Lab 5 – Smart Home Controller

Online Links

This lab is available on my personal website at: http://AaronNelson95.com/IT441Lab5.php
The code is also available on my GitHub repository at https://github.com/AaronNelson95/IT441

Objective

The purpose of this lab is to learn how to incorporate MQTT data from Arduino devices with an IoT controller. The Home Assistant controller will be saved to a Raspberry Pi and different integrations will be set up to view data. The value of MQTT topics will be recorded as sensors, and templates will be used to transform the way the values are worded into easily read descriptions. Cards will be added to the Home Assistant main screen to view MQTT data at a glance. Automations will be used to integrate home MQTT data with external services, such as IFTTT. This lab will help one learn:

- How to install Hassbian and Home Assistant onto a Raspberry Pi
- How to subscribe to MQTT topic values in Home Assistant as sensors
- How to read and display sensor values at a glance
- How to use IFTTT Webhooks to run custom-made Applets using automations
- How to connect Arduino devices to outside services with IFTTT

Materials

To complete this lab, you will need the following materials:

- A Raspberry Pi with a bootable SD card
- A Raspberry Pi power supply
- A computer with a SD card reader and a MQTT broker installed
- The three Arduino devices made in Lab 4

References

The following links may be helpful throughout this lab:

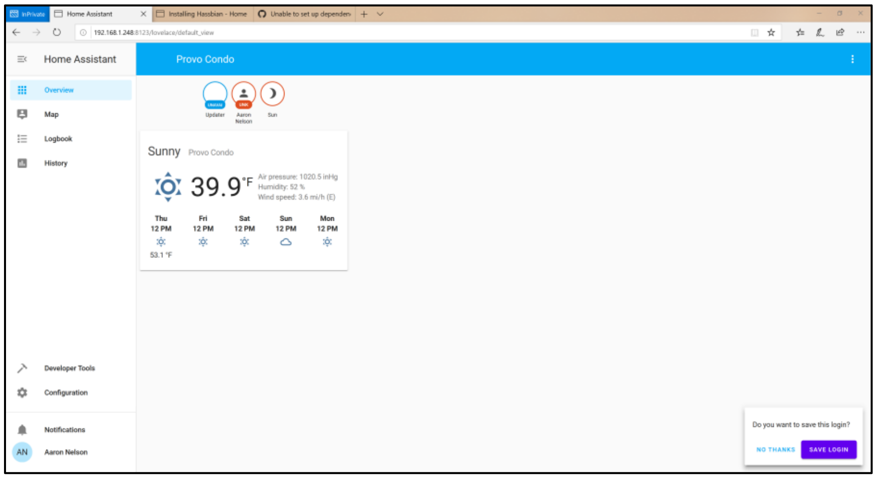
- http://aaronnelson95.com/IT441Lab4.php - Lab 4 Instructions on how to set up the Mosquitto MQTT broker on a computer and how to develop 3 Arduino devices to communicate with MQTT
- https://www.home-assistant.io/docs/installation/hassbian/installation/ - Steps on how to install Hassbian on a Raspberry Pi (as well as a link to the latest Hassbian image)
- https://www.juanmtech.com/install-home-assistant-simple-steps/ - Another helpful guide on installing Hassbian. This one includes more information about personally configuring the image and gaining access to it
- https://www.home-assistant.io/integrations/switch.mqtt/ - Adding MQTT capabilities to Home Assistant
- https://www.home-assistant.io/docs/configuration/devices/ - Saving MQTT topics as sensors in Home Assistant
- https://www.home-assistant.io/integrations/template/ - Working with templates to alter the way MQTT values are written
- https://www.home-assistant.io/integrations/ifttt/ - Creating IFTTT automations by sending a webhook event
- https://ifttt.com/maker_webhooks - The IFTTT Webhooks service, used to create applets with home assistant sensors

Procedures

1. Setup the Door Opener, Echo Sensor, and Stoplight devices from Lab 4. In addition, you should have a Mosquitto MQTT server running. Verify that these devices work together and that they publish data to the following MQTT topics. This information will be used and read by Home Assistant:

Topic	Message	Description	Published By	Subscribed By
device/doorSwitch	Activated	Device came online	Door Opener	
device/echoSensor	Activated	Device came Online	Echo Sensor	
device/stoplight	Activated	Device came online	Stoplight	
garage/doorSwitch	0	Magnets are disconnected	Door Opener	Echo Sensor
	1	Magnets are connected		
garage/echoSwitch/color	g	Distance read is > 35cm	Echo Sensor	Stoplight
	y	Distance read is 20-35cm		
	r	Distance read is 10-20cm		
	b	Distance read is 0-10cm		
	o	Received "0" from garage/doorSwitch		

2. Install the Hassbian operating system by following the instructions at https://www.home-assistant.io/docs/installation/hassbian/installation/.
   a. Download the latest Hassbian image and flash the image onto an empty SD card.
   b. Insert the card into a Raspberry Pi and either plug it into a monitor with a keyboard and ethernet cable or configure your Pi's wireless network by altering a file in the boot partition (as explained in the guide).
   c. Wait several minutes for the latest version of Home Assistant to be installed. When prompted, you can login to Hassbian with the default username of "pi" and password of "raspberry" (or you can SSH right into the Pi with its IP address if you are doing this without extra peripherals). When the installation is finished, you should be able to see some files populated in "/home/homeassistant/.homeassistant/".
   d. Type in "$ sudo raspi-config" to change the default password. You can also add a wifi network here and adjust other settings. If you would like to set multiple wifi networks for your device (I would highly recommend running it off of a hotspot hosted by the same computer your MQTT server is on), you can set those in the "/etc/wpa_supplicant/wpa_supplicant.conf" file.
   e. Using a device on the network, open a web browser and go to "http://<YOUR_PI_IP_ADDRESS>:8123". If Home Assistant appears, set an initial account up to log in with. If it does not appear, follow the troubleshooting steps in the Home Assistant instruction guide. Once you are in, you should see the main Home Assistant page, where you will add cards to for monitoring MQTT data.



Sunny Provo Condo
39.9 °F
Air pressure: 1020.5 inHg
Humidity: 52 %
Wind speed: 3.6 mi/h (E)
Thu 12 PM 53.1 °F
Fri 12 PM
Sat 12 PM
Sun 12 PM
Mon 12 PM
Do you want to save this login?

