# Lab 5 - Smart Home Controller

#### Online Link

The details and code to this lab can be found at: <https://github.com/Rhemsley/HomeAssistant-Garage>

#### Objective

The purpose of this lab is to learn how to create a smart home garage with a magnet, distance sensor, and stoplight using Home Assistant. This is to emulate a distance sensor that is used for parking your car in a garage and only turns on and runs the LEDS when the garage is open through the magnet sensor. This is accomplished as follows:

* Install an open-source home automation platform
* Interface this platform via MQTT or ESPHome with your various devices
* Use Home Assistant to monitor the state of the garage door, and whether the car is parked in the garage.

#### Materials

I built this Home assistant smart garage with the same physical build of this github <https://github.com/Rhemsley/MQTT-Garage-Distance-Stoplight>. The same coding strategy was followed and automations were built to get the system running automatically. On top of this, I also used a raspberry pi (4th gen) connected to my network over ethernet to run the Home Assistant Operating System, though a Home Assistant Container would also work. These instructions will be for implementing a raspberry pi as the host of Home Assistant.

#### References

I used the following resources to complete this lab:

<https://www.home-assistant.io/> Basic Home Assistant learning

<https://www.home-assistant.io/installation/raspberrypi/> Setting up the Home Assistant Operating System on my Raspberry Pi

<https://esphome.io/guides/getting_started_hassio.html> Setting up ESPHome and overview

<https://esphome.io/components/> Coding specific sensors/components in ESPHome

#### Baseline Information

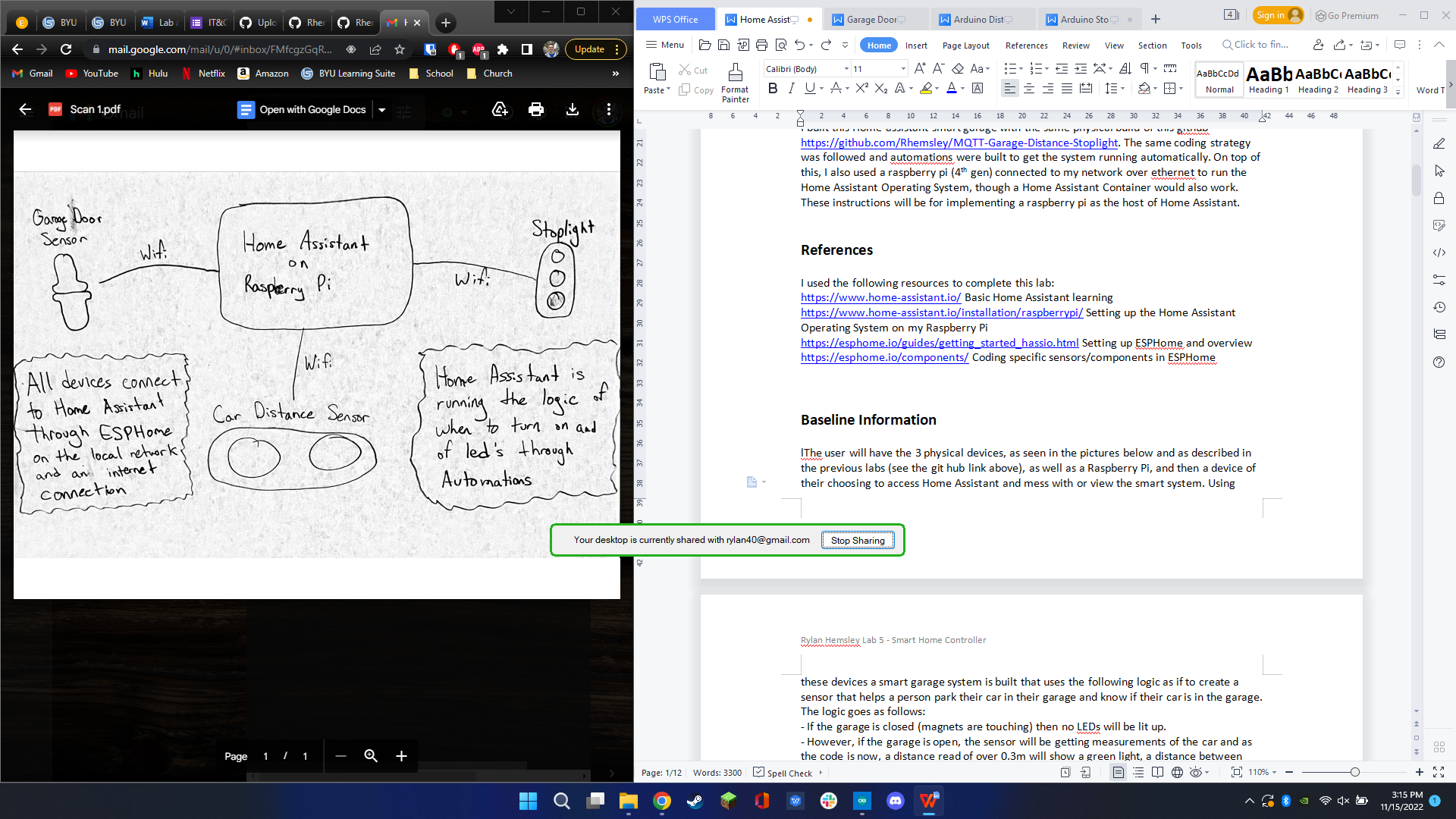
The user will have the 3 physical devices, as seen in the pictures below and as described in the previous labs (see the git hub link above), as well as a Raspberry Pi, and then a device of their choosing to access Home Assistant and mess with or view the smart system. Using these devices a smart garage system is built that uses the following logic as if to create a sensor that helps a person park their car in their garage and know if their car is in the garage. The logic goes as follows:

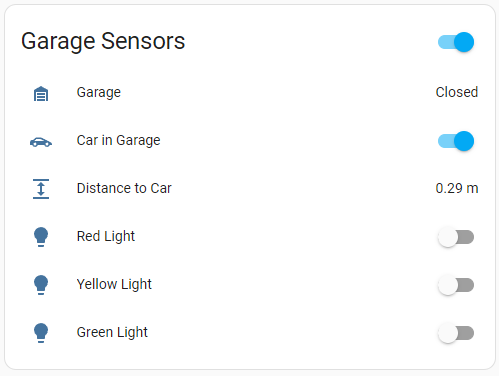
- If the garage is closed (magnets are touching) then no LEDs will be lit up.

- However, if the garage is open, the sensor will be getting measurements of the car and as the code is now, a distance read of over 0.3m will show a green light, a distance between 0.3m and 0.2m will show a yellow light, a distance between 0.2m and 0.1m will show a red light and bellow 0.1m will show a blinking red light. Then lastly, if the distance measured is below 0.3m then the car in garage boolean will indicate true to say there is a car in the garage.

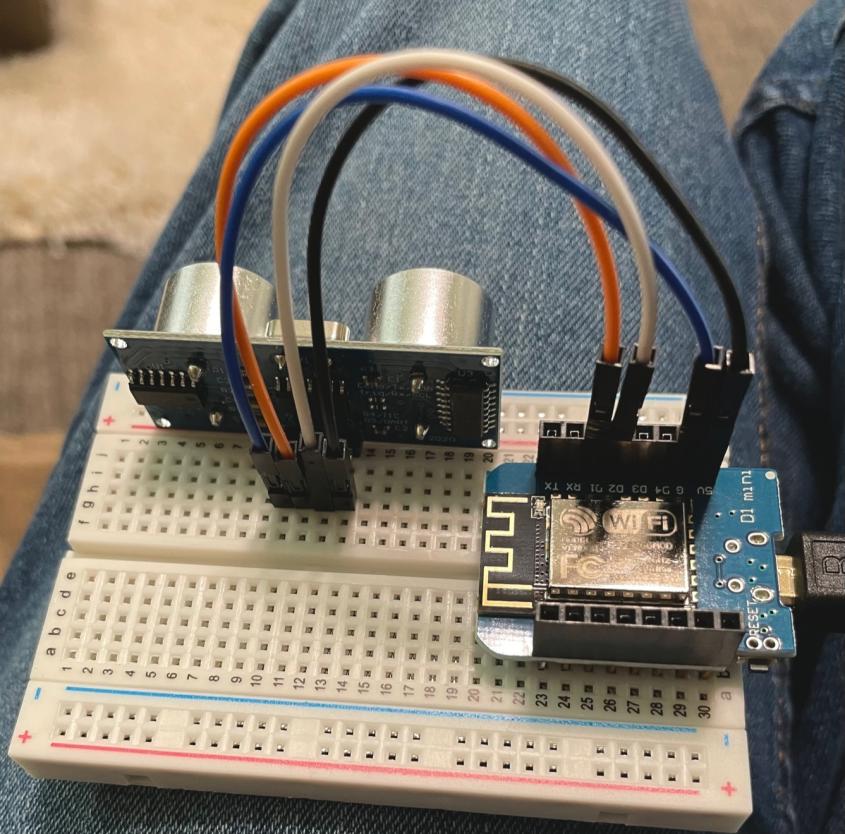
Those distances could easily be changed by modifying the Home Assistant Automation distance range values.

This first picture Shows the 4 devices in the system and how they communicate over the local network where Home Assistant controls the devices and logic as explained above using ESPHome’s Add-on.

