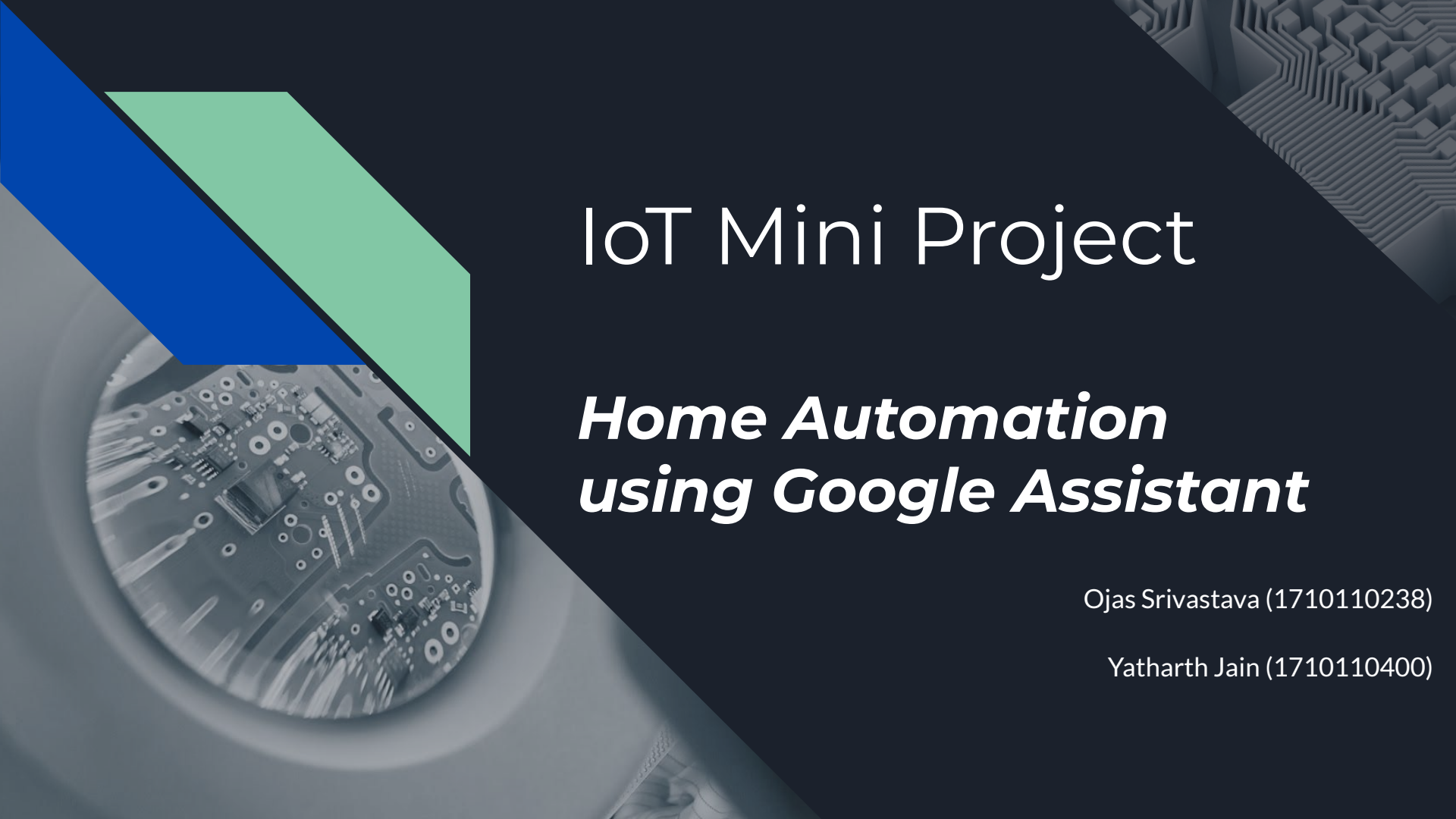


IoT Mini Project

Home Automation using Google Assistant

Ojas Srivastava (1710110238)

Yatharth Jain (1710110400)

Overview

01 We will try to control our Home Appliances using Google Assistant. The Google Assistant will be used for understanding the Human commands & language.

