International Journal of Modern Electronics and Communication Engineering (IJMECE) ISSN: 2321-2152
Volume No.-7, Issue No.-3, May, 2019

IOT Based Home Automation System using NodeMCU and MQTT

T. Aravinda Babu
Dept. of ECE,
Chaitanya Bharathi Institute of Technology
Hyderabad,India
Email Id: tummalaaravind@yahoo.co.in

Abstract: Internet of Things (IoT) conceptualizes the idea of remotely connecting and monitoring real world objects (devices) through the Internet. When it comes to our house, this concept can be aptly incorporated to make it smarter, safer and automated. We exploit this ubiquitous nature of Wi-Fi to connect our home devices to a cloud server as clients in order to get seamless access. MQTT is publish-subscribe-based messaging protocol and works on top of the TCP/IP protocol. Publish-subscribe messaging pattern requires a message broker which is implemented using cloud services. In this paper, Node MCU as a MQTT client which receives messages on topics which it is subscribed to from the cloud broker. We can control the appliances from any device which has an internet connection. Thus the existing infrastructure can be used to enhance the home appliances and make them smart. This implementation provides an intelligent, comfortable and an energy efficient home automation system.

Keywords: IOT, NodeMCU, MQTT

I. Introduction

IoT systems allow users to achieve deeper automation, analysis, and integration within a system. They improve the reach of these areas and their accuracy. IoT utilizes existing and emerging technology for sensing, networking, and robotics. IoT exploits recent advances in software, falling hardware prices, and modern attitudes towards technology. Its new and advanced elements bring major changes in the delivery of products, goods, and services; and the social, economic, and political impact of those changes. As automating of the appliances in our home makes the daily life easier to get on with, and because of its possibility of remote access of the appliances we find it highly useful. Advantages of home automation typically fall into a few categories, including savings, safety, convenience, and control. Additionally, some consumers purchase home automation for comfort. smart thermostats and smart light bulbs save energy, cutting utility costs over time. Some home automation technologies monitor water usage, too, helping to prevent exorbitant water bills. Certain devices even offer rebates. Many home automation technologies fall under the umbrella of home security. Consumers purchase these devices because they want to make their homes safer and more secure. Automated lighting thwarts would-be burglars, and motion sensors help people enter doors and walk hallways late at night. Because home automation technology performs rote tasks automatically, end users experience great convenience. Lots of smart gadgets are compatible with one another, and you can set different triggers between devices to automate regular home process. For instance, you could set your smart locks to turn on your smart lighting when you unlock the front door. Consumers also choose smart home devices to better control functions within the home. With home automation technology, you can know what's happening inside your home at all times. Some people use smart technology to record shows or to play music throughout the home. Connected devices can also help create a comfortable atmosphere they provide intelligent and adaptive lighting, sound, and temperature, which can all help create an inviting environment.

In recent years, wireless systems like Wi-Fi have become more and more common in home networking. Also, in home and building automation systems, the use of wireless technologies gives several advantages that could not be achieved using a wired network are reduced installation costs, System scalability and easy extension, Aesthetical benefits, Integration of mobile devices For all these reasons, wireless technology is not only an attractive choice in renovation and refurbishment, but also for new installations.

Implementing this paper will provide us the control of all appliances at our finger tips and the booming automation market wherein everyone wants a smart and automated house will find our project inspiring.

Organization of the paper, NodeMCU and its pin layout. Study of various modules used in the project and their contribution. MQTT protocol and its architecture. Overview of tool and platforms used in this project. Describes the implementation Procedure and the code, results and conclusion.

II. MQTT

Message Queuing Telemetry Transport (MQTT) is a light weight transport protocol. MQTT works on TCP and assures the delivery of messages from node to the server[1,2]. Being a message oriented information exchange protocol, MQTT is ideally suited for the IoT nodes which have limited capabilities and resources. MQTT is a publish/subscribe based protocol. Any MQTT connection typically involves two kinds of agents: MQTT clients and MQTT public broker or MQTT server. Data that is being transported by MQTT is referred to as application message. Any device or program that is connected to the network and exchanges application messages through MQTT is called as an MQTT client. MQTT client can be either publisher or subscriber. A publisher publishes application messages and subscriber requests for the application messages. MQTT server is a device or program that interconnects the MQTT clients. It accepts and transmits the application messages among multiple clients connected to it. Devices such as sensors, mobiles etc. are considered as MQTT client. When an MQTT client has certain information to broadcast, it publishes the data to the MQTT broker as shown in figure1. MQTT broker is responsible for data collection and organization. The application messages that are published by MQTT client is forwarded to other MQTT clients that subscribe to it. MQTT is designed to simplify the implementation on client by concentrating all the complexities at the broker. Publisher and subscriber are isolated, meaning they need not have to know the existence or application of other.

