Voice controlled home automation using google assistant

By

Ojas Srivastava

And

Yatharth Jain

For fulfillment of requirements towards

EED379 : Internet of Things

Under supervision

Of

Dr. Rohit Singh & Prof. Prem Chand Jain

Overview

- We will try to control our Home Appliances using Google Assistant. The Google Assistant will be used for understanding the Human commands & language.
- We will try to control our Home Appliances using Google Assistant. The Google Assistant will be used for understanding the Human commands & language.
- In this project we have explore two different types of IoT communications standards ,i.e, Cloud-based communication & MQTT
- This will trigger our LED on our Raspberry Pi to switch ON or OFF on our command

---

Flow of the Project

1. ***ThingSpeak***
    a. Google Assistant
        i. Giving command to Google Assistant in Natural Language.
    b. IFTTT Service
        i. We trigger an applet using IFTTT service to generate a WebHook to send data to ThingSpeak Cloud
    c. ThingSpeak Cloud
        i. The data pushed from the Applet is written in our private channel field.
    d. RPi Processing
        i. he data on ThingSpeak cloud is read by the Raspberry Pi and processed to switch ON/OFF the LED

2. ***MQTT***

    a. Google Assistant
        i. Giving command to Google Assistant in Natural Language.

    b. IFTTT Service
        i. We trigger an applet using IFTTT service to write to a feed in our personal MQTT account.

    c. Adafruit MQTT
        i. The data pushed from the Applet is written in our private feed.

    d. RPi Processing
        i. The data on MQTT feed is read by the Raspberry Pi and processed to switch ON/OFF the LED

Google Assistant

In this project we have used Google Assistant to understand the natural language of humans. Google Assistant helped us make custom commands for it using the IFTTT platform.

We used three different commands for different operations and assigned it to different integer values.

Following are the commands and their encoded values to determine the operation of Raspberry Pi:

1. Ok Google, "Turn ON light" == 1
2. Ok Google, "Turn Off the light" == 2
3. Ok Google, "End my Experiment"==3

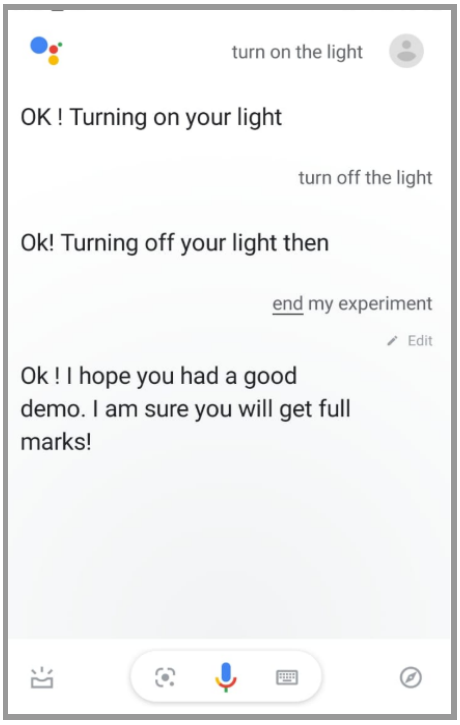

Figure: Demo of Google Assistant

IFTTT

IFTTT: If This Then That is a web-based Applet making service which is used to make to integrate Google Assistant with other services.

IFTTT gives us an option to make custom command for Google Assistant, which When fed to Google Assistant, generate a trigger to the IFTTT Applet. In this project We have used it to trigger two different services : WebHooks & Adafruit IO

1. **Webhook**
    a. Upon getting the trigger from Google Assistant, we deploy a webhook which is a standard Web communications helper.
    b. A webhook initializes a connection between the sender and receiver nodes and then listens to the messages sent by the sender. It works like a message highway between two nodes.

c. Here webhooks were used to send the data to ThingSpeak API using GET web communication protocol.
d. One can find the write API URL of their private ThingSpeak Channel after creating a channel and then creating a field.
e. The URL provided by ThingSpeak API was then used to send data by embedding the data in the URL itself as it used GET protocol

