Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control
Journal homepage: http://kinetik.umm.ac.id
ISSN: 2503-2267
Vol. 4, No. 3, August 2019, Pp. 197-206

Internet of things platform for manage multiple message queuing telemetry transport broker server

Aulia Arif Wardana*[1], Andrian Rakhmatsyah[2], Agus Eko Minarno[3], Dhika Rizki Anbiya[4]
Telkom University, Indonesia [1,2]
Universitas Muhammadiyah Malang, Indonesia [3]
Badan Pengkajian dan Penerapan Teknologi, Indonesia [4]

Article Info

Keywords:
Internet of Things Platform, Internet of Things, Message Queuing Telemetry Transport, MQTT Broker Server

Article history:
Received 26 May 2019
Revised 17 June 2019
Accepted 9 July 2019
Published 30 July 2019

Cite:
Wardana, A., Rakhmatsyah, A., Minarno, A., & Anbiya, D. (2019). Internet of Things Platform for Manage Multiple Message Queuing Telemetry Transport Broker Server. Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control, 4(3). doi:http://dx.doi.org/10.22219/kinetik.v4i3.841

* Corresponding author.
Aulia Arif Wardana
E-mail address:
auliawardan@telkomuniversity.ac.id

Abstract

This study proposed the Internet of Things (IoT) monitoring platform model to manage multiple Message Queuing Telemetry Transport (MQTT) broker server. The Broker is a part of the MQTT protocol system to deliver the message from publisher to subscriber. The single MQTT protocol that setup in a server just have one broker system. However, many users used more than one broker to develop their system. One of the problems with the user that use more than one MQTT broker to develop their system is no recording system that helps users to record configurations from multi brokers and connected devices. This can cause to slow the deployment process of the device because the configuration of the device and broker not properly managed. The platform built is expected to solve the problem. This proposed platform can manage multiple MQTT broker server and device configuration from different product or vendor. The platform also can manage the topic that connects to a registered broker on the platform. The other advantages of this platform are open source and can modify to a specific business process. After usability testing and response time testing, the proposed platform can manage multiple MQTT broker server, functional to use, and an average of response time from the platform page is not more than 10 seconds.

1. Introduction

Internet of Things (IoT) is a technology that connects multiple sensor and device with the cloud over internet connection [1]. The right choice of communication model in IoT system is the key of success because many service and device in IoT system are communicated to each other with a large number of connection and exchange data [2]. There are many communication models in the IoT system, one of them is the publish/subscribe communication model. IoT system is part of a distributed system and the data that produce from the connection is very huge. The publish/subscribe communication model very suitable with IoT system to handling huge connection of data in the distributed system [3].

One of protocol that implements publish/subscribe communication model in the IoT system is Message Queuing Telemetry Transport (MQTT). This protocol is popular in IoT system and applied in many industries [4]. Nowadays, IoT with MQTT protocol is implemented in many domains such as health, transportation, farming, disaster management, and other domain that use the large connection to handle [1]. MQTT is specifically designed for the IoT system and suitable for resource-constrained device [5]. MQTT protocol has three main entities on the system, there are the publisher, subscriber, and broker. The publisher is an entity that sending data to broker with specific topic and subscriber is an entity that receiving data from a broker with a specific topic. In IoT system, publisher represented a device that periodically publishes sensor data, subscriber represented the specific application that subscribe data from publishes sensor data, and the broker is an entity that hold and route data from the publisher to the subscriber [6],[7].

One of an important entity in MQTT protocol is a broker. The broker is central to communication and handling huge data from many publishers and subscribers connection [8]. Large connections from publisher and subscriber can cause the availability of MQTT broker to decrease. When the reliability from the broker is decreased, the single failure will occur to MQTT broker. The single failure in the MQTT broker can cause packet loss or the broker connection is down. When the broker connection is down, the subscriber need to re-subscribe the data from broker and publisher need to re-publish the data to the broker. In the increase of re-subscribe and re-publish load connection, the broker eventually starts dropping packets [9],[10].

