International Journal of Research in Engineering, Science and Management
Volume-2, Issue-2, February-2019
www.ijresm.com | ISSN (Online): 2581-5792

Home Automation System using ESP8266 based MQTT

S. Balakrishnan[1], B. Madhurekha[2], N. Shobana[3], S. Sherlyn Selshiya[4], G. Sathyabama[5]
[1]*Assistant Professor, Department of ECE, The Kavery Engineering College, Salem, India*
[2,3,4,5]*Student, Department of ECE, The Kavery Engineering College, Salem, India*

***Abstract*: In this paper we discuss and create a MQTT based Secured home automation system, by using temperature sensors and using ESP8266model as the network gateway, here we have implemented MQTT Protocol for transferring & receiving sensor data and finally getting access to those sensor data, also we have implemented ACL (access control list) to provide encryption method for the data and finally monitoring those data on webpage or any network devices. ESP8266 has been used as a gateway or the main server in the whole system, which has various sensor connected to it via wired or wireless communication.**

***Keywords:* Message Queuing Telemetry Transport (MQTT), ESP8266, Mosquitto, Home automation**

1. Introduction

Home automation refers to remotely monitoring the conditions of home and performing the required actuation. Through home automation, household devices such as TV, light bulb, fan, etc. are assigned a unique address and are connected through a common home gateway. These can be remotely accessed and controlled from any PC, mobile or laptop. This can drastically reduce energy wastage and improve the living conditions besides enhancing the indoor security.

Owing to the rapid growth in technology, the devices in the recent past are becoming smart. The real world devices are being equipped with intelligence and computing ability so that they can configure themselves accordingly. Sensors connected to embedded devices along with the low power wireless connectivity is facilitates to remotely monitor and control the devices. This forms an integral component of Internet of Things(IoT) network. Internet of Things can be considered as a network of devices that are wirelessly connected so that they communicate and organize themselves based on the predefined rules. However these devices are constrained in terms of their resources. Hence light weight protocols such as MQTT, CoAP etc. are used for the data transmission over wireless connectivity. There are so many kinds of radio modules out of which GSM, 3G, WiFi, Bluetooth, Zigbee, etc. are common. However, owing to the surging number of WiFi hotspots and range sufficient to perform the required control and monitoring, WiFi is chosen as the mode of communication in the prototype and the devices are controlled through MQTT protocol implemented using ESP8266.

Organization of the paper is as follows: A brief overview about the MQTT protocol is presented in section II. The related work that has already been done in this area is discussed in section III. In section IV the implementation details about the network setup, hardware and software used is briefed. Results from developed prototype are discussed in section V. Section VI presents the conclusions and future scope of work.

2. Message queuing telemetry transport

Message Queuing Telemetry Transport (MQTT) is a light weight transport protocol that efficiently uses the network bandwidth with a 2 byte fixed header [1]. MQTT works on TCP and assures the delivery of messages from node to the server. Being a message oriented information exchange protocol, MQTT is ideally suited for the IoT nodes which have limited capabilities and resources. MQTT was initially developed by IBM [2] in 1999 and recently has been recognized as standard by Organization for the Advancement of Structured Information Standards (OASIS) [3].

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. 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

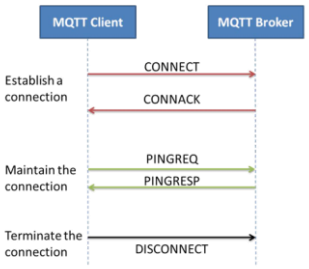

Fig. 1. Establishing, maintaining and terminating MQTT connection

Packets are exchanged before transmitting the application messages, control 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 [4] exchanged between MQTT clients and MQTT server. 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.