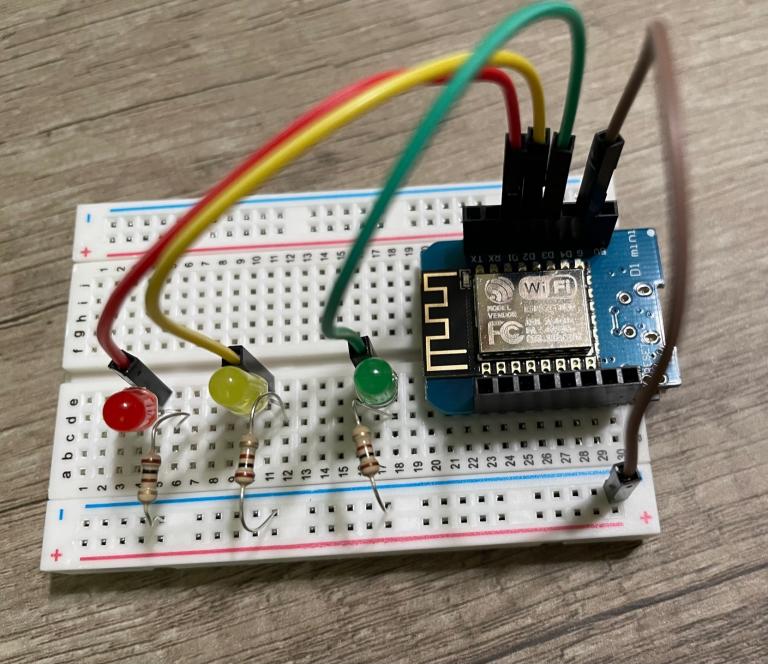
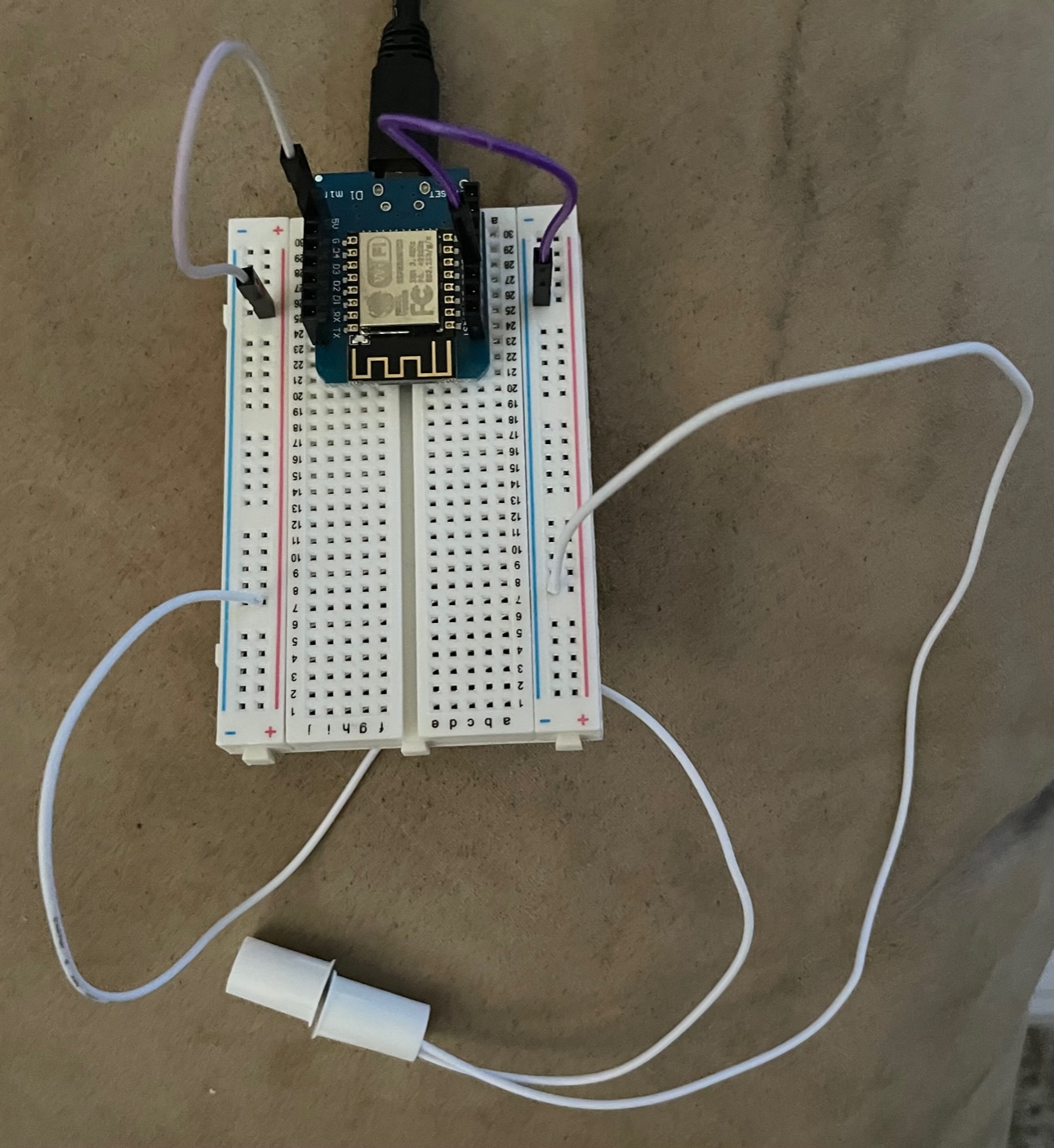




The second picture is the custom built Home Assistant Dashboard card to show all of the sensor data and allow for manual light configuration.



The third, fourth, and fifth pictures are the already seen build pictures showing how each sensor is built. For more specific details see the previous git hub repositories for each of the labs found by going to <https://github.com/Rhemsley/MQTT-Garage-Distance-Stoplight> and reading about each of the previous labs.



