Linnæus University
Sweden

Report

Control speakers with Google home

-Project course 2DT301

Author: Ola Lindsten (ol222es)
Supervisor: Francesco Flammini
Semester: VT 2020
Course code: 2DT301

Linnæus University
Sweden

Table of contents

1. Introduction

This report is about my project in the course 2DT301. I will talk about what my project is, what went good and what kind of setbacks I had.

My project works as following, I will say or write something to my google home, for example turn on speakers, or increase volume by 10. The google home will then send an HTTP request to the raspberry pi with the help of custom commands using IFTTT and webhooks. After this the raspberry pi will interpret the HTTP request it got from the google home. In order to not go in too deep on small complicated things I decided to not worry about threading and instead I created another class where I can add buttons and store them in the json file with the corresponding IR-code from the remote.

The program will then look through that json file and see if there is a button there that matches the button we got from the google home. The raspberry will then take that ir code for the specific button and then send it via the ir-transmitter to the speaker receiver. The program will then print the different commands on the display which is connected via USB to the Arduino which then displays it on and LCD for e.g. “Speakers muted”

2. Background

Smart devices have been around for a while now, but they have become more popular in the recent years when Google and Amazon released their smart assistants google home and Alexa.

I have watched the development of the smart assistants since their released and always wanted to buy one for myself. So, a couple of months ago I decided that I should finally buy one. I decided to buy the google home nest mini 2nd generation.

I have watched a lot of YouTube videos on this subject and I have seen people do all kinds of cool things with their google home, like custom commands, turning on lights and so on. Therefore, when this course started, I knew that I wanted to do something with my google home, the question was just what.

So, I started to think about what I wanted to do. My first idea for this project was to connect my google home with my lamp using a relay for the socket, but because I do not have that much experience in electronics I felt that this would be too hard to do in this timeframe of this course, so I scratched that idea. My other idea was to turn my computer on or off, but this felt like to small of a project. Therefore, I started to look for a middle ground between those two ideas.

I then came up with the idea to either connect my google home to my smart tv or my speakers, and after some consideration I decided to connect it to my speakers, because I always forget to turn them off.

Hardware

- *Google home nest mini 2nd generation*
  Google home is a smart speaker developed by google. The product is integrated with google assistant which allows voice control.

- *Arduino LCD*
  Liquid crystal display, that is developed for an Arduino.

- *Arduino Wi-Fi module (ESP 8266)*
  ESP 8266 is a Wi-Fi microchip that is developed for an Arduino.

- *Arduino Uno*
  Arduino is a microcontroller card.

- *Raspberry pi 3*
  Raspberry pi is a low-cost small computer.

- *Ir Receiver*
  A module that can get an ir-signal.

- *Ir Transmitter*
  A module that can transmit an ir-signal.

- *Logitech z906*
  My speakers that have a remote which transmit an ir-signal and a control unit which can receive an ir-signal.

3. Implementation

When I had decided which project to do, it was time to think about how I should do it. Because my speakers don't have Bluetooth, Wi-Fi or something like that, my options where very limited, so therefore I made an easy decision to send an ir signal to the device.

When I had figured out what I wanted to do and how I should do it I started by writing down the different components that was required in order to complete this project, and then checked if I needed to buy something. I bought an ir-receiver, ir-transmitter and an ESP8266 module.

Unfortunately it took a fairly long time to get all the parts and therefore I got a late start to this project and this set back isn't really anything I could do anything about either, because I needed the parts in order to get started with the project. When I finally got all the different components I started almost immediately with the project because I was excited to get started and felt like the time was already running away.

The first practical thing I started with was trying to connect the ESP8266 module to the Arduino so I could connect it to the internet. Already here I unfortunately faced a problem and probably the most time-consuming problem in this project. For some unknown reason that I couldn't figure out the Arduino couldn't connect to the internet. I tried googling and watched countless videos about how to connect Arduino with the ESP8266 module and my conclusion on this issue is that the ESP8266 module was not working correctly. I don't know if I did something to cause it or if it was broken when I got it.

After I had tried to get the Wi-Fi-module to work for quite a sometime with no success I decided to switch from having the server on the Arduino to instead try to set it up on my raspberry pi which already has Wi-Fi integrated into it. So, I started googling to see if I could find any good libraries to use for my project.

When I switched to raspberry pi the first thing I did was to install a buster lite image on the raspberry. Buster lite is an operative system for raspberry without a GUI, so therefore it won't take unnecessary computing power to load an GUI. I instead configured SSH. So, I could remotely connect to the raspberry. I also configured the network, so it would automatically connect to my network on boot.

When I had installed the buster image on the raspberry and had configured the settings, I started by installing a library called pigpio which is a library in python which allows you to control the GPIOs on the raspberry pi. I also looked up another library called ircodec which is a library for handling IR-signals.