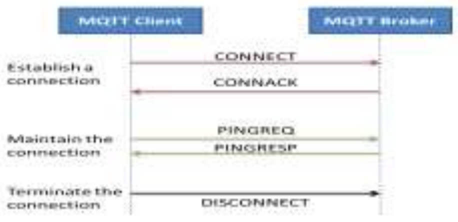


Figure1: Establishing, maintaining and terminating MQTT connection

Before transmitting the application messages, control packets are exchanged based on the QoS associated with them. An MQTT control packet consists of a fixed header, a variable header and payload. CONNECT, CONNACK, PUBLISH, PUBACK, PUBREC, PUBREL, SUBSCRIBE, SUBACK, etc. are some of the MQTT control packets exchanged between MQTT clients and MQTT server[3]. “Topic” in MQTT provide the routing information. Each topic has a topic name and topic levels associated with it. There may be multiple topic levels separated by / in a topic tree. Wildcard characters such as # and + are used to match multiple levels in a topic. Featuring the queuing system, MQTT server buffers all the messages if client is offline and delivers them to the client when the session is enabled.

A. Establishing a connection

Upon the successful establishment of network between the MQTT client and the MQTT server, control packets are exchanged between the client and the server. The client that wishes to connect to the MQTT server sends a CONNECT packet to the server specifying its identifier, flags, protocol level and other fields. The server acknowledges the client with the specified identifier through CONNACK packet with a return code denoting the status of connection [2].

B. Subscribing to a topic

If the MQTT client wants to subscribe to the application messages published on topic, a topic is a UTF-8 string, which is used by the broker to filter messages for each connected client. A topic consists of one or more topic levels. Each topic level is separated by a forward slash.

In comparison to a message queue a topic is very lightweight. There is no need for a client to create the desired topic before publishing or subscribing to it, because a broker accepts each valid topic without any prior initialization, it sends the SUBSCRIBE packet along with the topic name indicated in UTF-8 encoding. The server acknowledges the subscription with SUBACK packet along with a return code denoting the status of request. Once the subscription is successful, the application messages on the specified topic are forwarded to the client with the maximum QoS. To unsubscribe a topic, the client sends an UNSUBSCRIBE packet to the server which acknowledges it with the UNSUBACK packet.

C. Maintaining the connection alive

After a certain time-out, the connection between the client and the server is terminated. To maintain the connection, the client indicates that it is alive by transmitting a PINGREQ packet to the server[5]. The MQTT server responds to the client with the indicated identifier with a PINGRESP packet and maintains the connection alive as shown in figure2.

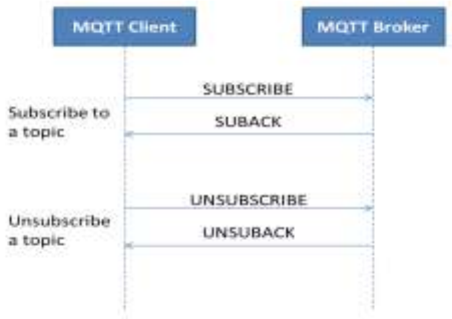


Figure2: Client subscribing and unsubscribing to the topic

D. Clean sessions and reliable connections

At the point when a subscriber associates with the broker, clean session association is considered as permanent, if its value is false. In this task, consecutive messages which come out conveying a highest QoS assignment are reserved for delivery when the association is resumed. Use of these flag is optional.

E.Terminating the connection

To terminate the connection, the MQTT client sends a DISCONNECT packet to the server. The server does not acknowledge this packet. However all the application messages related to the client will be flushed off and the client is disconnected from the server.

MQTT PUBLISH / SUBSCRIBE

MQTT pub/sub is the decoupling of publisher and receiver, which can be differentiated in more dimensions: Space decoupling: Publisher and subscriber do not need to know each other. Time decoupling: Publisher and subscriber do not need to run at the same time. Synchronization decoupling: Operations on both components are not halted during publish or receiving. In summary publish/subscribe decouples publisher and receiver of a message, through filtering of the messages it is possible that only certain clients receive certain messages. The decoupling has three dimensions: Space, Time, Synchronization.