#### Procedures

I will assume the steps given in the above MQTT Garage Distance Stoplight Git repository and prerequisite repositories on creating the individual devices have already been followed to create the Stoplight, Distance sensor, and magnet D1 Mini systems. Once those 3 devices have been built, the following steps can be followed to get Home Assistant running, ESP Home in place, the devices coded, and the smart system fully implemented. Note: you only need to follow the building steps in the previous labs and none of the previous downloading or coding steps will be needed as the below steps will go over the new code implementation.

1. Setup Home Assistant
   1. Please refer directly to the instructions given at <https://www.home-assistant.io/installation/raspberrypi/> to get your Raspberry Pi (or a container) running Home Assistant. As it indicates, this may take some time as you wait for things to load.
   2. Follow those steps until you are able to access your Home Assistant at homeassistant:8123 on an internet browser on your network. Once it has all been configured to your liking and therefore the setup is done, move to the next step.
2. Add ESPHome Add-on
   1. To add the ESPHome Add-on which will be used for coding your D1 Minis, hit “Settings” then “Add-ons” and finally the blue button at the bottom “ADD-ON STORE.” Within this page you can search for “ESPHome” and install it. You will first want to hit the “Show in sidebar” option to on and may also want to set the ESPHome Add-on to “Start on boot”.
   2. For additional info on initial setup, see <https://esphome.io/guides/getting_started_hassio.html>
3. Add the 3 sensor devices using ESPHome
   1. Now pull up the ESPHome Tab either through the quick sidebar button ”ESPHome”. Within this page you are able to add your devices with the green “+ NEW DEVICES” button at the bottom right. The first device may have you put in a couple extra things like WiFi username and password but the below steps should take you through what to do to add your devices.
   2. Upon hitting that “+ NEW DEVICES” button, hit continue to start setting up your sensor and then put in your Sensor Name and hit “NEXT”. For this project I used the sensor names of “stoplight”, “distance-sensor”, and “garage.” If you use other names then make sure to adjust the given code to match these name changes.
   3. After picking the name, hit “Pick specific board” and then select the board option of “Wemos D1 and Wemos D1 mini.”
   4. Now install the initial configuration with the “INSTALL” button. Then I would recommend plugging the device into your raspberry pi if possible and selecting the “Plug into the computer running ESPHome Dashboard” option then selecting the Port the device is plugged into. Now wait for the installation to completely finish (this will likely take several minutes, just wait for when it says “setup() finished successfully!” in green) and then you can hit “STOP”. I will note that the “Plug into this computer” option can also work, it will just require downloading something and going to another browser as it directs.
   5. Now that the device has been configured, you can open its YAML code configuration with the “EDIT” button under the new devices box. This will pull up the YAML code already installed on the device and where you can add the YAML code I have in github for that device in line with 2 lines below the “captive\_portal:” text at the bottom. For example, the distance sensor should look something like,

captive\_portal:

sensor:

- platform: ultrasonic

trigger\_pin: D2

echo\_pin: D1

name: "Ultrasonic Sensor"

update\_interval: 100ms

* 1. If you have any confusion at what these code additions are doing, feel free to check out <https://esphome.io/components/> to find your sensor and read about what you can do with it.
  2. Now just repeat steps b through f for the remaining two devices and once a device is installed, feel free to power it with a USB anywhere in the network as it is now connecting over WiFi to your network.

