Programming Internet of Things

Pranav Shrikhande
SUNY Albany
pshrikhande@albany.edu

***Abstract*—This is a report of the project in ICSI 680. Here we study what is Internet of things, its application. How to program a device. How a device can make smart decisions based on data available. How embedded device can impact the overall functioning. In this project we have used Lua script. Embedded device used was ESP8266 Node MCU. We understand overall architecture of ESP8266 Node MCU and how it is programmed and how it can be used to retrieve data. We also see how sensors could be connected and high volumes of data could be obtained that is valuable and gives insight to various decisions. We also study about network devices and various other protocols such as HTTP, MQTT, IFTTT**

Keywords:- IoT, Network devices, sensors with ability to collect data, actuators, Internet, Embedded computing devices, Unique identifiers, Processing and utilizing data, protocols

I. Introduction

IoT represents a general concept for the ability of network devices to sense and connect data from the world around us and then share data across Internet, where it can be processed and utilized for various interesting purposes. IoT is internetworking of physical devices embedded with electronics, software, sensors, actuators and network connectivity that enable these objects to collect and exchange data. IoT is a system of interrelated computing devices mechanical or digital machine, object, animals or people that are provided with unique identifiers and the ability to transfer data identifiers and the ability to transfer data over network without requiring human to human or human to computer instruction or interaction.[3]

IoT Components could be
1 sensors.
2. Acutators
3. Embeded Computing device
4. Unique identifiers
5. Network
Iot devices are tied together by a network using various wireless or wire line technologies, standards and protocol to provide connectivity. For example few protocols are Zigbee protocol, MQTT protocol etc. Wifi, Wired LAN, Bluetooth etc can be used

Now its not required for the sensors individually to be connected to internet. All the sensors might be connected to single device sharing the temperature values with it. Main device is connected to internet sending data over the internet cloud for processing [3]

II. Related Work

Providing a consistent and composable interface for standalone devices is critical for device usability and creating applications as the devices become more numerous. Another system that targets these goals is the Thing System [6]. The Thing System provides a web server that presents a common GUI for interacting with a range of devices. Each supported device has device interface into the Thing System server. While the Thing System is motivated by the same general problem, our approach differs in several key aspects. First, we focus on simplifying the task of writing the per-device Lua script wrapper by eliminating any framework level API that the wrapper must be compatible with. This reduces the number of lines of code in a wrapper for the same device by half or more. Second, our system provides a mechanism for easily creating synthetic devices with more useful, higher-level interfaces from existing devices. Third, our system is designed as a kernel that can be embedded in other applications such as smartphone apps or web services.

III. Standard Interface Model

For this project we use several hardware and software components. To start we use ESP8266 Node MCU development board. Bread Board, Resistors of 220 ohm, Light Emitting Diodes, Jumping wires, DHT22 Sensor. Android Cell Phone so that we can configure voice command with help of Google Assistant or we could also use Google mini and Personal Computer where we code and push on to the development board.

On Software side we make use of firmware bundle that comes with NodeMCU documentation where we use different modules such as DHT, Encoder, GPIO, I2C, Wi-Fi etc. We use JDK 8.0. To write scripts we use Lua Script. We use ESPLorer IDE to push code onto Development board and for testing and debugging we use ZeroBrane studio. Further in project we then access Thingspeak cloud for data visualization that is dynamically obtained data is plotted on it. We then move on to MQTT cloud that act as a MQTT Broker for our clients so as to ensure the publisher/subscriber model of MQTT(Message Queuing Telemetry Transport). We use IFTTT.com(If this then that protocol) so as to send messages while setting up the trigger according to our project. We also use VS code for NodeJS for sending and receiving data on Amazon cloud. Finally we use AWS Lambda where we deploy and get api so that all this project is running over internet.

IV. Understanding IoT

Section II: Understanding Iot Suppose you have smart AC at home, lets assume that office is 20 miles from home, at time you leave office, if outside temperature is more than 80 deg Fahrenheit, AC immediately turns on when you are at distance of 5 miles from home. Here AC is able to make decision hence its smart. Embedded system with computational capabilities is part of IoT which detects location using internet, checks outside temperature and smartly takes decision when to turn on AC

Embedded IoT Platforms:- There are various boards available on embedded IoT platforms such as Raspberry Pi, Aurdino, ESP 8266, Linkit, BeagleBone, Intel Galileo etc. We use ESP 8266 extensively in this project.