MQTT ARCHITECTURE

The typical MQTT architecture can be divided into two main components as shown in figure 3. Each component briefly described below.

1) **Client**: Client could be a Publisher or Subscriber and it always establishes the network connection to the Server (Broker). It can do the following things [7]:

- Publish messages for the interested users.
- Subscribe in interested subject for receiving messages.
- Unsubscribe to extract from the subscribed subjects.
- Detach from the Broker.

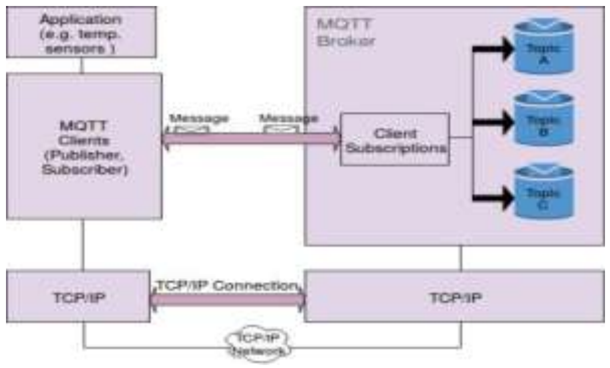


Figure 3: MQTT Architecture

*2) **Broker**:* Broker controls the distribution of information aand mainly responsible for receiving all messages from ppublisher, filtering them, decide who is interested in it and ttthen sending the messages to all subscribed clients[7]. It can do ttthe following things:

- Accept Client requests.
- Receives Published messages by Users.
- Processes different requests like Subscribe and Unsubscribe from Users as shown in figure4.

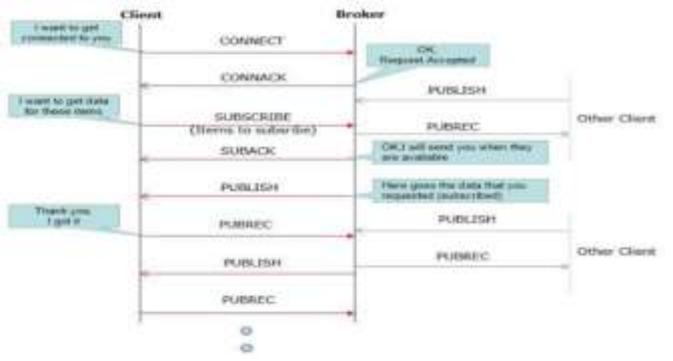

Client
Broker
I want to get connected to you
CONNECT
OK Request Accepted
CONNACK
PUBLISH
I want to get data for these items
SUBSCRIBE (Items to subscribe)
Other Client
PUBREC
SUBACK
OK,I will send you when they are available
PUBLISH
Here goes the data that you requested (subscribed)
Thank you I got it
PUBREC
PUBLISH
PUBLISH
PUBREC
Other Client
PUBREC

Figure 4: Working of MQTT

III. NodeMCU

An open-source firmware and development kit that helps you to prototype your IOT product within a few Lua script lines. It is based on the eLua project, and built on the ESP8266 SDK 1.4. It uses many open source projects, such as lua-cjson, and spiffs. It includes firmware which runs on the ESP8266 Wi-Fi SoC, and hardware which is based on the ESP-12 module [5].

Arduino-like hardware IO: Advanced API for hardware IO, which can dramatically reduce the redundant work for configuring and manipulating hardware. Code like arduino, but interactively in Lua script.

Nodejs style network API: Event-driven API for network applicaitons, which faciliates developers writing code running on a 5mm*5mm sized MCU in Nodejs style. Greatly speed up your IOT application developing process.

Lowest cost WI-FI: WI-FI MCU ESP8266 integrated and esay to prototyping development kit. We provide the best platform for IOT application development at the lowest cost.