1. Add devices to Home Assistant
   1. Now that the devices have been created in ESPHome we will add those devices to the Home Assistant controller itself. To do this, go to your ESPHome page and click on the 3 stacked dots at the bottom right of one the device of your choosing, then hit “Show API Key”, and copy that key to clipboard with the “COPY” button.
   2. Next go to “Settings” then “Devices & Services” and you should see there a couple boxes labeled “Discovered” with the name of one of your devices in each one. Under the device of the API Key you just copied, hit “CONFIGURE” then “SUBMIT” and then paste in your API Key from your clipboard (Ctrl + v) and hit “SUBMIT” again. If done properly, your device will now be accessible from your Home Assistant allowing for lots of flexibility.
   3. Now repeat steps a and b for the remaining two devices so that all 3 of the devices have been configured and added to your Home Assistant.
2. Create Car status Boolean
   1. While on this “Devices & Services” page in “Settings” lets add a Toggle Helper boolean that will be used to indicate if the car is currently in the garage.
   2. To do this, head over to the “Helpers” tab at the top and then hit the blue “+ CREATE HELPER” button and select “Toggle”. Then name it “Car in Garage” and if desired, set the icon to a car icon and hit “CREATE”.
3. Add devices to Dashboard
   1. Lets go make a visual for your newly added devices on your dashboard so you can manually make changes or see live indicators of your devices.
   2. To do this, head to the “Overview” page and if first time setup, allow for custom configuration of this page. Then go into the Page configuration screen with the “Edit Dashboard” button in the 3 stacked dots and then hit the “+ ADD CARD”. You can then make your own cards if desired, or if you would like to make an all in one card like I did, feel free to select any example card it shows you and then hit the “SHOW CODE EDITOR” button and paste in the Garage Sensors Code found in github/below and hit “SAVE”

type: entities

entities:

- entity: binary\_sensor.garage

- entity: input\_boolean.car\_in\_garage

- entity: sensor.ultrasonic\_sensor

name: Distance to Car

- entity: light.red\_light

- entity: light.yellow\_light

- entity: light.green\_light

title: Garage Sensors

* 1. You should now see all of your sensor and light information so long as they are powered and connected to your network. Make sure that any name changes you made to my code are reflected in all areas of the code.

1. Create Automation
   1. All that is left is to get the devices working together to create a smart garage. This smart garage’s logic goes as follows: If the garage is closed (magnets are touching) then no LEDs will be lit up. However, if the garage is open, the sensor will be getting measurements of the car and as the code is now, a distance read of over 0.3m will show a green light, a distance between 0.3m and 0.2m will show a yellow light, a distance between 0.2m and 0.1m will show a red light and bellow 0.1m will show a blinking red light. Then lastly, if the distance measured is below 0.3m then the car in garage boolean will indicate true to say there is a car in the garage.
   2. The above automation logic is accomplished in two Automations that will both need to be created. This is done by going to “Settings” then “Automations and Scenes”. Next hit the blue “+ CREATE AUTOMATION” button at the bottom right and select the “Start with an empty automation” button. From here hit the 3 stacked dots at the top right and select the “Edit in YAML” option and paste in the automation code I gave. Do this for both the “Garage Open Automation YAML” and “Garage Closed Automation YAML” making sure to again update any name changes you made in this code.
   3. These automations will now trigger upon the change of the garage door open or closed status and the sensor system will run continuously as the garage door is open.
   4. These can be tested physically to make sure they are working properly and can even be seen on that combined card on the overview dashboard.

#### Observations

I seriously loved this lab. I have loved all of the labs but I think this one took number one especially because of how much it opened my eyes to what is possible with Home Assistant. I absolutely want to implement this in my house with my different systems and am excited to implement my own creation in the final project. Getting this lab up and going and then finished was really not too bad. As one would expect, I ran into several road bumps along the way but luckily none of them were really that bad and soon made sense why I ran into them as I will hit on a bit in the next paragraph. For this lab I decided to go with the ESPHome route as I was really intrigued with it and wanted to learn what it was all about. I enjoyed MQTT the other week and lab but figured now was a good time to try out a new system and I am so glad I did. It clicked pretty quickly and big time helped me accomplish what I was trying to do.