Seeing the Home Assistant page come up is an indication that the installation worked

3. Add the MQTT Integration to Home Assistant to be able to work with your Mosquitto Server. This can either be done through the UI, or by editing the configuration yaml file for Home Assistant. Change this file with the command "$ sudo nano /home/homeassistant/.homeassistant/configuration.yaml" (we will be editing this file frequently throughout the lab). Simply insert the code below to connect to Mosquitto hosted at 192.168.137.1 (replace with your server's IP):

```
# Sets up the MQTT broker, which in this case is my computer's Mosquitto server
mqtt:
    # The IP of the Mosquitto server
  broker: 192.168.137.1
    # What the Home Assistant client should check in as
  client_id: homeassistant
  discovery: true
    # The MQTT Topic to watch (subscribe to)
  discovery_prefix: garage
```

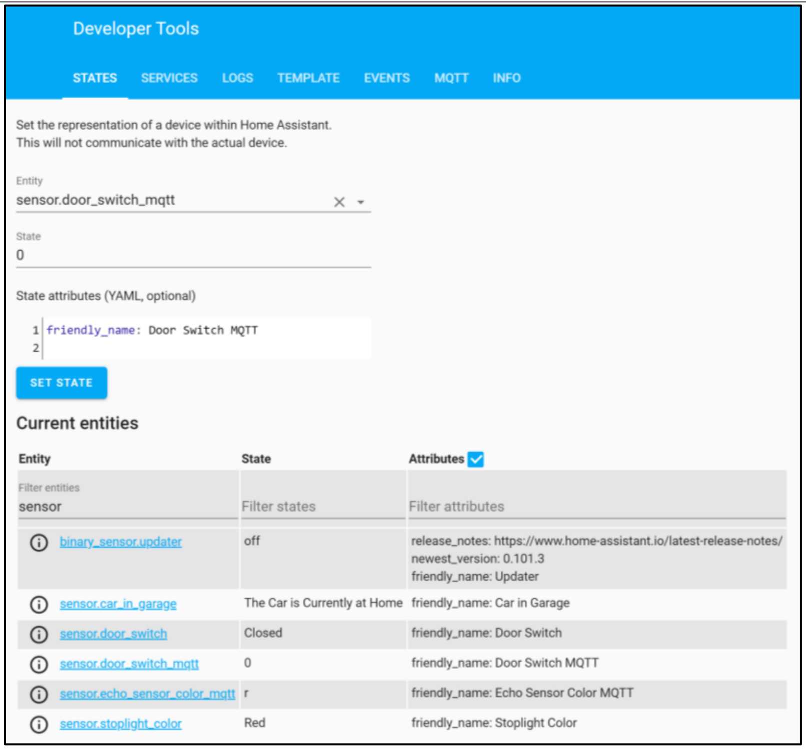
Afterwards, restart home assistant with the command "$ sudo service home-assistant@homeassistant restart" (this will also be done frequently, so memorize this command!). In the browser, under the Configuration tab and then Integrations, you should now see a listing for "MQTT: configuration.yaml". You can test that this works by going to the Developer Tools tab, selecting MQTT, and then listen to the topic "#", which should then populate when you use your Arduino devices.

4. Now that Home Assistant is connected to the MQTT server, the topics will be setup as different sensors. In Home Assistant, once a topic receives a different value, the sensor will reflect and also contain that value. This is done by editing the configuration.yaml file again and then restarting the Home Assistant service. You will add the following for each topic you wish to watch:

```
sensor:
    # Directly returns the value of garage/doorSwitch (a 0 or 1)
  - platform: mqtt
    name: "Door Switch MQTT"
    state_topic: "garage/doorSwitch"
```

5. Because the MQTT sends simple values, it may be beneficial to create another sensor that uses a template to assign friendly names to these topics. This will create another sensor device, and will not alter the original MQTT feed. To test the syntax of any template, you can go to the Developer Tools tab of Home Assistant and then select Template. You can examine if your code works like you intend. In addition, under the Developer Tools, you can also select the States tab to view your sensor values. As you adjust MQTT values, the state column for the sensor entity will change here. Create an easy to read sensor by following this example:

```
sensor:
    # A template "adjusts" the values from a MQTT sensor to make it more readable
      and user friendly
    # This makes the Door Switch MQTT either return "Opened" or "Closed"
  - platform: template
    sensors:
      door_switch:
        friendly_name: "Door Switch"
        value_template: >-
          {% if is_state('sensor.door_switch_mqtt', '1') %}
            Opened
          {% else %}
            Closed
          {% endif %}
```



Set the representation of a device within Home Assistant.
This will not communicate with the actual device.

Entity: sensor.door_switch_mqtt

State: 0

State attributes (YAML, optional)

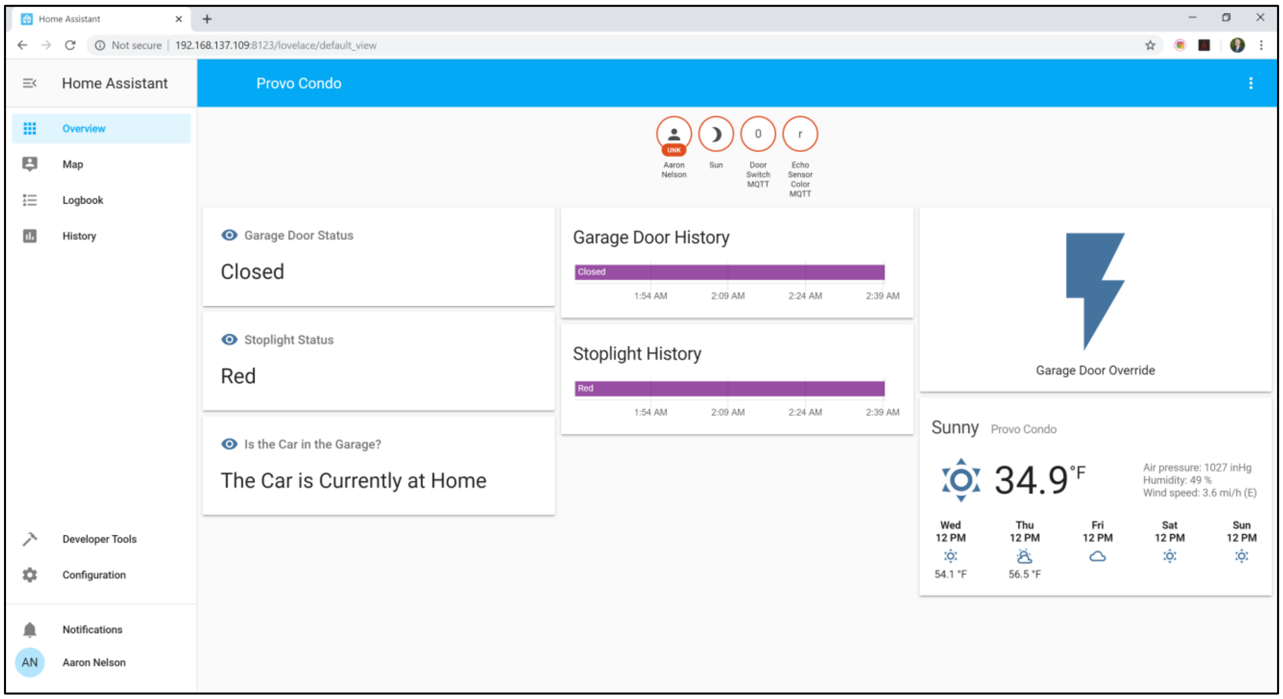
```
friendly_name: Door Switch MQTT
```

Current entities

Entity	State	Attributes
sensor	Filter states	Filter attributes
binary_sensor.updater	off	release_notes: https://www.home-assistant.io/latest-release-notes/ newest_version: 0.101.3 friendly_name: Updater
sensor.car_in_garage	The Car is Currently at Home	friendly_name: Car in Garage
sensor.door_switch	Closed	friendly_name: Door Switch
sensor.door_switch_mqtt	0	friendly_name: Door Switch MQTT
sensor.echo_sensor_color_mqtt	r	friendly_name: Echo Sensor Color MQTT
sensor.stoplight_color	Red	friendly_name: Stoplight Color