02 A custom command will be used to instruct the Google Assistant

03 This will trigger our LED on our Raspberry Pi to switch ON or OFF on our command

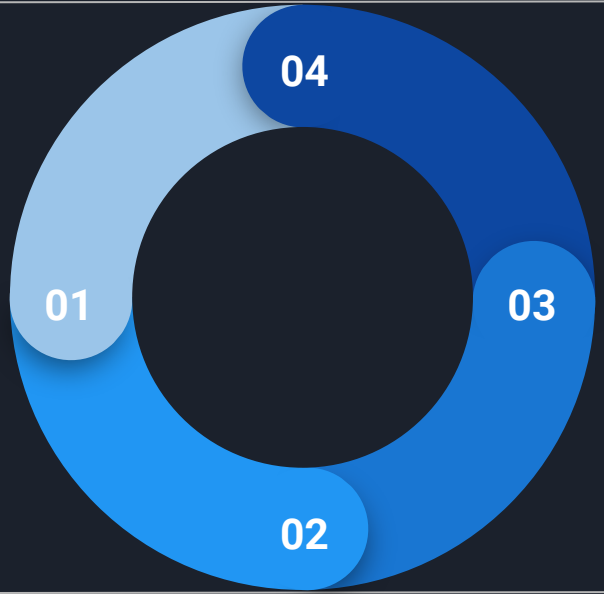
Flow diagram: ThingSpeak

Google Assistant

Giving command to Google Assistant in Natural Language.

IFTTT Service

We trigger an applet using IFTTT service to generate a WebHook to send data to ThingSpeak Cloud



RPi Processing

The data on ThingSpeak cloud is read by the Raspberry Pi and processed to switch ON/OFF the LED

ThingSpeak Cloud

The data pushed from the Applet is written in our private channel field.

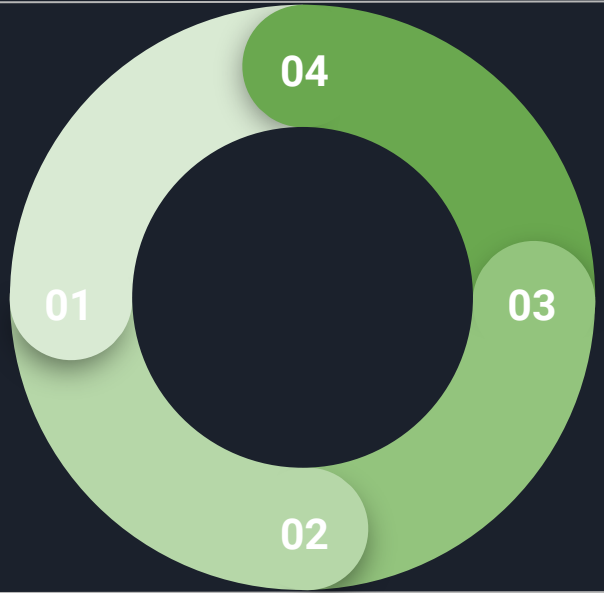
Flow diagram: MQTT

Google Assistant

Giving command to Google Assistant in Natural Language.

IFTTT Service

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



RPi Processing

The data on MQTT feed is read by the Raspberry Pi and processed to switch ON/OFF the LED

Adafruit MQTT

The data pushed from the Applet is written in our private feed.

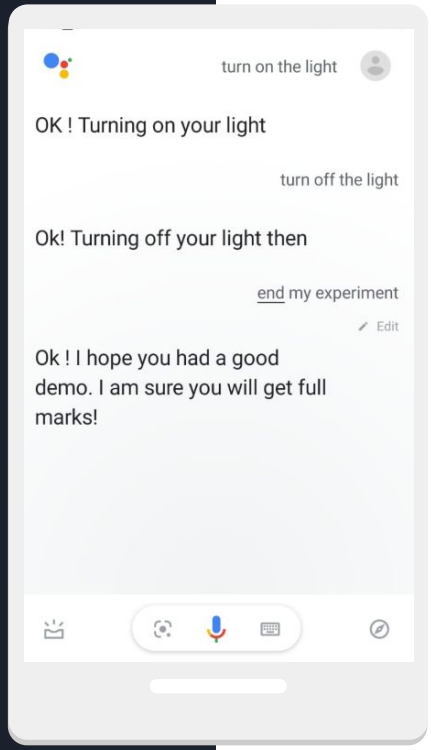
Commands to Google Assistant

We have entered three custom commands to Google Assistant :

