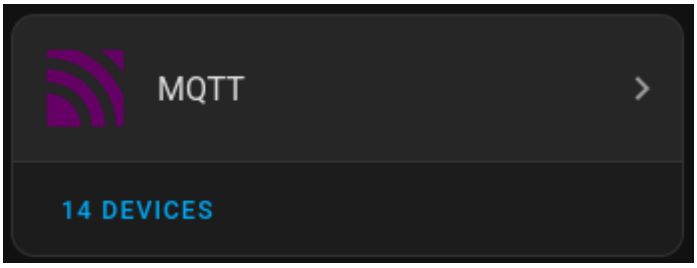
```
# This is an empty template for configuring rtl_433. mqtt information will be
# automatically added. Create multiple files ending in '.conf.template' to
# manage multiple rtl_433 radios, being sure to set the 'device' setting. The
# device must be set before mqtt output lines.
# https://github.com/merbanan/rtl_433/blob/master/conf/rtl_433.example.conf

output mqtt://${host}:${port},user=${username},pass=${password},retain=${retain}
report_meta time:iso:usec:tz

# To keep the same topics when switching between the normal and edge versions,
# use this output line instead.
# output
mqtt://${host}:${port},user=${username},pass=${password},retain=${retain},devices=rtl_433/9b13b3f4
-rtl433/devices[/type][/model][/subtype][/channel][/id],events=rtl_433/9b13b3f4-
rtl433/events,states=rtl_433/9b13b3f4-rtl433/states
```

```
# Uncomment the following line to also enable the default "table" output to the
# addon logs.
output kv

frequency     433.92M
convert    si


# Disable TPMS sensors by default. These can cause an overwhelming number of
# devices and entities to show up in Home Assistant.
# This list is generated by running:
# rtl_433 -R help 2>&1 | grep -i tpms | sd '.*\[(\d+)\].*' 'protocol -/run.sh'
#    [59]  Steelmate TPMS
#    [60]  Schrader TPMS
#    [82]  Citroen TPMS
#    [88]  Toyota TPMS
#    [89]  Ford TPMS
#    [90]  Renault TPMS
#    [95]  Schrader TPMS EG53MA4, PA66GF35
#    [110]  PMV-107J (Toyota) TPMS
#    [123]* Jansite TPMS Model TY02S
#    [140]  Elantra2012 TPMS
#    [156]  Abarth 124 Spider TPMS
#    [168]  Schrader TPMS SMD3MA4 (Subaru)
#    [180]  Jansite TPMS Model Solar
#    [186]  Hyundai TPMS (VDO)
#    [201]  Unbranded SolarTPMS for trucks
#    [203]  Porsche Boxster/Cayman TPMS
protocol -59
protocol -60
protocol -82
protocol -88
protocol -89
protocol -90
protocol -95
protocol -110
protocol -123
protocol -140
```

```
protocol -156
protocol -168
protocol -180
protocol -186
protocol -201
protocol -203
```

- As you learn more, you can update this config file. This example config file shows lots of good options that can be configured
- Later on we'll edit this config file to listen to more than one frequency in case you have devices on other frequencies.
- Go back to Settings > Add-ons, select **rtl_433** and **restart** it.
- Check the Log tab and you should see your SDR receiver detected. Let it run for a few minutes and refresh the logs to see if it detects anything. It may not if you aren't nearby any sensors.

This setup process will automatically detect new sensors and add them to the MQTT integration within HA. There will not be a notification of a new device, they will just show up and you can Disable them if you don't want them. If you find too many devices are showing up, you can disable the **rtl_433 MQTT Auto Discovery** Add-on. Down below we'll discuss adding a sensor manually if needed.



Configure a Sensor

If you have the **rtl_433 MQTT Auto Discovery** Add-on enabled, you don't need to do anything. Any commonly recognized devices will show up automatically in the MQTT integration for you to begin using. If you aren't using Auto Discovery or your devices is not one that can be auto discovered, it can be manually configured.