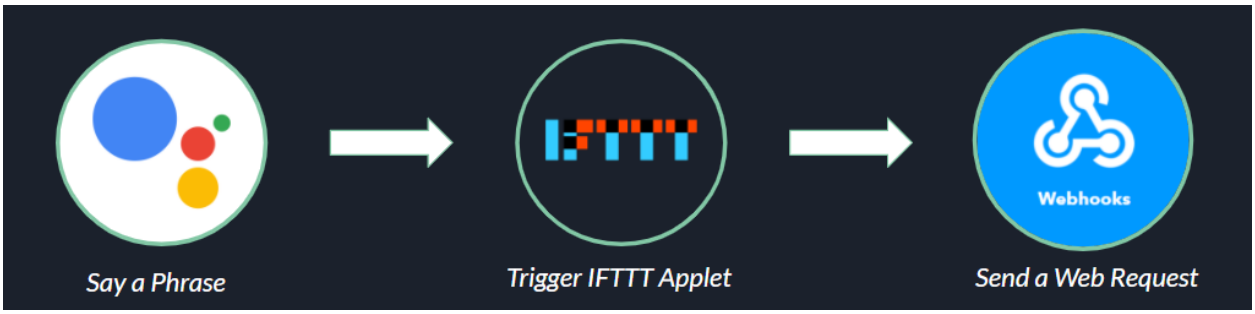

Figure: Flow for ThingSpeak

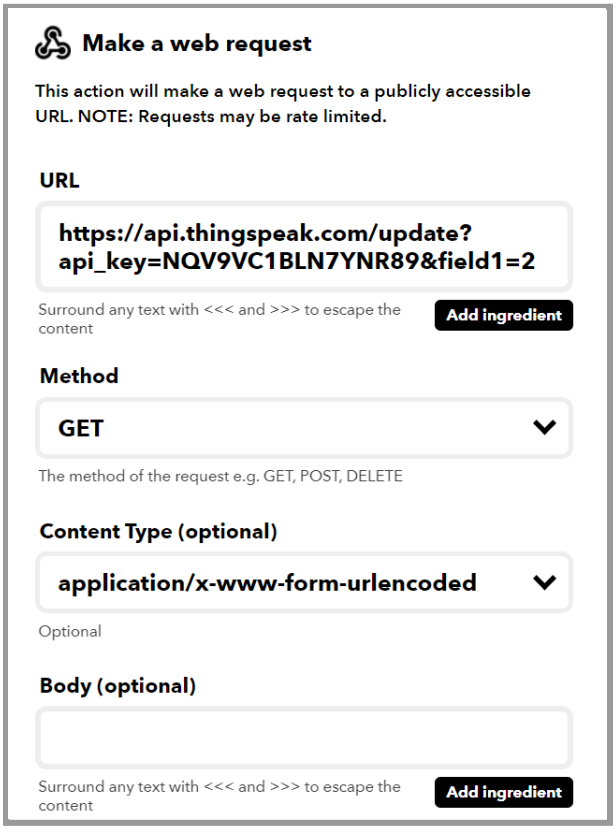

Figure : Parameters for making a Web Request from IFTTT

2. **Adafruit IO**
    a. Adafruit MQTT is directly integrated with IFTTT and hence can directly be triggered from there .
    b. It does not require any API calls or URL requests as in the ThingSpeak Case.

