A practical guide to home automation using open source tools

ABOUT OPENSOURCE.COM

What is Opensource.com?

OPENSOURCE.COM publishes stories about creating, adopting, and sharing open source solutions. Visit Opensource.com to learn more about how the open source way is improving technologies, education, business, government, health, law, entertainment, humanitarian efforts, and more.

Submit a story idea: opensource.com/story

Email us: open@opensource.com

Supported by Red Hat

STEVE OVENS

STEVE OVENS is a dedicated IT professional and Linux advocate. Prior to joining Red Hat, he spent several years in financial, automotive, and movie industries. Steve currently works for Red Hat as an OpenShift consultant and has certifications ranging from the RHCA (in DevOps), to Ansible, to Containerized Applications and more. He spends a lot of time discussing technology and writing tutorials on various technical subjects with friends, family, and anyone who is interested in listening.

Follow me at @linuxovens

CHAPTERS

Why I use Home Assistant for open source home automation

Home automation can be a slippery slope. The right open source tools can get you on firmer footing.

HOME AUTOMATION is a slippery slope; you have been warned! In this multipart series, I will discuss home automation using the open source project Home Assistant. This introductory article will cover my journey to Home Assistant [1], what the application does, and why it's important.

How my journey began

Some time ago, when I set out on this journey, my goal was not lofty. I was solving a need. You see, I have a fairly sizable homelab [2]. Nothing on the scale of some notable YouTubers, but I have eight machines ranging from 16GB RAM all the way up to 96GB. I have a Netgear 10G Ethernet switch as the backbone of my networking infrastructure. However, I have a small problem. Every once in a while, this switch's state table fills up, and then it crashes, taking the network with it. This is a known issue with this model (although it was not known to me ahead of time). The only way to resolve the issue, without replacing the switch, is to power it off for a few seconds and then power it back on.

This would not seem like a big deal. Especially because I live in an apartment, so I don't even have to travel up or down a flight of stairs to do this. However, I work for Red Hat as a senior OpenShift consultant, which means I am often on the road (at least pre-COVID). I use my lab almost every day for work-related activities, and my family uses the network as well for things like playing games, watching our Blu-ray collection, etc. So, it's a giant pain to have the network go down when I don't have physical access. My solution was to get a smart plug and join it to a completely different network with nothing else on it, completely firewalled off from other equipment. If the Netgear switch needs to be rebooted, I should be able to access the smart plug remotely to reboot the switch.

My journey away from the cloud

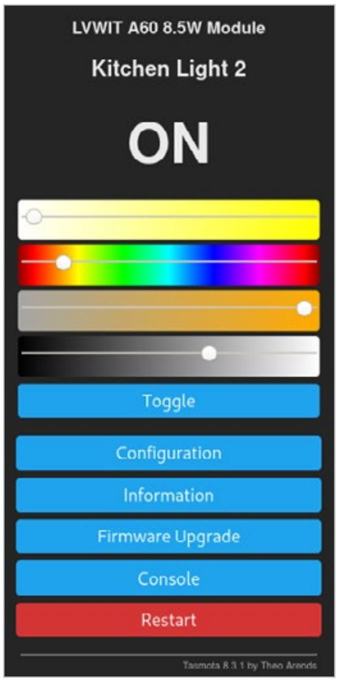
Being privacy-conscious, perhaps bordering on the tinfoil hat breed, I am immediately uncomfortable with a "cloud"-connected anything. We run Plex [3], Kodi [4], Nextcloud [5], and a host of other services because, well, I am "that guy." But I'm not "anti-cloud." In fact, a large part of my day job is working with the big three: AWS, Google Compute, and Azure. But when it comes to standing up services that I rely on, I have an almost irrational need to host things locally.

After doing a significant amount of digging and speaking to my coworker Alex Kretzschmar (who also hosts the Self-Hosted [6] podcast), I discovered alternative firmware projects, such as Tasmota [7] and ESPHome [8], available for certain wireless chipsets. I'll cover the different chipsets and protocols (Zigbee, WiFi, Z-Wave, and the like) in a future article, but suffice it to say, you are not stuck buying products that are dependent on the cloud. There are online stores such as CloudFree [9] that sell devices pre-flashed with Tasmota. There are even companies like Shelly [10] that produce high-quality products with an optional cloud component, but the buyer maintains local control.

Anyway, back to my story. I bought a plug, flashed the firmware to run Tasmota, and remotely power-cycled my switch on occasion, and that was the end of it, right? Well, if it was, I imagine I would not be writing this article. I currently have 43 Internet of Things (IoT) devices—from sensors I built to smart bulbs, smart LED strips, infrared blasters, and more. Remember how I said home automation is a slippery slope?

Did I mention that I have had virtually no experience with soldering, electronic theory/repairs, or anything remotely related to home automation? In fact, outside of Linux and related technology, I am one of the least "handy" people I know. Sure, I can punch a hole in a wall or put some screws in a board, but up to the age of 30, the only tools that I ever owned were a Dremel and some screwdrivers for installing computer components.

Why mention this? So you understand exactly how far behind square one I was when I started. If I can take this on, so can you!



(Steve Ovens, CC BY-SA 4.0)

Centralized local control

Great, I have a ton of IoT devices, and I can toggle things on and off from a phone. For a while, I was satisfied with simply calling up the Tasmota web UI and using its sparse but functional controls.

Even my wife, God bless her willingness and patience while I constantly learned and attempted new things, got used to using the Tasmota interface. After a while, as the number of devices grew, this method of managing things became untenable and did not scale well at all.

I began to poke around to learn how other people were managing their "smart homes." Let's be honest; at this point, it wasn't a smart home as much as a small but growing number of lights we could remotely control.

The big open source projects in this area are openHAB [11] and Home Assistant [12]. This series won't compare them in any manner. Partly, it's because I am not qualified for this, as I have only been using Home Assistant for the last year or so, but also because for me to feel comfortable making such a comparison, I would have to set up openHAB as a drop-in replacement and, frankly, I have too much invested in Home Assistant to explore this avenue.

About Home Assistant

You may be wondering what Home Assistant is and what it has to offer. Imagine a house that reacts to its inhabitants, like in sci-fi movies. Perhaps you have a couple of bedside lamps, a TV, some floor lighting, a fan, and a few other gizmos in your bedroom. As you walk into the bedroom, the lights turn on. Big deal, you say; we have had this technology for some time. Of course, you are right. But what if it's late and your significant other is already in bed? You might be in for a rather cold greeting if the lights come on as you enter the bedroom. Yikes!

What if, instead, when you walk into the room at night, the floor LED lights turn on to a soft glow—enough so that you can move around without killing yourself but not bright enough to wake a sleeping partner. You climb into bed, and a few minutes later, the floor lighting turns off. This is what Home Assistant can do for you.

"But wait!" I hear you screaming at your monitor. "I can just ask Alexa or Google to turn off my lights for me." You're correct again, of course. However, what happens when these services suffer some sort of disruption? Also, you may need to consider how loudly you need to speak to trigger the listening device. And remember that thing I said about privacy? Do you really want these big companies to learn your habits? Even discarding the privacy issue, consider that others may use your home as well. This means visitors need to know what commands you have available for controlling your smart devices.

Lots of "smart" products on the market can set timers, schedules, and scenes, but none of them can really react to any given situation. This is where a home automation hub like Home Assistant comes into play.

At its core, Home Assistant is software that helps centralize all of the sensors, gizmos, and gadgets you have in your home. With all of these products working together in concert, you can set all kinds of conditions that would not otherwise be possible.

Has someone entered the room? Is it a certain time of the day? Is the room warmer than a certain value? Is someone in bed? All of these data points are of limited use on their own, but together, you can use them to set the brightness of a light at dusk, turn a fan on because it's above 26ºC (80ºF) in the room, but don't turn on Steve's desk lamp because his side of the bed is occupied.

If you can imagine it, you can accomplish it using Home Assistant and the right sensor inputs.

Sustainable and open source

One thing I like about Home Assistant is that it seems to be reaching a critical mass of adoption from various vendors. But more important to me is that the company behind it, Nabu Casa [13], creates a focal point for the community. The company hired developers directly from the community, and it does not penalize you if you don't pay a monthly fee. The revenue Nabu Casa generates comes from value-added items, like being able to access your local Home Assistant easily and from anywhere in the world.

This is important because it means that there is a path forward for the project, and it has a sustainable income for future development. Unlike "open core" models, Home Assistant is fully open source, which means that if the community disagrees strongly enough with Nabu Casa's direction, it could fork Home Assistant at its current state and take a divergent path. Because my family has chosen to make this piece of software integral to how we interact with our devices, that Home Assistant has a revenue stream and is fully open source are invaluable to me.

In the next article, I will discuss some of the common standards for IoT devices as well as the benefits of local control. Future articles will walk through the basics of installing Home Assistant, setting up essential add-ons (e.g., MQTT), Node-RED, the Community Store, making backups, and more.

In the meantime, comment or tweet [14] at me if you would like to explore something specific in-depth. If I have used it, I can talk about it.

Links

[1] https://www.home-assistant.io/
[2] https://opensource.com/article/19/3/home-lab
[3] https://www.plex.tv/
[4] https://opensource.com/article/19/1/manage-your-media-kodi
[5] https://nextcloud.com/
[6] https://selfhosted.show/
[7] https://tasmota.github.io/docs/
[8] https://esphome.io/
[9] https://cloudfree.shop/
[10] https://shelly.cloud/
[11] https://www.openhab.org/
[12] https://opensource.com/article/18/3/smart-home-assistant
[13] https://www.nabucasa.com/
[14] https://twitter.com/linuxovens

Cloud control vs local control: What to choose for your home automation

Cloud may be more convenient, but local control gives you more privacy and other options in your Home Assistant ecosystem.

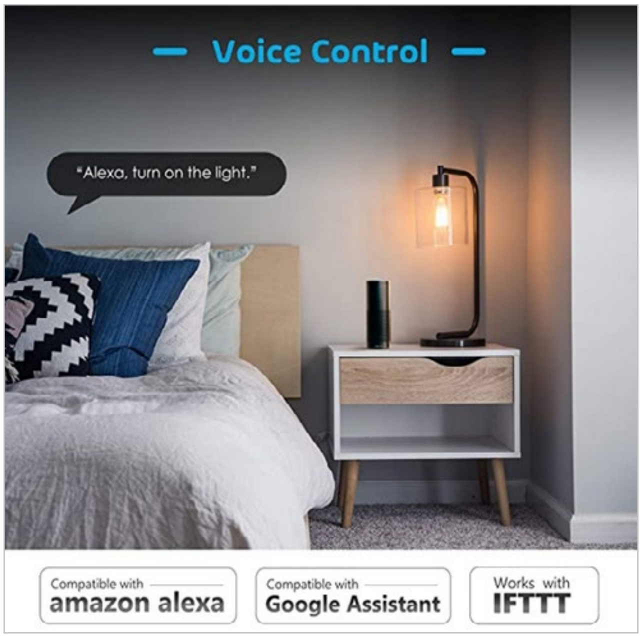
THERE ARE A LOT of factors to consider when investing in a home automation ecosystem. In my first article in this series, I explained why I picked Home Assistant [1], and in this article, I'll explain some of the foundational issues and technologies in home automation, which may influence how you approach and configure your Internet of Things (IoT) devices.

Cloud connectivity

Most devices you can buy today are tied to some type of cloud service. While the cloud brings a certain level of convenience, it also opens a host of problems. For starters, there are privacy issues related to a company having access to your personal habits—when you are home, what shows you watch, what time you go to bed, etc. Although most people are not as concerned about these issues as I am, privacy should still be a consideration, even if it is a small one.

Cloud access also creates issues around being reliant on something outside your control. In 2019, Sonos came under fire for remotely bricking [2] older smart speakers. Speakers usually continue to work for years after their warranty ends; in fact, they usually function until they physically break. There's also the case of Automatic, which produced a cloud-based car tracker. When it announced in May 2020 that it would be shutting down [3] its services, it advised customers to "please discard your adapter by following standard electronic recycling procedures."

Being dependent on a third-party provider for critical functionality can come back to bite you. IFTTT [4], a popular service for programming events based on external conditions, recently altered its free plan's terms and conditions [5] to severely limit the number of events you can create—from an unlimited number to three. This is even though IFTTT charges device manufacturers for certification with its system, which allows products like Meross smart bulbs [6] to proudly display their compatibility with IFTTT.


Voice Control
"Alexa, turn on the light."
Compatible with amazon alexa
Compatible with Google Assistant
Works with IFTTT

(Amazon screenshot by Steve Ovens, CC BY-SA 4.0)

Some of these decisions are purely financial, but there are more than a few anecdotal cases where a company blocks a person's access to a device they purchased simply because they did not like what the user said [7] about them. How crazy is that?

Another consideration with cloud connectivity is a device's responsivity if its signals must travel from your home to a cloud server (which may be halfway around the world) and then back to the device. This can lead to a two-second (or more) delay on any action. For some people, this is not a deal-breaker. For others, that delay is unbearable.

Finally, what happens if there is an internet outage? While most modern home internet connections are quite reliable, they do happen. Some large [8], well-known cloud service providers [9] have experienced outages this year. Are you OK trading convenience for possibly having your automations break and losing control of your smart devices for periods of time?

Local control

There are several ways you can regain control over your smart devices. Commercially, you could try something like Hubitat [10], which is a proprietary platform that emphasizes local control. I have no experience with these devices, as I don't like to rely on an intermediary.

