INFORMATION
Volume 21, Number 2, pp.581-588

ISSN 1343-4500 eISSN 1344-8994
©2018 International Information Institute

Design and Implementation ofMQTT based on Arduino

Am-Suk Oh*

*Department of Media Engineering, Tongmyong University, 428, Sinseon-ro, Nam-gu, Busan, Republic of Korea
E-mail: asoh@tu.ac.kr

Abstract

The maker movement and its culture demands appropriate digital tools for experiencing the "making" and testing the idea from the maker community. For pedagogical purpose, we design an IoT maker kit that can test students' ideas of utilizing IoT services and devices in the classroom. Within that design, the implementation of cloud server supporting MQTT is most critical. In this paper, we report the design of the MQTT cloud server and its implementation. The cloud server consists of MQTT broker and service web server. The websocket protocol is also used in order to obtain effective real time accessibility. The implementation of the proposed cloud server uses Nodjs, Mosquittos, and Amazon Web Server functions.

Key Words: Internet of Things, Websocket protocol, MQTT, Cloud server, MQTT Broker

1. Introduction

The maker movement refers broadly to the growing number of people who are engaged in the creative production of artifacts in their daily lives and who find physical and digital forums to share their processes and products with others [1]. Three key characteristics of that movement are digital desktop tools, a cultural norm of sharing designs and collaborating online, and the use of common design standards to facilitate sharing and fast iteration [2].

Recent advancement of Internet of Things (IoT) is taking us far beyond the Web and mobile computing and heterogeneous sets of computing devices are connected in IoT environment [3]. Combining the "maker community culture" and IoT technology together, we try to design an IoT maker kit for education in the classroom [4].

That proposed kit is to give proper understanding of IoT environment structures for students based on oneM2M standard IoT platform [5]. The technologies and applications consolidated under the vision of IoT and "Machine to Machine Communications" (M2M) are attracting the interest of businesses and poised to trigger next disruption in Information & Communication Technology (ICT) applications [6].

Message Queue Telemetry Transport (MQTT) is a typical mechanism of message-oriented paradigm. It is an open application protocol that is designed to use bandwidth and battery usage sparingly, which is why, Facebook Messenger currently uses it [7].

In MQTT protocol, it needs a broker (server) [8] that contains topics. Each client can be a publisher that sends information to the broker at a specific topic or/and a subscriber that receives automatic messages every time there is a new update in a topic he is subscribed.

In this paper, we design and implement a cloud server for IoT Maker Kit that contains a broker under MQTT. IoT cloud server for our application connects IoT devices and IoT service Apps to activate the IoT services [4]. We also use Websocket protocol [9] for real time accessibility.

2. Underlying Protocols

2.1 Message Queue Telemetry Transport (MQTT) Protocol

MQTT is a messaging protocol that was introduced by Andy Stanford Clark of IBM and Arlen Nipper of Arcom (now Eurotech) in 1999 and was standardized in 2013 at OASIS [10]. MQTT aims at connecting embedded devices and networks with applications and middleware. The connection operation uses a routing mechanism (one-to-one, one-to-many, many-to-many) and enables MQTT as an optimal connection protocol for the IoT and M2M [11]. The basic configuration of MQTT is shown as Fig. 1.

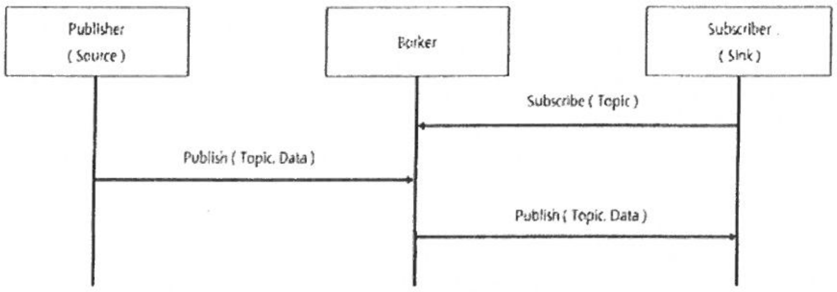


Fig. 1. Publish/subscribe process utilized by MQTT [10]

An interested device would register as a subscriber for specific topics in order for it to be informed by the broker when publishers publish topics of interest. There is an M to N relationship between topics and subscribers in that a subscriber may obtain multiple topics and one topic can be transferred to multiple subscribers. A topic is a string with hierarchy as shown in Fig.2 as an example of monitoring smart home devices so that it can handle bulky data efficiently with hierarchy.