B. *Publishing the application messages*

If the client desires to be a publisher, it sends a PUBLISH packet to the server. This packet contains details about the QoS level of transmission, topic name, payload, etc. MQTT supports three levels of Quality of Service (QoS) [5] to the client. If the application messages are transmitted at QoS 0, the client does not receive any acknowledgment for the published packet. For QoS 1, the server acknowledges the

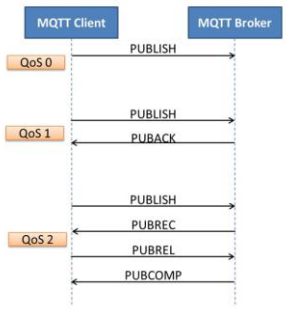

Fig. 2. Client publishing messages to the server with various QoS

Published packet with PUBACK including the packet identifier. However in QoS 2, four packets are exchanged. The server acknowledges the receipt of PUBLISH packet with the PUBREC packet. MQTT client then sends a packet to release publish with a PUBREL packet. The server then sends the fourth packet PUBCOMP, indicating the completion of publishing the application message on the given topic.

C. *Subscribing to a topic*

If the MQTT client want to subscribe to the application messages published on topic, 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.

D. *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. The MQTT server responds to the client with the indicated identifier with a PINGRESP packet and maintains the connection alive.

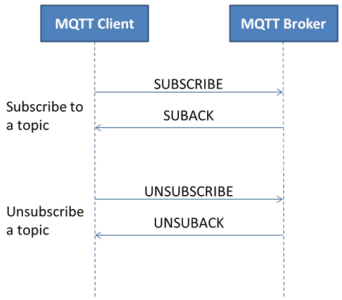

Fig. 3. Client subscribing and unsubscribing to the topic

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.

3. Related work

In [6], the authors discussed about the existing architectures for home automation and proposed a novel home automation architecture giving space to all the new IoT protocols. In [7], a prototype is designed to perform home automation through SMS. GSM network and the devices are bridged using a micro-controller. It also focuses on the security aspects in the networking and proposes a secure, reliable and adaptable home

automation system. The research work done in [8] proves that MQTT is better than HTTP for the nodes with constrained resources. It has been proven that data transmission through MQTT consumes only about 0.05% of battery/hour by using 3G for network connectivity.

4. Implementation details

A. Network setup

The intensity of light is sensed using LDR sensor connected to ESP8266 development board. ESP8266 development board processes the sensor data and performs actuation. It acts as a gateway for data transmission through WiFi. ESP8266 is configured as MQTT client publishing the sensor data to the MQTT broker and subscribing for the commands to control the actuation. LED and buzzer is used as actuators in the prototype. ESP8266 module publishes the sensor data under the topic 'esp\sense'. It subscribes for the topic 'esp\led' and 'esp\buzzer' to receive commands to control LED and

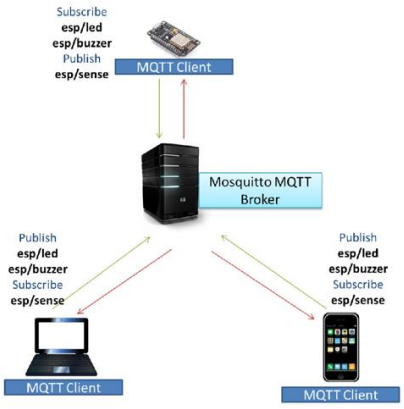

Fig. 4. Message transmission through MQTT

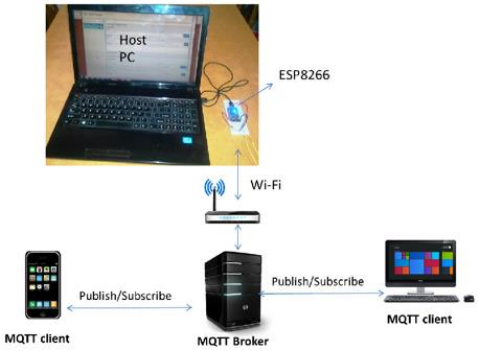

Fig. 5. Network Setup

Fig. 6. ESP8266 based Node MCU development board

Buzzer connected to the GPIOs of ESP8266. MQTT mosquitto broker is set up for ESP8266 to publish and subscribe to the application messages. Other MQTT clients such as PCs and Mobiles can connect to MQTT server through existing communication technologies such as Ethernet, 2G, 3G, WiFi etc.