Why we use Embedded system?

1. Autonomous:- we can build a system, specific to particular application.
2. Low cost.
3. Low Space.
4. Low Power: Devices generally operate at 5 volt

To program embedded system various languages could be used such as Java, Python, Javascript, LuaScript, C etc. we extensively use LuaScript

Also various IoT platforms are available such as Thingspeak, Cloud MQTT, Thingworx, IBM Watson, AWS IoT, Azure IoT, PubNub etc

V. ESP 8266

This board is also known as Node MCU development board. Here Node MCU is open source framework and ESP development board is hardware

ESP 8266 has a WiFi system on a chip. It is highly integrated chip designed to provide full internet connectivity in a small package.

ESP8266 chip can be programmed very easily. This chip is a standalone chip which means it doesnt need a support of Adruino or Raspberry Pi to take data from sensors.

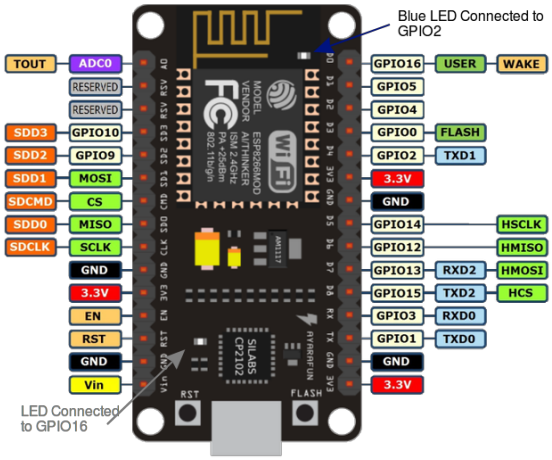

Fig. 1. [5]

APC	File	MQTT	SPI	WebSocket
BME280	GPIO	NET	TIMER	Wi-Fi
CJSON	HTTP	Node	U8G2	WS2801
DHT	I2C	PWM	UART	ENCODER
MDNS	SNTP	UCG		

Fig. 2.

A. *5.1 Understanding ESP8266*

ESP 8266 development board is a relatively simple board Pins are broken out into parallel rows that are compatible to breadboard

Maximum voltage is 3.6 volts. It has USB connector, RESET button which controls power of ESP8266, Flash button, it has LED that indicates power, charge and status of chip

Node MCU Firmware allows to do the programming using Lua script, with few lines of code we can establish a Wi-Fi connection, control GPIO pins, establish a Wi-Fi connection, control GPIO pins and turn development board into webserver.

We use following modules

Next we configure ESP8266 as web-server. we connect ESP8266 to Wi-Fi and obtain IP address, following is the code in Lua Script

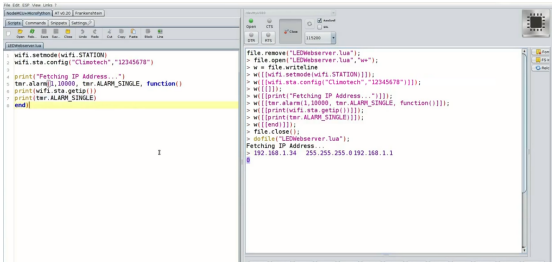

Fig. 3.

Now to create a webserver we use the custom in built function net.CreateServer(). Here we have to pass two parameters that is net.CreateServer(type, timeout). Type could be TCP and UDP. We use TCP and timeout is defined in seconds. Here we can only create one TCP server. More than one will be erroneous. Following is the code

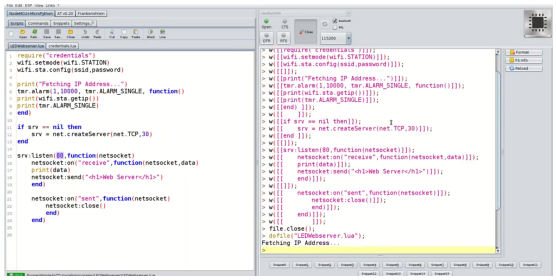

Fig. 4. []