TODO: documentation on manually configuring a sensor

Monitor Multiple Frequencies

To monitor multiple frequencies, you need to update your **rtl_433.conf.template** file.

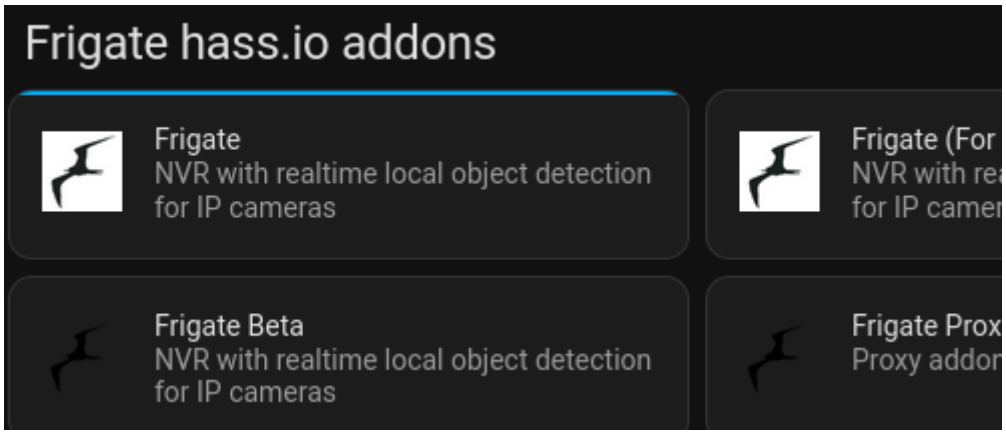
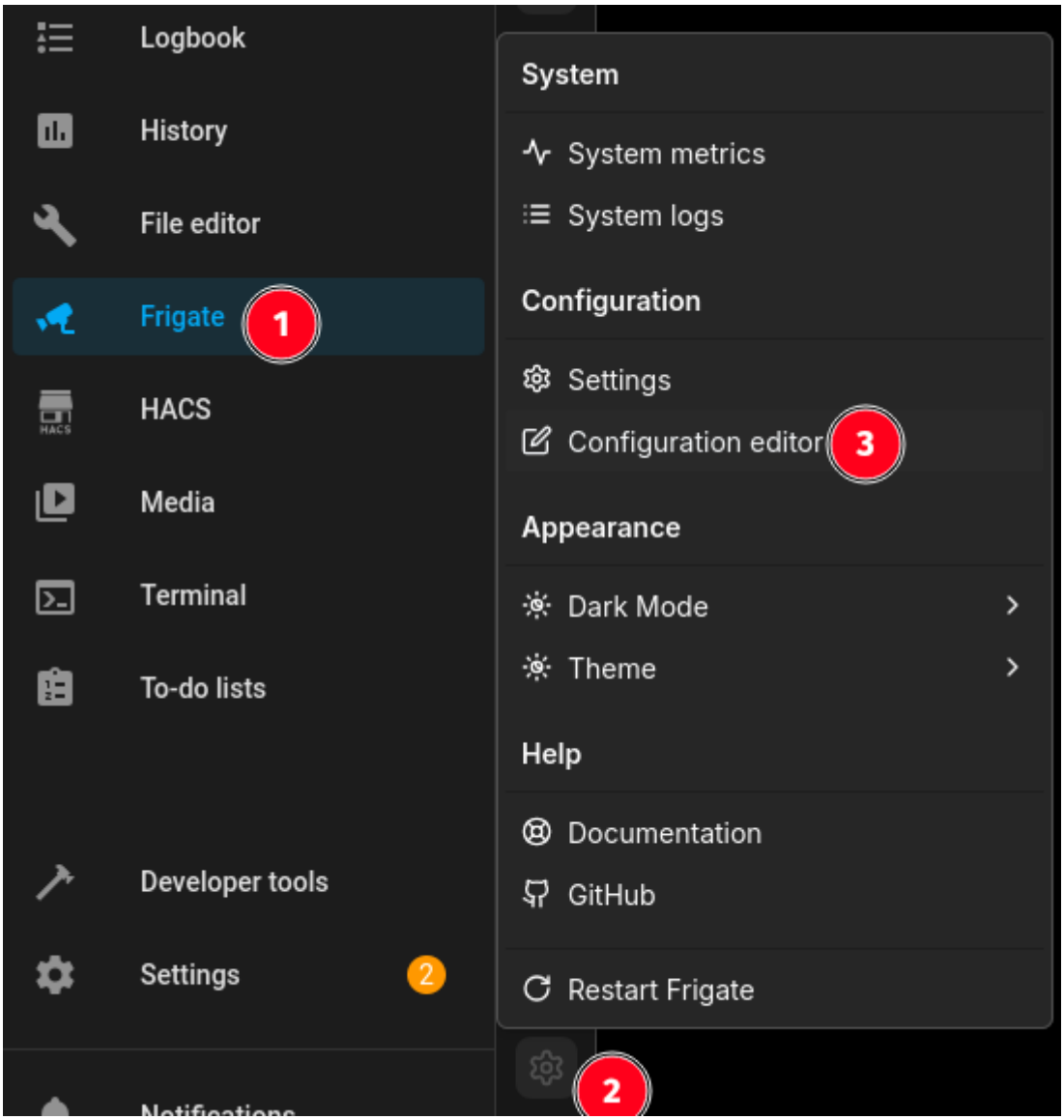
- Add the following two lines, then Save the file:
  - `hop_interval 30`
    `frequency 915M`
- Example:
  - ```
    output kv

    hop_interval 30
    frequency 433.92M
    frequency 915M
    convert si

    ```
- Restart the rtl_433 Add-on.

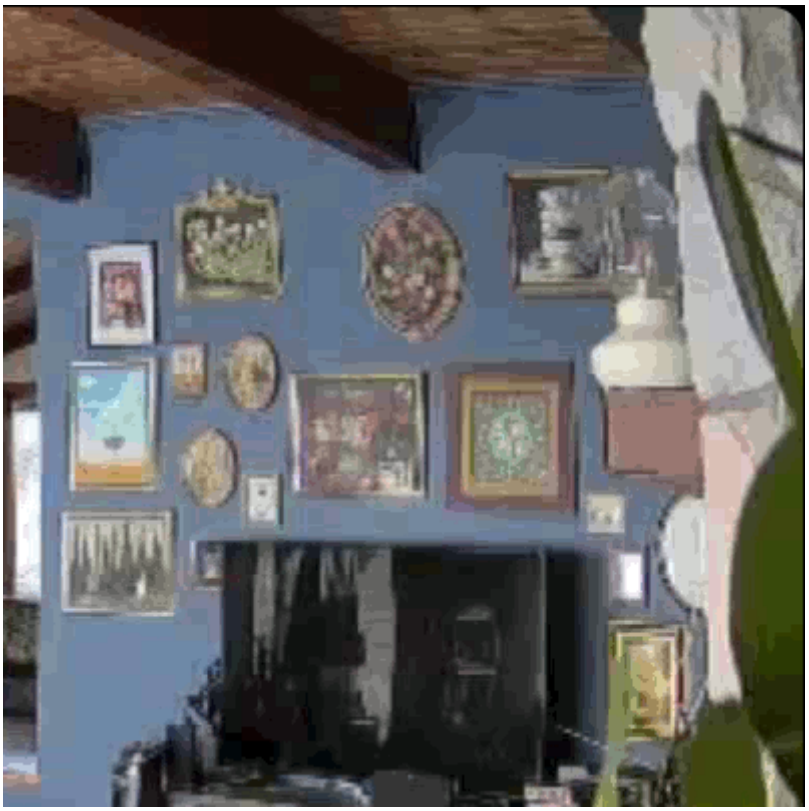
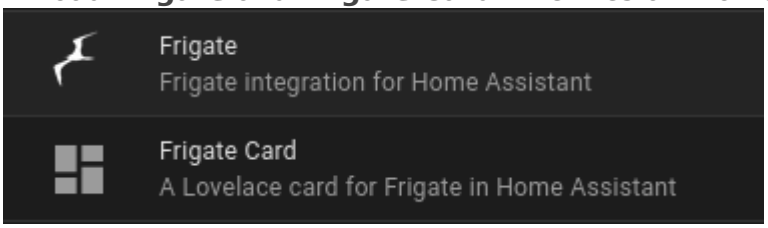
The hop_interval defines how many seconds each frequency will be scanned, before hopping to the next. If you want one frequency scanned more than another, you can list the same frequency multiple times.

The downside of this is that there is a period of time where you are not scanning each frequency and can miss messages. To scan each frequency without missing anything you would need to get an additional SDR receiver per frequency and configure multiple config files. This guide doesn't cover that setup (yet).

More Info

Here is a collection of resources and guides I used to help me get started with SDR and 433Mhz in Home Assistant:

- https://community.home-assistant.io/t/home-assistant-add-on-rtl-433-with-mqtt-auto-discovery/260665
- https://static.xtremeownage.com/blog/2021/433mhz-automation/#installing-rtl_433
- https://www.reddit.com/r/homeassistant/comments/10pkerb/i_have_no_idea_how_to_use_rtl_433/
- https://www.youtube.com/watch?v=_COwsvkxyFA
    ```

Frigate NVR

The goal with a Network Video Recorder (NVR) is to free yourself from cloud subscriptions for security cameras. You can (and probably should) run a standalone NVR, but in this guide we'll setup Frigate NVR as a Home Assistant Add-On.