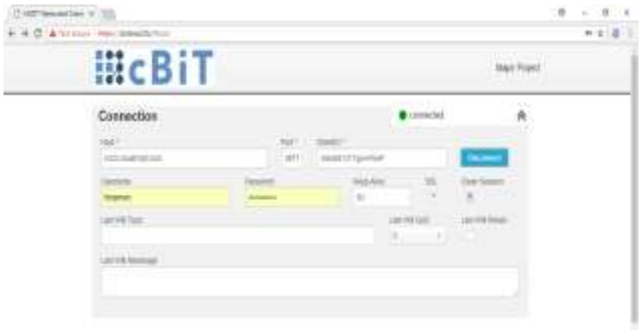
NodeMCU has general purpose input output pins on its board. We can make it digital high/low and control things like LED or switch on it. General-purpose input/output (GPIO) is a pin on an IC (Integrated Circuit). It can be either input pin or output pin, whose behaviour can be controlled at the run time. NodeMCU Development kit provides access to these GPIOs of ESP8266. The GPIO's shown in blue box (1, 3, 9, 10) are mostly not used for GPIO purpose. ESP8266 is a System on Chip (SoC) design with components like the processor chip. The processor has around 16 GPIO lines, some of which are used internally to interface with other components of the SoC, like flash memory.Since several lines are used internally within the ESP8266 SoC, we have about 11 GPIO pins remaining for GPIO purpose. Now again 2 pins out of 11 are generally reserved for RX and TX in order to communicate with a host PC through which compiled object code is downloaded. Hence finally, this leaves just 9 general purpose I/O pins i.e. D0 to D8. RX, TX, SD2, SD3 pins are not mostly used as GPIOs since they are used for other internal process. But we can try with SD3 (D12) pin which mostly like to respond for GPIO/PWM/interrupt like functions[6]. NodeMCU based ESP8266 has Hardware SPI with four pins available for SPI communication. With this SPI interface, we can connect any SPI enabled device with NodeMCU and make communication possible with it.

This ESP8266 Arduino core framework brings support for ESP8266 chip to the Arduino environment. It lets you write sketches using familiar Arduino functions and libraries, and run them directly on ESP8266, no external microcontroller required. ESP8266 Arduino core comes with libraries to communicate over WiFi using TCP and UDP, set up HTTP, mDNS, SSDP, and DNS servers, do OTA updates, use a file system in flash memory, work with SD cards, servos, SPI and I2C peripherals.

ESP8266WIFI LIBRARY

The Wi-Fi library for ESP8266 has been developed basing on ESP8266 SDK, using naming convention and overall functionality philosophy of Arduino WiFi library. Over time the wealth Wi-Fi features ported from ESP9266 SDK to esp8266/Adruino outgrew Arduino WiFi library. In the line WiFi.begin("network-name", "pass-to-network")replace network-name and pass-to-network with name and password to the Wi-Fi network you like to connect. Then upload this sketch to NodeMCU module and open serial monitor. Devices that connect to Wi-Fi network are called stations (STA). Connection to Wi-Fi is provided by an access point (AP),that acts as a hub for one or more stations. The access point on the other end is connected to a wired network. An access point is usually integrated with a router to provide access from Wi-Fi network to the internet. Each access point is recognized by a SSID (**S**ervice **S**et IDentifier),that essentially is the name of network you select when connecting a device (station) to the Wi-Fi.

ESP8266 module can operate as a station, so we can connect it to the Wi-Fi network. It can also operate as a soft access point (soft-AP), to establish its own Wi-Fi network. Therefore we can connect other stations to such ESP module. ESP8266 is also able to operate both in station and soft access point mode. This provides possibility of building e.g. mesh networks. The ESP8266WiFi library provides wide collection of C++ methods (functions) and properties to configure and

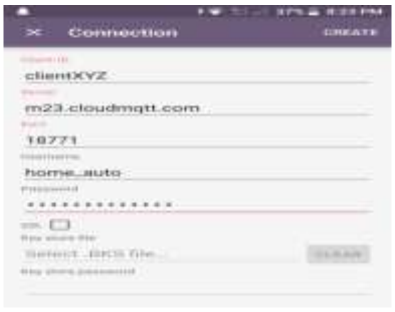


operate an ESP8266 module in station and / or soft access point mode.

IV. Implementation

There are two ways to access our Home automation system, one through a website and another through an Android application interfaces are shown in figure 5 and 6.

