Mini Smart Home with Huzzah, HASSio and Crickit
Created by Richard Albritton
Overview

Though the term Internet of Things covers a vast range of tech, most people think of Smart Homes. There are many ways doing a Smart Home, but I think that the word "smart" is used a bit too loosely. If you have ever looked at consumer Smart Home products, most of them require an app on your phone and probably another app that connects those apps. Even then most of the logic that goes into Automations is limited to something like: "if this one thing happens, then do this other one thing". Then there is the constant request for remote internet access that ties up network resources, creates a security risk, and gives control over your personal data to large companies.

All that craziness so that you can press a button on your phone to turn out the lights. However, it doesn't have to be this way. A true Smart Home uses your local network to handle most things and only connects remotely when it needs to. What if you could keep your own data and use it to create dynamic and complex Automations so that you hardly ever need to use an app. What if your Smart Home was smart enough to do things on its own?

The Mini Smart Home is a test bed that can let you move one step closer to making a truly smart Smart Home. When finished, you will have a completely independent system that hosts a customisable browser based User Interface, a device management system, usage and data logging, advanced automation tools, and user account security. This is all done with a Smart Home server OS called Home Assistant.

What is Involved?
First we need a house to set up. We won't get into this much because there are so many ways that you can build a house. The house that you will see me work with is one that I designed on a laser cutter.

Next you will want to set up a Smart Home Hub. This is the brain of the house and the only device that uses remote internet access. It should be noted that while you do need a network setup for this kind of Smart Home, it doesn't need Internet access once all devices are setup and configured. Remote access can be made available through this Smart Home Hub if needed but we will talk more about that later.

Last is installing the individual devices that will be controlled through the home. There are many standards for this in an actual Smart Home and they all depend on if you are building a new home, retrofitting an older home, or renting a home. The standard is that if you can run physical wire through your home that is the best option, but it is not practical to do so unless you are building a new home. The next best thing is wireless tech like Z-Wave, WiFi, BLE, or LoRa and that is what most people go with for older homes and apartment dwellers. Since we are building the Mini Smart Home, we will be running physical wires from each device to a central microcontroller that will use WiFi to connect to the Smart Home Hub.
Raspberry Pi Server Setup

For the Smart Home server that will drive the user interface, automations, and other cool stuff, we will be installing Home Assistant onto a Raspberry Pi and connect it to your home network. This can be done on a Pi Zero W but the setup and server runs much faster on a Pi 3B+ connected to your home router via Ethernet.

Basically this is the easiest and best way to set up a Smart Home Hub like this, as the Disk Image we will use automatically configures the server when connected to the internet through Ethernet. By using a wired connection for the server, we can be sure that we get the fastest and most reliable connection possible for our Smart Home.

To get this going, we will create a boot disk using a MicroSD card. This MicroSD card will then go into the Raspberry Pi, which is then connected to your home Router via Ethernet. After about 20 minutes, you should be able to access the user interface from any device connected to your home WiFi by simply going to http://hassio.local:8123 (https://adafru.it/DQD).

Here are the things we will need to get your server up and running with a Pi Zero W:

**Raspberry Pi Zero W**

**SD/MicroSD Memory Card - 16GB Class 10 - Adapter Included**
If you plan to use this server for an actual Smart Home server, it is worth the extra expense to get the Raspberry Pi 3 B+. That is what I use to control all 27 IOT devices in my home.

Here are the things we will need to get your server up and running with a Pi 3 B+:

- **5V 2.5A Switching Power Supply with 20AWG MicroUSB Cable**
  - OUT OF STOCK
- **Raspberry Pi 3 - Model B+ - 1.4GHz Cortex-A53 with 1GB RAM**
  - $35.00
  - IN STOCK
  - Add To Cart
- **SD/MicroSD Memory Card - 16GB Class 10 - Adapter Included**
  - OUT OF STOCK
  - Out Of Stock
As for the Software tools, you will want the following (Windows-based, there are similar tools for other OSes).

SD Card Formatting tool:

https://adafru.it/FKd
https://adafru.it/FKd

SD Card Boot Disk tool:

https://adafru.it/EMc
https://adafru.it/EMc

Ethernet Cable - 3 ft long

OUT OF STOCK

5V 2.5A Switching Power Supply with 20AWG MicroUSB Cable

OUT OF STOCK

Installing Home Assistant

Download the latest disk image for Hass.io from the Home Assistant page. We will be using the Raspberry Pi 3B+ for this project so that is the one you want to get, but if you want to use another Pi version, pick that one from the downloads page.