Equipment

To run an NVR alongside Home Assistant you will likely need more CPU power than a Raspberry Pi provides, but most mini PCs should have enough. You can always try it out and find out.

If you plan to use any detection features at all (object tracking, notifications for specific objects) then you will need some sort of hardware acceleration. The Google Coral USB Accelerator device is recommended and is what we'll cover in this guide.

The Frigate recommended hardware guide has more details on both hardware acceleration and CPU recommendations.

Cameras

> See list of cameras I've tested at the bottom

You will need cameras that support RTSP and optionally ONVIF. ONVIF is needed for PTC cameras if you want Frigate to be able to control their movement. You also ideally want cameras that can output 2 streams, 1 high and 1 low resolution. High resolution will be used for viewing and the lower resolution for detection tasks.

In this guide I will be using a Tapo C210 camera which can sometimes be found on sale under $20 and supports Pan & Tilt as well as 2K resolution. Tapo cameras support RTSP and ONVIF as well as 2 different quality streams. Unfortunately they also require the Tapo app to perform the initial setup. I will be searching for other cameras that can be used entirely offline in the future and update this guide.

Setup Cameras

The first step is to setup your cameras and enable RTSP. This will vary based on the brand.

- Be sure to set a secure random username and password for your cameras when enabling RTSP
- In your router, reserve the camera's IP address so that it won't change on you. **Do not** expose the camera to the internet.
- Test your camera's RTSP stream in something like VLC. The exact string will differ based on the brand, but common examples are:
  - Tapo: `rtsp://username:password192.168.0.101:554/stream1`
    - Using `stream2` will load the low quality 720p stream.
- Make sure your RTSP stream is working before moving forward

Install Frigate

- In HA go to Settings > Add-Ons > Add-On Store
- In the upper right click the 3 dots, select Repositories.
  - Enter `https://github.com/blakeblackshear/frigate-hass-addons` and click **Add**
- Now under the **Frigate haas.io addons** section, select **Frigate** and click **Install**
  - 

  - Frigate (Full Access) is another option that can be used if the regular Frigate is not able to properly connect to your devices. It is not recommended unless absolutely needed since it has unrestricted access to your computer. Install the regular Frigate for now and you can always install Full Access later without requiring any additional configuration. Unfortunately they are not clear about exactly what reasons the Full Access version may be needed for.

- Enable the **Start on boot**, **Watchdog**, and **Show in sidebar** options.
- Click Start to start Frigate

Add-On Configuration

We now need to update the configuration file for Frigate to tell it about any cameras. To do that you need a way to upload or edit files in Home Assistant. If you have a favorite way, use that. For this demo I will use the File Editor built into Frigate itself.

- From the Frigate menu option, click the gear icon at the bottom of the page and then select **Configuration editor**



- The default Frigate config file will be displayed in YAML format. We need to add our first camera to it.
- Under the **cameras:** section lets add our camera:

Expand to see frigate.yaml

Keep any code above and below the cameras section that is already in the config file