IoT system is integrated of heterogeneous device and system. The deployment of the device and system in IoT is complex and need a system that can manage the configuration of the system and device [11],[12]. Many IoT system uses multiple brokers to keep availability from the broker server. Since the increase of the broker server in an IoT system, the device that connected to the broker also increase. The deployment of the broker and device must be synchronized well. Many broker and device that connect to each other need to maintain correctly. The configuration of the broker server (e.g server name, port, credential, and etc) and the configuration of device profile (e.g device id, credential, and etc) need to save in a single system. The good maintenance from multiple broker and device will affect to deployment process [13], [14], [15].

Based on the background, to make the user easier to maintain multiple broker and device that connect to each other, then a system should be created that can manage it on a platform [16], [17]. MQTT protocol does not have multiple broker management system to manage multiple the broker server [3-10], [16], [18], [19]. Therefore, this study proposed a platform that manages multiple broker system. The platform will manage the broker connection and manage the topic from the publisher and subscriber that connect to the broker server.

2. Research Method

This research starts with a process of reviewing research papers in the field of MQTT protocol with the topic of MQTT broker server availability, then finds a problem that single MQTT broker server not good enough to handle a large number of publisher and subscriber connection. Some systems anticipate this by using many brokers on the IoT system that is built. The problem rises up again, the increase of broker means the increase of device too. The system that manages the multiple broker and device is needed. Later, this research identifies that the MQTT protocol does not have a system to manage multiple broker server in the literature review process. The proposed internet of things platform is expected to make easier for the user to manage multiple MQTT broker server. Also, the proposed platform expected to make the deployment process easier because the information of heterogeneous broker and device is managed well in one system.

A Platform is part of the Internet of Things system that is located on the highest layer because it deals directly with the user [3], [20]. The platform is usually used to carry out device management and protocols that are connected to each other on the internet of things system [16], [21]. The platform must also support different configurations between heterogeneous systems or devices on the Internet of Things [22], [23], [24]. Based on the internet of things platform system in general, this research creates a platform that can manage heterogeneous MQTT broker servers and devices [21], [22], [23].

The platform will be built using the waterfall method. The method is a method commonly used in building software. Most of the software engineer will make this method to build their software. This method also the simplest way to build software [24], [25]. The first phase is the analysis phase, this phase is done in the introduction section and the beginning of research method paragraph. The second phase is the design phase, this phase will be discussed in the proposed system section. The third phase is the implementation phase, this phase will be explained in the platform implementation phase. The last phase is the testing phase, this phase will be explained and discussed in the result and discussion section.

The test method used to test the platform uses the method of Quality of Service (QoS) testing and functional testing. The functional testing will measure the functionality from the platform to help the user to manage multiple MQTT broker server and device. The QoS testing will test the time access from the platform. The QoS from the platform also affects the Quality of Experience (QoE) from the user of the platform [26], [27], [28], [29], [30].

2.1 Proposed System

This research tries to build a prototype internet of things platform to manage multiple MQTT broker servers. The system architecture from the prototype platform shows in Figure 1. The network architecture illustrated in Figure 1 shows that the platform will connect every publisher and subscriber connected to many broker servers. Each publisher is an IoT device that has different specifications. The MQTT broker servers also have different vendors and configuration. The configuration method for each device and MQTT broker will certainly be different, but this platform will equalize the configuration of each device and MQTT brokers use common standard.

The configuration that has been entered by the user into the platform will be used as a liaison between the publisher, subscriber, and MQTT broker server. The software architecture from the platform can see in Figure 2, the diagram consists of 6 layers. The first layer is the web server layer that used to serve the platform. The second layer is profile layer that consists of The User Manager Module, Monitoring Data View, and Vendor Manager. The user manager module is used to manage access to users who will use the platform. The monitoring data view module is used to subscribe to data from the publisher and MQTT broker that has been managed by the platform. The vendor manager module is used to manage companies or individuals who want to use this platform for their needs.