In my home, I standardized on WiFi (although I may branch out to Zigbee [11] in the future) and Home Assistant [12]. Using WiFi means I need to buy or make my devices based on their compatibility with alternative open source firmware, such as Tasmota [13] or ESPHome [14]. I admit that neither of these options is "plug-and-play friendly" unless you buy devices from sources like Shelly [15], which is very friendly to the community, or CloudFree [16], which has Tasmota installed by default.

(As a small aside, I have both flashed my own devices and purchased them from CloudFree. There are some savings with the DIY approach, but I buy pre-flashed devices for my father's house because this eliminates a lot of hassle.)

I won't go into more detail about alternative firmware, how to flash it, and so on. I simply want to introduce you to the idea that there are options for local control.

Achieving local control with MQTT

A local control device probably uses either a direct API [17] call, where Home Assistant talks directly to the device, or Message Queuing Telemetry Transport (MQTT) [18].

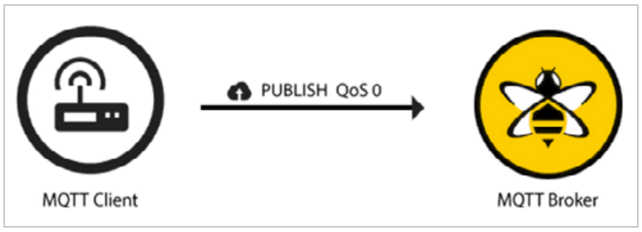
MQTT is one of the most widely used protocols for local IoT communication. I'll share some of the basics, but the Hook Up has an in-depth video [19] you can watch if you want to learn more, and HiveMQ has an entire series [20] on MQTT essentials.

MQTT uses three components for communication. The first, the **sender**, is the component that triggers the action. The second, the **broker**, is kind of like a bulletin board where messages are posted. The final component is the **device** that will perform the action. This process is called the *publish-subscribe* model.

Say you have a button on the wall that you want to use to turn on the projector, lower the blinds, and turn on a fan. The button (sender) posts the *message* **ON** to a specific section of the broker, called a *topic*. The topic might be something like `/livingroom/POWER`. The fan, the projector, and the blinds *subscribe* to this topic. When the `message` **ON** is posted to the topic, all of the devices activate their respective functions, turning on the projector, lowering the blinds, and starting the fan.

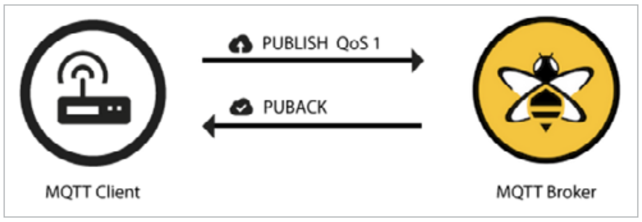
Unlike a message board, messages have different Quality of Service (QoS) states. The HiveMQ website has a good explanation of the three QoS levels [21]. In short:

- **QoS 0:** The message is sent to the broker in a fire-and-forget way. No attempt is made to verify that the broker received the message.



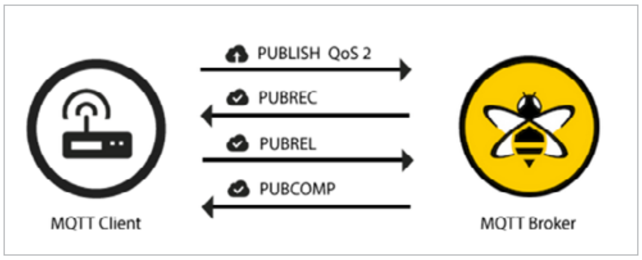
(© 2015 HiveMQ, reused with permission)

- **QoS 1:** The message is posted, and the broker replies once the message is received. Multiple messages can be sent before the broker replies. For example, if you are trying to raise the projector's brightness, multiple brightness bars may be inadvertantly adjusted before the broker tells the sender to stop publishing messages.



(© 2015 HiveMQ, reused with permission)

- **QoS 2:** This is the slowest but safest level. It guarantees that the message is received only once. Similar to TCP, if a message is lost, the sender will resend the message.



(© 2015 HiveMQ, reused with permission)

In addition, MQTT has a **retain** flag that can be enabled on the messages, but it is not set by default. Going back to the bulletin board analogy, it's like if someone posts a message to a bulletin board, but another person walks up to the board, takes the message down, reads it, and throws it away. If a third person looks at the bulletin board five minutes later, they would have no knowledge of the message. However, if the **retain** flag is set to true, it's like leaving the message pinned on the board until a new message is received. This means that no matter when people come to read messages, they will all know the latest message.

In home automation terms, whether or not the **retain** flag is set depends completely on the use case.

In this series, I will use Home Assistant's Mosquitto MQTT broker [22] add-on. Most of my devices use MQTT; however, I do have a couple of non-critical Tuya devices that require a cloud account. I may replace them with locally controllable ones in the future.

Wrapping up

Home Assistant is a large, wonderful piece of software. It is complex in some areas, and it will help you to be familiar with these fundamental technologies when you need to troubleshoot and coordinate your setup.

In the next article, I will talk about the "big three" wireless protocols that you are likely to encounter in smart devices: Zigbee, Z-Wave, and WiFi. Don't worry—I'm almost done with the underlying theories, and soon I'll get on with installing Home Assistant.

Links

[1] https://opensource.com/article/20/11/home-assistant
[2] https://www.bbc.com/news/technology-51768574
[3] https://www.cnet.com/roadshow/news/automatic-connected-car-service-dead-may-coronavirus/
[4] https://ifttt.com/
[5] https://ifttt.com/plans
[6] https://www.amazon.ca/meross-Dimmable-Equivalent-Compatible-Required/dp/B07WN2J3C7
[7] https://www.techrepublic.com/article/iot-company-bricks-customers-device-after-negative-review/
[8] https://www.theverge.com/2020/9/28/21492688/microsoft-outlook-office-teams-azure-outage-down
[9] https://www.cnn.com/2020/08/30/tech/internet-outage-cloudflare/index.html
[10] https://hubitat.com/
]11] https://zigbeealliance.org/
[12] https://www.home-assistant.io/
[13] https://tasmota.github.io/docs/
[14] https://esphome.io/
[15] https://shelly.cloud/
[16] https://cloudfree.shop/
[17] https://en.wikipedia.org/wiki/API
[18] https://en.wikipedia.org/wiki/MQTT
[19] https://www.youtube.com/watch?v=NjKK5ab0-Kk
[20] https://www.hivemq.com/tags/mqtt-essentials/
[21] https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/
[22] https://mosquitto.org/

How to choose a wireless protocol for home automation

Which of the three dominant wireless protocols used in home automation—WiFi, Z-Wave, and Zigbee—is right for you? Consider the options in part three of this series.

IN THE SECOND ARTICLE in this series, I talked about local control vs. cloud connectivity [1] and some things to consider for your home automation setup.

In this third article, I will discuss the underlying technology for connecting devices to Home Assistant [2], including the dominant protocols that smart devices use to communicate and some things to think about before purchasing smart devices.

Connecting devices to Home Assistant

Many different devices work with Home Assistant. Some connect through a cloud service, and others work by communicating with a central unit, such as a SmartThings Hub [3], that Home Assistant communicates with. And still others have a facility to communicate over your local network.

For a device to be truly useful, one of its key features must be wireless connectivity. There are currently three dominant wireless protocols that smart devices use: WiFi, Z-Wave, and Zigbee. I'll do a quick breakdown of each including their pros and cons.

A note about wireless spectra: Spectra are measured in hertz (Hz). A gigahertz (GHz) is 1 billion Hz. In general, the larger the number of Hz, the more data can be transmitted and the faster the connection. However, higher frequencies are more susceptible to interference and do not travel very well through solid objects. Lower frequencies can travel further and pass through solid objects more readily, but the trade-off is they cannot send much data.

WiFi

WiFi [4] is the most widely known of the three standards. These devices are the easiest to get up and running if you are starting from scratch. This is because almost everyone interested in home automation already has a WiFi router or an access point. In fact, in most countries in the western world, WiFi is considered almost on the same level as running water; if you go to a hotel, you expect a clean, temperature-controlled room with a WiFi password provided at check-in.

Therefore, Internet of Things (IoT) devices that use the WiFi protocol require no additional hardware to get started. Plug in the new device, launch a vendor-provided application or a web browser, enter your credentials, and you're done.

It's important to note that almost all moderate- to low-priced IoT devices use the 2.4GHz wireless spectrum. Why does this matter? Well, 2.4GHz has been around so long that virtually all devices—from cordless phones to smart bulbs—use this spectrum. In most countries, there are generally only about a dozen channels that off-the-shelf devices can broadcast and receive on. Like overloading a cell tower when too many users attempt to make phone calls during an emergency, channels can become overcrowded and susceptible to outside interference.

While well-behaving smart devices use little-to-no bandwidth, if they struggle to send/receive messages due to overcrowding on the spectrum, your automation will have mixed results. A WiFi access point can only communicate with one client at a time. That means the more devices you have on WiFi, the greater the chance that someone on the network will have to wait their turn to communicate.

Pros:
- Ubiquitous
- Tend to be inexpensive
- Easy to set up
- Easy to extend the range
- Uses existing network
- Requires no hub

Cons:
- Can suffer from interference from neighboring devices or adjacent networks
- Uses the most populated 2.4GHz spectrum
- Your router limits the number of devices
- Uses more power, which means less or no battery-powered devices
- Has the potential to impact latency-sensitive activities like gaming over WiFi
- Most off-the-shelf products require an internet connection

Z-Wave

Z-Wave [5] is a closed wireless protocol controlled and maintained by a company named Zensys. Because it is controlled by a single entity, all devices are guaranteed to work together. There is one standard and one implementation. This means that you never have to worry about which device you buy from which manufacturer; they will always work.

Z-Wave operates in the 0.9GHz spectrum, which means it has the largest range of the popular protocols. A central hub is required to coordinate all the devices on a Z-Wave ecosystem. Z-Wave operates on a mesh network [6] topology, which means that every device acts as a potential repeater for other devices. In theory, this allows a much greater coverage area. Z-Wave limits the number of "hops" to 4. That means that, in order for a signal to get from a device to a hub, it can only travel through four devices. This could be a positive or a negative, depending on your perspective.

On the one hand, it reduces the ecosystem's maximum latency by preventing packets from traveling through a significant number of devices before reaching the destination. The more devices a signal must go through, the longer it can take for devices to become responsive.

On the other hand, it means that you need to be more strategic about providing a good path from your network's extremities back to the hub. Remember, the lower frequency that enables greater distance also limits the speed and amount of data that can be transferred. This is currently not an issue, but no one knows what size messages future smart devices will want to send.

Pros:
- Z-Wave compatibility guaranteed
- Form mesh network
- Low powered and can be battery powered
- Mesh networks become more reliable with more devices
- Uses 0.9GHz and can transmit up to 100 meters
- Least likely of the three to have signal interference from solid objects or external sources

Cons:
- Closed protocol
- Costs the most
- Maximum of four hops in the mesh
- Can support up to 230 devices per network
- Uses 0.9GHz, which is the slowest of all protocols

Zigbee

Unlike Z-Wave, Zigbee [7] is an open standard. This can be a pro or a con, depending on your perspective. Because it is an open standard, manufacturers are free to alter the implementation to suit their products. To borrow an analogy from one of my favorite YouTube channels, The Hook Up [8], Zigbee is like going through a restaurant drive-through. Having the same standard means you will always be able to speak to the restaurant and they will be able to hear you. However, if you speak a different language than the drive-through employee, you won't be able to understand each other. Both of you can speak and hear each other, but the meaning will be lost.

Similarly, the Zigbee standard allows all devices on a Zigbee network to "hear" each other, but different implementations mean they may not "understand" each other. Fortunately, more often than not, your Zigbee devices should be able to interoperate. However, there is a non-trivial chance that your devices will not be able to understand each other. When this happens, you may end up with multiple networks that could interfere with each other.

Like Z-Wave, Zigbee employs a mesh network topology but has no limit to the number of "hops" devices can use to communicate with the hub. This, combined with some tweaks to the standard, means that Zigbee theoretically can support more than 65,000 devices on a single network.

Pros:
- Open standard
- Form mesh network
- Low-powered and can be battery powered
- Can support over 65,000 devices
- Can communicate faster than Z-Wave

Cons:
- No guaranteed compatibility
- Can form separate mesh networks that interfere with each other
- Uses the oversaturated 2.4GHz spectrum
- Transmits only 10 to 30 meters

Pick your protocol

Perhaps you already have some smart devices. Or maybe you are just starting to investigate your options. There is a lot to consider when you're buying devices. Rather than focusing on the lights, sensors, smart plugs, thermometers, and the like, it's perhaps more important to know which protocol (WiFi, Z-Wave, or Zigbee) you want to use.

Whew! I am finally done laying home automation groundwork. In the next article, I will show you how to start the initial installation and configuration of a Home Assistant virtual machine.

Links

[1] https://opensource.com/article/20/11/cloud-vs-local-home-automation
[2] https://opensource.com/article/20/11/home-assistant
[3] https://www.smartthings.com/
[4] https://en.wikipedia.org/wiki/Wi-Fi
[5] https://www.z-wave.com/
[6] https://en.wikipedia.org/wiki/Mesh_networking
[7] https://zigbeealliance.org/
[8] https://www.youtube.com/channel/UC2gyzKcHbYfqoXA5xbyGXtQ

Set up Home Assistant to manage your open source smart home

Learn how to install and configure Home Assistant in the fourth article in this series on home automation.

