Project Guideline

This project enables us to send commands from Google Assistant to Raspberry Pi to achieve home automation tasks with IFTTT, Gmail and Python.

IFTTT (If This Then That) is a web-based service which makes it possible to connect many different apps and services like Gmail, Twitter, Amazon, Google Assistant and more in order to write new rules and achieve some automated tasks, these tasks are named as applets. For example, you may write an applet for yourself which sends a tweet when you say your Google Assistant *"Hey Google, tweet 'Hello, world!'".* For this applet, "this" block stands for saying Google Assistant *"Hey, Google tweet 'Hello, world!'"* and "that" block stands for sending tweet as "Hello, world!". When "this" event is triggered, IFTTT allows us to proceed "that" block automatically.

There are many projects about home automation with Raspberry Pi and Google Assistant. Some of them are required to install different operating systems (Particle etc.) rather than Raspbian on Raspberry Pi, which limits the number of possible scenarios that you can achieve. Some of them use a service called Webhooks in IFTTT applets, this service provides you a special URL to make get and post web requests. You may write an applet to notify the user when a sensor which is wired to Raspberry Pi detects an object such as burglar-alarm. For this task, you write a Python script that checks the status of the pin that is wired to the sensor and when it is high value, you send a post request to Webhooks URL with just one-line code. You create an applet for this task, which contains "this" block to listen if any web request on this URL is triggered and "that" block to send a notification on the user's smartphone when any request is triggered. However, there are no examples about the reversed situation, in which you send post request with Google, Amazon or your smartphone and Python script listens to check any web request on the URL is triggered and do some tasks if any request exists. There are merely projects for this scenario with Django and Flask, but these are quite complicated if you don't have enough background for web applications, just like me.

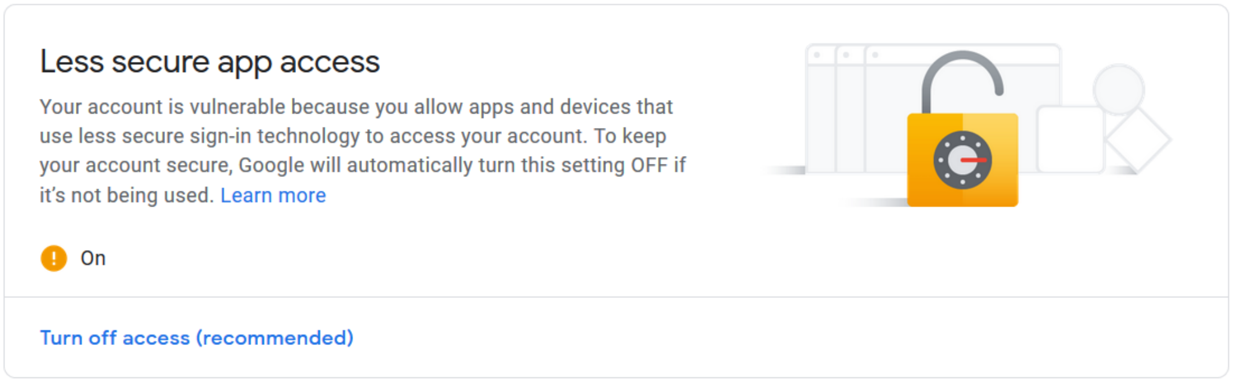
In this project, the main task is to send some requests from Google Assistant to Raspberry Pi to do some tasks according to these requests and this will be done through emails on Gmail account.

First Step: Creating Gmail Account for Home Automation

We will send our requests from Google Assistant to Raspberry Pi via emails. When we say a command to Google Assistant with a keyword, it will send an email which contains that keyword to Gmail account through the applet we will create. Python script will listen for unseen emails in Gmail with imaplib library and if there is an unseen email from IFTTT, it will read the content of the email and do some tasks according to the keyword in the email.

It is encouraged to create a new Gmail account for this project because of two main reasons: First, Python script will check for unseen messages in the inbox folder and it will mark the message as seen when it reads. In other words, it will mark all messages as seen as soon as it is executed, so you may miss some important messages in the inbox if you use your main Gmail account. Second, you should remove some security rules from Google account, which Google says it is unsecure in order to allow you access your emails with imaplib module in Python. For these reasons, you can create a new Google account which does not include any personal information and use this account for the project.