If needed, use the SD Card Formatter to format the MicroSD Card and remove any partitions.

Use Etcher to install the boot disk image onto your MicroSD card.

Once finished, eject the MicroSD card and put it into the MicroSD slot for the Raspberry Pi.

Use the Ethernet cable to connect your Raspberry Pi into an available network port on the back of your WiFi Router.

Plug the Raspberry into power using the Micro USB connector.

It can take up to 20 min for Hassio to install. After that time you should be able to access the user interface by going to http://hassio.local:8123 from and web browser on your local network WiFi or Ethernet connection.

You can connect the Raspberry Pi to your internet using WiFi if you do not have access to an Ethernet connection. However this process is much more complex and should be avoided if possible.
Once connected, you will be prompted to create a user account. This will serve as the administrative account for your Smart Home Hub so be sure to remember the information that you use for the login. **There is no password recovery system** in place.

Once finished, click **CREATE ACCOUNT**.
Now you will be asked to name your home and set it's location.

The Location part helps with things like weather reports, sun position data, and other things that can be added latter.

You can also set your **Time Zone, Elevation**, and unit of measurements.

Click **NEXT** when finished.

If you have any smart devices already connected to your network, Home Assistant may detect them and let you set them up. Devices like Philips Hue, Lifx, WeMo, Google Cast, and a few others are quite easy to set up using this user interface. You may choose to set these up now or do so later using the Integrations tool in the Configurations menu.

Click **FINISH** when done.
Now you are all set up and any smart devices that you added from the previous screen will now be available and ready to control using Home Assistant.

On the left you should see the Menu where you can see a Map view, Logbook, History, Hass.io, Configuration, and some Developer tools.

The Configuration menu is where you can add many supported devices and services using Integrations. You can also add new user accounts including non Admin users that can not access the full menu structure.

The General menu is where you can change location info along with reloading and restarting the core services if needed. Area Registry lets you define rooms for better use with Google Home, Alexa, or Home Kit. Automation allows you to create complex automations using a fairly easy form based GUI. Script is similar to Automation, but there is no event trigger. You use Script mostly to create complex combinations of tasks that you want to call repeatedly with an automation or with a button in the user interface.
MQTT Setup

What is MQTT?

MQTT is a kind of secure machine-to-machine message protocol that is made for the internet of things (IoT). For MQTT to work you need to have a computer running as an MQTT Broker and all other devices connect to that computer as an MQTT Client.

MQTT Clients can Subscribe to a Feed on the MQTT Broker and will receive an update if the content of the Feed it is Subscribed to changes. Publishing to a Feed is also allowed by the MQTT Client and the MQTT Broker will pass that published data along to any device that is Subscribed to that particular Feed. Any MQTT Client can Subscribe and Publish to a Feed, but they can also just Subscribe or Publish depending on what your device needs to do.

Adafruit.io hosts an MQTT server, as do many other companies, and they work very well for testing out devices or creating global networks of devices. However, these servers typically have Feed limits or subscription fees that may not always jell with what you want to do.

So why not run your own MQTT Broker using Mosquitto? It is not quite as powerful as Adafruit.io, but it works just as well for most non-industrial projects. Plus it is free and secure.

Install Mosquito Broker

From the Menu, click Hass.io

Go to the ADD-ON STORE

From the ADD-ON STORE, click on Mosquitto broker.
Add the following text to the **Config** text box.

```json
{
    "logins": [
        {
            "username": "yourMQTT",
            "password": "yourMQTTpass"
        }
    ],
    "anonymous": false,
    "customize": {
        "active": false,
        "folder": "mosquitto"
    },
    "certfile": "fullchain.pem",
    "keyfile": "privkey.pem",
    "quiet_logs": true
}
```

Change the **username** and **password** to whatever you want to use for connecting to the MQTT Broker.

Click on **SAVE** to save your new settings.

Now press **START** to start your new MQTT Broker.
Click on the Configuration menu.

Select Integrations from this menu.

You should see a list of devices waiting to be set up.