I then looked up a schematic for the ir-receiver to try to get the different codes for the remote. The ircodec library were a bit of tricky to understand because there was some configuration that needed to be done before it worked. The problem I had was that the pigpio library where running on python 2.7 and the ircodec was running on python 3. So, after I found that problem it was an easy fix.

I then looked at the schematic for the ir-transmitter in order to try to send a signal. The issue I had was that the schematic was connected to an Arduino, so I had to investigate the different GPIOs on the raspberry and connect it to the right pin. When All of that was done, I just tried to transmit a signal from the ir-transmitter to my speakers

After this I looked into how I should structure the server and decided that I would have a *"queryparam"* in the url so I only need one method for transmitting an ir-signal for all the buttons instead of a one method for each button. I started setting up the server and tried to just print some words to see that the server worked. To set up the server I used the python library flask, which is a web application framework. The url for the server looks like e.g. this:

$$\text{ip: port/send} - \text{button?button} = \$\&\text{amount} = £$$

where ip is my public ip address and the port is specified in the code. The $ is the button that we are going to send and the £ are how many times e.g. volume up will run.

Then I started configuring the different IFTTT commands with this structure in mind. Because the google home have some predefined commands therefore I needed to choose something that couldn't be misinterpreted by the google home. I decided to make it simple for myself and choose the following commands, where $ is a number

- down $ → Volume Down
- up $ → Volume Up
- speaker mute / speaker unmute → Mute or Unmute
- speaker on / speaker off → Speaker On or Off

But in order to let the google home communicate with the raspberry pi I needed to open the chosen port on my route to allow traffic to flow through the router to the raspberry. When I open the port, I added some different methods and classes where you need to run an initializer the first time you run the program in order to create the json file for the different ir-signals. I also create a new class for adding a json file and a new method for deleting a button.

The next step was to implement the LCD and because my original idea was to use an Arduino as a base for the project this became an issue. I thought about different ways to approach the situation and after some consideration without having to order any new parts I came up with the idea to use the Arduino as a middleman between the raspberry pi and the display. In order to do this, I connected them with an USB cable and imported a library called Serial and which helped me to transfer serial data to the display.

In order to show any text on my display, there is an Arduino library called Liquid Crystal which I've worked with before, so the setup was pretty easy. I simply just showed the data I got from the raspberry and made so when the screen has reached the end of the first line it will continue to the next line. The problem that occurred when I tried to transfer the data from the raspberry to the Arduino was that the Arduino didn't get enough power and it took me some time to realise this but when I figured it out I simply swapped to another adapter that could output a higher voltage and ampere.

Lastly, I thought about security which turned out to be quite hard. I thought in the beginning of the project that you could simply allow the IFTTT ip to make requests. But I then noticed that when I get the request from IFTTT they are using different ips every time. So, what I did is simply just to check if the body is correct when the request is made and that isn't a very secure solution but it's better than nothing.

4. Discussion and final remarks

If I had more time, there is two things I would like to improve

First off, I would have wanted to create an ui for the project so it would have been easy to do the same thing as with the google home, but without having to use the

google home app or voice commands, instead just go to the website and input it from there.

The second thing is I wanted to make it more secure because right now it's easy for someone from the outside of my network to make a request to my server. Unfortunately, I couldn't come up with a good solution in the timeframe of this course, but I would like to improve it in the future.

5. Conclusions

I am happy about the result it was really interesting to learn about how to make custom commands with google home, even though I had some setbacks.