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

Spotlight on mobile - Google Assistant

Ok Google Commands & Responses



This is our Google Assistant in action , understanding custom commands and giving custom outputs

STEP-2

IFTTT Applet: ThingSpeak

IFTTT -If This Then That service is used to make an Applet for:

- processing Google Assistant commands.
- A webhook is then used to send the information to the ThingSpeak Cloud using GET function.

Flow of Information is like follows :

Say a Phrase

Trigger IFTTT Applet

Send a Web Request

STEP-2

IFTTT Applet: MQTT

IFTTT -If This Then That service is used to make an Applet for:

- processing Google Assistant commands.
- The applet publishes data to the MQTT feed on Adafruit Platform.

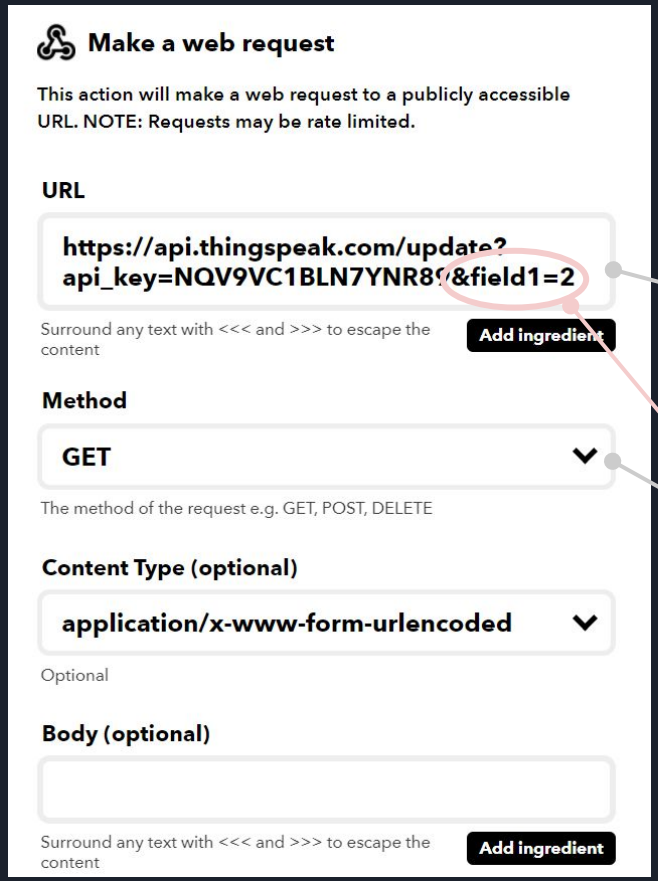
Flow of Information is like follows :

Say a Phrase

Trigger IFTTT Applet

Send a Web Request

WebHook Working: ThingSpeak



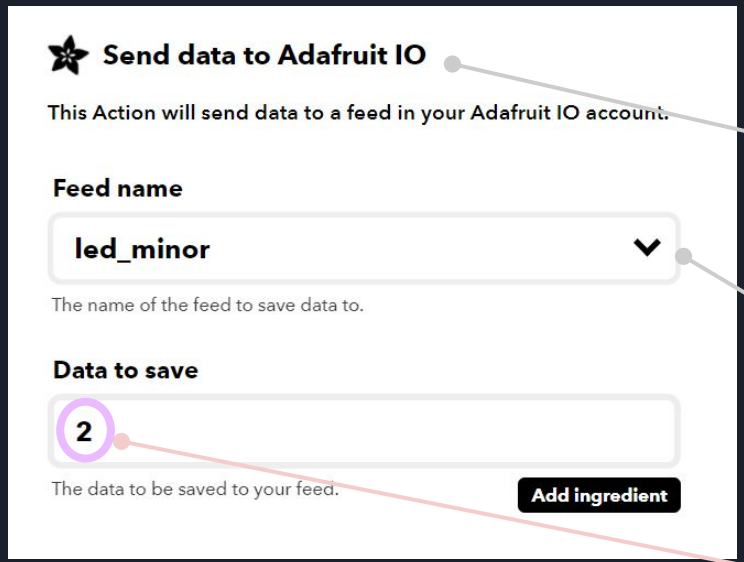
This is the URL is of our ThingSpeak Cloud Channel for writing data on to the channel. It has the following things:

1. Link of Thingspeak.com
2. Channel Write API Key
3. Value for field1, which here is set to 2

Method =GET , as is accepted form of web communication by ThingSpeak Cloud

1 = Switch ON; 2=Switch OFF ; 3=exit

MQTT Working



IFTTT has an in-built integration to call and connect to Adafruit IO platform.

One does not need to make an explicit API request as before.

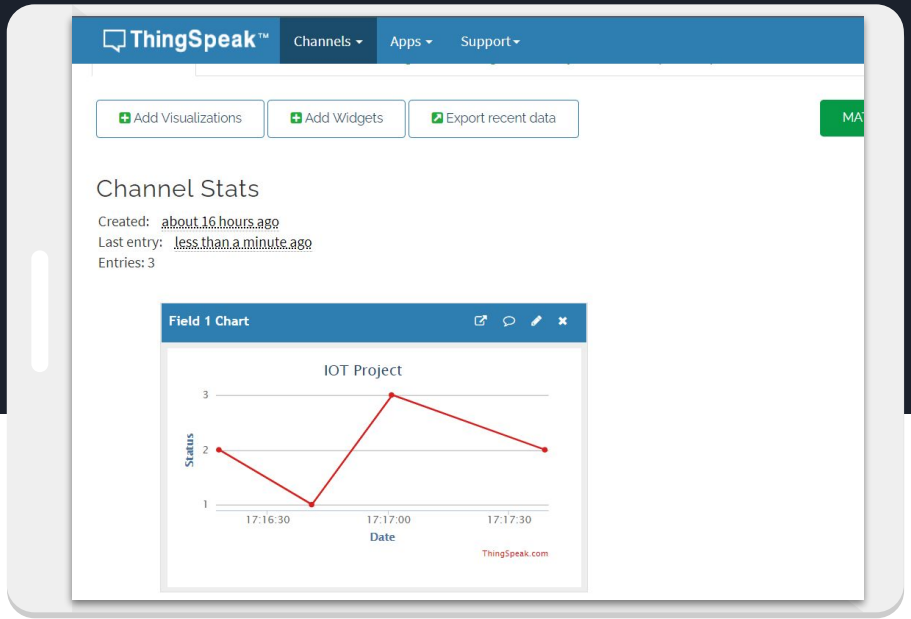
Name of our Private Feed

1 = Switch ON; 2=Switch OFF ; 3=exit

STEP-3

Data Loaded to ThingSpeak Cloud

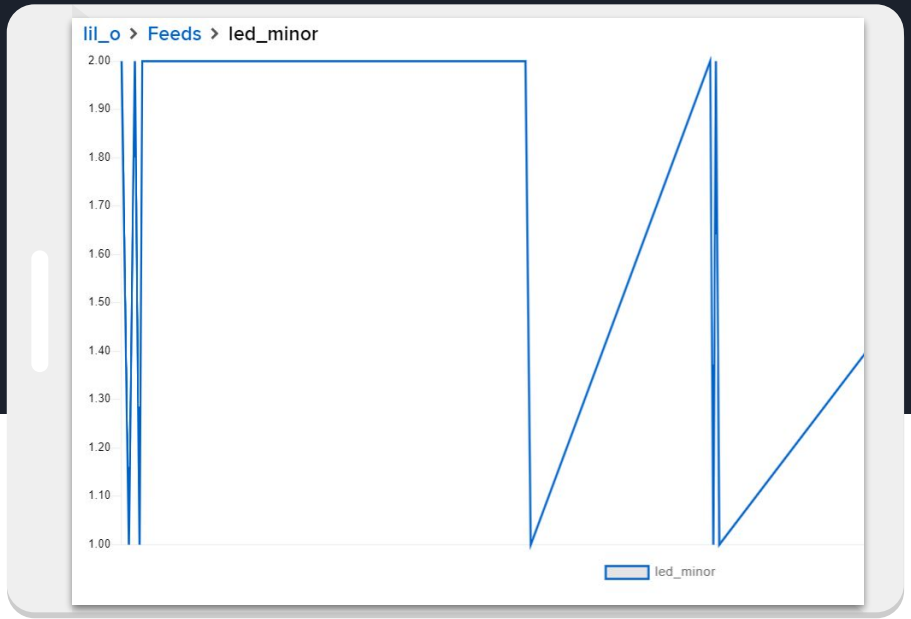
By WebHooks



One can see from the Field 1 chart that upon giving different commands to google assistant we are able to change the data in the field.

STEP-3

Data Loaded to MQTT Feed



One can see from the Field “led_minor” chart that upon giving different commands to google assistant we are able to change the data in the field.

STEP-4

Reading ThingSpeak Cloud Data to Raspberry Pi

The code follows the following steps to access the data :

1. Establishing a connection to the “READ” API function of ThingSpeak Cloud. This is done by using `urllib2.urlopen()`
2. Reading the response from the server via GET web communication protocol using `conn.read()`
3. Unpacking the server response in the form of JSON type to Python Object using `json.loads()`
4. Finally accessing the “Field Data” from the Python Object (Dictionary in this case)

```
import urllib.request as urllib2
import json

conn =
urllib2.urlopen("https://api.thingspeak.com/channels/1
223567/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

temp=data['feeds'] #accessing "feeds" key
dic = temp[0]
f=dic["field1"]
print ("Status is:" + str(f)) #printing the current
status
```

STEP-4

Reading MQTT Feed Data to Raspberry Pi

The code follows the following steps to access the data :

1. Establishing a connection to the API function of MQTT Feed.