```
cameras:
  C210: # <------ Name your camera
    enabled: true
    ffmpeg:
      inputs:
        - path: rtsp://username:password@192.168.0.101:554/stream1 # The High Quality stream you want to use for recording
          roles:
```

```
      - record
    - path: rtsp://username:password@192.168.0.101:554/stream2 # The Lower Quality stream you want to use for detection
      roles:
        - detect
  detect:
    enabled: false # Disable until you have a working camera feed and hardware acceleration
    width: 1280 # The resolution of the detection camera feed.
    height: 720
    fps: 5 # This can be raised later if we have the processing power
  record:
    enabled: false
    retain:
      days: 7 # The number of days a recording will be kept for after a motion is detected
      mode: motion
    events:
      retain:
        default: 30 # The event data will be kept for 1 day. After this period, the event data will be automatically deleted.
        mode: motion
# Leave out this section if camera is not PTZ or does not support ONVIF
  onvif:
    host: 192.168.0.101
    port: 2020 # Port may differ for your camera brand!
    user: username
    password: password
    autotracking:
      enabled: false # Disable until you have hardware acceleration
      calibrate_on_startup: true
      zooming: disabled # Can enable if camera supports Zoom
      track:
        - person

  objects:
    track:
      - person
      - car
```

- Click the **Save & Restart** button
  - 
- If Frigate fails to restart due to an error in your configuration file, in HA go to: **Settings** > **Add-Ons** > **Frigate** > **Log** to see the error message. Be very careful about your file formatting and spacing. Indentation and whitespaces matter in YML files.
- Going back to **Frigate** from the HA side menu, you should now see your camera feed. Clicking the camera feed opens it in full view and If you enabled ONVIF will also show your PTZ controls
  - 

> Additional Setup and usage instructions are coming soon. In the meantime go ahead and setup more cameras and play around with Frigate. You can try enabling detection in your Frigate config file, but without Hardware acceleration it may bring everything to a grinding halt.

Using go2rtc - Viewing High Quality Streams

Wait, you setup the high quality camera stream in your Frigate config, but when viewing it from the web UI you're seeing a low resolution stream!?

While this setup is optional, it will allow for viewing higher quality streams in the Frigate UI and also reduce the number of direct concurrent connections to your cameras. This also allows for more connection types to cameras than just RTSP. You can see the full list of supported connection types here.

Open your `frigate.yaml` file again for some editing. We are going to add a new main section titled `go2rtc:` and also add the cameras to this section, then we will update the `cameras:` section to point to this stream instead of directly to the camera.

Expand to see frigate.yaml

Add the following section right above the existing `cameras:` section. If using rtsp, you can copy the rtsp connection string from the cameras section to this section.