Fig5: Android App user interface



Connection

clientXYZ

m23.cloudmqtt.com

18771

home_auto

****

Figure6: Website user interface

These two user interfaces are used to give the commands to the home automation system according to the requirement of the user will build system block diagram as shown in fig.7. The commands to switch ON/OFF the appliances are given either from the android app or from the website. NodeMCU takes these commands and does the appropriate action, i.e

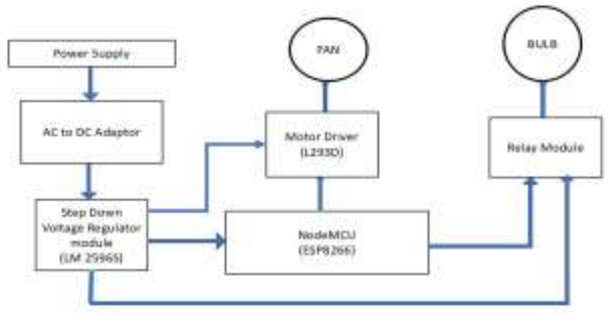


Figure7: Block Diagram of Home automation system.

sending logic 1 or 0 to the intermediate modules (Motor driver and Relay) which ultimately control the Fan and Bulb. The power supply of 5v is provided to all the modules through a Step-down voltage regulator (LM 2596S).

Hardware Implementation

First we have to setup Arduino ide so that we can burn the code into NodeMCU board. The following steps are to be followed:

- Start Arduino and open Preferences window.
- Enter http://arduino.esp8266.com/stable/package_esp8266com_index.json into Additional Board Manager URLs field. You can add multiple URLs, separating them with commas.
- Open Boards Manager from Tools > Board menu and find esp8266 platform.
- Select the version needed from a drop-down box.
- Click install button.
- Select ESP8266 board from Tools > Board menu after installation.

In order to make our ESP8266 a MQTT client we use PubSubClient. The pubsubclient provides many functions like connecting to a mqtt broker, subscribing to a topic, publishing to a topic, setCallback, reconnect, etc. We create a pubsubclient object then use that object to call the functions defined in pubsubclient library. So the code above uses this library to implement the functionality required.

Web Sockets Client Ui In Web-Browser

We will be using cloudmqtt, a mosquito broker hosted in the cloud. In order to communicate with the broker and tell it to send commands to our NodeMcu we have designed a Web sockets UI which will act as a client to publish data to cloudmqtt[4]. Then this published data will be routed to our NodeMCU which is using a pubsubclient library and appropriate action will be taken.

The steps required to designing a websocket client which works on all major browsers is as follows:

- First we will require the paho java script client which can be downloaded from the official website.
- This script includes all the major functions required to connect to a MQTT broker.
- We have to include this “mqttws31.js” javascript library in the pages where we want to use the functionality using : <script src="js/mqttws31.js"></script>.
- Now we can include jquery library by downloading it from jquery.com or use one of the many CDNs available.
- Include relevant cascaded style sheets which make the webpage look attractive.
- Design the website using css box model and include boxes, tables, drop down menus as per requirements.
- Now the website is ready to be deployed. To test the proper working we first hosted it on our computer using xampp.

After testing it extensively and eliminating all possible errors we were able to access it using local hosting. Since it is a front end application we did not require any data bases. After this process we searched for a place which would provide hosting for free. We came up across Firebase which is a platform which was implemented to make building mobile applications easier. It provided many services as listed in the previous chapter. We needed hosting and firebase was capable of hosting front end websites which did not require back end functionalities.

Deploying website on Firebase servers:

First we have to login into our google account on firebase.google.com. Then go to our console where we will have to create a new firebase project. After creating we will find many services which firebase offers.

These are the steps we followed to deploy our website:

- After creating a new project we have to install firebase command line tools using npm. For npm visit nodejs.org and download nodejs.
- Use the command: npm install -g firebase-tools to install the firebase tools.
- Create a folder which will contain the remote offline version of the website.
- Open command prompt and go to the location of newly created folder. Use the command firebase login and enter your credentials.
- Initiate a new project using firebase init. While initiating we should select hosting and link this remote folder to the actual project created on firebase server.
- Public directory will be the default directory, we didn’t make any changes to it.
- Now after going back to firebase console and we will see that a website has been hosted and we will have a website link to access that website. We created a free custom domain name using www.dot.tk, named cbitece2.tk. Now we have to connect this domain name to the website hosted on firebase.
- Copy and paste the website folders into the public directory and use the command firebase deploy to host the website.
- Click on connect domain and add your domain name. Go to your DNS manager in my.freenom. com and add a txt record with the information provided by firebase.
- It will take some time for firebase to verify that it your domain name and after that both the domain names will be linked together.
- When we entered our .tk domain name it got redirected under the hood to our firebase domain name.
- Thus we have deployed our website on firebase.