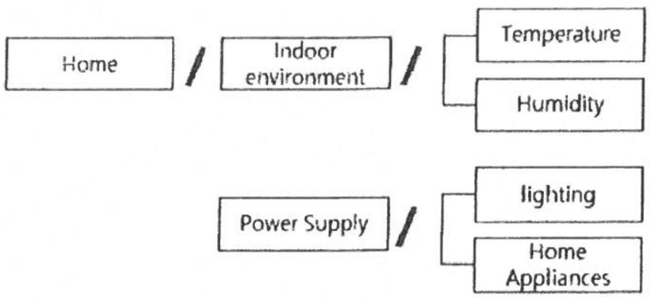


Fig. 2. Example Topic Hierarchy: Smart Home Monitoring Case

In [12], the functionality of MQTT and the QoS levels are well documented as following;

Besides acting as remote databases, M2M clouds also offer the following key services:

1. They offer Application Programming Interfaces (API) with built-in functions for end-users, thus providing the option to monitor and control end-devices remotely from a client device.

2. They act as asynchronous intermediate nodes between the end-devices and final applications running on devices such as smart phones, tablets or desktops.

MQTT ensures reliability by providing the option of three QoS levels:

1. Fire and forget: A message is sent once and no acknowledgement is required.

2. Delivered at least once: A message is sent at least once and an acknowledgement is required.

3. Delivered exactly once: A four-way handshake mechanism is used to ensure the message is delivered exactly one time.

Fig. 3 demonstrates QoS handling in MQTT.

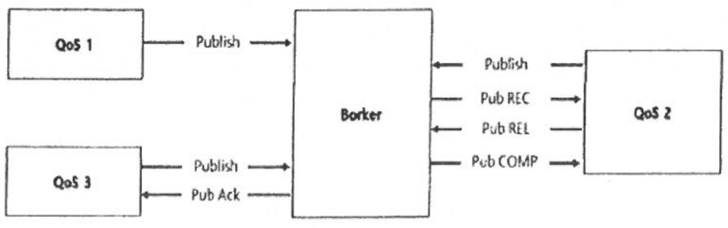
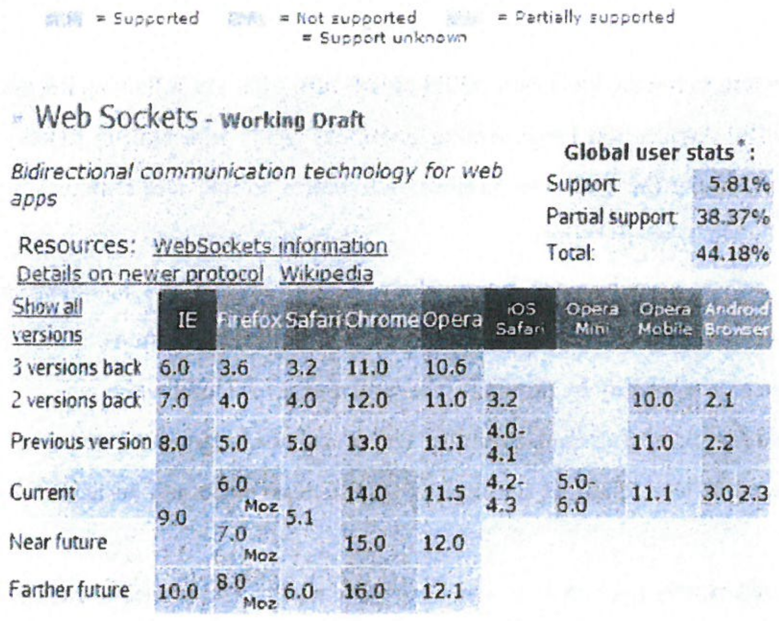


Fig. 3. QoS in MQTT

2.2 Websocket Protocol

Websocket protocol can support a single-socket full-duplex (or bi-directional) connection for pushing and pulling information between the browser and server. It is implemented on a web server and web browser to support real-time interaction between two points. Thus, it provides a scalable real-time web application effectively especially when the real-time property is important like IoT services. Web browsers that support websocket are summarized as shown in Fig. 4.


= Supported = Not supported = Partially supported
= Support unknown

Web Sockets - Working Draft

Bidirectional communication technology for web apps

Resources: WebSockets information Details on newer protocol Wikipedia

Global user stats*:

Support:	5.81%
Partial support:	38.37%
Total:	44.18%

Show all versions	IE	Firefox	Safari	Chrome	Opera	iOS Safari	Opera Mini	Opera Mobile	Android Browser
3 versions back	6.0	3.6	3.2	11.0	10.6				
2 versions back	7.0	4.0	4.0	12.0	11.0	3.2		10.0	2.1
Previous version	8.0	5.0	5.0	13.0	11.1	4.0-4.1		11.0	2.2
Current	9.0	6.0 Moz	5.1	14.0	11.5	4.2-4.3	5.0-6.0	11.1	3.0 2.3
Near future		7.0 Moz		15.0	12.0				
Farther future	10.0	8.0 Moz	6.0	16.0	12.1				

Fig. 4. Websocket supporting Brow [13]

As one can see in Fig. 4, almost all browsers except OperaMini support websocket protocol. It is also available in mobile platform since Android and iOS Safari support the protocol.