B. *5.2 Working with DHT22 Sensor*

We use DHT22 Sensor for obtaining temperature and humidity. we use DHT_read() module that is custom built. This is available from NODE MCU module online. Now DHT_read returns five variables that are 1) status 2) temperature 3) humidity 4) temperature in decimal 5) humitidy in decimal. Following is the lua script for configuring DHT22 sensor

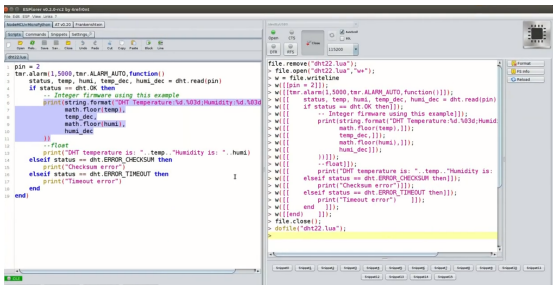

Fig. 5.

VI. Visualization

Here we use Thing-speak cloud data visualization. Thing-speak is an open data platform for IoT.
It allows to collect data in own channel and publish it in the form of easy graphs
We can collect, store, analyze, visualize and act on data from sensors.
Typical Thing-speak work flow

1) lets you create a channel and collect data.
2) Analyze and visualize the data
3) Act on Data

Following is the code where we access Thing-speak API which through Luascript where we pass API key and values through HTTP

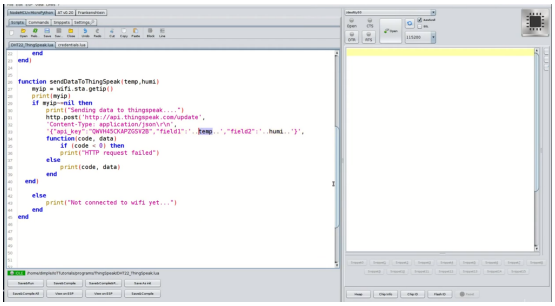

Fig. 6.

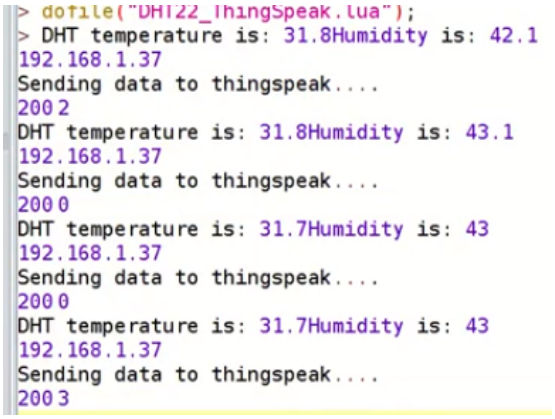

Fig. 7.

And data is sent to Thing-speak in following way

After sending data, Thing-speak Cloud data visualization plots two graphs.

6.1 Temperature

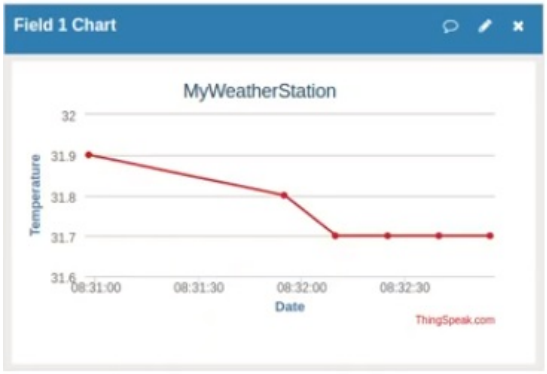

Fig. 8. Temperature Graph

6.2 Humidity

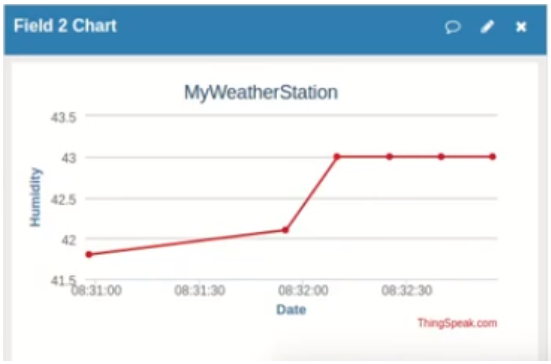

Fig. 9. Humidity graph

VII. MQTT

MQTT stands for message queuing telemetry transport. MQTT is a protocol for machine to machine and IoT applications. This is based on publisher/subscriber model. In MQTT there can be several clients. Here ESP8266 Node MCU developer board could be client. Raspberry pi with temperature sensor could be another client.