IN THE FIRST ARTICLE [1] in this series, I introduced Home Assistant [2] and why you might be interested in it. In short, Home Assistant is an automation hub for some of the most common smart devices on the market today. It enables centralized coordination of disparate hardware. By using it, you no longer have to choose suboptimal tech from a single vendor to manage your smart home from a single app. It also means you will no longer struggle with a hundred different apps that all function slightly differently to manage all your devices. One program to rule them all… or at least that's the dream.

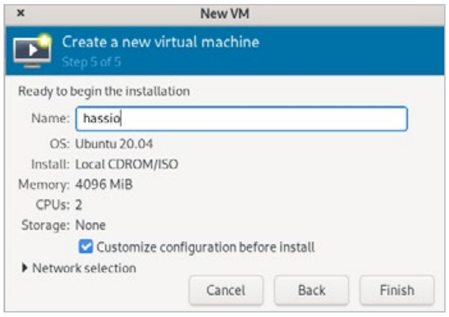
In the second and third articles, I looked at some of the decisions to make when developing home automation, namely local vs. cloud control [3], and whether to choose Zigbee, Z-Wave, or WiFi [4], just to hit the high points. This fourth article will be much more hands-on than the previous ones by walking you through setting up a virtual machine (VM) with the Home Assistant-provided image.

Set up the VM

I won't cover all of the methods available for installing Home Assistant (HA). I run HA in a virtualized environment, and the official installation page [5] provides VMDK, VHDX, VDI, QCOW2, and OVA downloads. I have a libvirt-based [6] homelab, so I chose the QCOW2 image.

Regardless of which hypervisor you use, you need to make sure that the boot type is set to UEFI instead of the traditional BIOS. In Arch Linux or Fedora, you need to install the package `edk2-ovmf` to have the option available in Virt-Manager [7]. In Ubuntu, the package is called ovmf. After the package is installed, restart libvirt.

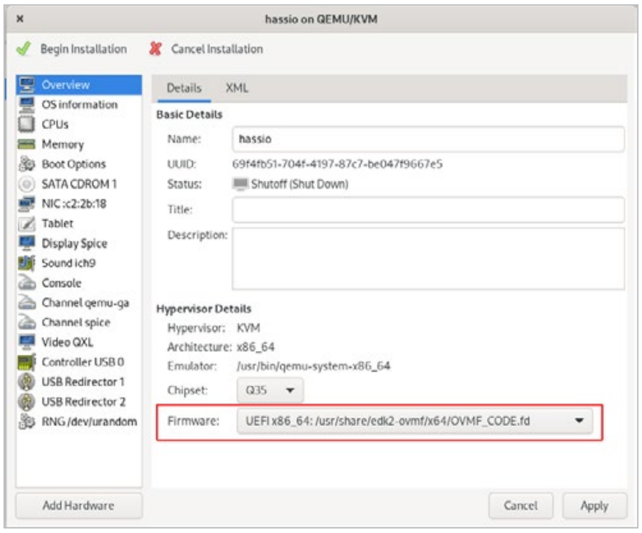
When you create your VM, select the HA image you downloaded. You can accept the default options that Virt-Manager selects *until* the *confirmation* screen. Make sure you check the box that says **Customize configuration before install**.



Ready to begin the installation
Name: hassio
OS: Ubuntu 20.04
Install: Local CDROM/ISO
Memory: 4096 MiB
CPUs: 2
Storage: None
[✓] Customize configuration before install
Network selection

(Steve Ovens, CC BY-SA 4.0)

When you do this, make sure to change the firmware to UEFI:



Basic Details
Name: hassio
UUID: 69f4fb51-704f-4197-87c7-be047f9667e5
Status: Shutoff (Shut Down)
Title:
Description:
Hypervisor Details
Hypervisor: KVM
Architecture: x86_64
Emulator: /usr/bin/qemu-system-x86_64
Chipset: Q35
Firmware: UEFI x86_64: /usr/share/edk2-ovmf/x64/OVMF_CODE.fd

(Steve Ovens, CC BY-SA 4.0)

Important note: You cannot change the firmware type after the VM has been created. If you choose BIOS, the VM will not boot!

If you need to expand the amount of disk available to HA's VM, shut down the VM and run:

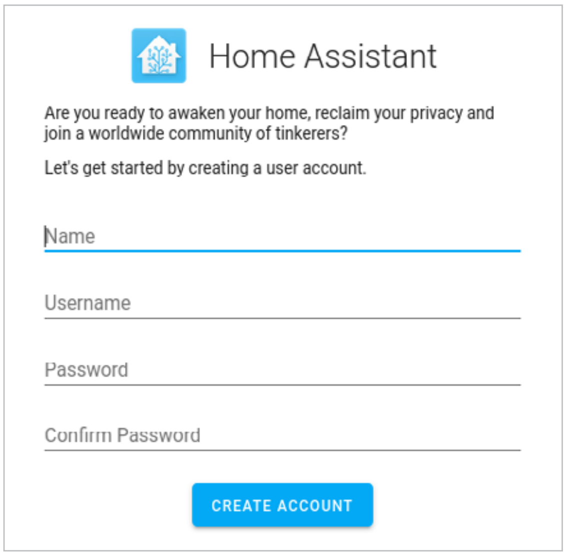
```
qemu-img resize hassos_ova-4.13.qcow2 +40G
```

Upon boot, the new space will be automatically allocated to the appropriate partitions.

The first boot can take a significant amount of time, as HA pulls down the latest versions of software from the internet and prepares them for initial launch and configuration. To be on the safe side, walk away for 10 minutes or so before attempting to pull up the webpage for the first time. In my experience, it often requires less than five minutes, but 10 minutes is a good amount of time for the system to spin up and settle.

First-time setup

You should now be able to access the HA interface by pointing a browser to `http://homeassistant.local:8123`. However, this relies on your router to support automatic DNS registration. You can also access the webpage via its IP. In my case, that's `http://192.168.122.90:8123`.



Home Assistant

Are you ready to awaken your home, reclaim your privacy and join a worldwide community of tinkerers?

Let's get started by creating a user account.

Name

Username

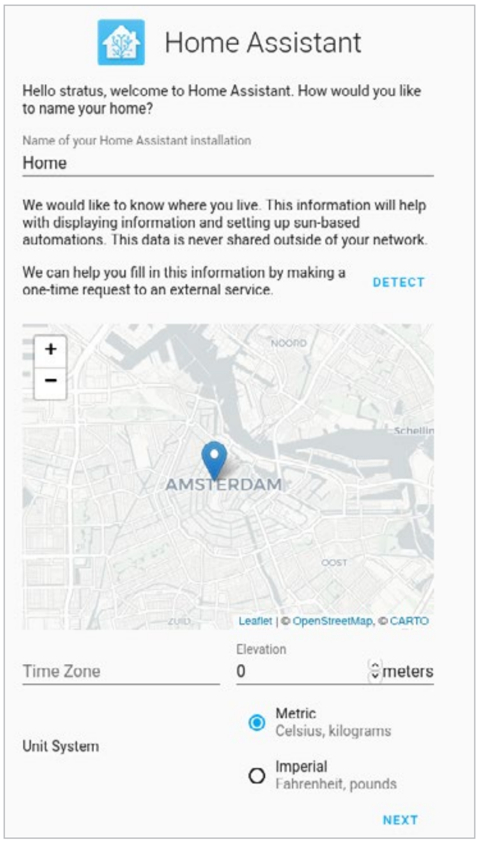
Password

Confirm Password

CREATE ACCOUNT

(Steve Ovens, CC BY-SA 4.0)

Enter your username and password for the administrative HA account. Then it will prompt you to select a location.



Home Assistant

Hello stratus, welcome to Home Assistant. How would you like to name your home?

Name of your Home Assistant installation

Home

We would like to know where you live. This information will help with displaying information and setting up sun-based automations. This data is never shared outside of your network.

We can help you fill in this information by making a one-time request to an external service. DETECT

Time Zone

Elevation 0 meters

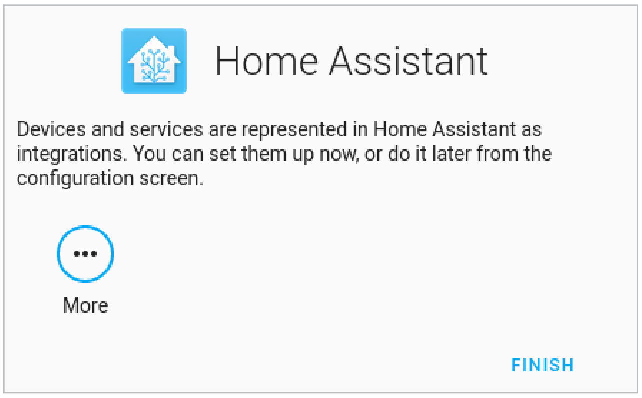
Unit System [✓ Metric Celsius, kilograms] Imperial Fahrenheit, pounds

NEXT

(Steve Ovens, CC BY-SA 4.0)

As you can see, your location is used for "sun-based automations." This means it uses your longitude and latitude to determine sunrise and sunset times and your time zone. If, for example, you have an automation that says, "turn on desk light 20 minutes before sunrise and turn off an hour after sunrise," HA uses this location information to determine what time to activate the lights. Unfortunately, you have to use the graphical map to set this information, and it may not function properly without an active internet connection.

After you have completed this, you will see a confirmation screen.



Home Assistant

Devices and services are represented in Home Assistant as integrations. You can set them up now, or do it later from the configuration screen.

More

FINISH

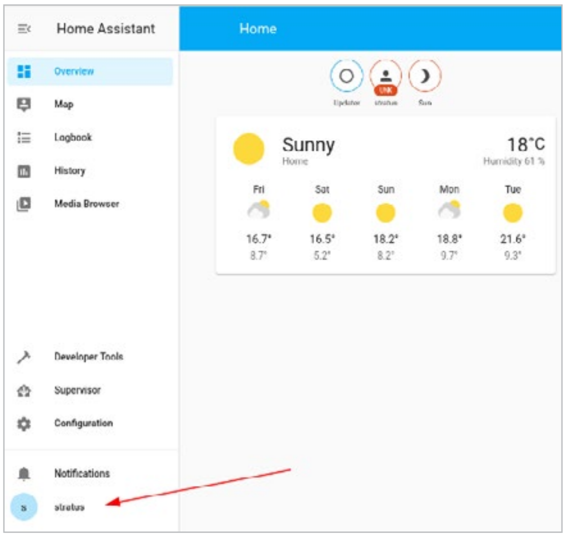
(Steve Ovens, CC BY-SA 4.0)

Rather than setting up devices here, I prefer to click **Finish** and use the full UI to configure my devices. This is optional, of course. This screen may prepopulate some integrations, depending upon whether HA has automatically discovered devices on your network.

Initial user settings

After you finish the initial configuration, you will see the Overview page. HA's default interface is called Lovelace [8]. It is a powerful YAML-described [9] interface. This means that even if you can't choose certain user interface (UI) elements in the graphic interface, you can open the built-in YAML editor and add them yourself.

Lovelace's default view has a single card that displays the weather based on the location you entered. Click your username (*stratus* in this example) in the bottom-left panel.

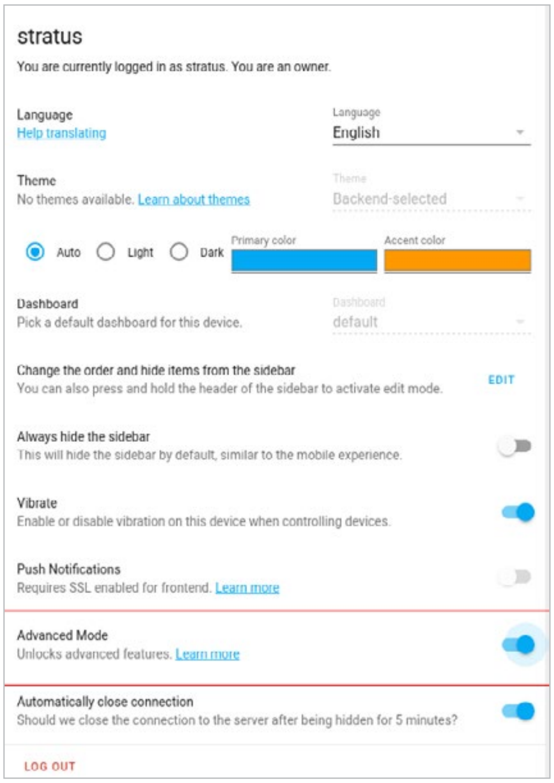


Sunny 18°C
Home Humidity 61 %

Fri	Sat	Sun	Mon	Tue
16.7°	16.5°	18.2°	18.8°	21.6°
8.7°	5.2°	8.2°	9.7°	9.3°

(Steve Ovens, CC BY-SA 4.0)

This brings up another screen with several options. Find **Advanced Mode** and make sure it is toggled on.


stratus

You are currently logged in as stratus. You are an owner.

Language
Help translating
Language: English

Theme
No themes available. Learn about themes
Theme: Backend-selected

Auto [✓] Light Dark
Primary color
Accent color

Dashboard
Pick a default dashboard for this device.
Dashboard: default

Change the order and hide items from the sidebar
You can also press and hold the header of the sidebar to activate edit mode.
EDIT

Always hide the sidebar
This will hide the sidebar by default, similar to the mobile experience.