If you see MQTT in this list, click CONFIGURE. If MQTT is not in the list, go to Configuration -> General -> RESTART. Once reconnected, go back to Integrations and MQTT should be there.
Click **SUBMIT**

Click **FINISH**
You now have your own MQTT server linked to the Home Assistant server. From any MQTT client, you can interact with feeds using the Home Assistant IP address as the Host and the MQTT Username and Password that you set up.

This will only work within your local network unless you use Port forwarding using the MQTT port to point internet traffic to your Home Assistant IP address. I do not recommend this unless you have some experience with network administration as it could pose a security risk if not set up properly.
Configurator

While many consumer products can be set up using the Interactions tool, other devices like those that use MQTT will need to be set up in the Configuration.YAML file. The code that goes in there will depend on what you are setting up and we will get to that after setting up the Configurator add-on.

1. From the ADD-ON STORE, find Configurator and click it.
2. Click the INSTALL link.
3. Once installed click START.
4. Click to turn on the Show in Sidebar option.

You can now access the file system by clicking on the OPEN WEB UI link or through the Configurator menu option on the left.
Now you are ready to edit files for Home Assistant. Next you will find and edit the Configuration.YAML file so that you can add all sorts of fun custom Arduino and MQTT devices to the user interface.
Node Red

Node RED is a Flow based programming interface that is somewhere between block and text based coding. It can be an easy way to program automations for Home Assistant that are a bit more advanced than what can be done with the Automations menu.

Add Node-RED

From the Hass.io ADD-ON STORE, scroll down to find Node-Red and click on it.

1. Click the INSTALL link.
2. Once Installed, scroll down to the Config section.
3. Set a credential_secret, which is used to encrypt sensitive data. This is just a password, which you should save in a secondary location.
4. Next set ssl and require_ssl to false.
   ○ Note: SSL can be added, more information in the documentation (https://adafruit.it/FKh).
5. Click the SAVE link
6. Now scroll up and click the START link.
7. This can take a few moments to start up, but you can check the log at the bottom of the page to see when Node-RED is up and running.
8. If desired, turn on the Show in sidebar
9. Once finished, you can now click on the OPEN WEB UI link or the Node-RED menu link to open Node-RED.
If you get an error “502: Bad Gateway” this could indicate that Node-RED is not fully loaded. Check the Log to see Node-RED is still loading or to read any errors while loading.
Setup Node-RED MQTT Client

Since we have already installed the MQTT Broker we may as well set up Node-RED to use that for sending and receiving data from connected devices right?

- Scroll down to find the mqtt node under input or output.
- Move the mqtt node to Flow 1
- Double click on the mqtt node to open its Properties window.

Click the Edit icon to add a new MQTT broker.
Under the **Connection** tab, add your **MQTT Server** address. This will be the same IP address as your Raspberry Pi unless you are using another MQTT server like Adafruit.IO

Next click the **Security** tab.

Add the **Username** and **Password** that you set for your MQTT server.

Now click **Add**

If you misplaced your MQTT credentials, they can be found by going to the Hass.io menu and click on Mosquitto broker. The Password and Username should be in the Config field.

That will add your connection and you can now set a topic to interact with. As a test, I will usually add and inject node to send a time stamp to the **mqtt** node under **output** with a **Topic** of "**test**". Then I will get another **mqtt** node from the **input** section, set it's **Topic** to "**test**" and connect it to a **debug** output.
When you click the tab next to the timestamp **inject** node it will send the Unix time to the topic "test" via MQTT. Once that happens, the **mqtt** node set as **input** will receive that update and send it to be printed in the **debug** log.
Smart Home hardware Setup

Now that we have the Smart Home Hub up and running, it is time to get the actual Smart Home tech installed.

This is all of the things that will get installed into the house and how it will all get connected to the Crickit board. There is plenty of space for other devices to be added, but we will just start with this stuff as each one will act similar to some sort of consumer smart home product available today.

The wiring for each type of device is broken out into their own sections from here on.
Wiring the Home

When it comes to Smart Homes, or any other network of devices, physical wires are always the best way to go if you can use them. Even though a lot of IoT and Smart Home companies push wireless tech, most recommended only going wireless if it is too expensive or too difficult for wires.

Basically older houses or apartments cannot have more wires installed without ripping out the walls, so they use WiFi or Z-Wave to connect things. Not to say that wireless tech is bad, but it often uses more power, is susceptible to signal interference, and could pose a security risk. If you are building a new house or plan to rip out all of the walls anyway, be sure to run plenty of wire to each room to make sure you can install whatever tech you will need later. So we will be running lots of wire because this is a new house.