Each client must have unique identification. Its similar to unique email id in a Google group. Publisher/Subscriber in MQTT is same as that of google group. Group name in google group is analogous to term Topic in MQTT.

In google group we create new topic, in MQTT we would send and receive messages on topic. Google server is one where thousands of groups are running. In MQTT we call it MQTT broker. MQTT broker is one which makes it possible to run thousands of topics concurrently

Google Group	MQTT (Publish/Subscribe)
User1, User2.....	Client1, Client2.....(e.g: ESP8266)(Unique Id)
Publisher	Publisher
Subscriber	Subscriber
Group Name (Programming Internet of things)	Topic
New Topic	Message
Google Server	MQTT Broker

Fig. 10.

A. 7.1 Scenario 1: When the motion is detected, switch on LED and turn on buzzer

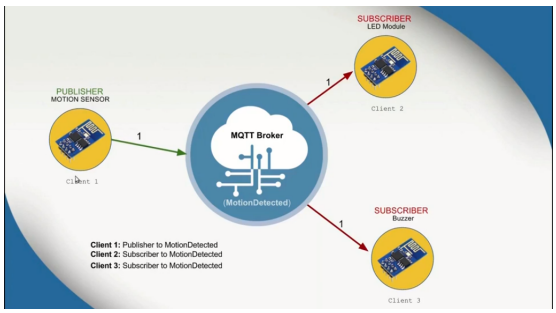

Fig. 11.

Here we have client 1 with unique ID. This act as a publisher, once motion is detected it will send message Motion Detected to broker. MQTT broker will publish message to client 2 and client 3 as they are subscribers of topic Motion Detected. When message reaches client 1, it turns on LED module and when it reaches client 3 it turns on Buzzer

7.2 Scenario 2: When the motion is detected, switch on LED and turn on buzzer. When LED is turned ON, display status of LED

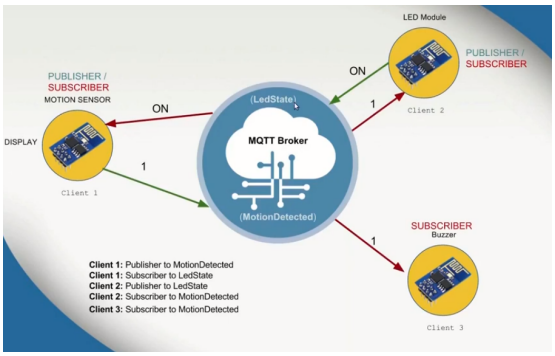

Fig. 12.

Client 1, We have attached motion sensor with display, when motion is detected, client 1 sends message with a value 1 with a topic Motion detected. Since client 2 and client 3 are subscribers to topic motion detected. 1 will be pushed to both clients. Client 3 will turn on the buzzer, Client 2 will switch on LED. Once LED is switched on it will publish message ON to topic LED State. Now as soon as message is published MQTT broker will push message is published. MQTT broker will push message to client 1 as client 1 is subscriber to topic LED state. This way client 1 will come to know LED is turned ON and will display current state of LED on display Module.