Vibrate
Enable or disable vibration on this device when controlling devices.

Push Notifications
Requires SSL enabled for frontend. Learn more

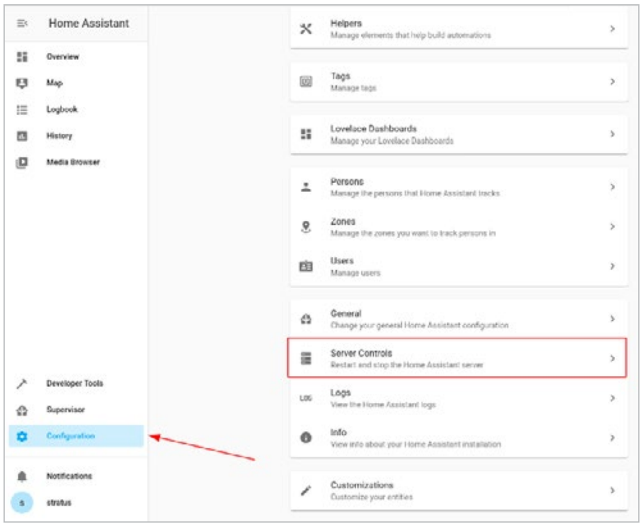
Advanced Mode
Unlocks advanced features. Learn more

Automatically close connection
Should we close the connection to the server after being hidden for 5 minutes?

LOG OUT

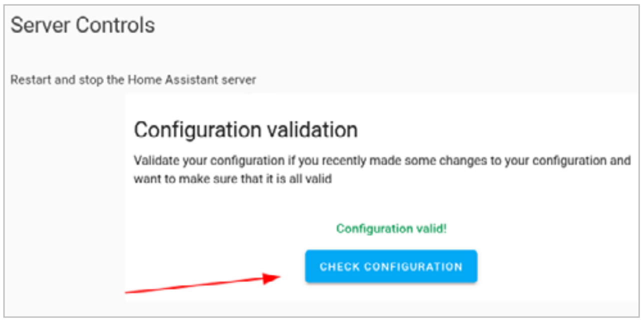
(Steve Ovens, CC BY-SA 4.0)

This setting is toggled per user, so if you have other administrative users, only this one (e.g., *stratus*) will have advanced settings turned on. There are quite a few options exposed when toggling **Advanced Mode**, but the one you want is the ability to run a syntax check against the HA configuration files. To see this in action, navigate to **Server Controls** by clicking on the **Configuration** option in the bottom-left panel, and then click on **Server Controls**.


Home Assistant
Overview
Map
Logbook
History
Media Browser
Developer Tools
Supervisor
Configuration
Notifications
stratus
Helpers
Manage elements that help build automations
Tags
Manage tags
Lovelace Dashboards
Manage your Lovelace Dashboards
Persons
Manage the persons that Home Assistant tracks
Zones
Manage the zones you want to track persons in
Users
Manage users
General
Change your general Home Assistant configuration
Server Controls
Restart and stop the Home Assistant server
Logs
View the Home Assistant logs
Info
View info about your Home Assistant installation
Customizations
Customize your entities

(Steve Ovens, CC BY-SA 4.0)

Clicking on the **Check Configuration** button will check all the HA configuration files for syntax errors. If it finds no errors, you will see a message in green that says **Configuration Valid**!


Server Controls

Restart and stop the Home Assistant server

Configuration validation

Validate your configuration if you recently made some changes to your configuration and want to make sure that it is all valid

Configuration valid!

CHECK CONFIGURATION

(Steve Ovens, CC BY-SA 4.0)

Looking ahead

Now that HA is set up and configured, you are ready to start really digging into it. In future articles, I will explain how to:

- Install and configure add-ons
- Create snapshots and run HA updates
- Install the Home Assistant Community Store (HACS)
- Configure entities via the built-in options
- Work with MQTT
- Create automation flows with NodeRed

And much more.

Links

[1] https://opensource.com/article/20/11/home-assistant
[2] https://www.home-assistant.io/
[3] https://opensource.com/article/20/11/cloud-vs-local-home-automation
[4] https://opensource.com/article/20/11/home-automation-part-3
[5] https://www.home-assistant.io/hassio/installation/
[6] https://libvirt.org/
[7] https://virt-manager.org/
[8] https://www.home-assistant.io/lovelace/
[9] https://en.wikipedia.org/wiki/YAML

Integrate devices and add-ons into your home automation setup

Learn how to set up initial integrations and install add-ons in Home Assistant in the fifth article in this series.

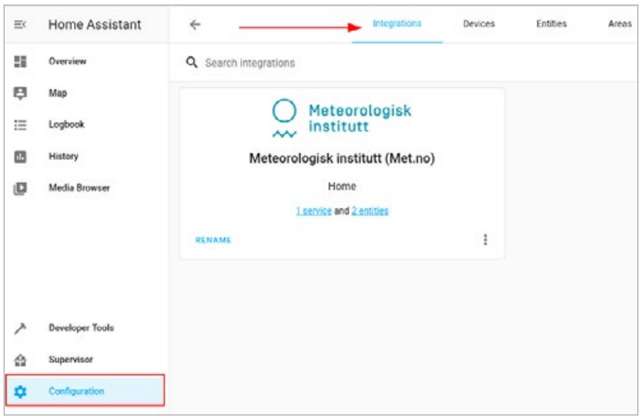
IN THE FOUR PREVIOUS ARTICLES in this series about home automation, I have discussed what Home Assistant is [1], why you may want local control [2], some of the communication protocols [3] for smart home components, and how to install Home Assistant [4] in a virtual machine (VM) using libvirt. In this fifth article, I will talk about configuring some initial integrations and installing some add-ons.

Set up initial integrations

It's time to start getting into some of the fun stuff. The whole reason Home Assistant (HA) exists is to pull together various "smart" devices from different manufacturers. To do so, you have to make Home Assistant aware of which devices it should coordinate. I'll demonstrate by adding a Sonoff Zigbee Bridge [5].

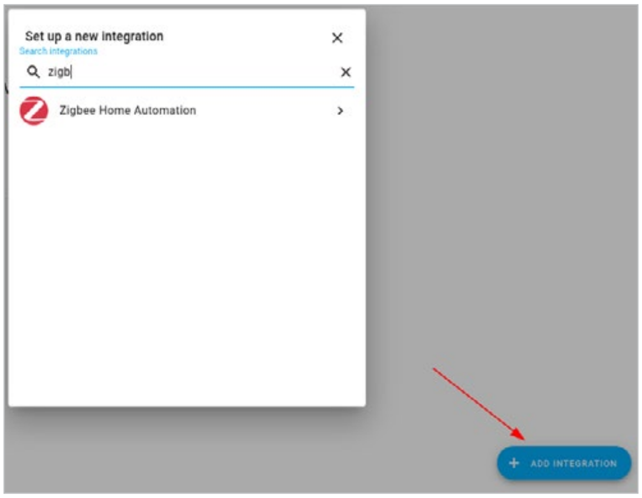
I followed DigiBlur's Sonoff Guide [6] to replace the stock firmware with the open source firmware Tasmota [7] to decouple my sensors from the cloud. My second article [8] in this series explains why you might wish to replace the stock firmware. (I won't go into the device's setup with either the stock or custom firmware, as that is outside of the scope of this tutorial.)

First, navigate to the **Configuration** menu on the left side of the HA interface, and make sure **Integrations** is selected:



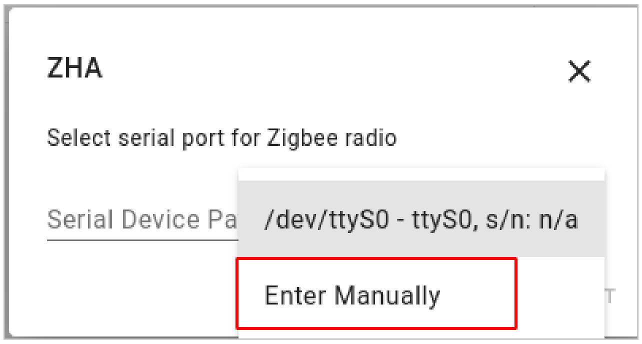
(Steve Ovens, CC BY-SA 4.0)

From there, click the **Add Integration** button in the bottom-right corner and search for Zigbee:



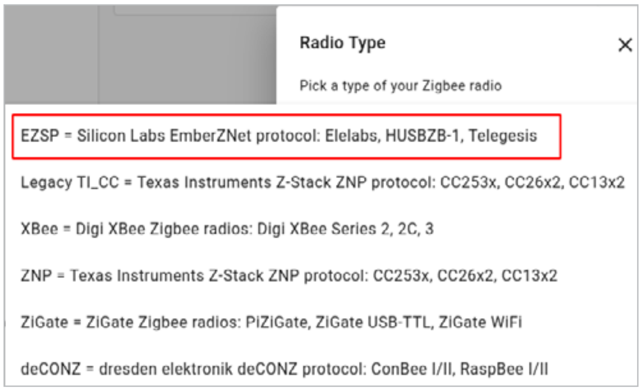
(Steve Ovens, CC BY-SA 4.0)

Enter the device manually. If the Zigbee Bridge was physically connected to the Home Assistant interface, you could select the device path. For instance, I have a ZigBee CC2531 USB stick that I use for some Zigbee devices that do not communicate correctly with the Sonoff Bridge. It attaches directly to the Home Assistant host and shows up as a Serial Device. See my third article [9] for details on wireless standards. However, in this tutorial, we will continue to configure and use the Sonoff Bridge.


ZHA
Select serial port for Zigbee radio
Serial Device Pa
/dev/ttyS0 - ttyS0, s/n: n/a
Enter Manually

(Steve Ovens, CC BY-SA 4.0)

The next step is to choose the radio type, using the information in the DigiBlur tutorial. In this case, the radio is an EZSP radio:


Radio Type

Pick a type of your Zigbee radio

EZSP = Silicon Labs EmberZNet protocol: Elelabs, HUSBZB-1, Telegesis

Legacy TI_CC = Texas Instruments Z-Stack ZNP protocol: CC253x, CC26x2, CC13x2

XBee = Digi XBee Zigbee radios: Digi XBee Series 2, 2C, 3

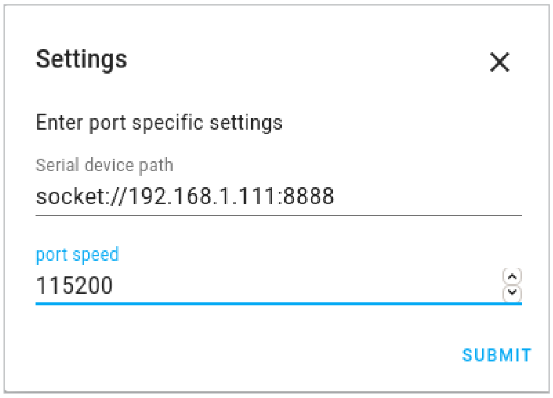
ZNP = Texas Instruments Z-Stack ZNP protocol: CC253x, CC26x2, CC13x2

ZiGate = ZiGate Zigbee radios: PiZiGate, ZiGate USB-TTL, ZiGate WiFi

deCONZ = dresden elektronik deCONZ protocol: ConBee I/II, RaspBee I/II

(Steve Ovens, CC BY-SA 4.0)

Finally, you need to know the IP address of the Sonoff Bridge, the port it is listening on, and the speed of the connection. Once I found the Sonoff Bridge's MAC address, I used my DHCP server to ensure that the device always uses the same IP on my network. DigiBlur's guide provides the port and speed numbers.


Settings

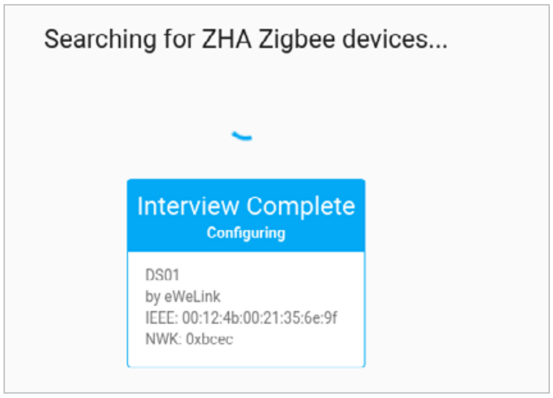
Enter port specific settings

Serial device path
socket://192.168.1.111:8888

port speed
115200

(Steve Ovens, CC BY-SA 4.0)

Once you've added the Bridge, you can begin pairing devices to it. Ensure that your devices are in pairing mode. The Bridge will eventually find your device(s).

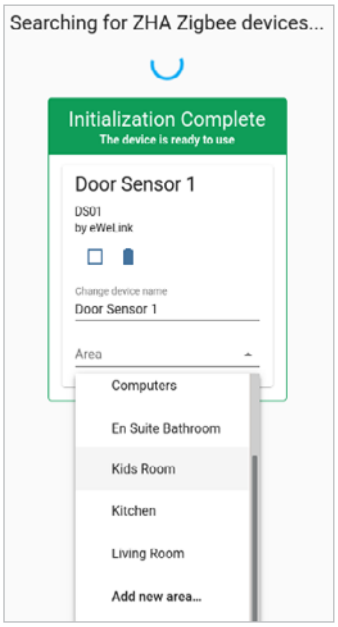

Searching for ZHA Zigbee devices...