Viewing MQTT Sensor Values in Developer Tools -> States

6. Now that we have sensors reading data from the MQTT broker, we can add cards to the Overview tab of Home Assistant to view this data easily at a glance. In this tab, click the three dots in the top right corner and select the option to configure the UI. In the bottom right, click the + button to add a card. To display the sensor data plainly, select the Sensor card. The Entity will be the name of your easy to read sensor and you can adjust other properties here. Feel free to test out other cards as well, such as the history graph. When you make changes and add cards here, they are recorded to a JSON stored in /home/homeassistant/.homeassistant/.storage/lovelace. Alternatively, you can manually insert and adjust card data in the code here.



Garage Door Status
Closed

Stoplight Status
Red

Is the Car in the Garage?
The Car is Currently at Home

Home Assistant Overview showing Sensor Values, histories, and other cards

7. Experiment with other sensors devices and cards. For example, I used a “switch” to listen to a MQTT topic. It also provides a button I can push to post a value into that topic as well, acting as a “manual override” mode for my garage door switch. I also created a sensor with a template to show whether or not my car is parked in the garage based on sensor readings.

8. Now we will integrate our sensors to work with IFTTT. IFTTT allows you to connect multiple devices, applications, and other technologies together with applets. You can add different services you would like to connect and then follow the pattern for creating applets: if a trigger happens on a service, then do a specified action on another service. If you have not done so already, create an account through IFTTT and then add the Webhooks service (https://ifttt.com/maker_webhooks). Click on the Documentation link near the top of the service page and copy your key value. Back in your configuration.yaml file, create an integration for IFTTT. Once again, restart the Home Assistant service.

```
# To use the IFTTT webhook integration, your key should be placed here (obtained
      from https://ifttt.com/maker_webhooks)
ifttt:
  key: <IFTTT KEY GOES HERE>
```

9. When Home Assistant calls the IFTTT Webhook, it will send an event over as well as optional values you can specify. The event will be a unique name so IFTTT will know what applet you wish to call. You can plan out your event names and automations you wish to incorporate. For example, I used an event “GarageLeftOpen” and “CarArrived”. In the IFTTT website, create a new applet with the If This portion of the code receiving a web request from your event name. Feel free to run any action you would like. As some examples, you can have IFTTT send you an email or a text if your garage door is left open. You can also have a log file tracked in a Google Sheet for times when the car arrives home. Once you have developed your applets, it is time to setup an automation to trigger the events.

10. You can create automations by altering the /home/homeassistant/.homeassistant/automations.yaml file or through the Home Assistant browser page (which we will use). First, test that your IFTTT integration works. Go to the Developer Tools tab then select Services. Here, pick the service “ifttt.trigger” and enter in service data that calls your event: “{"event" : "CarArrived"}”. Select the call service button and your IFTTT Applet should run. When you are satisfied with your trigger data here (you can also add fields for the values), copy it for your automations. Click on the Configuration tab and then select Automation. Here, you can give your automation a name and then specify the trigger to cause the event to happen. For example, you can have the automation respond when there is a state change made to the door switch sensor. When the value of the state changes to “Opened” for 1 minute (specified as “00:01:00 in the for field), it will then call an action. In the Actions section, you can have it call the ifttt.trigger service. The service data would be the service data you tested earlier. After saving the automation, when the garage door remains opened for a minute, the IFTTT applet will run. You can now use Home Assistant and IFTTT to manage and automate your Arduino devices.

Thought Questions

1. Which version of Home Assistant did you choose to install? (Docker-based Hass.io, Raspbian-based Hassbian, or install it yourself?) Why did you choose this particular version?

   I installed the Raspbian-based Hassbian image to run Home Assistant. I was unfamiliar with Docker containers and I was eager to jump right into using the Home Assistant application, rather than waiting to learn how docker works. Hassbian was advertised as being extremely easy to setup and being able to run right out of the box. Plus, I had access to a fresh SD card to use and image. However, when I began this project, I realized that I made a very big mistake in choosing to install Hassbian *this week*. I went through the steps and waited for Home Assistant to install, but I was not able to pull it up on the web browser. I reimaged my SD card and tried again, still with no luck. I decided to start digging deeper, and I tried older Hassbian images that others have used successfully before. I did not have similar luck. After a very long and trying time of testing, I learned that there was a bug in the latest version of Home Assistant and that lots of people were also having the same issue just this week as well: https://github.com/home-assistant/home-assistant/issues/28361. Basically, the latest version of Home Assistant doesn't properly install its frontend, the webpage that we view in the browser. There wasn't much I could do as *each* version of Hassbian is setup to download the *latest* version of Home Assistant as soon as it connects to the internet. I tried the suggestions others tried- such as to downgrade to version 0.100.3. I used multiple methods to install a lower version, but even though the installer said it successfully downgraded, my image was still stubbornly using version 0.101.3. I also tried manually installing the missing frontend file, but it still wasn't picked up immediately. Eventually, after I was ready to quit and move onto the Docker version (which probably would also have had an issue because it is Home Assistant that was having the problem), Hassbian finally started to work and run properly! I was thrilled to begin setting up Home Assistant, that is until I noticed that once I signed out of it, I could never use my same credentials to log back in… Probably associated with another bug, I tried enabling the legacy password and trusted networks, but everything was still having an issue. It wasn't until I added a single, "auth:" to my configuration.yaml, that I was finally able to log in properly to Home Assistant. All in all, the quick Hassbian setup that should have taken half an hour, took me about 12 hours just to set up and troubleshoot. I still think Hassbian would be an easier version to install and use, but I was just caught trying to do it at the wrong time.

2. How should you decide which logic to perform in Home Assistant versus coding the logic directly into the devices? What guiding principles would you establish for future devices?

   Now that we have used Home Assistant and seen the automations and logic it can perform, it would make sense to use our Arduino devices to just broadcast simple, basic information. In the future, I think it would be helpful to have our Arduino devices publish simple, small values that would be easy to be universally read by many other devices. Right now, my echo sensor device performs logic within its own program. After obtaining distance readings, it calculates the color another device should light up. It then posts the color information into a MQTT topic, where the stoplight reads it and runs a quick script. This works great on a small scale, only when those two devices are communicating with each other, but if you were to add another device which would like to see the distance data, the echo sensor would be completely worthless or would need to be altered to handle publishing to multiple topics. Instead, we can use the echo sensor to simply publish its distance values. Home Assistant can then receive these values in a sensor and then templates can be written to manipulate the format of this data to customize it specifically for other devices. The echo sensor, and other devices, do not need to perform their own logic, and as a result, they will be compatible in many new ways for a larger variety of devices.