7.3 Features of MQTT

1) Push instead of pull messages
2) Reliable even when used with unreliable network.
3) Ideal for constraint network. (Low bandwidth, high latency etc)
4) Easy to implement
5) Three Quality of service levels
QOS 0 :- Atmost once delivery
QOS 1 :- Atleast once delivery of message
QOS 2:- Exactly once delivery of message
As per requirement we can choose 0,1,2 for programming 6) Last will and testament
a) Its client who defines last will and testament
b) Broker sends the message on behalf of the client after clientdied
c) Real PUSH

7.4 Working with Cloud MQTT

CloudMQTT.com provides a free service. So we use this in our project. we create instance on that website, following are the credentials that are required later to code MQTT

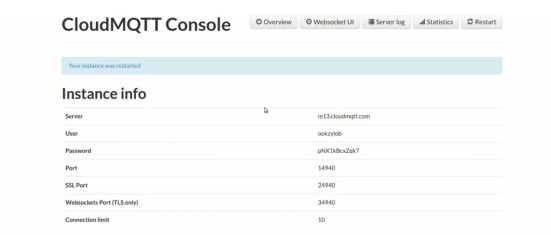

Fig. 13.

7.5 MQTT Implementation

We implement following scenario.

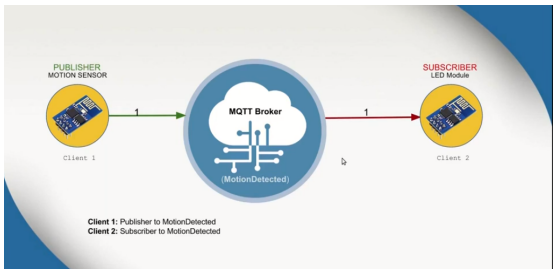

Fig. 14.

Here we created two topics 1)Motion LED 2) LED State. Here our MQTT broker is ”cloudMQTT.com”

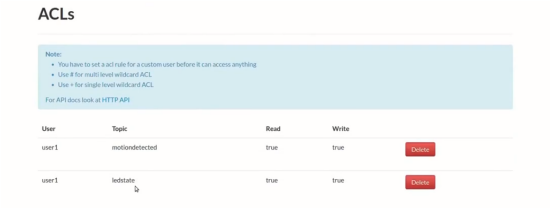

Fig. 15.

.

LuaScript to implement MQTT

```
require("credentials")
require("mqttconfig")
wifi.setmode(wifi.STATION)
wifi.sta.config(SSID,PASSWORD)

print("Fetching IP Address...")
tmr.alarm(1,10000, tmr.ALARM_SINGLE, function()
    myip = wifi.sta.getip()
    if myip~=nil then
        print(myip)
        mqttStart()
    else
        print("Error in connecting to wifi")
    end
end)

function mqttStart()
    m = mqtt.Client(CLIENTID1,120,"user1","password")
    m:connect(HOST,PORT,0,0,function(client)
        print("Connected...")
    end,
    function(client,reason)
        print("Reason.."..reason)
    end
    )

end
```

Fig. 16.

7.6 Problem of half open TCP connections and keep alive

1) Half open connection is a state where one of communicating parties is crashed or gets out of sync.
2) To keep alive functionally assures that connection is still open, and both broker and client are connected to on another
3) As long as messages are exchanged frequently and keep alive interval is not exceeded, there is no need to send a message to ensure that connection is still open.
4) Broker disconnects client which doesnt send any message in one and half times of keep alive interval.

VIII. IFTTT

IFTTT stands for If This Then That. IFTTT.com is an open website. It provides hundreds of services. Here we have to define IF condition. We here work with Google assistant. We select Google assistant module and then we define say a phrase. Here we define IF condition.
That condition: we choose custom service, we select SMS Connect SMS
So now we have set up our conditions. Next we install IFTTT application on cellphone. Now by giving a voice command Emergency that we defined in IF condition, we get message notification on cellphone stating Emergency at Home. .