Interview Complete
Configuring

DS01
by eWeLink
IEEE: 00:12:4b:00:21:35:6e:9f
NWK: 0xbcec

(Steve Ovens, CC BY-SA 4.0)

You can name the device(s) and assign an area (if you set them up).


Searching for ZHA Zigbee devices...

Initialization Complete
The device is ready to use

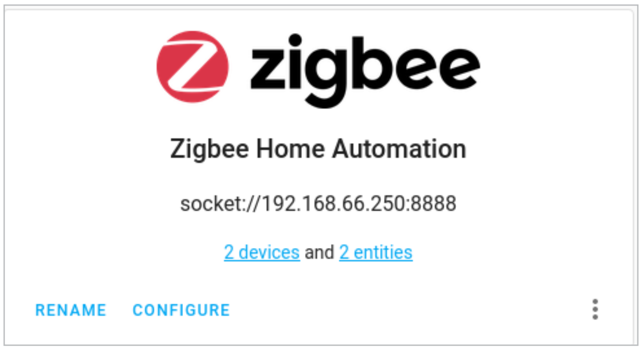
Door Sensor 1
DS01
by eWeLink

Change device name
Door Sensor 1

Area

(Steve Ovens, CC BY-SA 4.0)

The areas displayed will vary based on whether or not you have any configured. Bedroom, Kitchen, and Living Room exist by default. As you add a device, HA will add a new Card to the **Integrations** tab. A Card is a user interface (UI) element that groups information related to a specific entity. The Zigbee card looks like this:


Zigbee Home Automation

socket://192.168.66.250:8888

2 devices and 2 entities

(Steve Ovens, CC BY-SA 4.0)

Later, I'll come back to using this integration. I'll also get into how to use this device in automation flows. But now, I will show you how to add functionality to Home Assistant to make your life easier.

Add functionality with add-ons

Out of the box, HA has some pretty great features for home automation. If you are buying commercial-off-the-shelf (CoTS) products, there is a good chance you can accomplish everything you need without the help of add-ons. However, you may want to investigate some of the add-ons, especially if (like me) you want to make your own sensors.

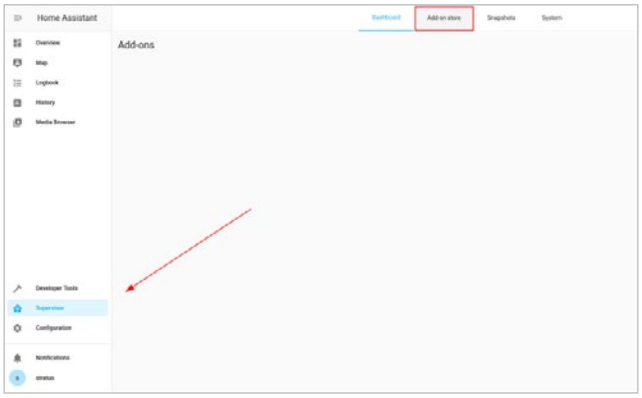
There are all kinds of HA add-ons, ranging from Android debugging (ADB) tools to MQTT brokers to the Visual Studio Code editor. With each release, the number of add-ons grows. Some people make HA the center of their local system, encompassing DHCP, Plex, databases, and other useful programs. In fact, HA now ships with a built-in media browser for playing any media that you expose to it.

I won't go too crazy in this article; I'll show you some of the basics and let you decide how you want to proceed.

Install official add-ons

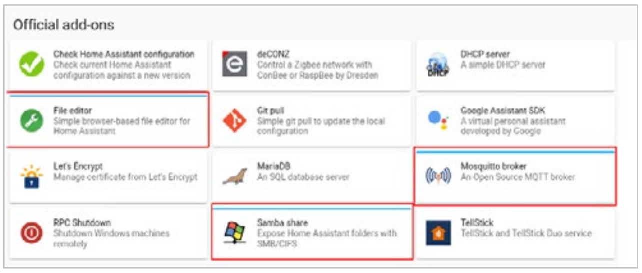
Some of the many HA add-ons are available for installation right from the web UI, and others can be installed from alternative sources, such as Git.

To see what's available, click on the **Supervisor** menu on the left panel. Near the top, you will see a tab called **Add-on store**.



(Steve Ovens, CC BY-SA 4.0)

Below are three of the more useful add-ons that I think should be standard for any HA deployment:


Official add-ons

Check Home Assistant configuration
Check current Home Assistant configuration against a new version

deCONZ
Control a Zigbee network with ConBee or RaspBee by Dresden

DHCP server
A simple DHCP server

File editor
Simple browser-based file editor for Home Assistant

Git pull
Simple git pull to update the local configuration

Google Assistant SDK
A virtual personal assistant developed by Google

Let's Encrypt
Manage certificate from Let's Encrypt

MariaDB
An SQL database server

Mosquitto broker
An Open Source MQTT broker

RPC Shutdown
Shutdown Windows machines remotely

Samba share
Expose Home Assistant folders with SMB/CIFS

TellStick
TellStick and TellStick Duo service

(Steve Ovens, CC BY-SA 4.0)

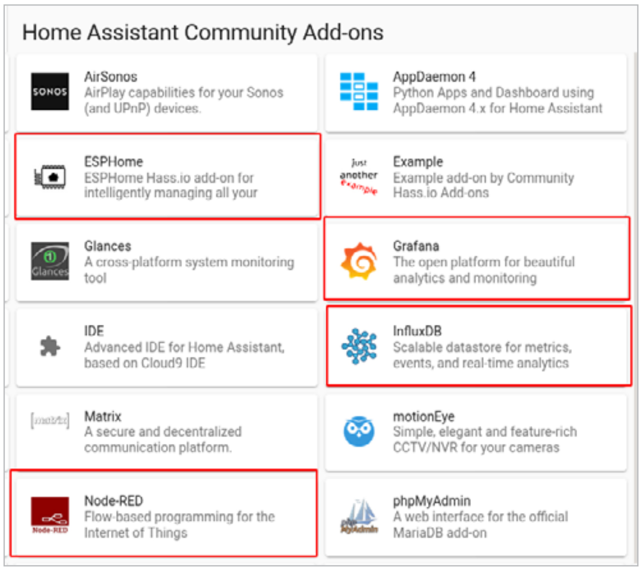
The **File Editor** allows you to manage Home Assistant configuration files directly from your browser. I find this far more convenient for quick edits than obtaining a copy of the file, editing it, and pushing it back to HA. If you use add-ons like the Visual Studio Code editor, you can edit the same files.

The **Samba share** add-on is an excellent way to extract HA backups from the system or push configuration files or assets to the **web** directory. You should never leave your backups sitting on the machine being backed up.

Finally, **Mosquitto broker** is my preferred method for managing an MQTT [10] client. While you can install a broker that's external to the HA machine, I find low value in doing this. Since I am using MQTT just to communicate with my IoT devices, and HA is the primary method of coordinating that communication, there is a low risk in having these components vertically integrated. If HA is offline, the MQTT broker is almost useless in my arrangement.

Install community add-ons

Home Assistant has a prolific community and passionate developers. In fact, many of the "community" add-ons are developed and maintained by the HA developers themselves. For my needs, I install:


Home Assistant Community Add-ons

AirSonos
AirPlay capabilities for your Sonos (and UPnP) devices.

AppDaemon 4
Python Apps and Dashboard using AppDaemon 4.x for Home Assistant

ESPHome
ESPHome Hass.io add-on for intelligently managing all your

Example
Example add-on by Community Hass.io Add-ons

Glances
A cross-platform system monitoring tool

Grafana
The open platform for beautiful analytics and monitoring

IDE
Advanced IDE for Home Assistant, based on Cloud9 IDE

InfluxDB
Scalable datastore for metrics, events, and real-time analytics

Matrix
A secure and decentralized communication platform.

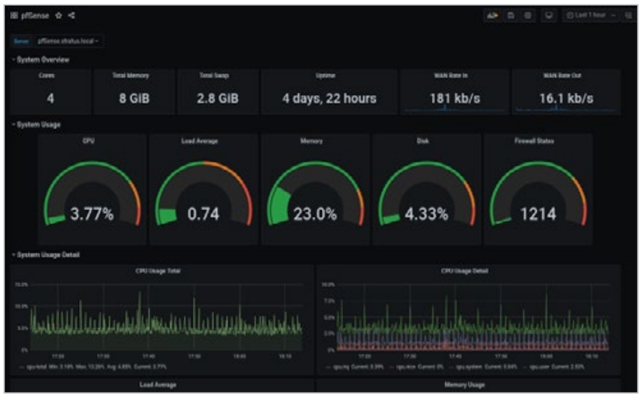
motionEye
Simple, elegant and feature-rich CCTV/NVR for your cameras

Node-RED
Flow-based programming for the Internet of Things

phpMyAdmin
A web interface for the official MariaDB add-on

(Steve Ovens, CC BY-SA 4.0)

Grafana (graphing program) and **InfluxDB** (a time-series database) are largely optional and relate to the ability to customize how you visualize the data HA collects. I like to have historical data handy and enjoy looking at the graphs from time to time. While not exactly HA-related, I have my pfSense firewall/router forward metrics to InfluxDB so that I can make some nice graphs over time.



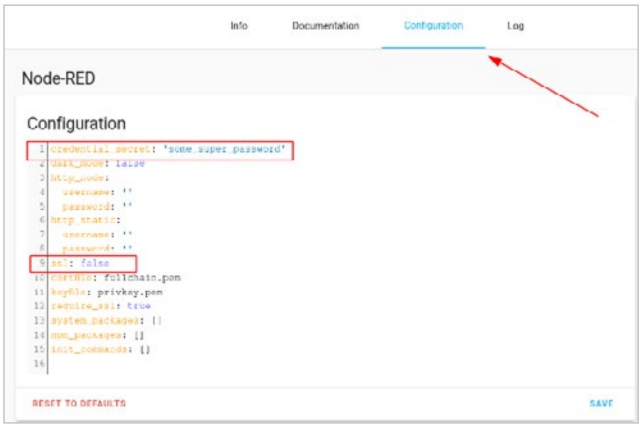
(Steve Ovens, CC BY-SA 4.0)

ESPHome is definitely an optional add-on that's warranted only if you plan on making your own sensors.

NodeRED is my preferred automation flow-handling solution. Although HA has some built-in automation, I find a visual flow editor is preferable for some of the logic I use in my system.

Configure add-ons

Some add-ons (such as File Editor) require no configuration to start them. However, most—such as Node-RED—require at least a small amount of configuration. Before you can start Node-RED, you will need to set a password:



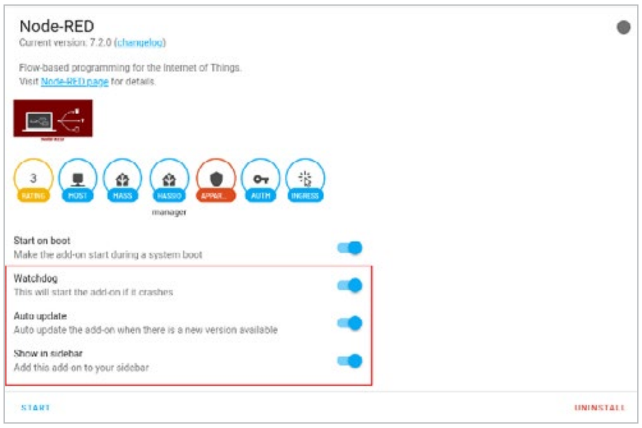
```
credential_secret: 'some_super_password'
dark_mode: false
http_node:
  username: ''
  password: ''
http_static:
  username: ''
  password: ''
ssl: false
certfile: fullchain.pem
keyfile: privkey.pem
require_ssl: true
system_packages: []
npm_packages: []
init_commands: []
```

(Steve Ovens, CC BY-SA 4.0)

IMPORTANT: Many people will abstract passwords through the secrets.yaml file. This does not provide any additional security other than not having passwords in the add-on configuration's YAML. See the official documentation [11] for more information.

In addition to the password requirement, most of the add-ons that have a web UI default to having the ssl: true option set. A self-signed cert on my local LAN is not a requirement, so I usually set this to false. There is an add-on for Let's Encrypt, but dealing with certificates is outside the scope of this series.

After you have looked through the **Configuration** tab, save your changes, and enable Node-RED on the add-on's main screen.



Current version: 7.2.0 (changelog)

Flow-based programming for the Internet of Things.
Visit Node-RED page for details.

Start on boot — Make the add-on start during a system boot
Watchdog — This will start the add-on if it crashes
Auto update — Auto update the add-on when there is a new version available
Show in sidebar — Add this add-on to your sidebar

(Steve Ovens, CC BY-SA 4.0)

Don't forget to start the plugin.

Most add-ons follow a similar procedure, so you can use this approach to set up other add-ons.

Wrapping up

Whew, that was a lot of screenshots! Fortunately, when you are doing the configuration, the UI makes these steps relatively painless.

At this point, your HA instance should be installed with some basic configurations and a few essential add-ons.

In the next article, I will discuss integrating custom Internet of Things (IoT) devices into Home Assistant. Don't worry; the fun is just beginning!