To work with imaplib on Gmail, the following privacy setting should be checked as "on" from "Manage Your Google Account / Security" section:



Less secure app access

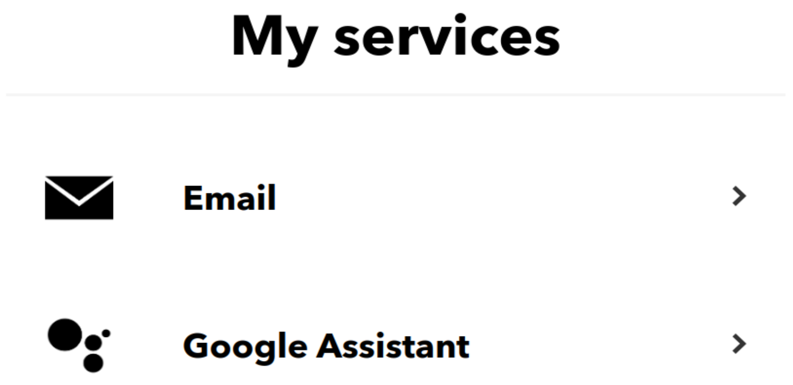
Your account is vulnerable because you allow apps and devices that use less secure sign-in technology to access your account. To keep your account secure, Google will automatically turn this setting OFF if it's not being used. Learn more

On

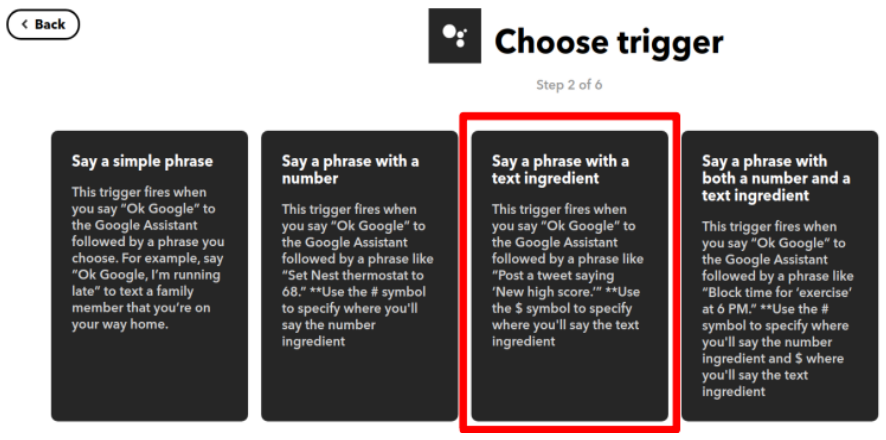
Turn off access (recommended)

Second Step: Creating Applet on IFTTT

We will first sign up on www.ifttt.com if we haven't before to use IFTTT applets. After successfully sign-in process, we search "Google Assistant" on the search bar, from "Services" tab we choose "Google Assistant" and connect our Google account which Google Assistant is related to. Similarly, we search "Email" on the search bar, we go to "Services" tab and choose "Email" service and click "Connect" button. We write Gmail email address that we created on the first step and "Email" services will be ready after the activation process. If all these steps are worked correctly, you should see your services on "Explore/My Services" section which is placed at the top-right corner of the screen as below:



Up to now, we added required services to our profile for accomplishing this task, we will create our applet which use these services for the next step. From "Explore" section at the top-right corner we choose "Create" section. The page appears to create a new applet. We click "This" section to specify our "this" block and choose "Google Assistant" service. In the page, it asks you to choose trigger type for



Say a simple phrase
This trigger fires when you say "Ok Google" to the Google Assistant followed by a phrase you choose. For example, say "Ok Google, I'm running late" to text a family member that you're on your way home.