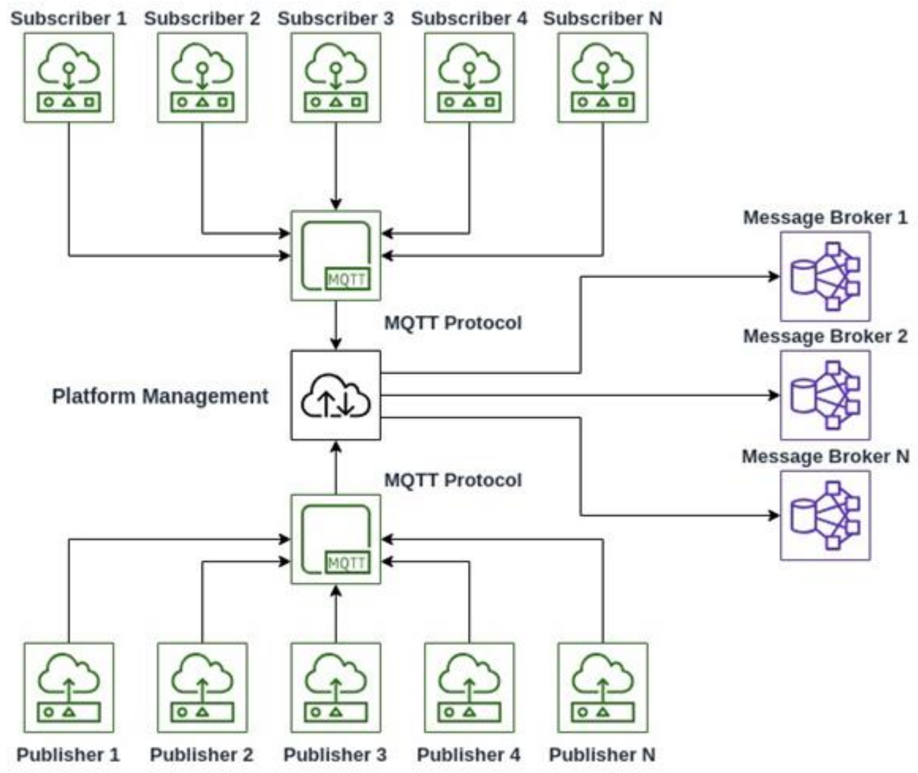


Figure 1. System Architecture of The Platform

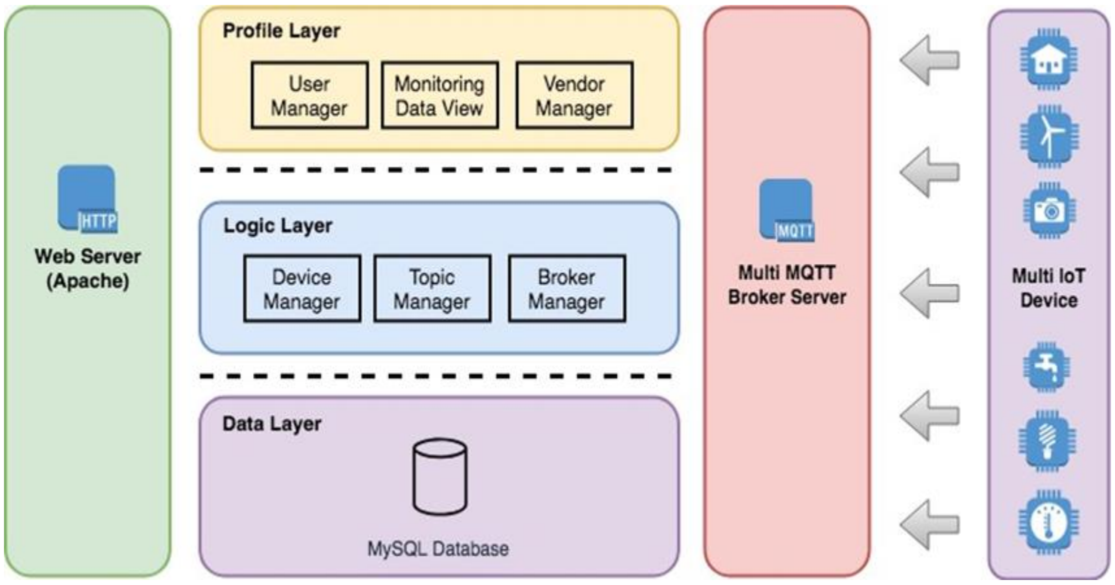


Figure 2. Software Architecture of The Platform

The third layer is the logic layer that consists of Device Manager, Topic Manager, and Broker Manager. The device manager module is used to manage devices that will be connected to the MQTT broker. Registered devices will have a unique ID that is used to identify the device. the topic module and broker module are used to manage topics that will be used on a broker. Both modules are a unit. The broker module can also configure security such as SSL/TLS or username and password when the broker uses a security mechanism. After the user makes an input to the device module and the broker module in the platform, then each device will be assigned to a broker with a specific topic that set in the broker. All devices that will be connected to the broker must be programmed to adjust the configuration that has been done on the platform. The fourth layer is The Data Layer that consists of a MySQL database. The database design for this platform shows in Table 1.