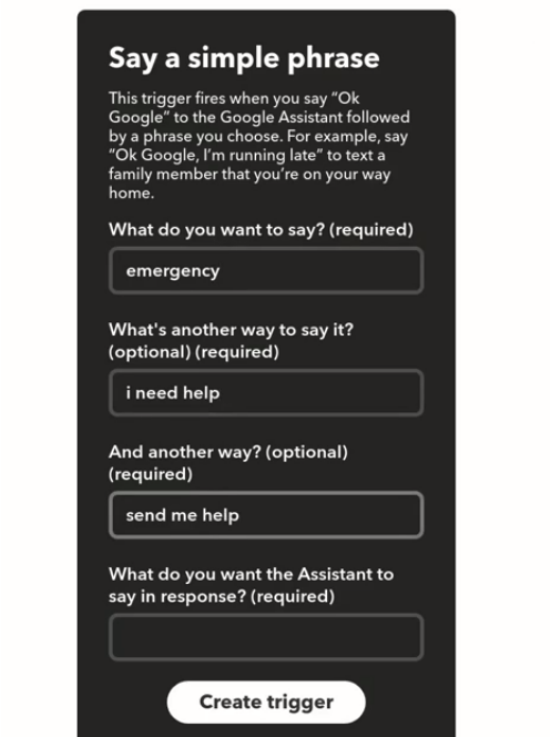

Fig. 17. IF Condition

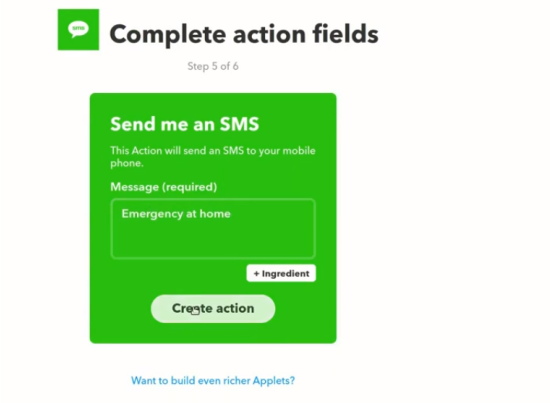

Fig. 18. That Condition

8.1 IFTTT - MQTTT We already covered scenario where client 1 communicates with client 2 using MQTT broker.

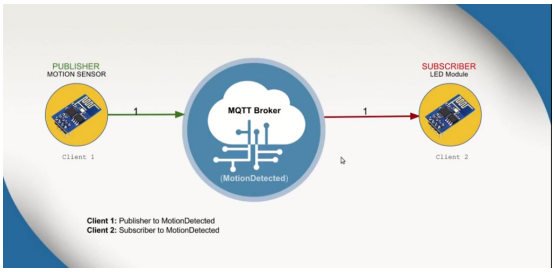

Fig. 19.

We also covered services of IFTTT

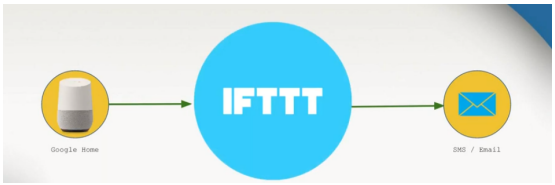

Fig. 20.

Now here we do the vive versa of the condition.

Fig. 21.

That is we give command on Google Assistant and it should turn ON/OFF LED on ESP8266 development board. Here we have to convert HTTP request to MQTT
Now here we need a middleware to convert HTTP request to MQTT and that should be recognized by MQTT broker.

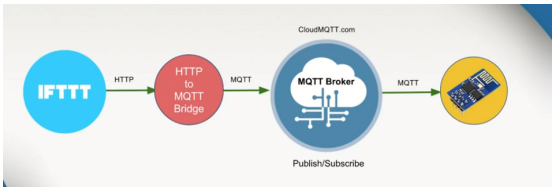

Fig. 22.

8.2 HTTP to MQTT Bridge using AWS Lambda
Why AWS Lambda?
1) AWS Lambda automatically runs your code without requiring you to provide or manage server
2) With Lambda you can run your code for virtually any type of application or backend service all with zero administration
3) Just write code and upload it to lambda
4) AWS lambda automatically scales application by running

code in response to each trigger
5) Your code runs parallel and process each trigger individually scaling precisely with the size of the workload
6) With AWS lambda you are charged for every 100 ms. Your code executes and number of times your code is triggered
7) You dont pay anything when code isnt running.
Now to configure AWS lambda we give name HttpToMqtt and we set runtime as NodeJS. NodeJS will be further used while accessing AWS lambda. Next on local computer we create a basic package.JSON file. This file is used to give information to NPM that allows to identify the project as well as enter projects dependencies. Next we write index.JS file that is expected by AWS Lambda.