```
go2rtc:
  streams:
    C210: # <-- Name the camera, use the same name as in the cameras section
      - rtsp://username:password@192.168.0.101:554/stream1 # <-- stream which supports video & aac audio
      - "ffmpeg:C210#audio=opus" # <-- Optional, This creates a copy of the stream which transcodes audio to opus for webrtc support
    C210_sub: # <-- Optional low-quality substream
      - rtsp://username:password@192.168.0.101:554/stream2 # <-- Low Quality stream which supports video & aac audio
      - "ffmpeg:C210_sub#audio=opus" # <-- Optional if auto is needed for substream
```

If you have issues with the rtsp connection string, you can try adding ffmpeg: to the start of it. This supports more encoding types, but does cause the camera connection to be slower. Example: `ffmpeg:rtsp://username:password@192.168.0.101:554/stream1`

Save the changes to this file and restart Frigate to test this stream and see if it works.

Below is an example of quality difference when viewed in the Frigate UI before and after enabling go2rtc. This is just one corner of the camera view and there is quality lost from converting this to a gif for display here, but it still makes the quality difference clear.



You can now change the `path:` element under the `cameras:` section to point to the go2rtc stream instead of directly to the camera. This will again reduce the number of connections made directly to the camera. No credentials need to be specified here. Example: `path: rtsp://127.0.0.1:8554/C210`

More go2rtc docs: https://docs.frigate.video/guides/configuring_go2rtc/

Notifications & MQTT Integration

Now that Frigate is installed and working, you'll need to install the Frigate HACS integration to integrate it with the rest of Home Assistant. If you have not yet enabled HACS, see here first. This also assumes you have already installed the MQTT Add-On and Integration. If not, see here.

- From HACS, search **Frigate**
- Download **Frigate** and **Frigate Card**. Then restart Home Assistant
  - 

- In HA, go to **Settings** > **Devices & Services** > **Add Integration** > **Frigate**
- Leave the default URL in place (`http://ccab4aaf-frigate:5000`) and click **Submit**
  - The default URL works if you've install Frigate as an Add-On in Home Assistant. If you have installed Frigate elsewhere, you need to enter the URL/IP to access it here.
- Your configured camera(s) should pop up so you can add them to an Area.

- Go to **Settings** > **Add-ons** > **Mosquitto broker** > **Configuration**
- If you haven't setup a user here yet, add one using this format in the **Logins** field
  - Setting whatever username and password you want. This user is only for applications to access MQTT and Frigate requires a user to exist, it cannot use MQTT unauthenticated like some other services can.
  - ```
    - username: mqtt-user
      password: mqtt-pwd
    ```
  - Click **Save** to save your new MQTT user
- Open your `frigate.yaml` file again for more editing. We are going to edit the section titled `mqtt:` so that Frigate can access MQTT. If the mqtt section is missing you can add it.

Expand to see frigate.yaml

The mqtt section is usually at the very top of the frigate.yml file and is disabled by default. Enable it as shown here and add your Home Assistant server's local IP address as well as the user and password you just setup in the MQTT Mosquitto broker config.

```
mqtt:
  enabled: true
  host: 192.168.1.100 # <-- Local IP of your HA server. Does not work with localhost/
  port: 1883
  user: mqtt-user
  password: mqtt-pwd
```