Say a phrase with a number
This trigger fires when you say "Ok Google" to the Google Assistant followed by a phrase like "Set Nest thermostat to 68." **Use the # symbol to specify where you'll say the number ingredient

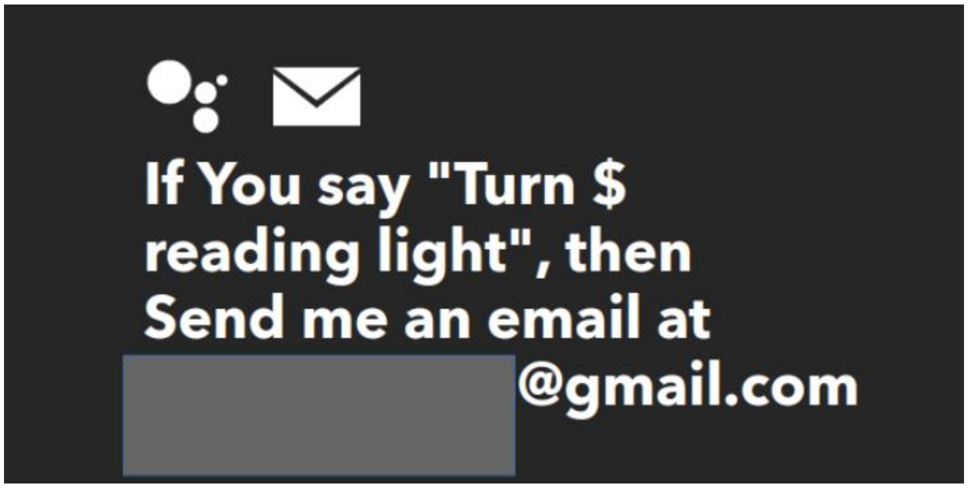
Say a phrase with a text ingredient
This trigger fires when you say "Ok Google" to the Google Assistant followed by a phrase like "Post a tweet saying 'New high score.'" **Use the $ symbol to specify where you'll say the text ingredient

Say a phrase with both a number and a text ingredient
This trigger fires when you say "Ok Google" to the Google Assistant followed by a phrase like "Block time for 'exercise' at 6 PM." **Use the # symbol to specify where you'll say the number ingredient and $ where you'll say the text ingredient

your assistant. If you want to trigger an event without any keyword, you would choose “say simple phrase” trigger type. If you want to trigger an event and send a keyword to the receiver service, you choose “say a phrase with a text ingredient”. In our project, we will send different keywords to make Raspberry Pi do different actions, that’s why we choose “say a phrase with a text ingredient” trigger type.

In next page, we will configure our “this” block to interact with Google Assistant. In the sample project, I will use Google Assistant to turn on and off my reading light on my table which is powered by USB hub. If I say Google Assistant *“Hey Google, turn on/off reading light”*, Python script will power on/off Raspberry Pi USB Hub and the reading light on the hub will turn on or off. The keyword for this task is “on” and “off” words so I write “Turn $ reading light” in “What do you want to say?” section. “$” here stands for keyword or text ingredient which will be sent in the email. We complete all sections for the trigger as we want.