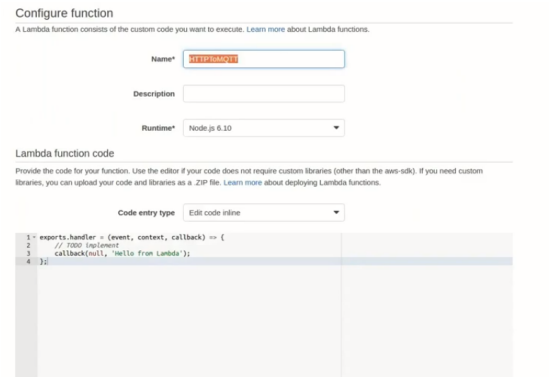

Fig. 23.

8.3 Coding HTTP to MQTT Bridge

We write following code in Javascript using NodeJS framework and then we zip with all dependencies and upload it on AWS Lambda.

Till now we have created a lambda function which has

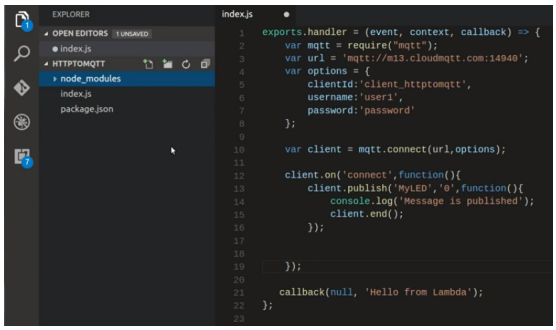

Fig. 24.

successfully published message to cloud MQTT server. Next we need url such as www.abc.com that should call our lambda function
i.e www.abc.com → Lambda Function → Cloud MQTT
So our Lambda Function is HTTP to MQTT bridge. Now we create HTTP URL, from there we pass parameter which will be received by lambda function.

8.4 Working with AWS API Gateway and Postman Service

Now here we have to create a new api and give name HttpToMqttApi. We go to resource and write it as HttpToMqttBridge and our lambda function HttpToMqtt. And in resource we scroll down to body and upload the zip file. Next we choose Deploy API and hence we see URL.

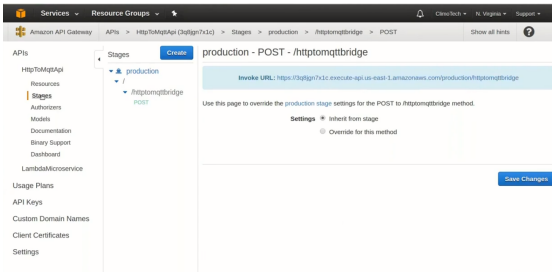

Fig. 25.

Now we goto postman.com to test service.

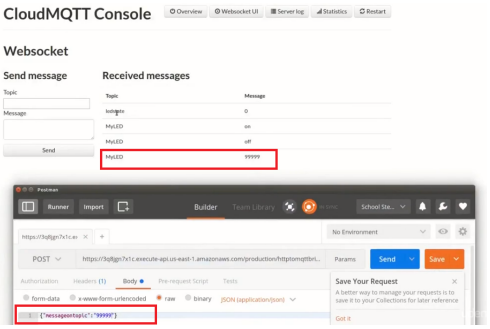

Fig. 26.

This means we have created HTTP to MQTT bridge. Now that we have tested our URL, we will use this in IF service next.

8.5 Integrating IFTTT with HTTP-MQTT bridge

We go same with IF Service that we saw earlier and describe phrase Turn the Light ON. In THAT service we select webhooks and post our API that we created. Now by giving voice commands we see changes in Cloud MQTT

8.6 HTTP to MQTT (Controlling LCD using voice Commands)

Here we write Lua Script to create HTTP to MQTT bridge that could work for ESP8266 Node MCU development kit. After this we give instruction to google assistant to Turn the Light ON and following is the output

After this we give instruction to Google assistant to Turn the Light OFF and following is the output.