2. Reading the response from the server via GET web communication protocol using `conn.read()`
3. Unpacking the server response in the form of JSON type to Python Object using `json.loads()`
4. Finally accessing the “Field Data” from the Python Object (Dictionary in this case)

Code on Next Slide -->

```
# Import Adafruit IO MQTT client.
from Adafruit_IO import MQTTClient

ADAFRUIT_IO_KEY = 'aio_Kjuo38CtXi803ZMMhMsnTJBAL0WS'
ADAFRUIT_IO_USERNAME = 'lil_o'
FEED_ID = 'led_minor'

# Define callback functions which will be called when
certain events happen.
def connected(client):
    print('Connected to Adafruit IO!  Listening for
    {0} changes...'.format(FEED_ID))
    client.subscribe(FEED_ID)

def subscribe(client, userdata, mid, granted_qos):
    print('Subscribed to {0} with QoS
    {1}'.format(FEED_ID, granted_qos[0]))

def disconnected(client):
    print('Disconnected from Adafruit IO!')
    sys.exit(1)
```

```
def message(client, feed_id, payload):
    print('Feed {0} received new value:
    {1}'.format(feed_id, payload))
    if (payload=="2"):
        switch_on()
    elif(payload=="1"):
        switch_off()

# 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()
client.loop_blocking()
```

STEP-5: Common to MQTT & ThingSpeak

Raspberry Pi acting on data

The code follows the following steps to process the data :

1. Based on the data received from the cloud/Feed we initiate either of the two function definitions.
2. If `status =2` then we run `switch_on()`
3. If `status =1` then we run `switch_off()`

```
import RPi.GPIO as GPIO

#####GPIO
GPIO_pin = ##Pin Number
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")
```

Results

- 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 few milliseconds or we can say that there was negligible delay. And because of this reason MQTT feed performed much better than ThingSpeak Cloud.

Thank you!