- Save the changes to this file and restart the Frigate Add-on
- Go to **Settings** > **Devices & Services** > **Frigate**, and select one of your cameras.
  - You should now see a bunch of Sensors available.
    - If MQTT is not working, these sensors will show unavailable. If that's the case, check the Frigate logs (Settings > Add-Ons > Frigate > Log) and search for anything referencing `mqtt`. Also make sure Frigate was restarted after the latest changes. Make sure the host value you entered in frigate.yml is correct and is the IP address and not something like `localhost` or `homeassistant.local` as those won't work.
- Now you need to import this blueprint which will allow you to finally setup notifications:
  - Import blueprint
- To setup a notification rule after importing this blueprint: Go to **Settings** > **Automations & Scenes** > **Create Automation** > **Frigate Notifications**
- As a basic notification example, Select a camera and a device to be notified.
  - If not devices show up, make sure you've installed the Home Assistant phone app and logged in with it
  - If your device shows up twice, it's probably the last one listed, but you may need to try and come back and change it if it doesn't work.
```

- For **Base URL**, you can start by leaving it blank. A value of `homeassistant://` will open links in the app but not work for TVs. Setting your public home assistant URL will open links in a browser, but will work for TVs.
- No other fields are required. You can come back later to customize your notifications further or setup additional notifications for other devices/groups.
- Click **Save** and name your notification. It is now active. You can find it in the **Settings** > **Automations & scenes** page.
- Try walking in front or your camera and see if it triggers.
  - If it does not trigger there are a few things to check:
    - Check frigate.yml and make sure you have the `detect:` element under your camera enabled.
    - Check the Frigate Review page to see if it's detecting events. If it's not, it's a camera config issue.
    - Open your camera in Frigate and make sure Detection is enabled there. This option seems to work separately from the frigate.yml setting, and both need to be enabled.
    - Make sure the correct device is selected in the Automation you just setup

Motion Tracking

Coming Soon, detailed instructions on setting up motion tracking and other features

Cameras Tested

Camera Brand	Model	Features	Substream	Works?	Notes
Tapo	C210	2k, Pan & Tilt	720p	✓	Internet Required for Setup P&T work in Frigate w/ manual controls but not automatic tracking.
Tapo	C110 / C120	2k, Cheap	720p	✓	Internet Required for Setup. C120 has better performance and clarity. I recommend it.

Camera Brand	Model	Features	Substream	Works?	Notes
Reolink		Cheap	360P	✓	Substream quality lower than ideal for object detection. Some models may have higher substream resolution. RTSP sometimes unstable Untested, but supports RTMP and may be more reliable
Eufy			X	X	No longer supports RTSP. Eufy pulled RTSP from cameras previously supporting it via a firmware update. Some models may still support RTSP, but it's a gamble.
Wyze		Cheap	360P	X	No RTSP Support. Hacky workaround available but it's not super reliable

What is a substream?

A sub-stream is a secondary camera stream at a lower resolution. Without a substream, object detection needs to happen on the main stream, which can take a lot of resources of it is high quality. However, if the substream resolution is too low then it's not very useful. Usually 720p is a good middle ground. If you have the resources to perform detection on the full resolution stream, go for it!

Common Issues

Devices Missing in Settings > Devices & Services > Frigate

The Frigate Devices don't update automatically until Home Assistant has been restarted. The most reliable way to get this updated is:

1. Restart Frigate
2. Restart Home Assistant

If you don't restart just Frigate first, then it won't pick up the Frigate config changes in time to update the Device list.

Changing Camera Names

Changing camera names in Frigate is a bit difficult as you need to update all the references manually. Note, the connection to the Frigate Camera Devices in Home Assistant will break and new devices will be created. These are the steps you need to take:

1. Update ALL references to the camera name (and any sub-stream names) in your frigate.yml file
2. Restart Frigate and wait for everything to load back up
3. Verify in the Frigate UI that all cameras are loading and the settings show the new camera names
4. Restart Home Assistant
    1. Yes, do this
5. After Home Assistant is back up, go to: Settings > Devices & Services > Frigate
    1. Verify that all the new names are displaying and the old ones are gone
6. Go to Settings > Automations & scenes
7. Manually update (and Save) every automation or scene that references your cameras so reference the new name

ESP Device Flashing Issues

If you are having difficulties flashing your ESP based device following any of the many online tutorials (I used this one), try these common solutions.

ESP Device Not Entering Flash Mode

First, don't expect any lights to come on when your device enters flash mode. So no lights may be a good thing.

Verify the device has power by powering it from your USB Serial device without holding down power first. It should light up if it gets power.

When booting into flash mode:

1. **Do not** have your ESP device plugged into the wall!
2. Start with the USB serial device unplugged
3. Ensure all connections are correct and you're using 3 or 3.3 V power (not 5V)
4. Press and hold the ESP device power button
5. Plug in the USB still holding the ESP device power button
6. Hold for up to 12 seconds (but usually 3 seconds is enough)
7. Attempt to connect and flash your device

USB Device Not Showing Up - Linux

In linux the USB to serial driver should be installed by default, no install needed. However if your kernel has been updated without a reboot it can cause this driver to not work.

You might see error messages like this: `failed to validate module [usbserial] BTF: -22`

The solution to this one is simple, just reboot.

Failed to open serial port - Linux

If you receive this error message while flashing: `Failed to execute 'open' on 'SerialPort': Failed to open serial port.`

The `brltty` config may be stealing the USB connection. BRLTTY is for supporting braille displays for the blind. If you do not use brial displays you can safely disable this config permanently. To do so, we link the config file to /dev/null like so:

```
sudo ln -sf /dev/null /etc/udev/rules.d/85-brltty.rules
```

You then need to restart.

Switch TX and RX

Your USB serial and ESP device should generally have TX and RX crossed. For example the USB serial devices Transmits and the ESP device receives. Some ESP devices have their TX and RX reversed, so try matching TX to TX and RX to RX to see if that solves your problem

Tesla Fleet Setup

Instructions for setting up the Tesla Fleet integration in Home Assistant. This is an advanced setup process (for now).

1. Sign in with your Tesla Account credentials and create a new app. Give it a unique name to ensure no one else has used the same name:
    1. https://developer.tesla.com/dashboard
2. In the Tesla app setting, set the **Redirect URL** to: `https://my.home-assistant.io/redirect/oauth`
3. For the **Allowed Origin(s)** setting you need a public domain where you can host a file. Hopefully soon we can simplify this step.
    1. *Update:* You can use https://fleetkey.cc/ if you don't have your own domain.
4. When asked, select all available scopes.
5. Generate a Public/Private Key Pair. See below:
    1. Linux Instructions, run these commands:
        1. `openssl ecparam -name prime256v1 -genkey -noout -out private-key.pem`
        2. `openssl ec -in private-key.pem -pubout -out public-key.pem`
    2. You now have two files: **private-key.pem** and **public-key.pem** which you'll need later
    3. You must place the public key on the Allowed Origins domain you specified at the following location:
        1. *https://your-domain.com/***.well-known/appspecific/com.tesla.3p.public-key.pem**
6. Use Postman or a similar app, we need to make some API calls to Tesla now.
    1. Here is a Postman collection you can import: Tesla Fleet.postman_collection.json
7. For the **Tesla Auth Token** call, fill in your **client_id** and **client_secret** from your Tesla Developer account, then click **Send** in postman.
    1. This assumes Tesla North America or Asia-Pacific. If you're in a different region like Europe, find your URL here and update the **audience** field:

       https://developer.tesla.com/docs/fleet-api/getting-started/base-urls
8. You'll get back a response with an **access_token** in quotes. Copy the value in the quotes.
9. Now open the **Tesla Partner Account** in postman. Go to the **Authorization** tab and enter it into the **Token** field.
10. Click the **Body** tab and in the quotes next to **domain** enter the same domain you entered in **Allowed Origin(s)** when you setup your app with Tesla (excluding the https:// portion).
    1. Remember, you must place the public key you created on the Allowed Origins domain you specified at the following location:
        1. *https://your-domain.com/***.well-known/appspecific/com.tesla.3p.public-key.pem**
11. **Send** the request in postman. If your file was in the correct place, you should get back a response. Otherwise you'll receive an error message with what went wrong.

12. Create new Application Credentials in your HA: https://my.home-assistant.io/redirect/application_credentials/
    1. Select **Tesla Fleet** as the Integration and Enter your app name, client_id, client_secret, then click **Add**
13. Reconnect the device in HA **Settings > Devices & services > Tesla Fleet**

Notes:

- Questions/Comments? Reach out on Mastodon: **@JaggedJax@hostux.social**
- This guide does not yet cover Command Signing which may be required by newer vehicles to perform actions. It is not required just to read stats: https://www.home-assistant.io/integrations/tesla_fleet#command-signing
- It doesn't appear that the public key or Allowed Origin URL is actually validated, beyond one just existing.