B. ESP8266

ESP8266 [9] is a low cost development board that consolidates GPIOs, I2C, UART, ADC, PWM and WiFi for rapid prototyping. Powered by 3.3V supply, ESP8266 together with voltage regulator and USB to serial is packaged as ESP-12 module. Applications can be developed on this board through Arduino IDE or Lua based Esp lorer.

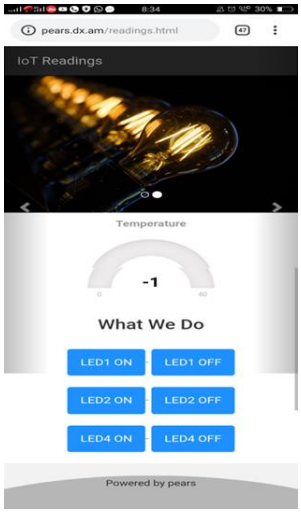

Fig. 7. Application UI on My MQTT Android application

C. Software setup

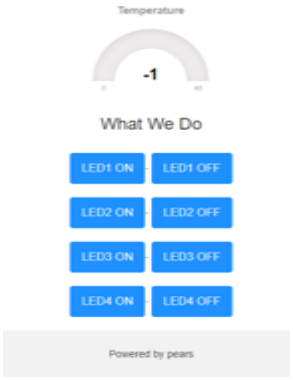

Fig. 8. Application UI on Computer

Arduino IDE is used to program the ESP8266 module as MQTT client. Mosquitto [10], an open source MQTT broker is implemented on Windows PC. It uses two services mosquitto Pub and mosquitto Sub to publish and subscribe to the application messages. MQTT broker is set up with the broker URL of the host IP of the PC on which Mosquitto broker is installed on port 1883. MQTTL ens [11], a Google Chrome based application is used as MQTT client that subscribes for the

sensor data and publishes the commands to control GPIOs of ESP8266. This sniffs the application messages that is being transmitted between Mosquitto MQTT broker and ESP8266. My MQTT, an android application is also another MQTT client that connects to the Mosquitto MQTT broker and publishes or subscribes to a topic.

5. Conclusion

This paper presented implementation of home automation system using ESP8266 based MQTT.

References

[1] MQTT v3.1 protocol specification. http://public.dhe. ibm.com/software/dw/webservices/ws-MQTT/MQTT-v3r1.html
[2] Hivemq. [Online]. Available: http://www.hivemq.com/blog/ mqtt-essentials-part-1-introducing- mqtt
[3] MQTT version 3.1.1 becomes an oasis standard. [On-line]. Available: https://www.oasis-open.org/news/announcements/mqtt-version-3-1-1-becomes-an-oasis-standard
[4] Oasis MQTT version 3.1.1. [Online]. Available: http://docs.oasis-open.org/MQTT/MQTT/v3.1.1/os/mqtt-v3.1.1-os.html
[5] Mqtt version 3.1.1 oasis standard. [Online]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html
[6] S. Nasrin and P. J. Radcliffe, "Novel protocol enables diy home automation," in Telecommunication Networks and Applications Conference (ATNAC), 2014 Australasian, Nov 2014, pp. 212–216.
[7] H. ElKamchouchi and A. ElShafee, "Design and prototype implementation of sms based home automation system," in Electronics Design, Systems and Applications (ICEDSA), 2012 IEEE International Conference on, Nov. 2012, pp. 162–167.
[8] A. Kumar and S. Johari, "Push notification as a business enhancement technique for e-commerce," in 2015 Third International Conference on Image Information Processing (ICIIP), Dec 2015, pp. 450–454.
[9] Node mcu–an open-source firmware based on esp8266 wifi-soc. [Online]. Available: http://nodemcu.com/index en.html/
[10] Eclipse. Mosquitto an open source mqtt v3.1/v3.1.1 broker. [Online]. Available: http://mosquitto.org/
[11] Mqtt lens-chrome web store. https://chrome.google.com/webstore/detail/mqttlens/hemojaaeigabkbcookmlgmdigohjobjm?hl=en