No Soldering Options

Jumper wires come in a lot of lengths and you can get a bunch to avoid having to solder things together. Be sure to keep things color coded because it can be hard to keep track of all those wires. You will need the following types if you want to go that route.
To connect any wires that need to be cut and connected, these terminal blocks should do the job.

Premium Male/Male Jumper Wires - 20 x 12" (300mm)

$3.95
IN STOCK
Add To Cart

Premium Female/Male ‘Extension’ Jumper Wires - 40 x 12" (300mm)

OUT OF STOCK
Out Of Stock

Terminal block - 4 pin Euro-Style

$2.50
IN STOCK
Add To Cart
Soldering your own wires

I do recommend getting a set of color coded Stranded Core wire as this will make it easy to keep track of what wire is connected to what device.

Since some of the devices use more than just two wires to connect, like servos and NeoPixels, I like to use a small ribbon cable like the following two.
You will also want to have some Heat Shrink tubing on hand to be sure that you do not get a short circuit.

All the other stuff

I find it useful to have a few Break-apart Male Header strips on hand to make some quick connectors by soldering wired to the top of them.
You will also want a good wire stripper and a good soldering iron.

So that is it and we are ready to start wiring things together.
Smart Lights

For our Smart Lights we will be using NeoPixels so that we can make one long string of lights throughout our mini smart home.

Smart Lights are some of the most common things you will find in any Smart Home and they typically have the most controllable options. Most Smart Lights made by Hue and Lifx give you control over the color, brightness, on/off, and some animations.

To connect the NeoPixels, we will be using the 5V and GND NeoPixel terminals on the Adafruit Crickit controller board.

Rather than connect the signal wire to the same terminal, we will be connecting it to the outside header socket for the Feather connector so that our signal will connect to Pin 13 on the Huzzah.
Each light will need to be connected in a string of lights that will wind its way through the house. The nice thing about NeoPixels is that we only need three wires to go from one light to the next because we can target any single LED in the string and tell it what to do.

As for the individual NeoPixels, there are a few form factors that you can use depending on how fancy you want to get. I tend to just get a roll of the 30 LEDs per meter strips and cut each single LED off where indicated so I can solder wire between them. Then you also have plenty of NeoPixels for other projects. There are a few other NeoPixel form factors that I have listed below if you want.

It is a good idea to measure out the length of wire needed to get each NeoPixel to where it needs to go.
Solder the wires to each copper pad on the NeoPixels being sure that you have the output from one NeoPixel is connected to the input of the next NeoPixel.

When finished, attach the NeoPixels to the ceiling inside the Mini Smart Home where ever you think it should go. I used just a little hot glue, but regular glue or tape would work as well.

Other NeoPixels that you can wire together:
Adafruit NeoPixel LED Dots Strand - 20 LED 4" Pitch

OUT OF STOCK

NeoPixel Mini Button PCB - Pack of 5

$4.95
IN STOCK
Add To Cart

Flora RGB Smart NeoPixel version 2 - Pack of 4

$7.95
IN STOCK
Add To Cart

Breadboard-friendly RGB Smart NeoPixel - Pack of 4

$7.95
IN STOCK
Add To Cart
Smart Lock

For our Smart Lock, we will be using a servo to lock and unlock the door. This is not really how most of these devices work, but it will do the job for this mini smart home.

Micro servo

Micro Servo

You will want to connect this servo to Servo 1 on the Crickit.
To allow for the servo head to stick out through the door frame, I cut a hole in the wall next to where the door opens. Then you will want to attach the servo to the inside of the house so that the servo head sticks outside next to the door. Attaching one of the servo arms will let you lock the door when needed.

You will want to be sure to change the servo arm position so that it is horizontal when the door is locked and vertical when unlocked.
Smart Fan

Most Smart Homes have some sort of heating and air conditioning control. For this house we will be using a DC fan to get air flow into the house.