3. What features do you like the most about Home Assistant?

I greatly enjoyed the cards and automations parts of home assistant. They took me quite some time to figure out, but I thought it was neat being able to see my MQTT values automatically changing right away on the screen, where I could also see a graph of these values over time. Templates were tricky at first, but it was cool to see how I could use them to manipulate the data and contents of the sensor's values. The switch button was also neat, where I could both see the current value of a state, and change it, at the click of a button. It all seemed very practical. My favorite part of the lab was incorporating IFTTT with automations. I've been a huge fan of IFTTT for several years now, as I've always been fascinated on how it can expand the services and apps I already know and love. The capabilities to combine services into applets are endless! It was fun learning how to use Home Assistant sensors to act as an IFTTT trigger, because now I can connect my home-focused devices with the wide variety of features available on the Internet. I can now receive notifications for things only seen on my home network. I can also log data quickly and easily and view this information anywhere. I can potentially incorporate features of IFTTT to activate a device when my location approaches my home. There are so many cool things you can do with IFTTT, and now I have many more possibilities I can explore by incorporating it with Home Assistant automations.

From a user experience standpoint, I think Home Assistant is a great way to manage your devices. Rather than just looking at basic text or command prompt information go back and forth, you can assign flashy buttons and features. Our Arduino devices can do some really cool things, but unfortunately, many non-technical people may be too afraid that they may mess things up by using them. By setting up our devices through an interactive page like Home Assistant provides, family members can feel a lot more comfortable with controlling the devices around their home. Home Assistant is also very customizable with the information you choose to display as well, so you can create a screen that only shows devices and features that pertain to an individual. Home Assistant puts a nice look to the technical side of home automation, and this can be very appealing to a typical user.

4. Estimate the time you spent on this lab and report.

   The lab portion of this assignment took me a ton of time to complete… As I mentioned, I spent around 12 hours installing, troubleshooting, and exploring Hassbian and Home Assistant, *just to get it set up*. This was a new application to me as well, and I wasn't very familiar with yaml files, so it took a while for me to adjust to the new learning curve associated with Home Assistant. I was confused at many points on what to do next as well- like once I had MQTT integrated with the program, I had no idea that I needed to setup sensors next. Templates were easy to figure out when I was using the Developer Tools tab, but once I tried incorporating them into the yaml file, they just wouldn't work! I ran into many stumbling blocks, but as a result, I did learn a ton about Home Assistant and the features I specifically worked with. I also found out a ton about how Pi's, and how the Raspbian (Hassbian) operating system works. I spent well over 20 hours on this lab… In addition, I spent around 5½ hours on the report portion as well.

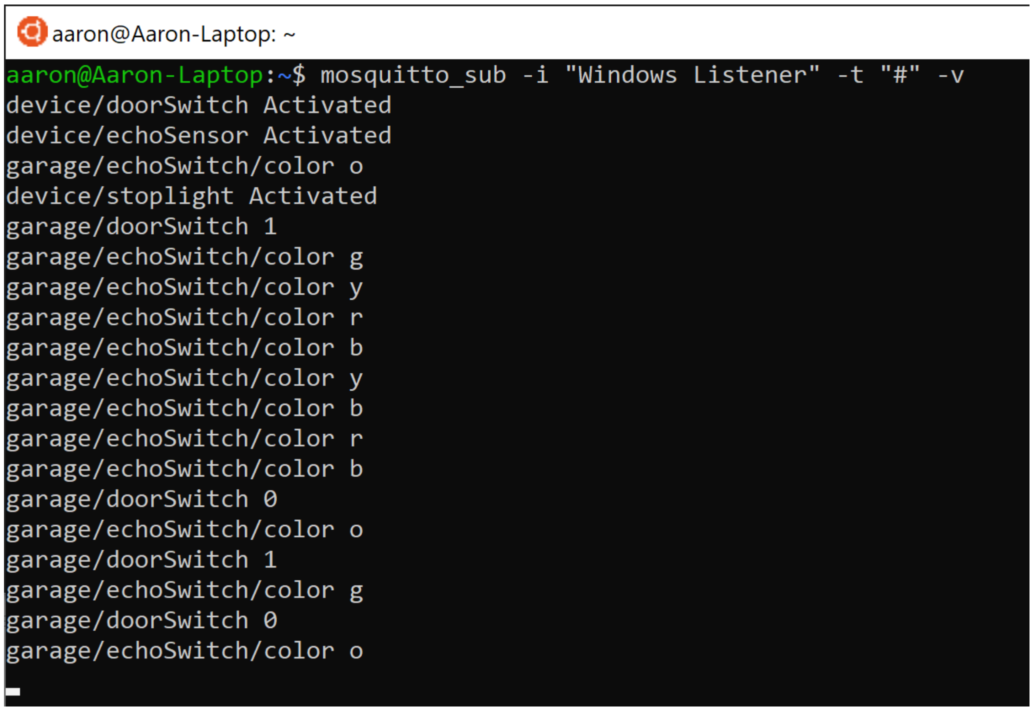
Certification of Work

I certify that this lab represents my own work. Any configuration code that I used was basic public knowledge provided from the Home Assistant documentation.

