# Lab 6 - Garage Door Controller

#### Online Link

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

#### Objective

The purpose of this lab is to implement a garage door controller that can be controlled remotely even while not at home. This will bring a level of creativity in how to accomplish this using Home Assistant as by default it is limited to a home network connection. This is accomplished as follows:

* Build a controller that could interface with an existing garage door opener
* Allow for remote control of the garage door opener with a smart phone (in this case iPhone using Siri/a Shortcut

#### Materials

I built this Garage Remote controller on top of the Home assistant smart garage outlined in this github <https://github.com/Rhemsley/HomeAssistant-Garage>. The exact same systems were used as well as the below equipment to make the remote and remote accessible remotely.

1 x D1 mini

4 x Male to Male Jumper Cables

1 x Relay built for D1 mini

1 x LED (Not entirely needed, only used as a visual to the relay)

1 x 100Ω 5% Resistor (BrBlBrGd) (For LED)

1 x Breadboard Button

1 x Breadboard

1 x Smart Outlet from TP-Link for Out of network remote control

1 x iPhone for controlling the remote with a Shortcut

1 x TP-Link App

#### References

I used the following resources to complete this lab:

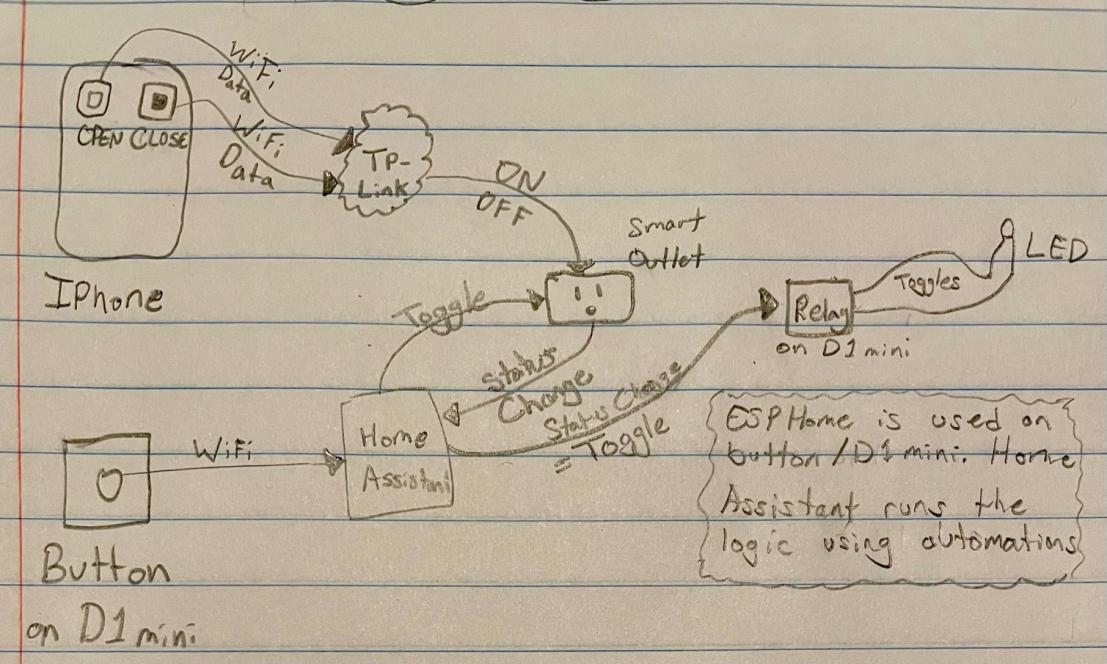
<https://esphome.io/components/> Coding the relay and button in ESPHome

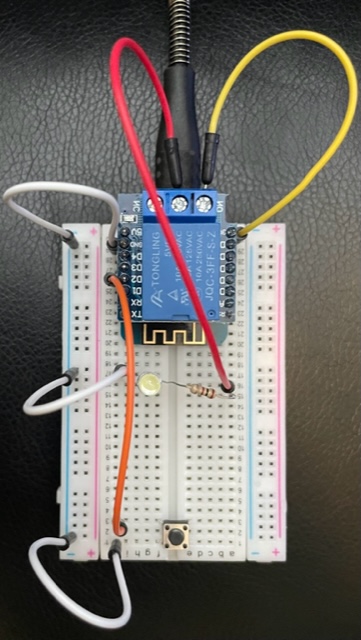
#### Baseline Information

This system is built just like the other ESP devices built in the previous lab but with a relay and button. Please see the github link above for the previous home assistant build lab. All understanding of that lab is assumed for the following writeup. This lab is adding a relay and button to the smart garage system. It is also adding a TP-Link Smart Plug that acts as a variable for controlling the Garage Remote while not on the network using the TP-Link App.