Links

[1] https://opensource.com/article/20/11/home-assistant
[2] https://opensource.com/article/20/11/cloud-vs-local-home-automation
[3] https://opensource.com/article/20/11/home-automation-part-3
[4] https://opensource.com/article/20/12/home-assistant
[5] https://sonoff.tech/product/smart-home-security/zbbridge
[6] https://www.digiblur.com/2020/07/how-to-use-sonoff-zigbee-bridge-with.html
[7] https://tasmota.github.io/docs/
[8] https://opensource.com/article/20/11/cloud-vs-local-home-automation
[9] https://opensource.com/article/20/11/wireless-protocol-home-automation
[10] https://en.wikipedia.org/wiki/MQTT
[11] https://www.home-assistant.io/docs/configuration/secrets/

How to set up custom sensors in Home Assistant

Dive into the YAML files to set up custom sensors in the sixth article in this home automation series.

IN THE LAST ARTICLE in this series about home automation, I started digging into Home Assistant. I set up a Zigbee integration [1] with a Sonoff Zigbee Bridge and installed a few add-ons, including Node-RED, File Editor, Mosquitto broker, and Samba. I wrapped up by walking through Node-RED's configuration, which I will use heavily later on in this series. The four articles before that one discussed what Home Assistant is [2], why you may want local control [3], some of the communication protocols [4] for smart home components, and how to install Home Assistant [5] in a virtual machine (VM) using libvirt.

In this sixth article, I'll walk through the YAML configuration files. This is largely unnecessary if you are just using the integrations supported in the user interface (UI). However, there are times, particularly if you are pulling in custom sensor data, where you have to get your hands dirty with the configuration files.

Let's dive in.

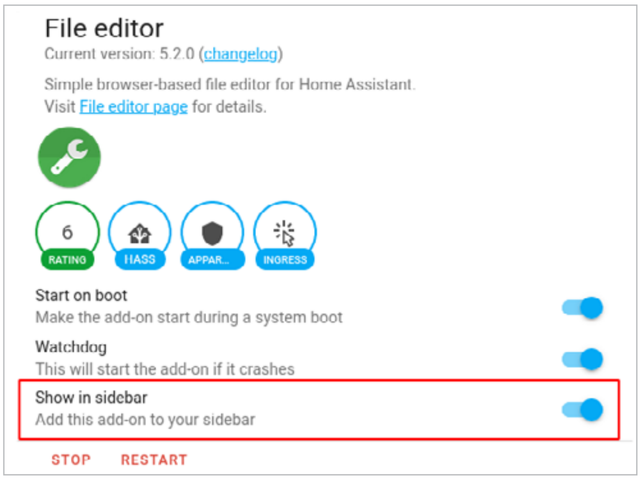
Examine the configuration files

There are several potential configuration files you will want to investigate. Although everything I am about to show you *can* be done in the main configuration.yaml file, it can help to split your configuration into dedicated files, especially with large installations.

Below I will walk through how I configure my system. For my custom sensors, I use the ESP8266 chipset, which is very maker-friendly. I primarily use Tasmota [6] for my custom firmware, but I also have some components running ESPHome [7]. Configuring firmware is outside the scope of this article. For now, I will assume you set up your devices with some custom firmware (or you wrote your own with Arduino IDE [8]).

The /config/configuration.yaml file

Configuration.yaml is the main file Home Assistant reads. For the following, use the File Editor you installed in the previous article. If you do not see File Editor in the left sidebar, enable it by going back into the **Supervisor** settings and clicking on **File Editor**. You should see a screen like this:


File editor
Current version: 5.2.0 (changelog)
Simple browser-based file editor for Home Assistant.
Visit File editor page for details.

Start on boot
Make the add-on start during a system boot

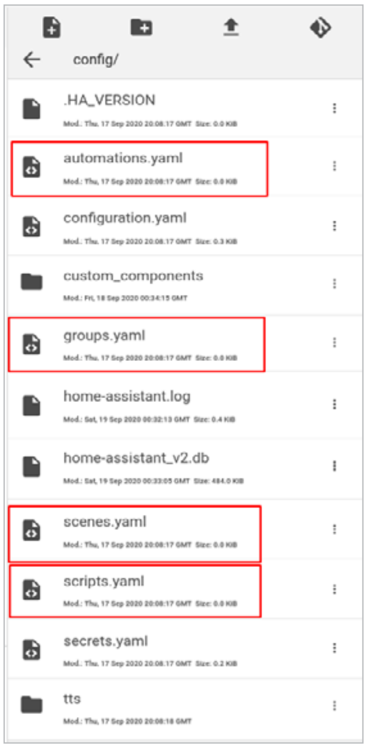
Watchdog
This will start the add-on if it crashes

Show in sidebar
Add this add-on to your sidebar

(Steve Ovens, CC BY-SA 4.0)

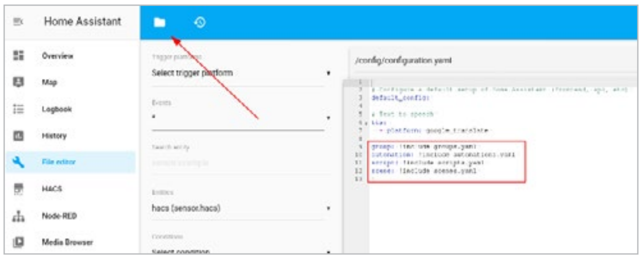
Make sure **Show in sidebar** is toggled on. I also always toggle on the **Watchdog** setting for any add-ons I use frequently.

Once that is completed, launch File Editor. There is a folder icon in the top-left header bar. This is the navigation icon. The `/config` folder is where the configuration files you are concerned with are stored. If you click on the folder icon, you will see a few important files:


```
config/

.HA_VERSION
Mod.: Thu, 17 Sep 2020 20:08:17 GMT  Size: 0.0 KiB
automations.yaml
Mod.: Thu, 17 Sep 2020 20:08:17 GMT  Size: 0.0 KiB
configuration.yaml
Mod.: Thu, 17 Sep 2020 20:08:17 GMT  Size: 0.3 KiB
custom_components
Mod.: Fri, 18 Sep 2020 00:34:15 GMT
groups.yaml
Mod.: Thu, 17 Sep 2020 20:08:17 GMT  Size: 0.0 KiB
home-assistant.log
Mod.: Sat, 19 Sep 2020 00:32:13 GMT  Size: 0.4 KiB
home-assistant_v2.db
Mod.: Sat, 19 Sep 2020 00:33:05 GMT  Size: 484.0 KiB
scenes.yaml
Mod.: Thu, 17 Sep 2020 20:08:17 GMT  Size: 0.0 KiB
scripts.yaml
Mod.: Thu, 17 Sep 2020 20:08:17 GMT  Size: 0.0 KiB
secrets.yaml
Mod.: Thu, 17 Sep 2020 20:08:17 GMT  Size: 0.2 KiB
tts
Mod.: Thu, 17 Sep 2020 20:08:18 GMT
```

(Steve Ovens, CC BY-SA 4.0)

The following is a default configuration.yaml:



/config/configuration.yaml

```
# Configure a default setup of Home Assistant (frontend, api, etc)
default_config:

# Text to speech
tts:
  - platform: google_translate

group: !include groups.yaml
automation: !include automations.yaml
script: !include scripts.yaml
scene: !include scenes.yaml
```

(Steve Ovens, CC BY-SA 4.0)

The notation `script: !include scripts.yaml` indicates that Home Assistant should reference the contents of scripts.yaml anytime it needs the definition of a script object. You'll notice that each of these files correlates to files observed when the folder icon is clicked.

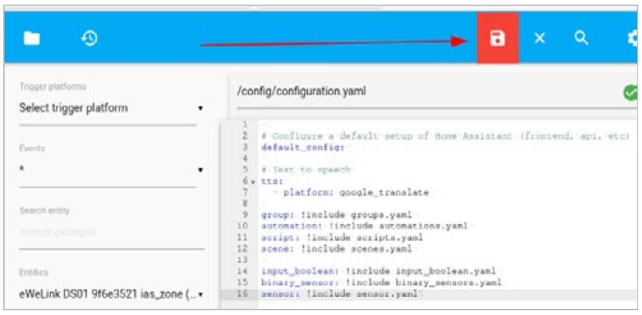
I added three lines to my configuration.yaml:

```
input_boolean: !include input_boolean.yaml
binary_sensor: !include binary_sensor.yaml
sensor: !include sensor.yaml
```

As a quick aside, I configured my MQTT settings (see Home Assistant's MQTT documentation [9] for more details) in the configuration.yaml file:

```
mqtt:
  discovery: true
  discovery_prefix: homeassistant
  broker: 192.168.11.11
  username: mqtt
  password: superpassword
```

If you make an edit, don't forget to click on the Disk icon to save your work.



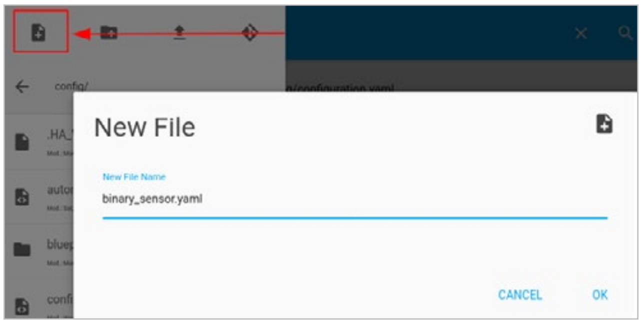
/config/configuration.yaml

```

# Configure a default setup of Home Assistant (frontend, api, etc)
default_config:

# Text to speech
tts:
  - platform: google_translate

group: !include groups.yaml
automation: !include automations.yaml
script: !include scripts.yaml
scene: !include scenes.yaml

input_boolean: !include input_boolean.yaml
binary_sensor: !include binary_sensors.yaml
sensor: !include sensor.yaml
```

(Steve Ovens, CC BY-SA 4.0)

The /config/binary_sensor.yaml file

After you name your file in configuration.yaml, you'll have to create it. In the File Editor, click on the folder icon again. There is a small icon of a piece of paper with a + sign in its center. Click on it to bring up this dialog:



New File

New File Name: binary_sensor.yaml

(Steve Ovens, CC BY-SA 4.0)

I have three main types of binary sensors [10]: door, motion, and power. A binary sensor has only two states: on or off. All my binary sensors send their data to MQTT. See my article on cloud vs. local control [11] for more information about MQTT.

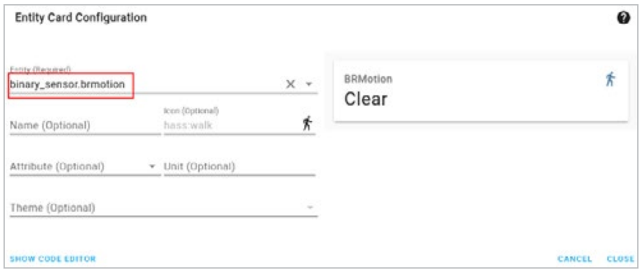
My binary_sensor.yaml file looks like this:

```
- platform: mqtt
  state_topic: "BRMotion/state/PIR1"
  name: "BRMotion"
  qos: 1
  payload_on: "ON"
  payload_off: "OFF"
  device_class: motion

- platform: mqtt
  state_topic: "IRBlaster/state/PROJECTOR"
  name: "ProjectorStatus"
  qos: 1
  payload_on: "ON"
  payload_off: "OFF"
  device_class: power

- platform: mqtt
  state_topic: "MainHallway/state/DOOR"
  name: "FrontDoor"
  qos: 1
  payload_on: "open"
  payload_off: "closed"
  device_class: door
```

Take a look at the definitions. Since `platform` is self-explanatory, start with `state_topic`.

- `state_topic`, as the name implies, is the topic where the device's state is published. This means anyone subscribed to the topic will be notified any time the state changes. This path is completely arbitrary, so you can name it anything you like. I tend to use the convention `location/state/object`, as this makes sense for me. I want to be able to reference all devices in a location, and for me, this layout is the easiest to remember. Grouping by device type is also a valid organizational layout.

- `name` is the string used to reference the device inside Home Assistant. It is normally referenced by `type.name`, as seen in this card in the Home Assistant Lovelace [12] interface:



Entity (Required): binary_sensor.brmotion
Name (Optional)
Icon (Optional): hass:walk
Attribute (Optional)
Unit (Optional)
Theme (Optional)

BRMotion
Clear

(Steve Ovens, CC BY-SA 4.0)

- `qos`, short for quality of service, refers to how an MQTT client communicates with the broker when posting to a topic.
- `payload_on` and `payload_off` are determined by the firmware. These sections tell Home Assistant what text the device will send to indicate its current state.
- `device_class`: There are multiple possibilities for a device class. Refer to the Home Assistant documentation [13] for more information and a description of each type available.

The /config/sensor.yaml file

This file differs from `binary_sensor.yaml` in one very important way: The sensors within this configuration file can have vastly different data inside their payloads. Take a look at one of the more tricky bits of sensor data, temperature.

Here is the definition for my DHT temperature sensor:

```
 - platform: mqtt
    state_topic: "Steve_Desk_Sensor/tele/SENSOR"
    name: "Steve Desk Temperature"
    value_template: '{{ value_json.DHT11.Temperature }}'

  - platform: mqtt
    state_topic: "Steve_Desk_Sensor/tele/SENSOR"
    name: "Steve Desk Humidity"
    value_template: '{{ value_json.DHT11.Humidity }}'
```

You'll notice two things right from the start. First, there are two definitions for the same `state_topic`. This is because this sensor publishes three different statistics.