You will want to use a 5V fan like this one:

![Miniature 5V Cooling Fan for Raspberry Pi (and Other Computers)](image)

$3.50
IN STOCK
Add To Cart

The wiring is simple and you will just want to connect the Black wire to the Motor GND and Red wire to Motor 1.
I attached the fan to a window, but you can put your fan wherever you want it to go.
Door Sensor

Magnetic contact switches are most used as a way to detect if a door or window is open. Basically one side has a Reed Switch in it and other side has a magnet in it. When the two side get close to each other, the two wires become connected and this can be read just like a binary button press using Arduino code.

This door sensor has two white wires but it doesn’t matter what wire you connect to what pin because the sensor basically just connects the two wires together when the magnet gets close to it.

Connect one of the White wires to the GND pin.

The other White wire should be connected to the Signal 5 pin on the Crickit.
Once wired in, attach the Door Sensor to the part of the door frame farthest from the door hinge. This will ensure that the door sensor has the best chance of detecting that the door is open. The other side to the door sensor is just a magnet, but due to the small scale of the mini smart home you will want to space it about 0.5 inches away from the other side of the sensor so that the two sides get farther away from each other faster.
Light Sensor

Systems like Home Assistant do tell you if it is day time or night time based on your physical location on the earth, but sometimes it is good to check just how much light you are actually getting on your home. This is typically part of a local weather station that can tell you if the sun is shining or it is overcast.

We will be using a photocell (light dependent resistor) to read the analog value of ambient light outside of your Mini Smart Home.

One side of the photocell will get connected to 3.3v in the Signal I/O section of the Crickit.

The other side of the photocell should be connected to the Signal 1 pin on the Crickit as well as to a 10K ohm Resistor.

The other end of the Resistor will connect to the GND pin on the Crickit.

I used a Tiny Breadboard (https://adafru.it/kft) to make this connection easy, but there are other ways to build this sensor if you want. Just be sure that the connections are correct.
You may want this to be on the roof of your Mini Smart Home so that you always know if the sun is out or not.
Motion Sensor

A passive infrared (PIR) sensor uses infrared light to detect movement in a room and are one of the most commonly used devices for human presence detection. These sensors look for a change in the amount of reflected infrared light and then send a binary signal to your microcontroller to indicate that the sensor detected movement. It can take these sensors a few seconds to reset before motion can be detected again, but they do work rather well.

The signal wire from the PIR sensor will be connected to Signal 3 on the Crickit and the Ground wire will need to be connected to GND 3.

This sensor requires 5V of power to operate, so we will want to connect the 5V pin on the sensor to the 5V terminal of the Drive section on the Crickit.
I like to mount this sensor so that it is facing into the Mini Smart Home and pointed slightly down. This will limit the likelihood that the sensor will trigger when things are moving outside the house.
The Code

We will be using the Adafruit Feather HUZZAH ESP8266 to control the Adafruit Crickit and connect over WiFi to our Home Assistant server. This will require a bit of Arduino programming but mostly we just need to get things set up so that we can flash the code to your HUZZAH with a few changes to add your WiFi information and MQTT server credentials. Here is a checklist to be sure that you are ready to go.

1. Install the Arduino IDE

   [Link]

2. Add hardware support for the ESP8266

   [Link]

3. You will also need to install a few Libraries:

   • PubSubClient library
   • Adafruit NeoPixel library
   • Adafruit_seesaw library

   [Link]

Once this is finished and you can download Arduino code to the Feather HUZZAH, we can get the final code updated and downloaded to your Smart Home.
Arduino Code

There is quite a bit of code that controls all of the devices connected to your Mini Smart Home, but all that really needs to be done is add your WiFi and MQTT information. The HuzzahCrickitIOT project contains two files, HuzzahCrickitIOT.ino has the main code and config.h for the network settings.

You will want to download the link below to your Arduino folder so that we can make edits and upload.

https://adafruit.it/FY2

Open the HuzzahCrickitIOT.ino file in the Arduino IDE and click the config.h tab at the top so that you can add your WiFi and MQTT information.

All text in quotations will need to be updated with your information.

- You will need to replace Your SSID with the SSID for your WiFi network.
- Replace Your Password should be replaced by the password used to connect to your WiFi.
- For the MyMDTTadress, this will be the IP address of the Raspberry Pi that you set up with Home Assistant.
- The space for yourMQTT will be replaced with the user name from your MQTT setup.
- Finally, you will want to add the MQTT password to replace yourMQTTpass.

Once finished, connect your HUZZAH Feather to your computer and download the code. When download is finished, click on the magnifying glass icon on the top right corner to open the Serial Monitor. If all went well you should see it connect and start to stream data.
Connecting to...