– Aaron Nelson

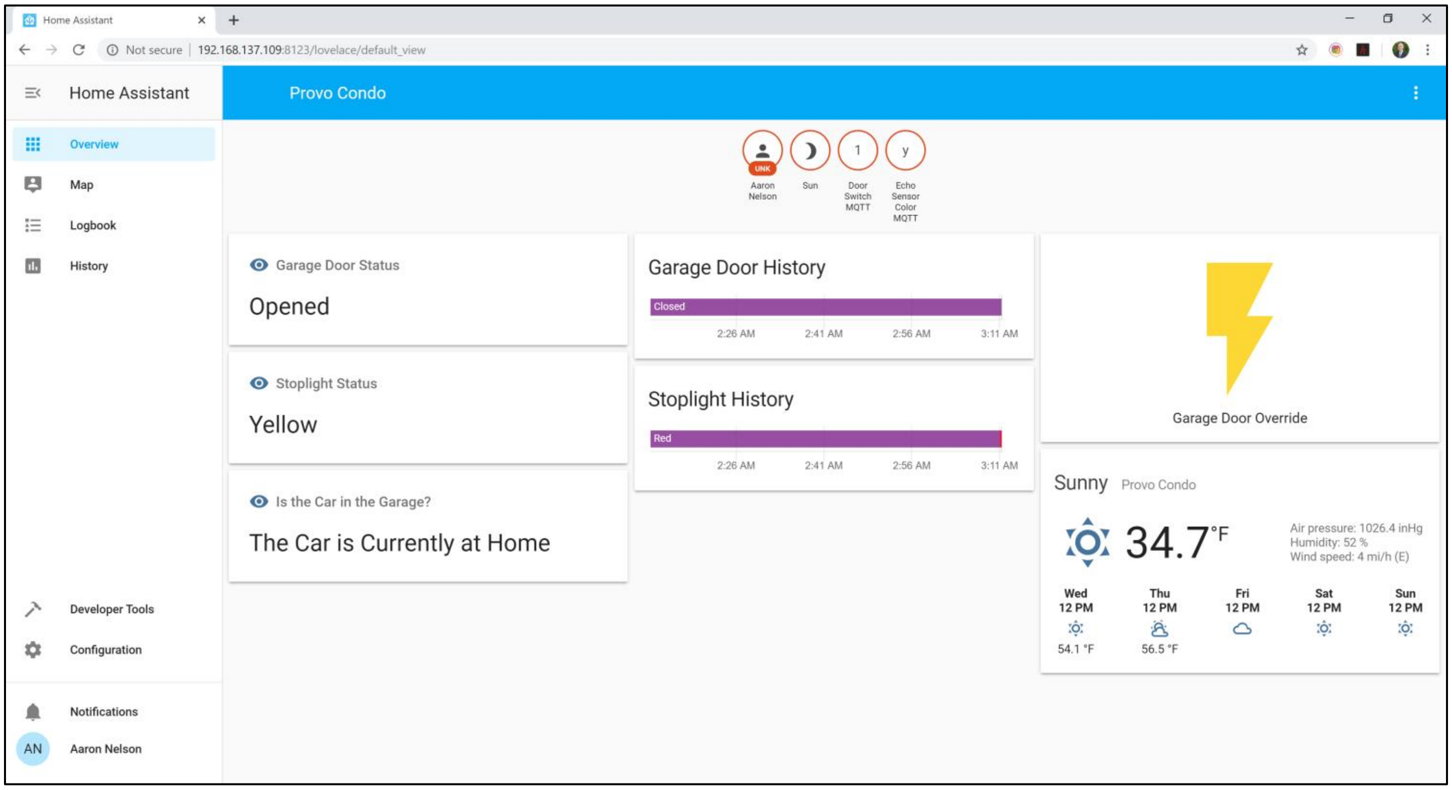
Appendix

Images of Final Product



```
aaron@Aaron-Laptop: ~
aaron@Aaron-Laptop:~$ mosquitto_sub -i "Windows Listener" -t "#" -v
device/doorSwitch Activated
device/echoSensor Activated
garage/echoSwitch/color o
device/stoplight Activated
garage/doorSwitch 1
garage/echoSwitch/color g
garage/echoSwitch/color y
garage/echoSwitch/color r
garage/echoSwitch/color b
garage/echoSwitch/color y
garage/echoSwitch/color b
garage/echoSwitch/color r
garage/echoSwitch/color b
garage/doorSwitch 0
garage/echoSwitch/color o
garage/doorSwitch 1
garage/echoSwitch/color g
garage/doorSwitch 0
garage/echoSwitch/color o
```

Our Mosquitto Server is properly setup from Lab 4. It listens to values passed by the Arduino devices



Home Assistant
Not secure | 192.168.137.109:8123/lovelace/default_view

Home Assistant — Provo Condo

Overview
Map
Logbook
History
Developer Tools
Configuration
Notifications
AN Aaron Nelson

Aaron Nelson (UNK) | Sun | Door Switch MQTT: 1 | Echo Sensor Color MQTT: y

Garage Door Status
Opened

Stoplight Status
Yellow

Is the Car in the Garage?
The Car is Currently at Home

Garage Door History
Closed
2:26 AM 2:41 AM 2:56 AM 3:11 AM

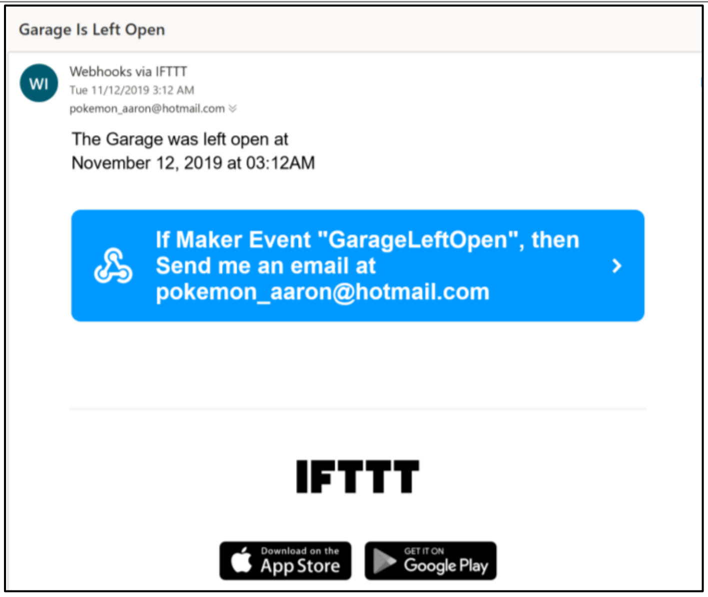
Stoplight History
Red
2:26 AM 2:41 AM 2:56 AM 3:11 AM

Garage Door Override

Sunny Provo Condo
34.7 °F
Air pressure: 1026.4 inHg
Humidity: 52 %
Wind speed: 4 mi/h (E)

Wed 12 PM	Thu 12 PM	Fri 12 PM	Sat 12 PM	Sun 12 PM
54.1 °F	56.5 °F			

When our devices are running and the distance from the sensor is relatively close, home assistant shows the current color of the stoplight, if the garage door is opened, and whether or not the car is likely still at home. Also, the garage door override button lights up, showing it is opened and that we can force it to close.



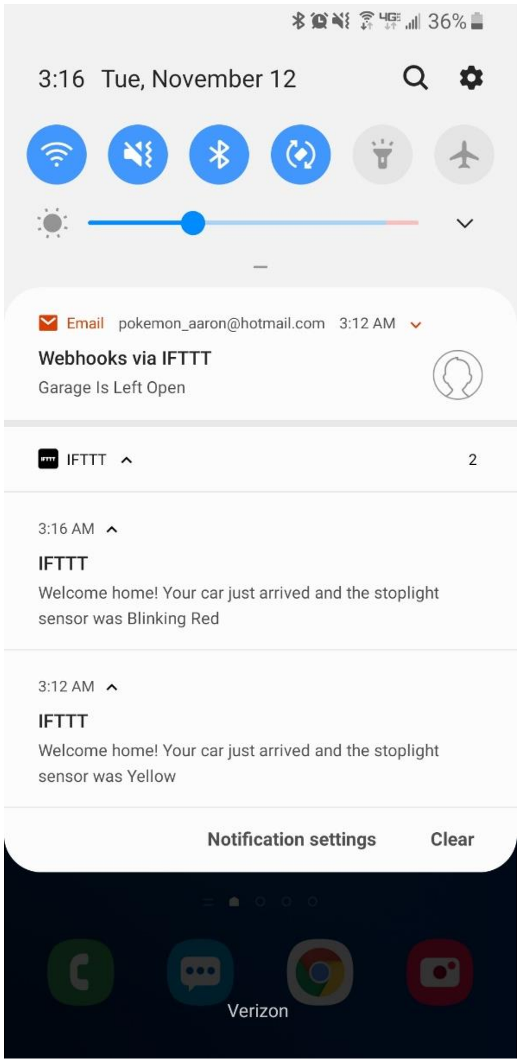
Garage Is Left Open

Webhooks via IFTTT
Tue 11/12/2019 3:12 AM
pokemon_aaron@hotmail.com

The Garage was left open at
November 12, 2019 at 03:12AM

If Maker Event "GarageLeftOpen", then Send me an email at pokemon_aaron@hotmail.com