Ok, now for difficulties I ran into. Luckily I didn’t run into too many. The first one was just sinking a bit of time into learning about Home Assistant and ESPHome. I didn’t spend tons of time but reading up on the basics on their websites was for sure handy and got me up on my feet quickly. The first hiccup I recall running into was actually after I got all the devices up and running and accessible with Home Assistant when I was then trying to figure out how to code the logic of the devices with Automations. I had to trial and error several of those to figure out when I wanted triggers vs conditions vs actions and then basically copy and paste half the logic within a while look. This luckily wasn’t too bad once I realized I could copy and paste the YAML. All in all the logic wasn’t too bad and it felt like solving a puzzle so I enjoyed it. The other hiccup I remember was getting the Car in Garage status to work because I figured out how to add a helper toggle boolean but then went and changed a name or two and broke the whole magnet system. I ended up just recreating the magnet device entirely and restored it but that was real dumb of me to changes the names in the ways that I did. Lesson learned on that one for sure.

#### Thought Questions

1. Which version of Home Assistant did you choose to install? (Home Assistant Operating System, Home Assistant Container or one of the more experienced versions) Why did you choose this version?

I installed the Home Assistant Operating System on my Raspberry Pi 4. I wasn’t really using the Raspberry Pi for anything and figured it would be nice to have a dedicated system to run Home Assistant as I may very well use this for who knows how long, especially as I started using it and figuring out how it all works.

1. 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?

I decided that I would like to do the logic coding within Home Assistant automations itself and let the devices just send their data. This was especially effective as I used ESPHome and was able to setup the devices to just send their data pretty easily. Then getting their data to Home Assistant and creating Cards for them was very easy. For sure the more difficult part was nailing down all the login within the Automations tool, especially as I hadn’t used automations before and had to learn what the Triggers, Conditions, Actions, and the sub categories of all of those meant and could do. For future esp devices or really any devices, I fully plan to add the device and get its raw data going to Home Assistant and then let home Assistant do the logic coding as it is easy to work with, turn on and off, and later modify if needed.

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

I love how many different devices and systems it works with! It was so so easy to get several of my systems added and it blew my mind scrolling through the list how many systems they work with. Another thing I loved was how it build out like a custom dashboard for me right away and then allowed me to go modify that or make a new dashboard. It felt like the possibilities were endless. Then lastly I really thought ESPHome was super nifty, cool, and capable. It did feel a bit like a poke to the side of how many hours I spent manually coding these devices however that also gave me a great appreciation for how cool ESPHome is.

4. Please estimate the total time you spent on this lab and report.

I would estimate I spent about 3 hours coding/building the project and then 4 hours creating the lab write up.

#### Certification of Work

I certify that the results and solution to this lab were my own work. For the resources of information I found through exploring the internet, I referenced the websites and what I pulled from it. All code written was of my own writing.

-Rylan Hemsley

#### Appendix

Code for reference:

**Garage Closed Automation YAML**

alias: Garage Closed

description: ""

trigger:

- type: not\_opened

platform: device

device\_id: d7376f5cfa13ff23b36bd041c26b2e43

entity\_id: binary\_sensor.garage

domain: binary\_sensor

condition: []

action:

- type: turn\_off

device\_id: d8768d804421463dd081023f60d097da

entity\_id: light.red\_light

domain: light

- type: turn\_off

device\_id: d8768d804421463dd081023f60d097da

entity\_id: light.yellow\_light

domain: light

- type: turn\_off

device\_id: d8768d804421463dd081023f60d097da

entity\_id: light.green\_light

domain: light

mode: single

**Garage Open Automation YAML**

alias: Garage Open

description: ""

trigger:

- type: opened

platform: device

device\_id: d7376f5cfa13ff23b36bd041c26b2e43

entity\_id: binary\_sensor.garage

domain: binary\_sensor

condition: []

action:

- repeat:

while:

- type: is\_open

condition: device

device\_id: d7376f5cfa13ff23b36bd041c26b2e43

entity\_id: binary\_sensor.garage

domain: binary\_sensor

sequence:

- if:

- type: is\_value

condition: device

device\_id: 9e72f8a9316496d66d7976dde7ff392a

entity\_id: sensor.ultrasonic\_sensor

domain: sensor

above: 0

below: 0.1

then:

- type: toggle

device\_id: d8768d804421463dd081023f60d097da

entity\_id: light.red\_light

domain: light

- type: turn\_off

device\_id: d8768d804421463dd081023f60d097da

entity\_id: light.yellow\_light

domain: light

- type: turn\_off

device\_id: d8768d804421463dd081023f60d097da

entity\_id: light.green\_light

domain: light

alias: Red Blink Distance

- if:

- type: is\_value

condition: device

device\_id: 9e72f8a9316496d66d7976dde7ff392a

entity\_id: sensor.ultrasonic\_sensor

domain: sensor

above: 0.1

below: 0.2

then:

- type: turn\_on

device\_id: d8768d804421463dd081023f60d097da

entity\_id: light.red\_light

domain: light

- type: turn\_off

device\_id: d8768d804421463dd081023f60d097da

entity\_id: light.yellow\_light

domain: light

- type: turn\_off

device\_id: d8768d804421463dd081023f60d097da

entity\_id: light.green\_light

domain: light

alias: Red Distance

- if:

- type: is\_value

condition: device

device\_id: 9e72f8a9316496d66d7976dde7ff392a

entity\_id: sensor.ultrasonic\_sensor

domain: sensor

above: 0.2

below: 0.3

then:

- type: turn\_off

device\_id: d8768d804421463dd081023f60d097da

entity\_id: light.red\_light

domain: light

- type: turn\_on

device\_id: d8768d804421463dd081023f60d097da

entity\_id: light.yellow\_light

domain: light

- type: turn\_off

device\_id: d8768d804421463dd081023f60d097da

entity\_id: light.green\_light

domain: light

alias: Yellow Distance

- if:

- type: is\_value

condition: device

device\_id: 9e72f8a9316496d66d7976dde7ff392a

entity\_id: sensor.ultrasonic\_sensor

domain: sensor

above: 0.3

below: 10

then:

- type: turn\_off

device\_id: d8768d804421463dd081023f60d097da

entity\_id: light.red\_light

domain: light

- type: turn\_off

device\_id: d8768d804421463dd081023f60d097da

entity\_id: light.yellow\_light

domain: light

- type: turn\_on

device\_id: d8768d804421463dd081023f60d097da

entity\_id: light.green\_light

domain: light

alias: Green Distance

- delay:

hours: 0

minutes: 0

seconds: 0

milliseconds: 100

- if:

- type: is\_value

condition: device

device\_id: 9e72f8a9316496d66d7976dde7ff392a

entity\_id: sensor.ultrasonic\_sensor

domain: sensor

above: 0

below: 0.3

then:

- service: input\_boolean.turn\_on

data: {}

target:

entity\_id: input\_boolean.car\_in\_garage

alias: Car in Garage Status

else:

- service: input\_boolean.turn\_off

data: {}

target:

entity\_id: input\_boolean.car\_in\_garage

mode: single

**Garage Sensors Card Code**

type: entities

entities:

- entity: binary\_sensor.garage

- entity: input\_boolean.car\_in\_garage

- entity: sensor.ultrasonic\_sensor

name: Distance to Car

- entity: light.red\_light

- entity: light.yellow\_light

- entity: light.green\_light

title: Garage Sensors

**Stoplight YAML after config**

light:

- platform: binary

name: "Green Light"

output: green\_led

- platform: binary

name: "Yellow Light"

output: yellow\_led

- platform: binary

name: "Red Light"

output: red\_led

output:

- id: green\_led

platform: gpio

pin: GPIO0

- id: yellow\_led

platform: gpio

pin: GPIO4

- id: red\_led

platform: gpio

pin: GPIO5

**Garage YAML after config**

binary\_sensor:

- platform: gpio

pin:

number: D5

mode:

input: true

pullup: true

name: "Garage"

device\_class: garage\_door

**Distance Sensor YAML after config**

sensor:

- platform: ultrasonic

trigger\_pin: D2

echo\_pin: D1

name: "Ultrasonic Sensor"

update\_interval: 100ms