Fig. 27. webhooks screen

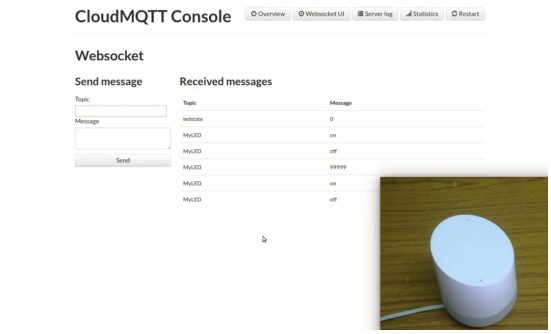

Fig. 28. voice commands and changes in Cloud MQTT

IX. Recap of Work Flow

1) On IFTTT.com we configured This and That condition, where this condition is Google assistant using voice command and That condition is makerwebhook service
2) Maker webhook service calls URL, now URL get parameters from makerwebhooks
3) URL belongs to AWS API gateway and calls POST method.
4) POST method further calls Lambda Function.
5) Lambda function is acting as HTTP to MQTT Bridge.
6) Now this lambda function publishes parameters as message to Cloud MQTT server on topic MYLED.
7) Now we have LED module configured with ESP8266 development board which is subscribed to the topic MYLED on cloud MQTT server.
8) This is how by voice command we control LED Module

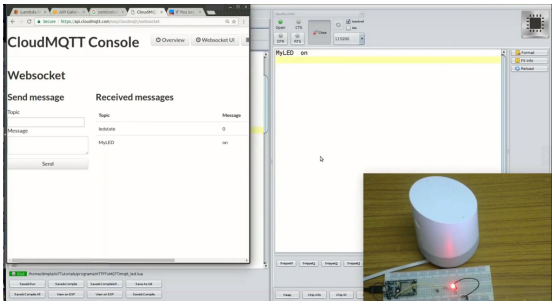

Fig. 29. When voice command given to turn the Light ON

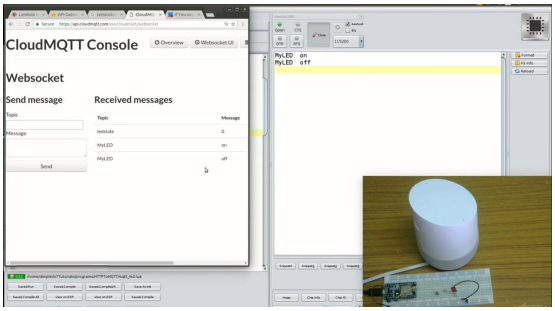

Fig. 30. When voice command given to turn the Light OFF

for ESP8266 development board.

X. Conclusion

Here we have programmed IoT devices. We used ESP8266 Node MCU development board, we worked we sensors and visualzied using Thingspeak cloud. Then we built MQTT protocol. We used IFTTT protocol so as to setup SMS/email push notification to understand PUB/SUB model. Then we developed AWS Lambda function and deployed our code there. Genreated API. By using API we again used IFTTT model to test voice commands and MQTT cloud and finally we configured our ESP8266 development board with MQTT HTTP bridge.

There are numerous benefits of using IoT devices as we devices can make decisions and become smart. Also edge computing such as Machine learning and Analytic could be applied with data obtained so as to improve performance

REFERENCES

[1] Internet Of Things Wikipedia, *9-May-2019 [Online]. Available: https:https://en.wikipedia.org/wiki/Internet_of _things [Accessed: 9-May-2019].*

[2] IFTTT,IFTTT, *9-May-2019 [Online]. Available: https://ifttt.com/ [Accessed: 9-May-2019].*

[3] Thigspeak*9-May-2019. [Online]. Available: https://thingspeak.com/ [Accessed: 9-May-2019].*

[4] Cloud MQTT,Wikipedia, *9-May-2019. [Online]. Available: https://www.cloudmqtt.com/ [Accessed: 9-May-2019].*

[5] Circuits for you . *[Online]. Available: https://circuits4you.com/2017/12/31/nodemcu-pinout/. [Accessed: 9-May-2019].*

[6] "AWS Lambda", AWS Lambda. [Online]. Available:*https://aws.amazon.com/lambda/ . [Accessed: 9-May-2019].*

[7] "Interfacing internet of trillion things", Interfacing internet of trillion things. [IEEE]. Available:*https://aws.amazon.com/lambda/ . [Accessed: 9-May-2019].*