I created an IFTTT automation so that if the garage door is opened for over a minute (the magnet is close together), then I will receive an email informing me that I may want to close the door



3:16 Tue, November 12

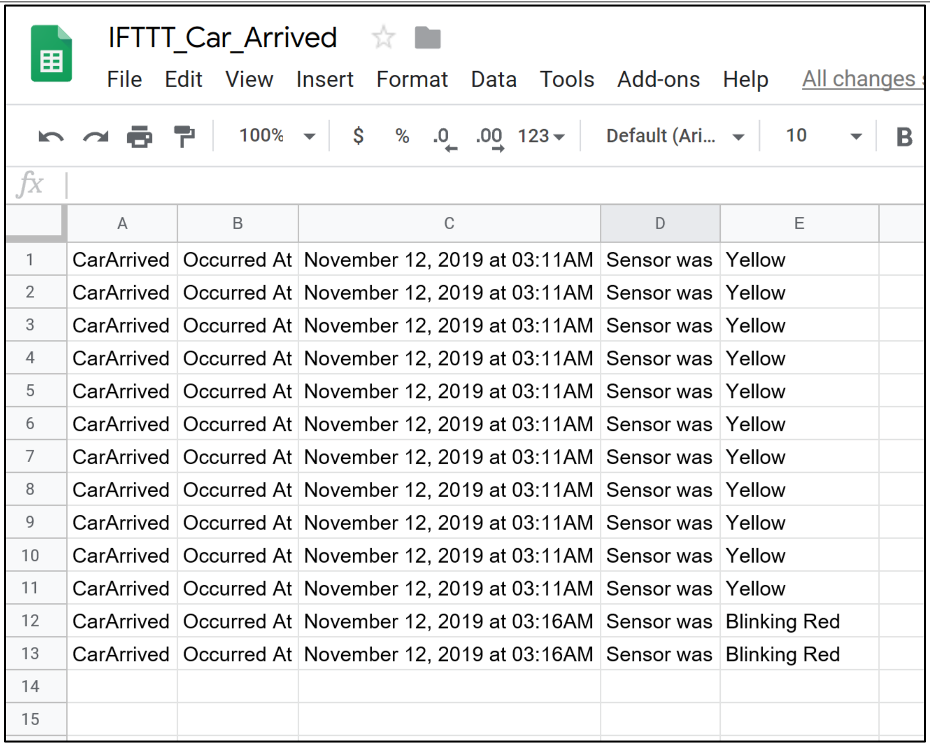
Email pokemon_aaron@hotmail.com 3:12 AM
Webhooks via IFTTT
Garage Is Left Open

IFTTT 2

3:16 AM
IFTTT
Welcome home! Your car just arrived and the stoplight sensor was Blinking Red

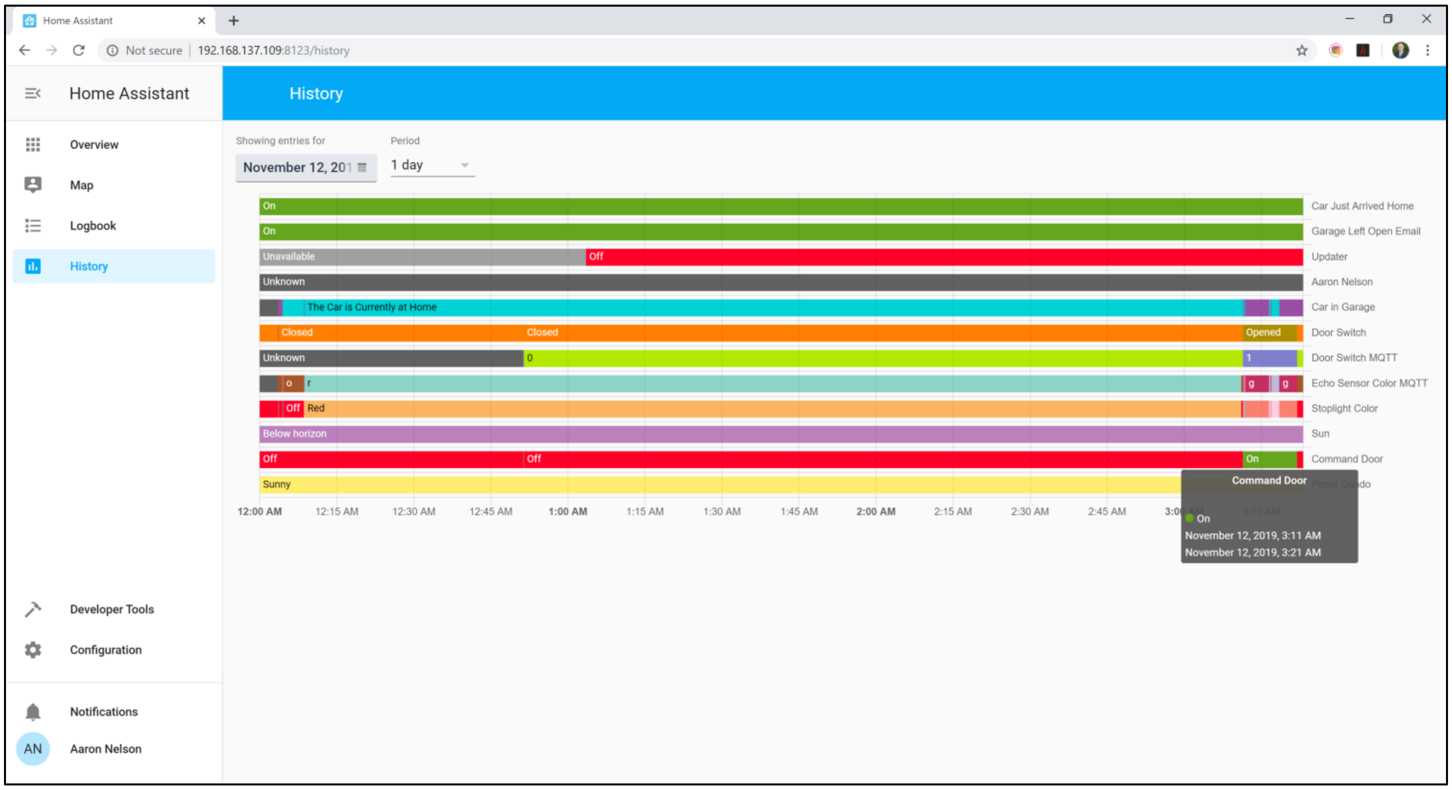
3:12 AM
IFTTT
Welcome home! Your car just arrived and the stoplight sensor was Yellow