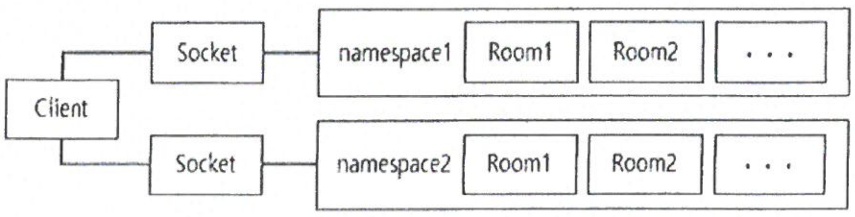


Fig. 5. Websocket Room

Websocket also supports room function that specifies a channel with a namespace as shown in Fig. 6. Room function enables only people in the room can transfer messages. Even if a device or an application does not support MQTT, we can use room function by creating a room with specific topic name and let other customers join so that it has similar functionality of publish/subscribe communication in MQTT.

3. Design and Implementation of IoT Cloud Server for IoT Maker Kit

With proposed IoT Maker Kit in [4], we design the MQTT cloud server that connects IoT kit with IoT service application under MQTT as shown in Fig. 6.

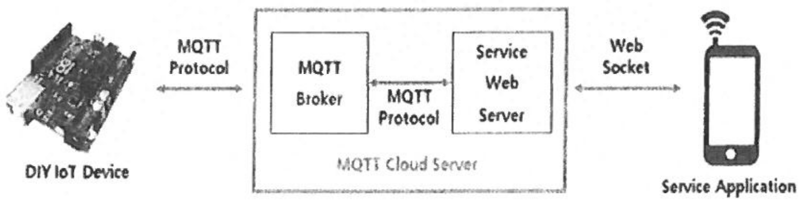


Fig. 6. Configuration of IoT Maker Kit and Cloud Server

In our design, the cloud server contains MQTT broker for supporting publish/subscribe communication. Then, the DIY IoT device and service web server become clients of that broker. The service web server is an application that transfers alarms and data via websocket protocol for real time accessibility.

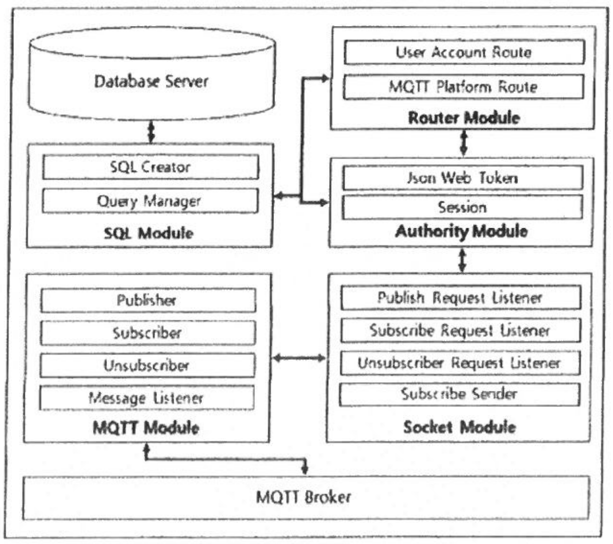


Fig. 7. Configurations of MQTT Cloud Server

Fig. 7 demonstrates the overall configurations of the cloud server proposed in this paper.

First, the Router Module performs the functions requested by API from the service application and report the result. Inside the module, the user account management API uses the User Account Route and the MQTT service management / provided API such as the topic service uses MQTT Platform Route.

The Authority Module verifies user's authority based on the information received through the API based on the existence of the session or the socket communication by verifying the decoded token with Json Web Token (JWT). It is helpful to reduce the transmission errors of messages when more than one service of different kind use the same topic or prevent a possible intrusion of data from unauthorized client.

The SQL module generates SQL statements according to the commands received by the module that performs queries such as DB inquiry and registration according to the request of the service application and returns the result information.

Socket module supports socket communication between MQTT Cloud Server and service application. The MQTT module is a client connected to the MQTT Broker and performing publish /subscribe function.

Fig. 8 demonstrates the publish/subscribe action in MQTT module.