WiFi connected
IP address:
192.168.1.65
Crickit started
Attempting MQTT connection...connected
Lux = 13
13
0 - 0
Motion Sensor = STILL
Door = CLOSED
Lux = 13
13
0 - 0
The last step is to tell Home Assistant how to interact with our new MQTT devices. Even though we only used one microcontroller, the HUZZAH Feather, the MQTT feeds will allow us to separate out control over all of our devices and sensors. Thankfully Home Assistant has an extensive lineup of different device types formatted for MQTT. There is one for an MQTT Lock, MQTT Binary Sensor, and even a special one for MQTT RGB Lights. Templates exist for all of these and unless you changed any of the Arduino code in the HuzzahCrickitIOT.ino file relating to the Feed names, you can just use the code below.

You will want to copy the following code and use it to replace all of the code in the configuration.yaml file using the Home Assistant Configurator tool (https://adafru.it/FXM).

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

# Uncomment this if you are using SSL/TLS, running in Docker container, etc.
# http:
#   base_url: example.duckdns.org:8123

# Text to speech
tts:
  - platform: google_translate

sensor:
  # Weather prediction
  - platform: yr
  - platform: mqtt
    name: "Light Sensor"
    state_topic: "house/lux"
    unit_of_measurement: 'Lux'
    icon: mdi:brightness-6
  - platform: mqtt
    name: "Door Sensor"
    state_topic: "house/door"
    icon:mdi:door
  - platform: mqtt
    name: "Motion Sensor"
    state_topic: "house/motion"
```

© Adafruit Industries
fan:
- platform: mqtt
  name: "Fan"
  state_topic: "house/fan"
  command_topic: "house/fan"
  speed_state_topic: "house/fan/speed"
  speed_command_topic: "house/fan/speed"
  qos: 0
  payload_on: "ON"
  payload_off: "OFF"
  payload_low_speed: "low"
  payload_medium_speed: "medium"
  payload_high_speed: "high"
  speeds:
    - low
    - medium
    - high

light:
- platform: group
  name: All Lights
  entities:
    - light.light_1
    - light.light_2
    - light.light_3
    - light.light_4
    - light.light_5
- platform: mqtt
  name: "Light 1"
  state_topic: "house/led/one"
  command_topic: "house/led/one"
  brightness_state_topic: "house/led/one/brightness"
  brightness_command_topic: "house/led/one/brightness"
  rgb_state_topic: "house/led/one/color"
  rgb_command_topic: "house/led/one/color"
  on_command_type: first
  state_value_template: "{{ value_json.state }}"
  brightness_value_template: "{{ value_json.brightness }}"
  rgb_value_template: "{{ value_json.rgb | join(',',) }}"
  qos: 0
  payload_on: "ON"
  payload_off: "OFF"
  optimistic: false
- platform: mqtt
  name: "Light 2"
  state_topic: "house/led/two"
  command_topic: "house/led/two"
  brightness_state_topic: "house/led/two/brightness"
  brightness_command_topic: "house/led/two/brightness"
  rgb_state_topic: "house/led/two/color"
  rgb_command_topic: "house/led/two/color"
  on_command_type: first
  state_value_template: "{{ value_json.state }}"
  brightness_value_template: "{{ value_json.brightness }}"
  rgb_value_template: "{{ value_json.rgb | join(',',) }}"
  qos: 0
  payload_on: "ON"
payload_off: "OFF"
optimistic: false
- platform: mqtt
  name: "Light 3"
  state_topic: "house/led/three"
  command_topic: "house/led/three"
  brightness_state_topic: "house/led/three/brightness"
  brightness_command_topic: "house/led/three/brightness"
  rgb_state_topic: "house/led/three/color"
  rgb_command_topic: "house/led/three/color"
  on_command_type: first
  state_value_template: "{{ value_json.state }}"
  brightness_value_template: "{{ value_json.brightness }}"
  rgb_value_template: "{{ value_json.rgb | join(',') }}"
  qos: 0
  payload_on: "ON"
  payload_off: "OFF"
  optimistic: false
- platform: mqtt
  name: "Light 4"
  state_topic: "house/led/four"
  command_topic: "house/led/four"
  brightness_state_topic: "house/led/four/brightness"
  brightness_command_topic: "house/led/four/brightness"
  rgb_state_topic: "house/led/four/color"
  rgb_command_topic: "house/led/four/color"
  on_command_type: first
  state_value_template: "{{ value_json.state }}"