I also receive phone notifications from IFTTT telling me when I arrive home and how close I was parked to the sensor


IFTTT_Car_Arrived
File Edit View Insert Format Data Tools Add-ons Help All changes
100% $ % .0 .00 123 Default (Ari... 10 B

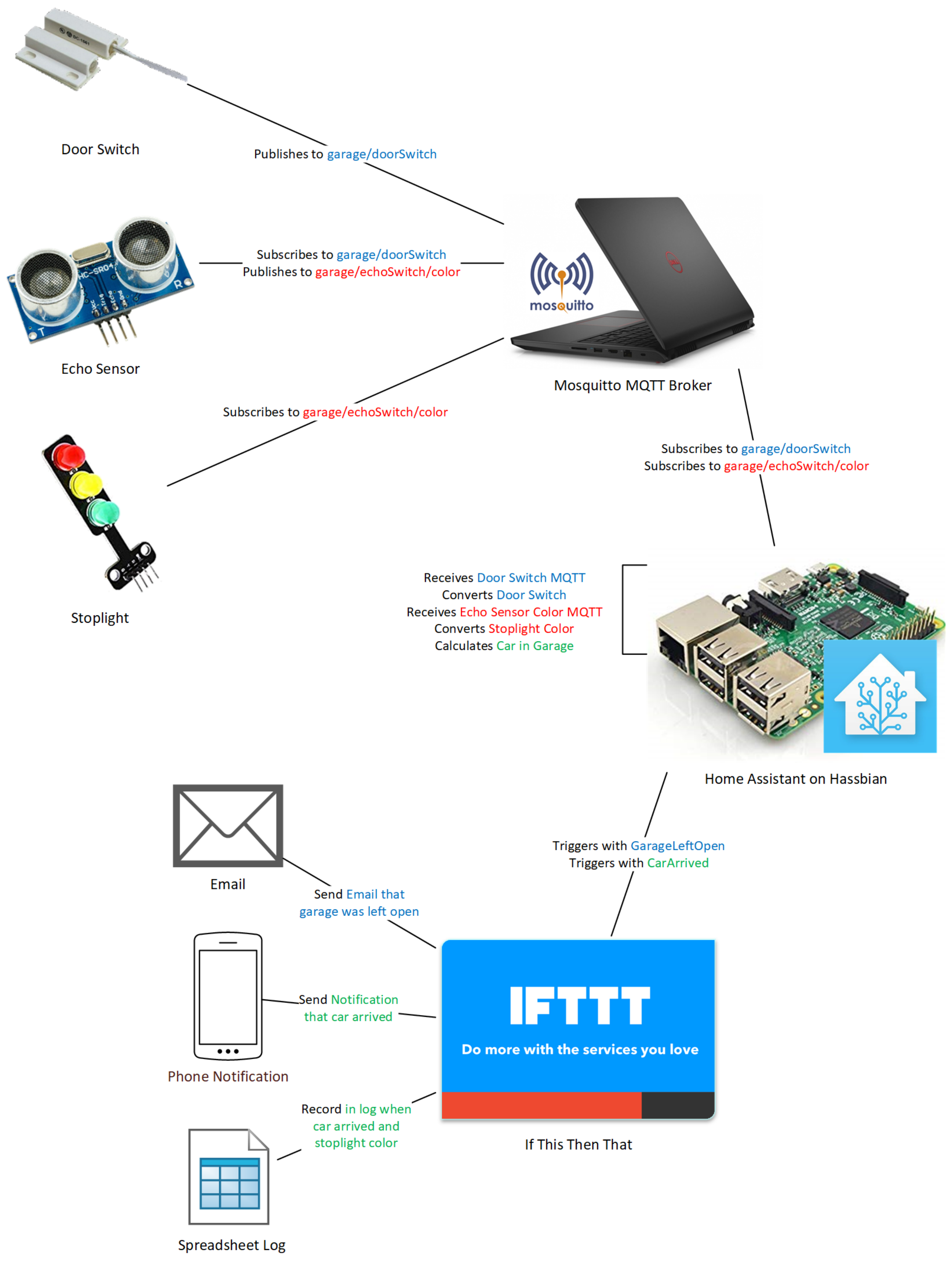
	A	B	C	D	E
1	CarArrived	Occurred At	November 12, 2019 at 03:11AM	Sensor was	Yellow
2	CarArrived	Occurred At	November 12, 2019 at 03:11AM	Sensor was	Yellow
3	CarArrived	Occurred At	November 12, 2019 at 03:11AM	Sensor was	Yellow
4	CarArrived	Occurred At	November 12, 2019 at 03:11AM	Sensor was	Yellow
5	CarArrived	Occurred At	November 12, 2019 at 03:11AM	Sensor was	Yellow
6	CarArrived	Occurred At	November 12, 2019 at 03:11AM	Sensor was	Yellow
7	CarArrived	Occurred At	November 12, 2019 at 03:11AM	Sensor was	Yellow
8	CarArrived	Occurred At	November 12, 2019 at 03:11AM	Sensor was	Yellow
9	CarArrived	Occurred At	November 12, 2019 at 03:11AM	Sensor was	Yellow
10	CarArrived	Occurred At	November 12, 2019 at 03:11AM	Sensor was	Yellow
11	CarArrived	Occurred At	November 12, 2019 at 03:11AM	Sensor was	Yellow
12	CarArrived	Occurred At	November 12, 2019 at 03:16AM	Sensor was	Blinking Red
13	CarArrived	Occurred At	November 12, 2019 at 03:16AM	Sensor was	Blinking Red
14					
15					

And a log of this data is recorded in a Google Sheet



I can also view a detailed history of my sensors and states in Home Assistant

System Diagram


Door Switch
Publishes to garage/doorSwitch
Echo Sensor
Subscribes to garage/doorSwitch
Publishes to garage/echoSwitch/color
mosquitto
Mosquitto MQTT Broker
Subscribes to garage/echoSwitch/color
Stoplight
Subscribes to garage/doorSwitch
Subscribes to garage/echoSwitch/color
Receives Door Switch MQTT
Converts Door Switch
Receives Echo Sensor Color MQTT
Converts Stoplight Color
Calculates Car in Garage
Home Assistant on Hassbian
Triggers with GarageLeftOpen
Triggers with CarArrived
Email
Send Email that garage was left open
Phone Notification
Send Notification that car arrived
IFTTT
Do more with the services you love
If This Then That
Record in log when car arrived and stoplight color
Spreadsheet Log

Code (this and lab 4 code is available at https://github.com/AaronNelson95/IT441)