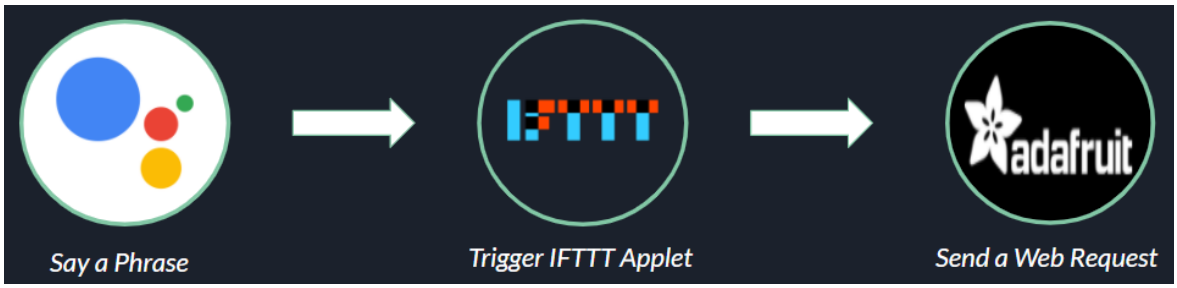

Figure: Flow of MQTT

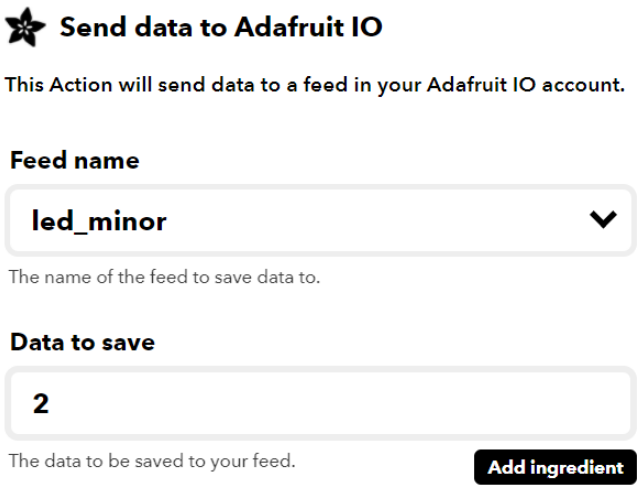

Figure: Parameters to be given for MQTT in IFTTT

Thing Speak Cloud

We made a free account on Thingspeak.com using our MATLAB accounts. We then Created a private channel and created one field inside it. The Write API key of the channel was then used to supply information from IFTTT and the read API key was used in the Rpi program to fetch the data from the cloud.

Adafruit IO MQTT

We made an account on *io.adafruit.com* and then created a feed named “led_minor”. Data was then supplied by IFTTT and then retrieved by Raspberry Pi by the Adafruit MQTT library Adafruit_IO, documentation for which is available on its Github repository.

Raspberry Pi Processing

Raspberry Pi was used as an interface between the LED light and the command that was uploaded either to the ThingSpeak cloud or MQTT feed. We used different inbuilt modules of python and raspberry pi to retrieve data from the cloud or feed then according to the data received power was altered at pin 6 of the board to which LED was already connected.

Code-ThingSpeak Cloud

```
"""
This File is for Mini Project in EED379 :Internet Of Things

Course : Internet Of Things EED379
Component : Mini Project
Topic: Home Automation using Google Assistant
```

```
Authors: Ojas Srivastava
        Yatharth Jain

Code Component Description : This code is client side for the Raspberry Pi
Node.
                             This program will fetch information from
ThingSpeak Cloud and load it on to the
                             program variables. Upon doing this the
program will use the status sent by the cloud to
                             switch ON or OFF our LED.
"""

# Importing Important Libraries
import urllib.request as urllib2  #urllib for opening any URL and fetching
data from there
import json  #All URLs return information in the form of JSON. This
library will help us decode it from JSON to Python object
import time

######Libraries specific to Rpi <TO BE FILLED>
import RPi.GPIO as GPIO

#Defining GLOBAL variables
READ_API_KEY= #Our READ API key for ThingSpeak Cloud
CHANNEL_ID= #Our Channel ID for ThingSpeak Cloud

#####GPIO
GPIO_pin = 6      #Pin connect to LED
GPIO.setmode(GPIO.BCM)
GPIO.setup(GPIO_pin,GPIO.OUT)

def switch_on():
    GPIO.output(GPIO_pin,GPIO.HIGH)
    print("LED is ON \n")

def switch_off():
    GPIO.output(GPIO_pin,GPIO.LOW)
    print("LED is OFF \n")
```

```
#main function is used as the fetcher of data and the command center for
the Rpi
def main():
    #print ("http status code=%s" % (conn.getcode()))  #used to get the
status code from the server, status code:200 is success
    i=1
    while(True):
        print(i)
        i+=1
        conn =
urllib2.urlopen("https://api.thingspeak.com/channels/1223567/fields/1.json
?api_key=WBETH6BSB6YPSMX4&results=2")
        #opening my ThingSpeak URL which gives data in GET mode.

        response = conn.read() #read the information presented in the URL
        data=json.loads(response) #the information is in JSON format and
hence needs to be unpacked

        #print (data)
        #The data is in dictionary format and we will access the "feeds"
Key
        temp=data['feeds'] #accessing "feeds" key
        dic = temp[0]
        f = dic["field1"]
        print ("Status is:" + f) #printing the current status


        if (f=="2"):
            switch_on()
        elif(f=="1"):
            switch_off()
        if (f=="3"): #ADD THIS FUNCTIONALITY TO IFTTT :Ojas
            switch_off()
            break
        #time.sleep(1) #pause for 1 second
    conn.close() #close connection

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

Code-Adafruit MQTT