  brightness_value_template: "{{ value_json.brightness }}"
  rgb_value_template: "{{ value_json.rgb | join(',') }}"
  qos: 0
  payload_on: "ON"
  payload_off: "OFF"
  optimistic: false
- platform: mqtt
  name: "Light 5"
  state_topic: "house/led/five"
  command_topic: "house/led/five"
  brightness_state_topic: "house/led/five/brightness"
  brightness_command_topic: "house/led/five/brightness"
  rgb_state_topic: "house/led/five/color"
  rgb_command_topic: "house/led/five/color"
  on_command_type: first
  state_value_template: "{{ value_json.state }}"
  brightness_value_template: "{{ value_json.brightness }}"
  rgb_value_template: "{{ value_json.rgb | join(',') }}"
  qos: 0
  payload_on: "ON"
  payload_off: "OFF"
  optimistic: false

lock:
- platform: mqtt
  name: Frontdoor
  state_topic: "house/lock"
  command_topic: "house/lock"
  payload_lock: "LOCK"
  payload_unlock: "UNLOCK"
  optimistic: true
  qos: 1
If there are no code errors, restart Home Assistant using the Server Control from the Configuration menu to update with these new devices.
Using Home Assistant

There are a lot of things that you can do with Home Assistant now that everything is setup. Here are a few basic things to get you started.

Add Devices to the User Interface

Home Assistant uses something called Lovelace to handle the UI. It makes things run fast but it is also easy to edit in real time.

Automation and Service calls

The key to any Smart Home is good Automations. Now we will get into how to create a simple Automation and making things happen.

Doing more

Honestly there are so many things that can be done with Home Assistant and it really all depends on how far you want to take it. There are many examples of massive UI hacks and elaborate automation sequences, but most of that is done using YAML and Python. I recommend sticking with the web interface if you are just starting out and you will find plenty of things to do.

When you are ready, the main Home Assistant website is a good resource on how to do custom edits, make your own integration app, creating a new card for Lovelace, or even using data templates. Here are a few links to help get you well into hacking your Home Assistant.
MQTT Feeds

All MQTT Feeds used for this project

The nice thing about MQTT is that you can use it any way you want so long as you use consistent Feed names and know what data to send. So here is a list of all device Feeds and what data they except.

MQTT Feeds:

1. Door Lock
   - house/lock
     - State: LOCK or UNLOCK

2. Window Fan
   - house/fan
     - State: ON or OFF
   - house/fan/speed
     - State: 0 to 255

3. RGB LED 1
   - house/led/one
     - State: ON or OFF
   - house/led/one/brightness
     - State: 0 to 255
   - house/led/one/color
     - State: three sets of 0 - 255 numbers separated by commas.
     - Template: “red,green,blue”
     - Example: 20,150,255

4. RGB LED 2
   - house/led/two
     - State: ON or OFF
   - house/led/two/brightness
     - State: 0 to 255
   - house/led/two/color
     - State: three sets of 0 - 255 numbers separated by commas.
     - Template: “red,green,blue”
     - Example: 20,150,255

5. RGB LED 3
   - house/led/three
- State: ON or OFF
  - house/led/three/brightness
    - State: 0 to 255
  - house/led/three/color
    - State: three sets of 0 - 255 numbers separated by commas.
    - Template: “red,green,blue”
    - Example: 20,150,255

6. RGB LED 4
   - house/led/four
     - State: ON or OFF
   - house/led/four/brightness
     - State: 0 to 255
   - house/led/four/color
     - State: three sets of 0 - 255 numbers separated by commas.
     - Template: “red,green,blue”
     - Example: 20,150,255

7. RGB LED 5
   - house/led/five
     - State: ON or OFF
   - house/led/five/brightness
     - State: 0 to 255
   - house/led/five/color
     - State: three sets of 0 - 255 numbers separated by commas.
     - Template: “red,green,blue”
     - Example: 20,150,255

8. Light Level Sensor
   - house/lux
     - State: 0 to 6000

9. Motion Sensor
   - house/motion
     - State: MOVE or STILL

10. Door Sensor
    - house/door
      - State: OPEN or CLOSED