WEB SOCKETS CLIENT ANDROID APPLICATION

We have used Eclipse Paho android service which is implemented already by eclipse developers, the entire source code is present on github so just cloning it was sufficient[6]. The Paho Android Service is an MQTT client library written in Java for developing applications on Android. We had to add the Eclipse Maven repository to your build.gradle file and then add the Paho dependency to the dependencies section. The source code comes along with a sample app which is contained in the ‘org.eclipse.android.sample’ package so we have used this sample app in our project as it was sufficient to control the NodeMCU.

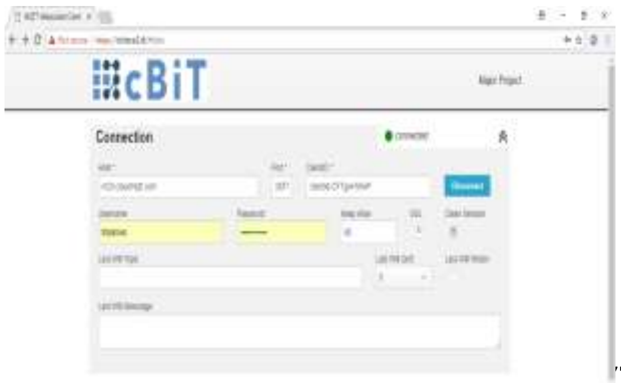
V. RESULT

First, webpage of the site created by us so as to get connected to the NodeMCU via cloud MQTT Broker.

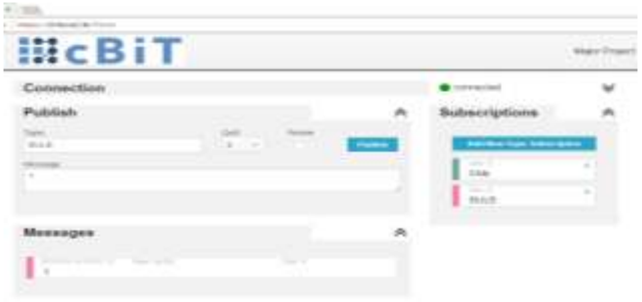
Figure 7: Homepage of web sockets client

The website designed by us can be divided into 4 major sections, they are

1) Connection 2) Publish 3) Subscribe 4) Messages



The first section (Connection) fulfills the function of connecting our web socket client to the cloud MQTT broker



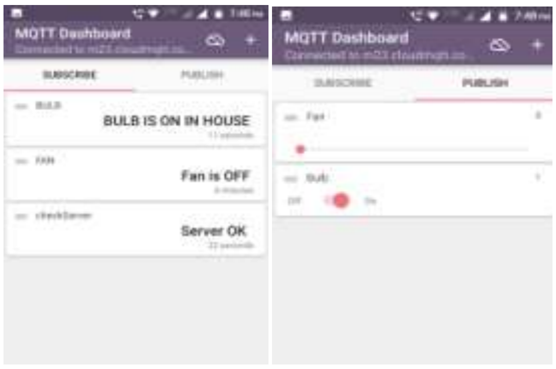
which is being used by us. Here there are certain input fields like username, password, host, port number etc which are to be filled by the user. The second section (Publish) contains the topic name and the qos with which the message has to be delivered to the broker. There is also an option if we want the message to be retained. The third section (Subscribe) is used to subscribe to any topic and will give the current message that is present in that topic. The fourth section is the messages section and it shows all the messages which are received by the client from the time of its connection.

CONTROLLING THE BULB

By subscribing to the topic "bulb" and publishing the message "1 " in the website the bulb is switched ON and similarly by publishing the message "0 " the bulb is switched OFF.

Figure8: Switching ON the Bulb from the website

By subscribing to the topic bulb and by keeping the button in ON state in the application the bulb is switched on is as shown in figure 8, 9 and 10.



BULB IS ON IN HOUSE

Fan is OFF

Server OK

Figure9: Switching ON the Bulb from the application

The command "ON" sent from the website or application is received by the NodeMCU and the bulb is switched on.