```
"""
This File is for Mini Project in EED379 :Internet Of Things

Course : Internet Of Things EED379
Component : Mini Project
Topic: Home Automation using Google Assistant
Authors: Ojas Srivastava
        Yatharth Jain

Code Component Description : This code is client side for the Raspberry
   Pi Node.
                             This program will fetch information from
   MQTT.
                             Upon doing this the program will use the
   status sent by mqtt to
                             switch ON or OFF our LED.
"""

# Edit the variables below to configure the key,
# username, and feed to subscribe to for changes.

# Import standard python modules.
import sys

# Import Adafruit IO MQTT client.
from Adafruit_IO import MQTTClient

# Libraries specific to Rpi <TO BE FILLED>
import RPi.GPIO as GPIO

# Set to your Adafruit IO key.
# Remember, your key is a secret,
# so make sure not to publish it when you publish this code!
ADAFRUIT_IO_KEY = #Add you Adafruit Key here
```

```
# Set to your Adafruit IO username.
# (go to https://accounts.adafruit.com to find your username)
ADAFRUIT_IO_USERNAME = #Add your username here

# Set to the ID of the feed to subscribe to for updates.
FEED_ID = #Add feed name here

################## RASPBERRY-PI CODE ########################
#####GPIO
GPIO_pin =7     #Pin connect to LED
GPIO.setmode(GPIO.BCM)
GPIO.setup(GPIO_pin,GPIO.OUT)
#
def switch_on():
    GPIO.output(GPIO_pin,GPIO.HIGH)
    print("LED is ON \n")

def switch_off():
    GPIO.output(GPIO_pin,GPIO.LOW)
    print("LED is OFF \n")



################# RASPBERRY- PI CODE ################

# Define callback functions which will be called when certain events
   happen.
def connected(client):
    # Connected function will be called when the client is connected to
   Adafruit IO.
    # This is a good place to subscribe to feed changes.  The client
   parameter
    # passed to this function is the Adafruit IO MQTT client so you can
   make
    # calls against it easily.
    print('Connected to Adafruit IO!  Listening for {0}
   changes...'.format(FEED_ID))
    # Subscribe to changes on a feed named DemoFeed.
    client.subscribe(FEED_ID)
```

```
def subscribe(client, userdata, mid, granted_qos):
    # This method is called when the client subscribes to a new feed.
    print('Subscribed to {0} with QoS {1}'.format(FEED_ID,
   granted_qos[0]))

def disconnected(client):
    # Disconnected function will be called when the client disconnects.
    print('Disconnected from Adafruit IO!')
    sys.exit(1)

def message(client, feed_id, payload):
    # Message function will be called when a subscribed feed has a new
   value.
    # The feed_id parameter identifies the feed, and the payload
   parameter has
    # the new value.
    print('Feed {0} received new value: {1}'.format(feed_id, payload))
    if (payload=="2"):
        switch_on()
    elif(payload=="1"):
        switch_off()
    if (payload=="3"):
        switch_off()
        disconnected()



# Create an MQTT client instance.
client = MQTTClient(ADAFRUIT_IO_USERNAME, ADAFRUIT_IO_KEY)

# Setup the callback functions defined above.
client.on_connect    = connected
client.on_disconnect = disconnected
client.on_message    = message
client.on_subscribe  = subscribe

# Connect to the Adafruit IO server.
client.connect()
```

```
# Start a message loop that blocks forever waiting for MQTT messages to
   be
# received.  Note there are other options for running the event loop
   like doing
# so in a background thread--see the mqtt_client.py example to learn
   more.
client.loop_blocking()
```

Final Comments

- We were able to automate the process of switching on and off of LED using both ThingSpeak Cloud and MQTT Feed.
- The Limitation that we encountered during the process were for the case when we were using ThingSpeak Cloud as because we were using the free version of the cloud service, we were getting a delay or lag while uploading the data to the cloud. Because of this reason the whole process was facing lag and hence the performance was hampered.
- On the other hand, while using MQTT feed for the exact same task there was a delay of only a few milliseconds or we can say that there was negligible delay. And because of this reason **MQTT feed performed much better than ThingSpeak Cloud.**

End of Document