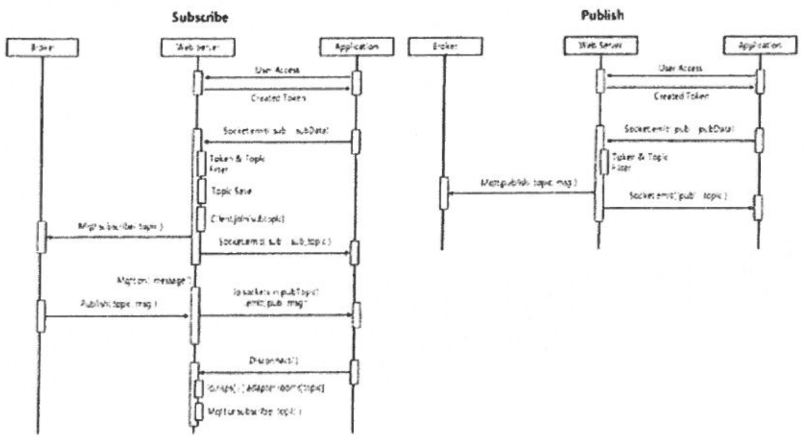


Fig. 8. Publish/Subscribe Sequence Diagram for MQTT in Cloud Server

The subscribe function generates the token and sends it to the application when the user authentication is requested. Subsequently, when an application subscribes via socket communication, it verifies the token received by the event listener and checks if the request topic exists in the database. When the verification is completed, socket room is created with the same topic name and client joins in that room. The MQTT broker subscribes to the topic and sends the result to the application. When publish event comes from the MQTT broker, it broadcasts the message to the clients joined to that socket room. The service server of the MQTT cloud server checks the number of clients in the socket room for the subscribed topic and the subscription terminates when the subscriber no longer exists. The Publish function carries out the procedure for authenticating the user right as the subscribe function. When the application makes a publish request through socket communication, the publish event listener verifies the received token and topic. When the verification is completed, the MQTT broker performs the event and send the result to the service application.

4. Concluding Remarks

In this paper, we demonstrate the design and implementation of cloud server subsystem under MQTT and Websocket protocol for an IoT maker kit tool that can be used in engineering class. IoT cloud server connects IoT devices and IoT service Apps to activate the IoT services.

Proposed MQTT cloud server consists of MQTT broker implemented by Mosquitto and service web server implemented by Amazon Web Services (AWS). The detailed design and activation procedure of MQTT cloud server is explained under MQTT principles. With such design and implementation, we hope the IoT maker kit can be used in engineering class for creating and testing new ideas under the principles of "maker movement".

5. Acknowledgments

This research was supported by the Tongmyong University Research Grants 2017.

References

[1] Halverson ER, Sheridan K. The maker movement in education. Harvard Educational Review. 2014 Dec 1;84(4):495-504.

[2] Anderson, C. (2012). Makers: The new industrial revolution. New York: Crown

[3] M. Selinger, A. Sepulveda, and J. Buchan, "Education and the Internet of Everything: How Ubiquitous Connectedness Can Help Transform Pedagogy", White Paper, Cisco, San Jose, CA, 2013

[4] Lim JY, Kim GH, Kwon OH, Oh AS. Research for Maker Board of Internet of Things. In International Conference on Future Information & Communication Engineering 2017 Jun (Vol. 9, No. 1, pp. 313-316).

[5] Datta SK, Da Costa RP, Bonnet C, Härri J. oneM2M architecture based IoT framework for mobile crowd sensing in smart cities. In Networks and Communications (EuCNC), 2016 European Conference on 2016 Jun 27 (pp. 168-173). IEEE.

[6] Balamuralidhara P, Misra P, Pal A. Software platforms for internet of things and M2M. Journal of the Indian Institute of Science. 2013 Aug 14;93(3):487-98.

[7] http://mqtt.org/2011/08/mqtt-used-by-facebook-messenger, cited 28 Jul 2014.

[8] 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.

[9] Fette, Ian. "The websocket protocol." (2011). https://tools.ietf.org/html/rfc6455

[10] MQTT V3.1 Protocol Specification
http://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/mqttv3r1.html

[11] Bandyopadhyay S, Bhattacharyya A. Lightweight Internet protocols for web enablement of sensors using constrained gateway devices. InComputing, Networking and Communications (ICNC), 2013 International Conference on 2013 Jan 28 (pp. 334-340). IEEE.

[12] Karagiannis V, Chatzimisios P, Vazquez-Gallego F, Alonso-Zarate J. A survey on application layer protocols for the internet of things. Transaction on IoT and Cloud Computing. 2015;3(1):11-7.

[13] websocket support browsers, http://caniuse.com/websockets

*Corresponding author: Am-Suk Oh, Ph.D.
Department of Media Engineering,
Tongmyong University,
428 Sinseon-ro, Nam-gu, Busan, Korea
Phone number: +82-51-629-1211
Fax number: +82-51-629-1129
E-mail: asoh@tu.ac.kr