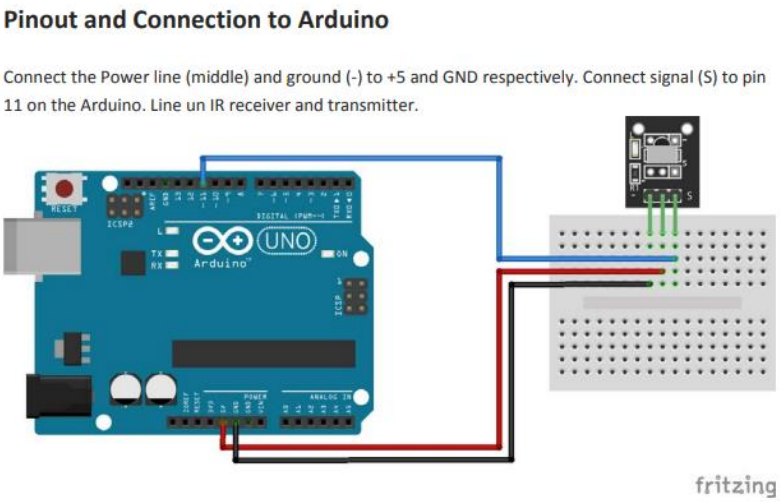
Appendix

Ir – Infrared

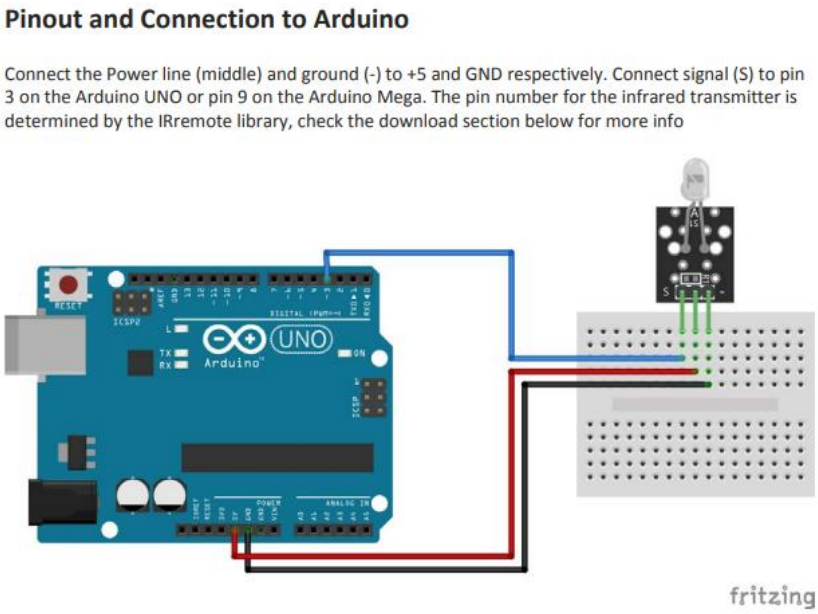
GPIO – General purpose input/output

IFTTT – If this then that

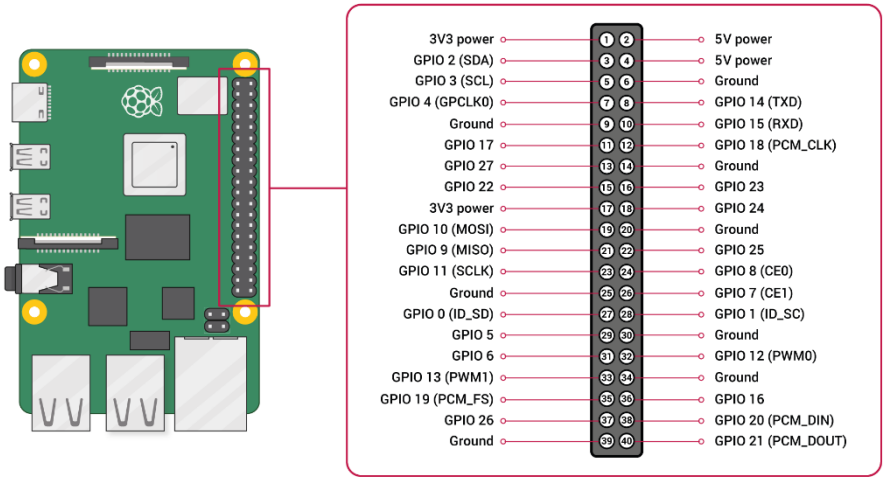
Schematic for ir-receiver



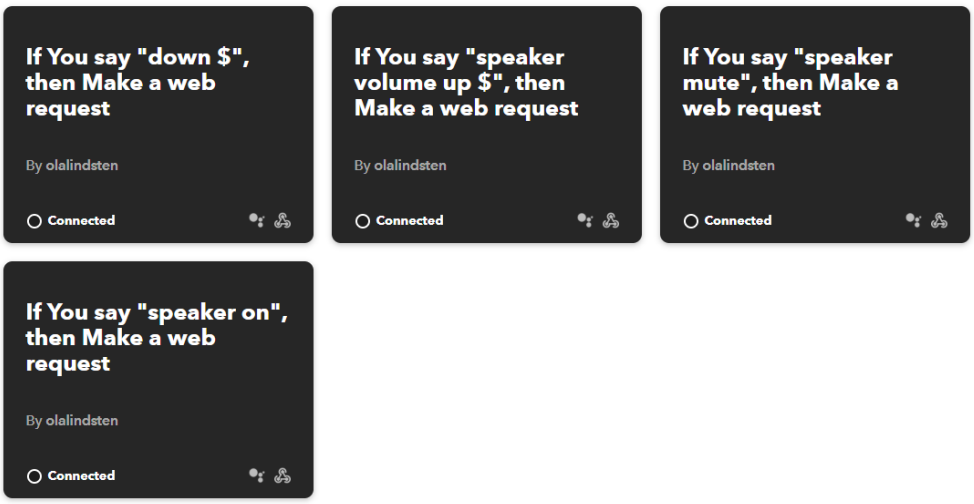
Schematic for ir transmitter



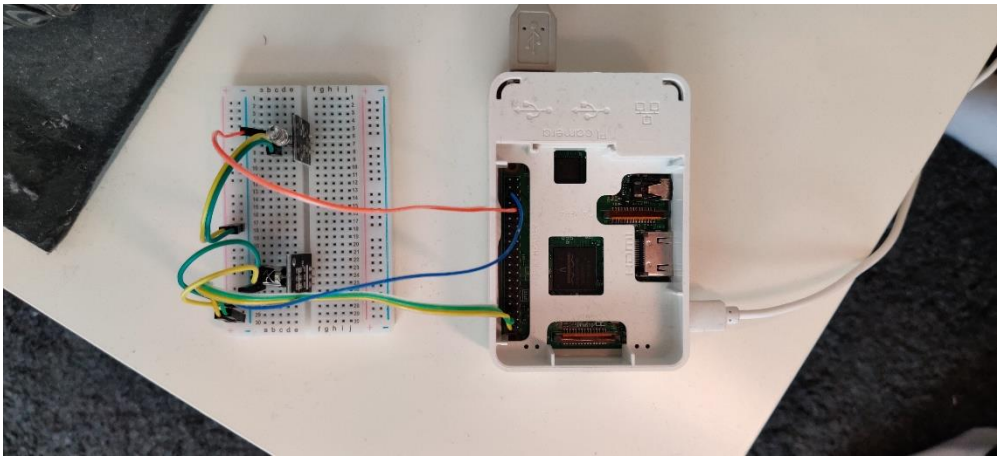