Two Home Assistant Automation run the logic. One says that if the physical button is pressed the smart outlet changes states. The other says if the smart outlet changes states then toggle the relay. This results in the button toggling the relay and the smart phone being able to send “On” or “Off” requests to the smart outlet through Siri/Shortcuts. Please see the diagram below for a visual of this system.

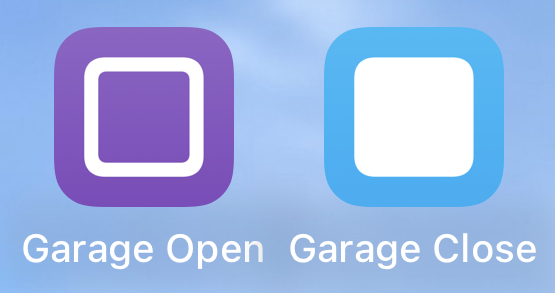
This first picture shows the logic of the system saying what was explained directly above as it shows the button and the iPhone controlling the relay through home assistant and the smart outlet.





The second picture is the build of the relay attached to a D1 mini with the wires going out of the relay to the LED and the button wired up as well. This has the button wired into D2 and ground and the LED wired to ground and the relay with the relay also wired to one of the 3 volt pins.

The third picture is the Shortcuts created on the iPhone desktop to control the Garage remotely.



#### Procedures

I will assume the steps given in the previous Lab of creating the Home Assistant Smart Garage system have already been followed and understood. On top of this, this remote garage opener uses a TP-Link Smart Outlet. Please set this outlet up (I used the name “garage\_remote”) in the TP-Link before hand and then in Home Assistant add the “TP-Link Kasa Smart” Integration in order for your Smart outlet to also show up as an available device. The following steps will explain how to get the relay up and running on home assistant, get the automations setup so the button starts working, and finally create the shortcuts to control your garage with a smart phone.

1. Add the D1 mini with the relay and button using ESPHome
   1. Pull up the ESPHome Tab 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.
   2. Upon hitting that “+ NEW DEVICES” button, hit continue to start setting up your relay and then put in the name and hit “NEXT”. For this project I used the name “garage-remote”. If you use other names then make sure to adjust the given code to match any 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 to install the initial configuration with the “INSTALL” button. 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 below YAML code/code in github under the “captive\_portal:” text at the bottom. Like this:

captive\_portal:

binary\_sensor:

- platform: gpio

pin:

number: D2

inverted: true

mode:

input: true

pullup: true

name: "Garage Door Button"

switch:

- platform: gpio

name: "Relay"

pin: D1

* 1. If you have any confusion at what these code additions are doing, feel free to check out <https://esphome.io/components/> to read more about the relay and button setups.

1. Add device to Home Assistant
   1. Now that the devices have been created in ESPHome we will add the device 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.
2. Add device to Dashboard if desired
   1. If desired you can head to the “Overview” page and then go into the Page configuration screen with the “Edit Dashboard” button in the 3 stacked dots.
   2. From here you can open up the Garage Sensors Card added in the previous lab and hit the “SHOW CODE EDITOR” button and then add the below code/github code inbetween “green\_light” and “title: Garage Sensors” as seen below and hit “SAVE”. Or simply replace the existing code with my new YAML code in github.

//- entity: light.green\_light

- entity: switch.garage\_remote

name: Garage Remote Bool

- entity: switch.relay

name: Garage Door Controller

- entity: binary\_sensor.garage\_door\_button

//title: Garage Sensors

* 1. Again, make sure that any name changes you made to my code are reflected in all areas of the code and in particular changing the name of the smart outlet (garage\_remote) to match the one you are using.

1. Create Automations
   1. Next we will get the Home Assistant Automations rolling. First we will make the Garage Button automation, then the Garage Remote automation.
   2. The Garage Button automation says that if the button is pushed it will toggle the garage door remote outlet. The Garage Remote automation says that when the garage remote outlet changes state to then toggle the relay. This gets the relay to always change when the garage remote outlet changes.
   3. 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 Button Automation YAML” and “Garage Remote Automation YAML” making sure to again update any name changes you made in this code and in particular again changing the name of the smart outlet to match the one you are using.
   4. These automations can now both be tested by simply pressing the button. Upon pressing the button, both the outlet should change states and the relay should follow suit in changing states. 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.
2. Create the iPhone Remote Controller
   1. At this point you should have the TP-Link app all setup and running and we will now create a iPhone Shortcut to run the device in the app.
   2. To do this we will open up the Shortcuts App and hit the “+” icon at the top right. Then from here type in the name at the top (I did “Garage Open” and “Garage Close”).
   3. Next search “Kasa” in the search box and select the prebuilt “Turn on/off a device” option with the blue plus icon. Now select the “on” state and then select the device of the Garage Remote Outlet you setup and have used in the previous automations.
   4. Repeat b and c for the “off state” and if desired the “Ask Each Time” prompt.
   5. Your iPhone/Siri should now be able to control the outlet and therefore garage relay with commands such as “Hey Siri, Run (name of shortcut like “Garage Open”).”
   6. To level this up one more level, you can go back into each of the shortcuts, hit the drop down arrow next to the name, change the icon, and then select the “Add to Home Screen” option to create a quick button shortcut on your home screen.
3. Your Smart garage just received its upgrade!
   1. Your remote should be all up and running now. If you are having any issues, make sure the names are all lined up correctly in code or otherwise google the issue with that specific device.
   2. I will note that there can be a slight delay in the shortcut actually toggling the relay but this is out of ones control and this was implemented to allow the iPhone to be able to control the garage from anywhere using data or WiFi.