Second, there is a new definition of `value_template`. Most sensors, whether custom or not, send their data inside a JSON payload. The template tells Home Assistant where the important information is in the JSON file. The following shows the raw JSON coming from my homemade sensor. (I used the program jq to make the JSON more readable.)

```
{
  "Time": "2020-12-23T16:59:11",
  "DHT11": {
    "Temperature": 24.8,
    "Humidity": 32.4,
    "DewPoint": 7.1
  },
  "BH1750": {
    "Illuminance": 24
  },
  "TempUnit": "C"
}
```

There are a few things to note here. First, as the sensor data is stored in a time-based data store, every reading has a Time entry. Second, there are two different sensors attached to this output. This is because I have both a DHT11 temperature sensor and a BH1750 light sensor attached to the same ESP8266 chip. Finally, my temperature is reported in Celsius.

Hopefully, the Home Assistant definitions will make a little more sense now. `value_json` is just a standard name given to any JSON object ingested by Home Assistant. The format of the `value_template` is `value_json.<component>.<data point>`.

For example, to retrieve the dewpoint:

```
value_template: '{{ value_json.DHT11.DewPoint}}'
```

While you can dump this information to a file from within Home Assistant, I use Tasmota's `Console` to see the data it is publishing. (If you want me to do an article on Tasmota, please let me know in the comments below.)

As a side note, I also keep tabs on my local Home Assistant resource usage. To do so, I put this in my sensor.yaml file:

```
 - platform: systemmonitor
    resources:
      - type: disk_use_percent
        arg: /
      - type: memory_free
      - type: memory_use
      - type: processor_use
```

While this is technically not a sensor, I put it here, as I think of it as a data sensor. For more information, see the Home Assistant's system monitoring [14] documentation.

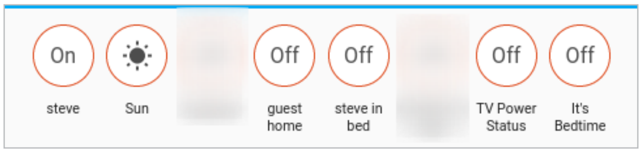
The /config/input_boolean file

This last section is pretty easy to set up, and I use it for a wide variety of applications. An input boolean is used to track the status of something. It's either on or off, home or away, etc. I use these quite extensively in my automations.

My definitions are:

```
steve_home:
    name: steve
 steve_in_bed:
    name: 'steve in bed'
 guest_home:

 kitchen_override:
    name: kitchen
 kitchen_fan_override:
    name: kitchen_fan
 laundryroom_override:
    name: laundryroom
 bathroom_override:
    name: bathroom
 hallway_override:
    name: hallway
 livingroom_override:
    name: livingroom
 ensuite_bathroom_override:
    name: ensuite_bathroom
 steve_desk_light_override:
    name: steve_desk_light
 projector_led_override:
    name: projector_led

 project_power_status:
    name: 'Projector Power Status'
 tv_power_status:
    name: 'TV Power Status'
 bed_time:
    name: "It's Bedtime"
```

I use some of these directly in the Lovelace UI. I create little badges that I put at the top of each of the pages I have in the UI:



(Steve Ovens, CC BY-SA 4.0)

These can be used to determine whether I am home, if a guest is in my house, and so on. Clicking on one of these badges allows me to toggle the boolean, and this object can be read by automations to make decisions about how the "smart devices" react to a person's presence (if at all). I'll revisit the booleans in a future article when I examine Node-RED in more detail.

Wrapping up

In this article, I looked at the YAML configuration files and added a few custom sensors into the mix. You are well on the way to getting some functioning automation with Home Assistant and Node-RED. In the next article, I'll dive into some basic Node-RED flows and introduce some basic automations.

Stick around; I've got plenty more to cover.

Links

[1] https://opensource.com/article/21/1/home-automation-5-homeassistant-addons
[2] https://opensource.com/article/20/11/home-assistant
[3] https://opensource.com/article/20/11/cloud-vs-local-home-automation
[4] https://opensource.com/article/20/11/home-automation-part-3
[5] https://opensource.com/article/20/12/home-assistant
[6] https://tasmota.github.io/docs/
[7] https://esphome.io/
[8] https://create.arduino.cc/projecthub/Niv_the_anonymous/esp8266-beginner-tutorial-project-6414c8
[9] https://www.home-assistant.io/docs/mqtt/broker
[10] https://www.home-assistant.io/integrations/binary_sensor/
[11] https://opensource.com/article/20/11/cloud-vs-local-home-automation
[12] https://www.home-assistant.io/lovelace/
[13] https://www.home-assistant.io/integrations/binary_sensor/
[14] https://www.home-assistant.io/integrations/systemmonitor

Protect your Home Assistant with these backups

Make sure you can recover quickly from a home automation failure with a solid backup and hardware plan in the seventh article in this series.

IN THE LAST TWO ARTICLES in this series on home automation with Home Assistant (HA), I walked through setting up a few integrations [1] with a Zigbee Bridge and some custom ESP8266 [2] devices that I use for automation. The first four articles in the series discussed what Home Assistant is [3], why you may want local control [4], some of the communication protocols [5] for smart home components, and how to install Home Assistant [6] in a virtual machine (VM) using libvirt.

Now that you have a basic home automation setup, it is a good time to take a baseline of your system. In this seventh article, I will talk about snapshots, backups, and backup strategies. Let's get right into it.

Backups vs. copies

I'll start by clearing up some ambiguity: A copy of something is not the same as a backup. Here is a brief overview of the difference between a copy and a backup. Bear in mind that this comes from the lens of an IT professional. I work with client data day in and day out. I have seen many ways that backups can go sideways, so the following descriptions may be overkill for home use. You'll have to decide just how important your Home Assistant data really is.

- **Copies**: A copy is just what it sounds. It is when you highlight something on your computer and hit **Ctrl+C** and paste it somewhere else with **Ctrl+V**. Many people may view this as backing up the source, and to some extent, that is true. However, a copy is merely a representation of a point in time. If it's taken incorrectly, the newly created file can be corrupt, leading to a false sense of security. In addition, the source may have a problem—meaning the copy will also have a problem. If you have just a single copy of a file, it's often the same as having nothing at all. When it comes to backup, the saying "one is none" is absolutely true. If you do not have files going back over time, you won't have a good idea of whether the system creating the backups has a problem.
- **Backups and snapshots**: In Home Assistant, it is a bit tricky to differentiate between a copy and a backup. First, Home Assistant uses the term "snapshot" to refer to what we traditionally think of backups. In this context, a backup is very similar to a copy because you don't use any type of backup software, at least not in the traditional sense. Normally, backup software is designed specifically to get all the files that are hidden or otherwise protected. For example, backup software for a computer (such as CloneZilla) makes an exact replica (in some cases) of the hard drive to ensure no files are missed. Home Assistant knows how to create snapshots and does it for you. You just need to worry about storing the files somewhere.

Set a good backup strategy

Before I get into how to deal with snapshots in Home Assistant, I want to share a brief story from a recent client. Remember when I mentioned that simply having a single copy of your files doesn't give you any indication that a problem has occurred? My client was doing all of the right things when it came to backups. The team was using the proper methodology for backups, kept multiple files going back a certain period of time, ensured there were more than two copies of each backup, and was especially careful that backups were not being stored locally on the machine being backed up. Sounds great, doesn't it? They were doing everything right. Well, almost. The one thing they neglected was testing the backups. Most people put this off or disregard it entirely. I admit I am guilty of not testing my backups frequently. I do it when I remember, which is usually once every few months or so.

In my client's case, a software upgrade created a new requirement from the backup program. This was missed. The backups continued to hum along, and the automated checks passed. There were files after every backup run, they were larger than a certain amount, and the magic file checks [7] reported the correct file type. The problem was that the file sizes shrunk significantly due to the software change. This meant the client was not backing up the data it thought.

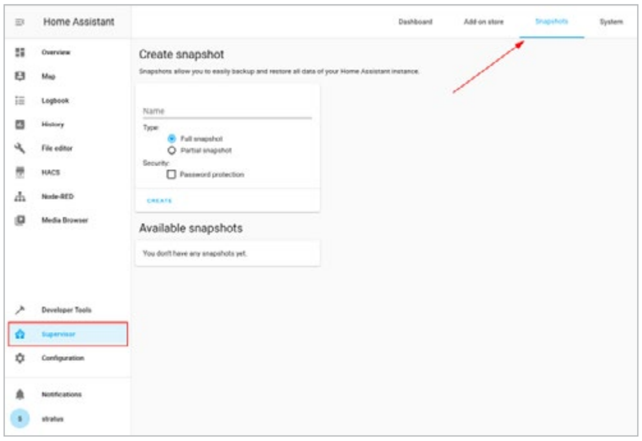
This story has a happy ending, which brings me to my point: Because the client was doing everything else right, we could go through the backups and identify the precise moment when something changed. From this, we linked this to the date of an upgrade some weeks back. Fortunately, there was no emergency that precipitated the investigation. I happened to be doing a standard audit when I discovered the discrepancy.

The moral of the story? Without proper backup strategies, we would have had a much harder time tracking down this problem. Also, in the event of a failure, we would have had no recovery point.

A good backup strategy usually entails daily, weekly, and monthly backups. For example, you may decide to keep all your daily backups for two weeks, four weekly backups, and perhaps four monthly backups. This, in my opinion, is overkill for Home Assistant after you have a stable setup. You'll have to choose the level of precision you need. I take a backup before I make any change to the system. This gives me a known-good situation to revert to.

Create snapshots

Great, so how do you create a snapshot in Home Assistant? The **Snapshots** menu resides inside the **Supervisor** tab on the left-size panel.



Create snapshot

Snapshots allow you to easily backup and restore all data of your Home Assistant instance.

Name

Type: [✓ Full snapshot] Partial snapshot

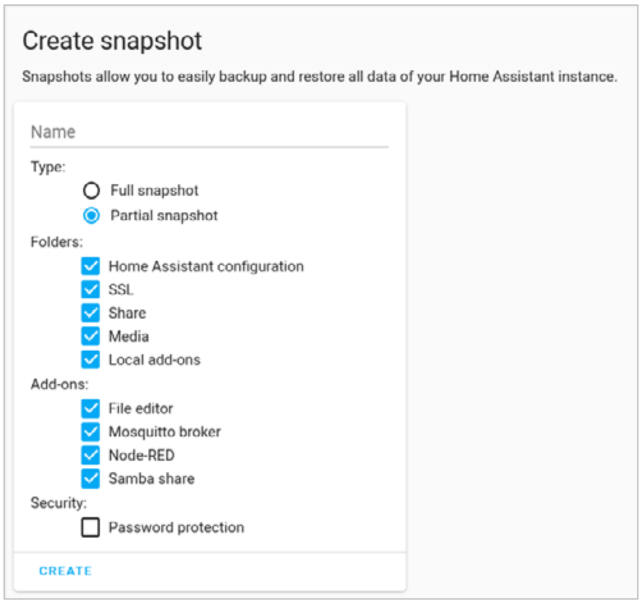
Security: ☐ Password protection

Available snapshots

You don't have any snapshots yet.

(Steve Ovens, CC BY-SA 4.0)

You have two options for creating a snapshot: *Full snapshot* or *Partial snapshot*. A Full snapshot is self-explanatory. You have some options with a Partial snapshot.



Create snapshot

Snapshots allow you to easily backup and restore all data of your Home Assistant instance.

Name

Type: Full snapshot [✓ Partial snapshot]

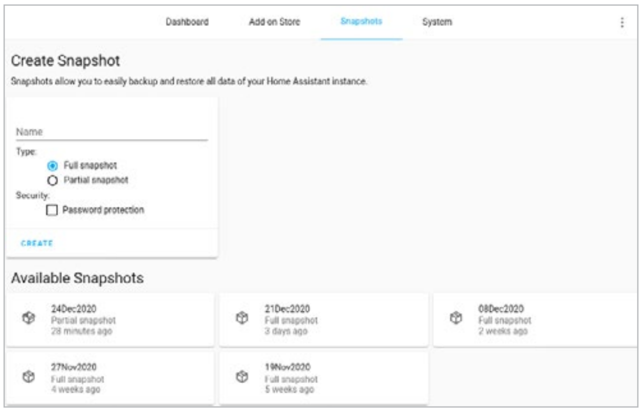
Folders:
- ✓ Home Assistant configuration
- ✓ SSL
- ✓ Share
- ✓ Media
- ✓ Local add-ons

Add-ons:
- ✓ File editor
- ✓ Mosquitto broker
- ✓ Node-RED
- ✓ Samba share

Security: ☐ Password protection

(Steve Ovens, CC BY-SA 4.0)

Any component you install in Home Assistant will populate in this menu. Choose a name for your backup and click **Create**. This will take some time, depending on the speed of the disk and the size of the backup. I recommend keeping at least four backups on your machine if you can afford the space.



Create Snapshot

Snapshots allow you to easily backup and restore all data of your Home Assistant instance.

Name

Type: [✓ Full snapshot] Partial snapshot

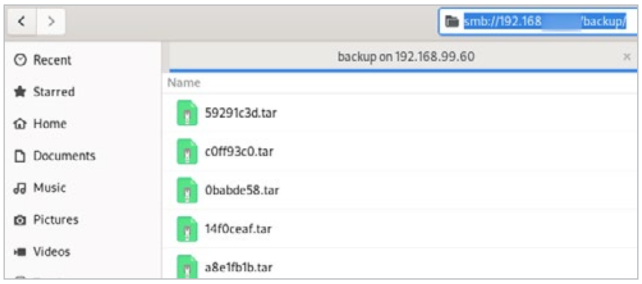
Security: ☐ Password protection

Available Snapshots

24Dec2020 Partial snapshot 28 minutes ago
21Dec2020 Full snapshot 3 days ago
08Dec2020 Full snapshot 2 weeks ago
27Nov2020 Full snapshot 4 weeks ago
19Nov2020 Full snapshot 5 weeks ago

(Steve Ovens, CC BY-SA 4.0)

You can retrieve these files from Home Assistant with File Browser if you set up the **Samba share** extension.