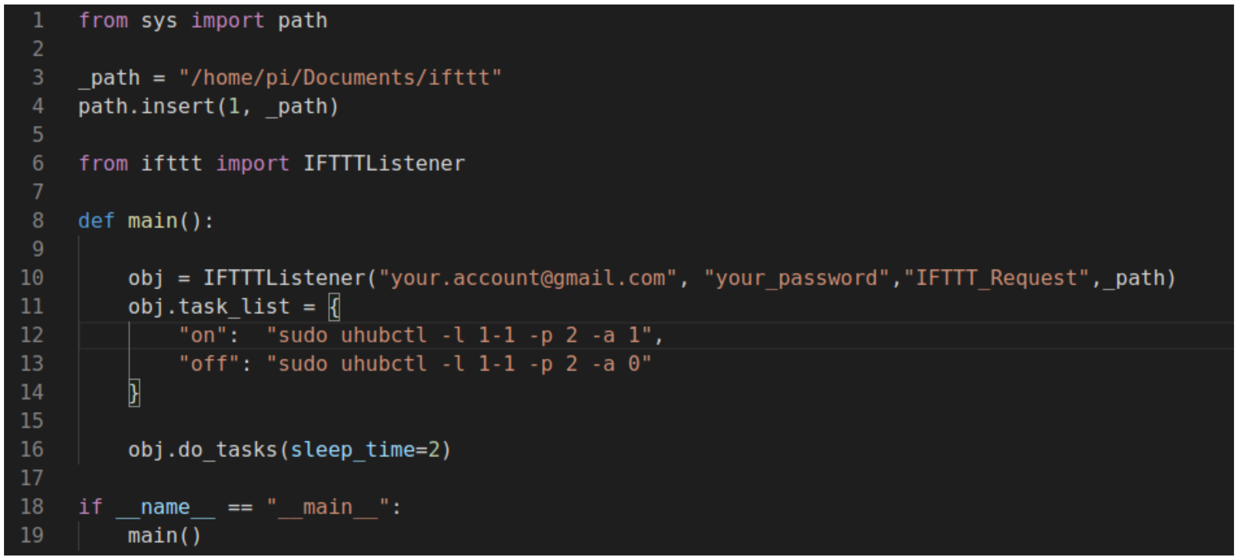
Next, we click “That” section to configure our action settings. We choose “Email” service and “Send me an email” action type. For the subject of email, we can name the subject to clear the email content, we will later use this subject title in Python script. I will write “IFTTT_Request” for the subject. We will not change the body section of the email, it will include “TextField” which represents the keyword for our events. When we click “create action” and finish editing, the applet will be ready to be used.



If You say "Turn $ reading light", then Send me an email at [redacted]@gmail.com

Final Step: Configuring Raspberry Pi for Home Automation

We will configure our Raspberry Pi to listen Gmail inbox and do some tasks according to new messages (or events). We copy ifttt folder and its content to our Raspberry Pi device at any directory we want. In my sample, I will move the folder to “/home/pi/Documents” directory. After you move ifttt folder, open “listener.py” script and set _path variable to the directory you moved the folder. In “obj” object, change class parameters according to your Gmail account address and the password and change “subj” parameter to the subject you named in your applet. In “obj.task_list” parameter (which is a dictionary), you will describe keyword-action pairs to do automated tasks. Key is the keyword you send with Google Assistant command (represented with “$” in the applet), and value is the command for terminal for the action you want to execute when an event with the keyword is received. In my sample project, when an event is received with “on” keyword, “sudo uhubctl -l 1-1 -p 2 -a 1” command is executed on the terminal, which powers on USB hub and turn on the reading light.



```python
from sys import path

_path = "/home/pi/Documents/ifttt"
path.insert(1, _path)

from ifttt import IFTTTListener

def main():

    obj = IFTTTListener("your.account@gmail.com", "your_password","IFTTT_Request",_path)
    obj.task_list = {
        "on":  "sudo uhubctl -l 1-1 -p 2 -a 1",
        "off": "sudo uhubctl -l 1-1 -p 2 -a 0"
    }

    obj.do_tasks(sleep_time=2)

if __name__ == "__main__":
    main()
```

We want to execute this Python script automatically when Raspberry Pi boots up. For this reason, we will use cron job. We open the terminal on Raspberry Pi and write this command:

```
sudo crontab –e
```

We will modify this file to execute “listener.py” script at the reboot. We move to the end of the file and write this command:

```
@reboot /usr/bin/python3 /home/pi/Documents/ifttt/listener.py &
```

If you move ifttt folder to a directory different than “/home/pi/Documents/”, change the command with your directory. “&” is important here because it makes the script run in the background. Because “obj.do_tasks()” contains while loop, Python program never ends until we cancel the process. If “&” does not exist in our command, the script will not be executed in the background so Raspberry Pi never completes the boot process. After you write this command at the end of the file, press “Ctrl + X” and press “y” to save the existing file. You write “reboot” command to restart Raspberry Pi and the script is executed automatically at the reboot.

Finally, you may cancel Python process after Raspberry Pi is rebooted. The script writes “config.txt” file in the ifttt folder, which contains “cancel:0” line. If you want to cancel the process, you may edit this line as “cancel:1” and save it. After Python reads this line, it will break while loop and the script will end. It will create “log.txt” file in ifttt folder, which contains the date and time of the cancellation process occur. If you reboot Raspberry Pi, it will run at the reboot again.

Happy coding…