Configuration Files

configuration.yaml (stored in /home/homeassistant/.homeassistant)

```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

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

homeassistant:
  # Specify what methods a user can use to log in
  auth_providers:
    - type: homeassistant
#    - type: legacy_api_password
      # Can be used if you would like a master password to be an authentication method
#      api_password: <UNIVERSAL PASSWORD HERE>
    - type: trusted_networks
      # If a device is attempting to log in from one of these networks, no password will
            be necessary
      trusted_networks:
        - 192.168.137.0/24
        - 127.0.0.1

# For some reason, I couldn't log in without this in the configuration file
auth:

# Sets up the MQTT broker, which in this case is my computer's Mosquitto server
mqtt:
    # The IP of the Mosquitto server
  broker: 192.168.137.1
    # What the Home Assistant client should check in as
  client_id: homeassistant
  discovery: true
    # The MQTT Topic to watch (subscribe to)
  discovery_prefix: garage

# These sensors either pull data directly from an MQTT Topic, or they use a template
      style to adjust the wording of a topic's value
sensor:
    # Directly returns the value of garage/doorSwitch (a 0 or 1)
  - platform: mqtt
    name: "Door Switch MQTT"
    state_topic: "garage/doorSwitch"
    # Directly returns the value of garage/echoSwitch/color (a g, y, r, b, or o)
  - platform: mqtt
    name: "Echo Sensor Color MQTT"
    state_topic: "garage/echoSwitch/color"
    # A template "adjusts" the values from a MQTT sensor to make it more readable and
      user friendly
    # This makes the Door Switch MQTT either return "Opened" or "Closed"
  - platform: template
    sensors:
      door_switch:
        friendly_name: "Door Switch"
        value_template: >-
          {% if is_state('sensor.door_switch_mqtt', '1') %}
            Opened
          {% else %}
            Closed
          {% endif %}

    # Another template to adjust values from a MQTT sensor
    # This returns the full color names of the stoplight's status
  - platform: template
    sensors:
      stoplight_color:
        friendly_name: "Stoplight Color"
        value_template: >-
          {% if is_state('sensor.echo_sensor_color_mqtt', 'g') %}
            Green
          {% elif is_state('sensor.echo_sensor_color_mqtt', 'y') %}
            Yellow
          {% elif is_state('sensor.echo_sensor_color_mqtt', 'r') %}
            Red
          {% elif is_state('sensor.echo_sensor_color_mqtt', 'b') %}
            Blinking Red
          {% else %}
            Off
          {% endif %}

    # This template detects how close the car is to guess if the car is parked in the
      garage (the last sensor reading was close by) or the car is away (the last reading
      was far away, so the garage is likely empty). It ignores the off state to continue
      showing the car's status while the door is shut.
  - platform: template
    sensors:
      car_in_garage:
        friendly_name: "Car in Garage"
        value_template: >-
          {% if is_state("sensor.echo_sensor_color_mqtt", "y") or
              is_state("sensor.echo_sensor_color_mqtt", "r") or
              is_state("sensor.echo_sensor_color_mqtt", "b") %}
            The Car is Currently at Home
          {% elif is_state("sensor.echo_sensor_color_mqtt", "g") %}
            The Car is Gone
          {% else %}
            {{ states.sensor.car_in_garage.state }}
          {% endif %}

# This switch reads the same garage/doorSwitch topic as a sensor, but we can press a
      button to send a 0 or 1 to the MQTT topic directly to "manually override" the
      sensor
switch:
  - platform: mqtt
    name: Command Door
    state_topic: "garage/doorSwitch"
    command_topic: "garage/doorSwitch"
    payload_on: "1"
    payload_off: "0"

# To use the IFTTT webhook integration, your key should be placed here (obtained from
      https://ifttt.com/maker_webhooks)
ifttt:
  key: <IFTTT KEY GOES HERE>
```

automations.yaml (stored in /home/homeassistant/.homeassistant)

This contains automations which are automatically written when created through the UI editor.

```
- id: '1573538073059'
  alias: Garage Left Open Email
  description: Send an email to myself after the garage door was left open
  trigger:
    # Happens 1 minute after the door_switch value changes to "Opened"
  - entity_id: sensor.door_switch
    for: 00:01:00
    from: Closed
    platform: state
    to: Opened
  condition: []
  action:
    # Sends a simple IFTTT trigger with an event name
  - data:
      event: GarageLeftOpen
    service: ifttt.trigger
- id: '1573540765244'
  alias: Car Just Arrived Home
  description: Send a notification and record a log entry when I arrive home with the
                car
  trigger:
    # Happens when the car just arrives at home in the car_in_garage state
  - entity_id: sensor.car_in_garage
    platform: state
    to: The Car is Currently at Home
  condition: []
  action:
    # Sends an IFTTT trigger that also contains the value the stoplight sensor was set
      to
  - data_template: {"event":"CarArrived", "value1":"{{
                                                 states.sensor.stoplight_color.state }}"}
    service: ifttt.trigger
```

lovelace (stored in /home/homeassistant/.homeassistant/.storage)

This Json file contains the cards seen on Home Assistant's main page. It is automatically generated when an entity is edited through the UI editor.

```
{
    "data": {
        "config": {
            "title": "Provo Condo",
            "views": [
                {
                    "badges": [
                        {
                            "entity": "person.aaron_nelson"
                        },
                        {
                            "entity": "sun.sun"
                        },
                        {
                            "entity": "sensor.door_switch_mqtt"
                        },
                        {
                            "entity": "sensor.echo_sensor_color_mqtt"
                        }
                    ],
                    "cards": [
                        {
                            "entity": "sensor.door_switch",
                            "name": "Garage Door Status",
                            "type": "sensor"
                        },
                        {
                            "entity": "sensor.stoplight_color",
                            "name": "Stoplight Status",
                            "type": "sensor"
                        },
                        {
                            "entities": [
                                {
                                    "entity": "sensor.door_switch"
                                }
                            ],
                            "hours_to_show": 1,
                            "refresh_interval": 0,
                            "title": "Garage Door History",
                            "type": "history-graph"
                        },
                        {
                            "entities": [
                                {
                                    "entity": "sensor.stoplight_color"
                                }
                            ],
                            "hours_to_show": 1,
                            "refresh_interval": 0,
                            "title": "Stoplight History",
                            "type": "history-graph"
                        },
                        {
                            "entity": "switch.command_door",
                            "hold_action": {
                                "action": "more-info"
                            },
                            "name": "Garage Door Override",
                            "show_icon": true,
                            "show_name": true,
                            "tap_action": {
                                "action": "toggle"
                            },
                                "theme": "default",
                                "type": "entity-button"
                            },
                            {
                                "entity": "weather.provo_condo",
                                "type": "weather-forecast"
                            },
                            {
                                "entity": "sensor.car_in_garage",
                                "name": "Is the Car in the Garage?",
                                "type": "sensor"
                            }
                        ],
                        "path": "default_view",
                        "title": "Home"
                    }
                ]
            }
    },
    "key": "lovelace",
    "version": 1
}
```