GPIOs for raspberry pi 3



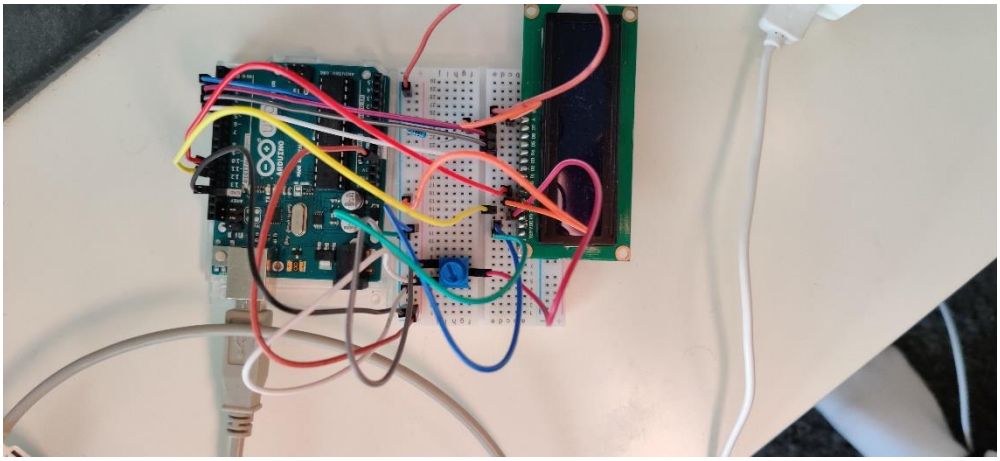
My different IFTTT commands



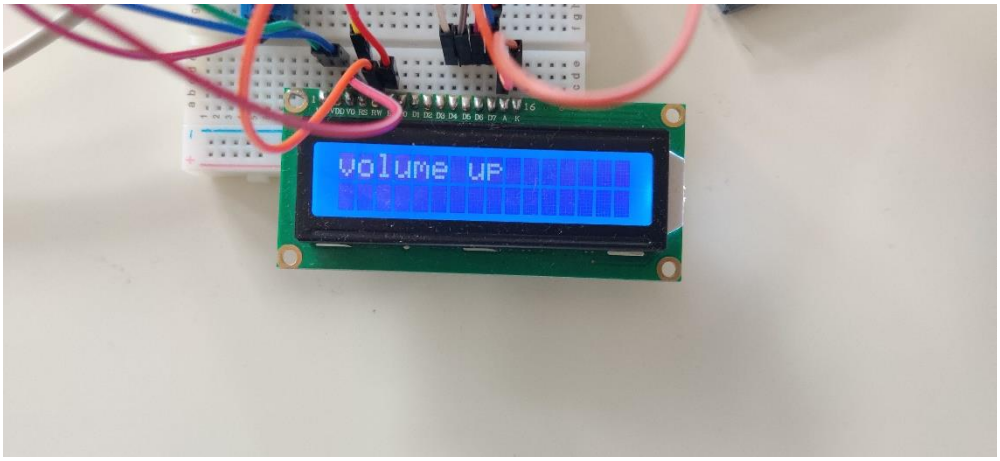
Picture of circuit raspberry pi + ir-receiver + ir-transmitter



Picture of circuit Arduino to LCD



Output when volume up is transmitted



Output when volume down is transmitted