59291c3d.tar
c0ff93c0.tar
0babde58.tar
14f0ceaf.tar
a8e1fb1b.tar

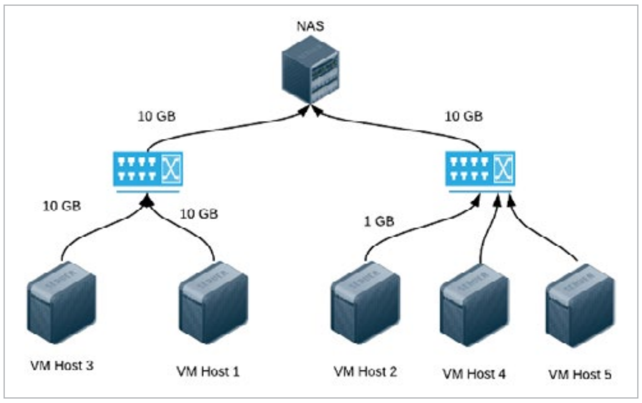
(Steve Ovens, CC BY-SA 4.0)

Save these files somewhere safe. The name you give the backup is contained in the metadata inside Home Assistant, and the file names are randomly generated. After I copy them to a different location, I usually rename them because when I test the restoration process on a different machine, the new file name does not matter.

My homelab strategy

I run my Home Assistant instance on top of KVM on a Linux host. I have had a few requests to go into a little more detail on this, so feel free to skip past this section as it's not directly related to HA.

Due to the nature of my job, I have a fairly large variety of hardware, which I use for a homelab [8]. Sometimes this is because physical hosts are easier to work with than VMs for certain clustering software. Other times, this is because I keep workloads isolated to specific hardware. Either way, this means I already have a certain amount of knowledge built up around managing and dealing with KVM. Not to mention the fact that I run *almost exclusively* open source software (with a few exceptions). Here is a basic layout of my homelab:



(Steve Ovens, CC BY-SA 4.0)

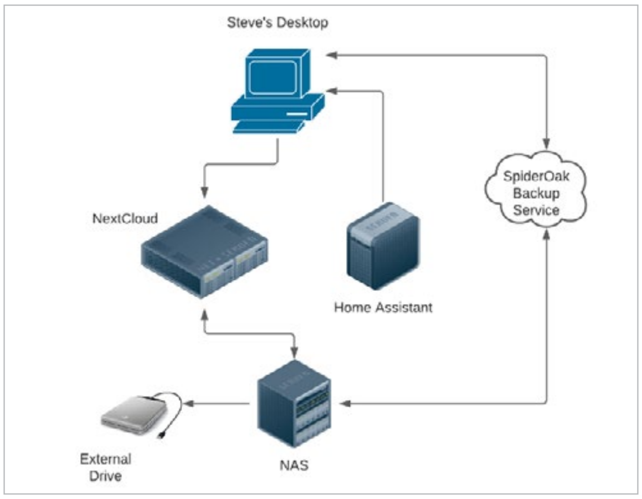
The network-attached storage (NAS) has dual 10GB network cards that feed into uplink ports. Two of the KVM hosts have 10GB network interface cards (NICs), while the hosts on the right have regular 1GB network cards.

For Home Assistant, this is well into overkill territory. However, this infrastructure was not designed for HA. HA runs just fine on a Raspberry Pi 4 (4GB version) at my parents' house.

The VM that hosts Home Assistant has three vCPU cores of a Broadwell Core I5 CPU (circa 2015) with 8GB of RAM. The CPU tends to remain around 25% usage, and I rarely use more than 2.2GB of RAM. This is with 11 add-ons, including InfluxDB and Grafana, which are heavier applications.

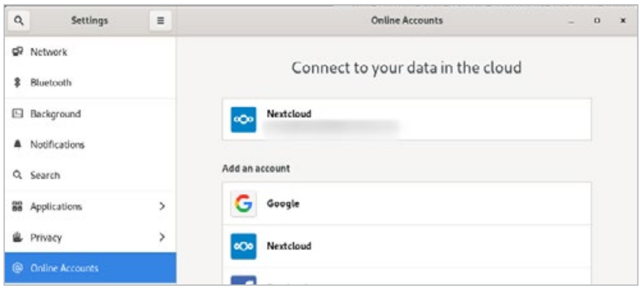
While I do have shared storage, I do not use it for live migration or anything similar. Instead, I use this for backend storage for specific mount points in a VM. For example, I store the data directory from Nextcloud on the NAS, divorcing the data from the service.

At any rate, I have a few approaches to backups with this setup. Naturally, I use the Home Assistant snapshotting function to provide the first layer of protection. I tend to store only four weeks' worth of data on the VM. What do I do with the files themselves? Here is a diagram of how I try to keep my backups safe:



(Steve Ovens, CC BY-SA 4.0)

Using the Samba add-on, I pull a copy of the snapshot onto my GNOME desktop. I configure Nextcloud using GNOME's **Online Accounts** settings.



(Steve Ovens, CC BY-SA 4.0)

Nextcloud takes a copy and puts it on my NAS. Both my desktop and the NAS use SpiderOak One Backup [9] clients to ensure the backups are linked to more than one host. In the unlikely event that I delete a device from my SpiderOak account, the file is still linked to another device. I chose SpiderOak because it supports a Linux client, and it is privacy-focused and has no insight into what files it stores. The files are encrypted before being uploaded, and only the owner has the ability to decrypt them. The downside is that if you lose or forget your password, you lose your backups.

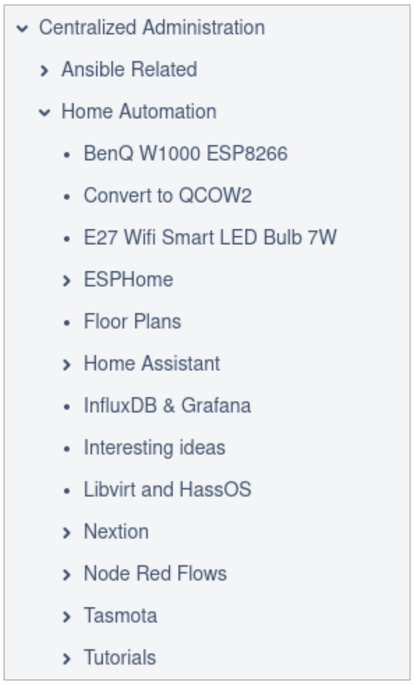
Finally, I keep a cold copy on an external hard drive. I have a 14TB external drive that remains off and disconnected from the NAS except when backups are running. It's not on the diagram, but I also occasionally replicate to a computer at my in-laws' house.

I can also take snapshots of the VM during critical operations (such as Home Assistant's recent upgrade from using a numbered point release to a month-based numbering system).

I use a similar pipeline for most things that I back up, although I recognize it is a bit overkill. Also, this whole process has the flaw that it relies on me. Aside from SpiderOak and Nextcloud, I have not automated this process. I have scripts that I run, but I do not run them in a cron or anything like that. In hindsight, perhaps I should work on that.

This setup may be considered extreme, but the built-in versioning in Nextcloud and SpiderOak, along with making copies in multiple locations, means that I am unlikely to suffer a failure that I can't recover from. At the very least, I should be able to dig up a close reference.

As a final precaution, I make sure to keep the important information about *how* I set things up on my private wiki on Wiki.js [10]. I keep a section just for home automation.



- Centralized Administration
  - Ansible Related
  - Home Automation
    - BenQ W1000 ESP8266
    - Convert to QCOW2
    - E27 Wifi Smart LED Bulb 7W
    - ESPHome
    - Floor Plans
    - Home Assistant
    - InfluxDB & Grafana
    - Interesting ideas
    - Libvirt and HassOS
    - Nextion
    - Node Red Flows
    - Tasmota
    - Tutorials

(Steve Ovens, CC BY-SA 4.0)

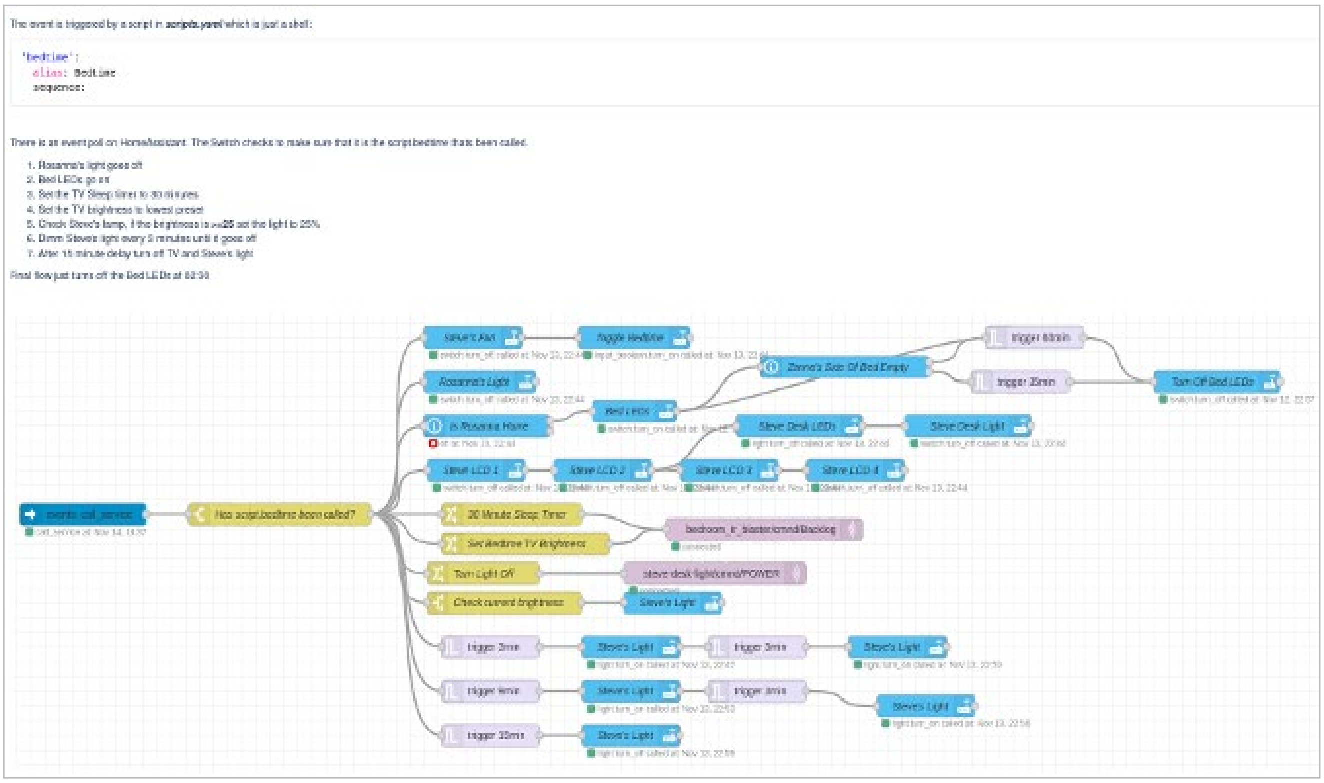
When you get into creating Node-RED automations (in the next article), I suggest you keep your own notes. I take a screenshot of the flow, write a brief description, so I know what I was attempting to achieve, and dump the flow to JSON (for brevity, I omitted the JSON from this screenshot):

Wrapping up

Backups are essential when you're using Home Assistant, as it is a critical component of your infrastructure that always needs to be functioning. Small downtime is acceptable, but the ability to recover from a failure quickly is crucial. Granted, I have found Home Assistant to be rock solid. It has never failed on its own; any problems I have had were external to HA. Still, if you are going to make HA a central part of your house, I strongly recommend putting a good backup strategy in place.

Links

[1] https://opensource.com/article/21/1/home-automation-5-homeassistant-addons
[2] https://opensource.com/article/21/1/home-assistant-6-custom-sensors
[3] https://opensource.com/article/20/11/home-assistant
[4] https://opensource.com/article/20/11/cloud-vs-local-home-automation
[5] https://opensource.com/article/20/11/home-automation-part-3
[6] https://opensource.com/article/20/12/home-assistant
[7] https://linux.die.net/man/5/magic
[8] https://opensource.com/article/19/3/home-lab
[9] https://spideroak.com/
[10] https://wiki.js.org/



The event is triggered by a script in **scripts.yaml** which is just a shell:

```
'bedtime':
  alias: Bedtime
  sequence:
```

There is an event poll on HomeAssistant. The Switch checks to make sure that it is the script bedtime that been called.

1. Rosanna's light goes off
2. Bed LEDs go on
3. Set the TV Sleep timer to 30 minutes
4. Set the TV brightness to lowest preset
5. Check Steve's lamp, if the brightness is >=25 set the light to 25%
6. Dimm Steve's light every 3 minutes until it goes off
7. After 15 minute delay turn off TV and Steve's light

Final flow just turns off the Bed LEDs at 03:30

(Steve Ovens, CC BY-SA 4.0)