#### Observations

I again loved this lab and am truly coming to love home assistant. I have already been adding small automations and other tools to my smart home and the ideas are coming more and more every day. This lab allowed me to be creative with how I combine devices and figure out what I can and can’t do remotely. Since doing this lab I added access to my home assistant from not just on my WiFi and feel like I could have found more solutions now.

What a unique objective to have to overcome with this lab. First of all, getting the relay to work as I would expect took a second as I needed to learn out the output of the relay truly worked but then the real fun came with figuring out how to get the system controllable while not on the network.

Interestingly, the idea came pretty quickly though I doubted its practical use case a bit. As seen above, I decided to use a smart outlet as a sort of middle man between my phone/the button and the relay. This allowed for simple manual control and phone control of my “garage.” As one might guess, the reason I feel like this might not be the most practical solution is because it requires turning on and off an outlet that has to be plugged in and then likely not used for anything else besides maybe a garage light source. The more I have thought about this the more I have thought about playing with the idea of maybe a Kasa shortcut that messes with the LED of the outlet or something like that. But none the less, I am pleased with my solution and very pleased with the simple shortcut app icons I have on my home screen. I sometimes mess with the apps just because I think it is so cool.

#### Thought Questions

1. What services did you consider integrating into your project and why?

I also considered what I could do with the Hue app/service as I have those and can control those remotely I think but didn’t really look far into this because then I thought about Kasa and quickly realized a potential solution, which I used. As seen I also decided to implement an LED for visual (though the ESPHome Relay already uses the onboard LED) and even a manual button as I figured one would want to be able to control this with a click of a button.

1. What services would you like to integrate in the future?

Moving forward I have a lot of ideas and exploration I would like to make with Home Assistant. I have already started putting those into play as I got a bathroom speaker playing a fan sound at the push of a button, I got plex running, and I hooked up my minecraft server to Home Assistant for simple stat viewing. I daily am exploring the potential integration options though and seeing what other devices I can mess with.

1. Consider internal and external security threats. Identify 3 likely attack vectors for someone to compromise this system? How did you mitigate these?

I think the number one thing is that an accidental trigger of this would result in my garage opening without maybe me knowing though I have not seen very many ways this could accidentally trigger and would actually have a better indicator of it being open than normal thanks to it being displayed on my dashboard. The next risk is if someone where to take my phone and somehow use it unlocked, they could open up my garage. Good think iPhones have pretty good security measures like face ID. Next would be if someone where to somehow hack into either the TP-Link app or my Home network, they could now open the garage.

1. What was the biggest challenge you overcame in this lab?

I would say it would be understanding how the relay worked properly, figuring out how to get the shortcut all built out how I wanted it and controlling the device as desired, or getting the automations working together properly.

1. 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 Remote Automation YAML**

alias: Garage Remote

description: ""

trigger:

- platform: state

entity\_id:

- switch.garage\_remote

condition: []

action:

- type: toggle

device\_id: 8e1f76f38cdb979f5abce305f69f390d

entity\_id: switch.relay\_2

domain: switch

mode: single

**Garage Button Automation YAML**

alias: Garage Button

description: ""

trigger:

- platform: state

entity\_id:

- binary\_sensor.garage\_door\_button\_2

from: "off"

to: "on"

condition: []

action:

- type: toggle

device\_id: 281b0845ddb7b9d517eadc36c18bb47b

entity\_id: switch.garage\_remote

domain: switch

mode: single

**New 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

- entity: switch.garage\_remote

name: Garage Remote Bool

- entity: switch.relay\_2

name: Garage Door Controller

- entity: binary\_sensor.garage\_door\_button\_2

title: Garage Sensors

**Garage-remote YAML after config**

binary\_sensor:

- platform: gpio

pin:

number: D2

inverted: true

mode:

input: true

pullup: true

name: "Garage Door Button"

switch:

- platform: gpio

name: "Relay"

pin: D1