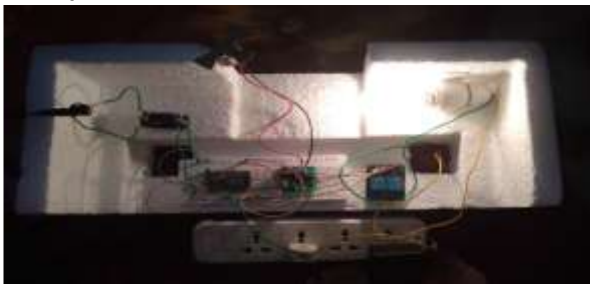
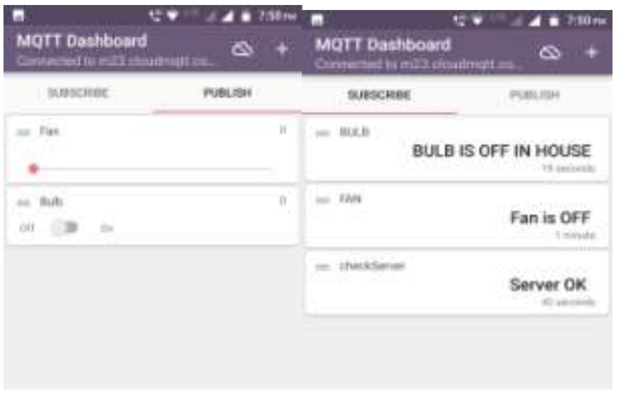


Figure10: Bulb switched ON

By subscribing to the topic bulb and by keeping the button in the application in OFF state, the bulb is switched OFF.



BULB IS OFF IN HOUSE

Fan is OFF

Server OK

Figure11: Switching OFF the bulb from the application

The command "OFF" sent from the website or the application is received by the NodeMCU and the bulb is switched off asshown in figure 11 and 12.

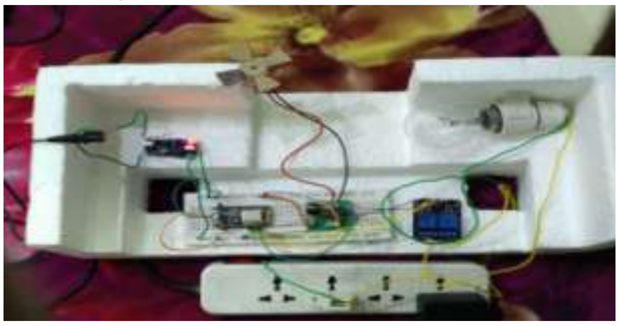


Figure12: Bulb switched OFF

CONTROLLING THE FAN

By subscribing to the topic fan and by publishing message with values "1 " to " 5 " in the website the fan is switched on and its speed is varied respectively and similarly by publishing the message "0 " the fan is switched off.

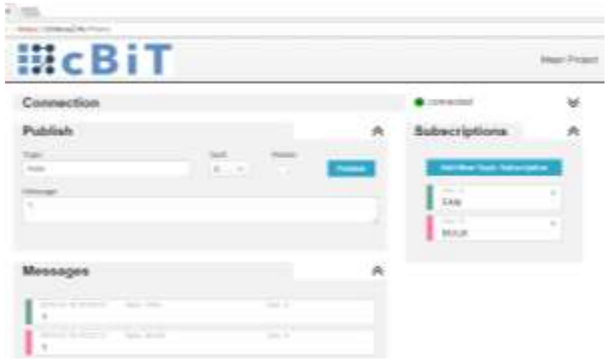
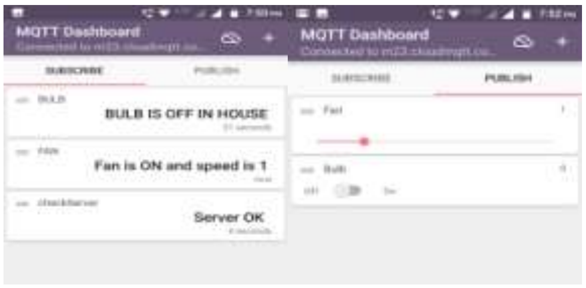


Figure13: Switching on the fan from website.



BULB IS OFF IN HOUSE

Fan is ON and speed is 1

Server OK

Figure14: Switching on the fan from the application

By subscribing to the topic "fan" and by varying the value published by using the slider in the application the fan is switched on and its speed is varied as shown in figure 13, 14 and 15. Five various levels of speed have been provided. The command "ON" sent from either the website or the application is received by the NodeMCU and the fan is turned on.

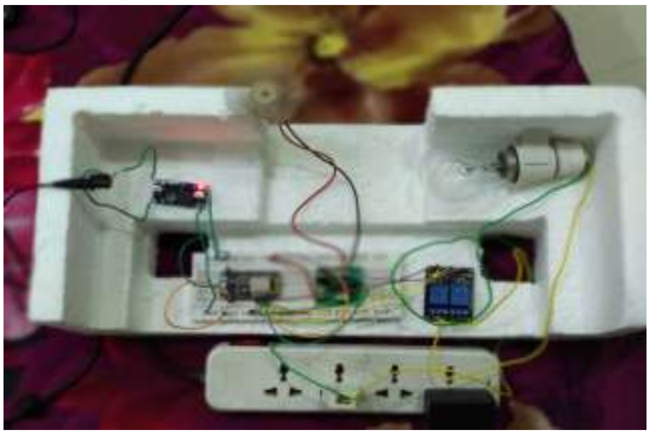
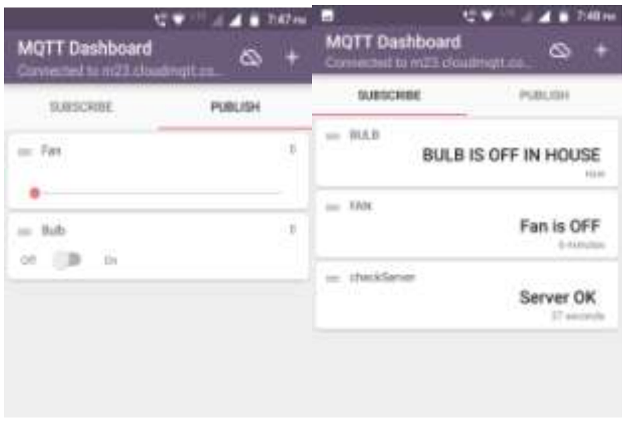


Figure15: Fan switched on

By subscribing to the topic fan and by keeping the slider in zero position in the application the fan is switched off.



BULB IS OFF IN HOUSE

Fan is OFF

Server OK

Figure16: Switching off the fan from application

The command "OFF" sent from either the website or the application is received by the NodeMCU and the fan is switched off as shown in figure16.

VI. Conclusion

IoT is to provide advanced connectivity of services, devices and systems which goes above machine-to-machine associations (M2M) and includes a range of applications, protocols and domains. MQTT's simplicity and open source code make this protocol suitable for constrained environments like IoT which has low power, limited computation capability and memory, and limited bandwidth.

A prototype of MQTT based home automation system is implemented on NodeMCU. The sensors and actuators connected to NodeMCU are remotely monitored and controlled through a common home gateway. Taking this further ahead, cloud platform can be used to aggregate, analyses and visualize data. Customized GUI can be developed to remotely access the devices to monitor and control them. This implementation provides an intelligent, comfortable and energy efficient home automation system. The feature to access and check the appliances at home from anywhere in the world just by the tap of a finger gives a modern ease of control to the user. It also assists the old and differently abled persons to control the appliances in their home in a better and easier way.

References

[1] Soni, Dipa&Makwana, Ashwin., A survey on MQTT: A protocol of Internet Of Things(IoT), ICTPACT - 2017, at bharath institute of higher education and research, chennai, india

[2] Kodali, Ravi &Soratkal, Sreeramya., MQTT based home automation system using ESP8266. 1-5. 10.1109/R10-HTC.2016.7906845,2016.

[3] M. Veeramanikandan and S. Sankaranarayanan, Publish/subscribe broker based architecture for fog computing. In 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS), pages 1024–1026, Aug 2017

[4] Light, Roger A., Mosquitto: server and client implementation of the MQTT protocol. Journal of Open Source Software, 2017.

[5] Shinho Lee, Hyeonwoo Kim, Dong-kweon Hong, Hongtaek Ju, Correlation Analysis of MQTT Loss and Delay According to QoS Level, International Conference on Information Networking (ICOIN), 28-30 Jan. 2013, pp. 714-717.

[6] I. Fette and A. Melnikov. The websocket protocol. RFC 6455, RFC Editor, December 2011.

[7] HiveMQ. HiveMQ - Enterprise MQTT Broker, 2015.