Table 1. Database Design

Table Name	Column Name	Type
devices	id_device	int(11)
	serial	varchar(65)
	username	varchar(65)
	product	varchar(65)
	date_inserted	timestamp
	date_registered	timestamp
mqtt_server	id_mqttserver	int(11)
	mqtt_host	varchar(65)
	mqtt_websocket	varchar(65)
	mqtt_user	varchar(65)
	mqtt_password	varchar(65)
	mqtt_port	varchar(65)
	mqtt_tls	varchar(65)
	mqtt_topic	varchar(65)
	mqtt_vendor	varchar(65)
	serial_device	varchar(65)

The database type used for this platform is a SQL database. The SQL database tables that are the main system on this platform are devices and mqtt_server. Device tables are used to store profiles of devices, while the mqtt_server table is used to store MQTT broker server profiles, including what topics should be assigned to devices and MQTT brokers. The mqtt_server tabel needs two type of port from MQTT broker server, the first port is MQTT TCP port (normally 1883 or 8883 when in TLS mode), the port is used by device to publish sensor data to MQTT broker server. The first port is saved in mqtt_host column. The second port is MQTT websocket port, the port is used by the platform to subscribe data from MQTT broker server. The second port is saved in mqtt_websocket column.

The fifth layer is multiple MQTT broker server that manage by vendor. The last layer is multi IoT device layer that acts as a publisher. Configuration patterns carried out on this platform are made based on the MQTT broker system in general. The configuration on this platform is not related to a particular vendor or implementation of MQTT broker server. This platform can make it easier for users to manage different broker vendors and IoT devices. The limitation of this platform is traffic from The MQTT broker is not directly control from the platform. This platform is just for broker management, the user must apply the same configuration with the platform in the deployment process.

2.2 Platform Implementation

The platform code can see in this git repository: https://gitlab.com/kijang-electric/low-cost-IoT-platform. Management of this platform is very cheap and easy because it is deployed in the hosting environment [12]. The software environment that uses in development can see in Table 2. This study uses 3 different MQTT broker as a sample for free MQTT broker server. The different vendor of MQTT broker system that used in this platform shows that this platform is suitable for manage different MQTT broker vendors.

Table 2. Software Environment used in Implementation

No	Name	Description
1.	Apache	Web Server to serve the platform
2.	PHP 7.0	Programming language for build the platform
3.	MySQL Database	Database for save data from the platform
4.	Cloud MQTT	MQTT Broker Server
5.	Hivemq	MQTT Broker Server
6.	Maqiatto	MQTT Broker Server

The platform system starts with the installation process to manage the platform vendor users and admin users when first access in the browser. The Vendor is a personal user or organization that used this platform for their business process. After the installation process, the user will log in with a super admin account, then the user will input the device profile (device ID and vendor device) and broker server profile (host, username, password, port, and SSL option) to the platform configuration. The next step of configuration in the platform assigns the broker server configuration to device configuration. The platform is supported subscriber system to monitor data flow from the publisher that send data to the MQTT broker server. The subscriber system monitors the flow of data by device connection.

The implementation architecture of this platform can see in Figure 3. This study uses 3 different of device vendor (Arduino Uno R3 with Ethernet Shield, Node MCU, and Raspberry Pi 3) and 3 different of MQTT broker server vendor. The configuration from 3 different MQTT broker server vendor in the platform will deploy to 3 different device vendor. The Node MCU and Arduino Uno R3 with Ethernet Shield device are programmed using Arduino IDE for deploying the configuration, then the Raspberry Pi 3 is programmed using python for deploy the configuration. The deployed configuration used by device to connect to the MQTT broker server.

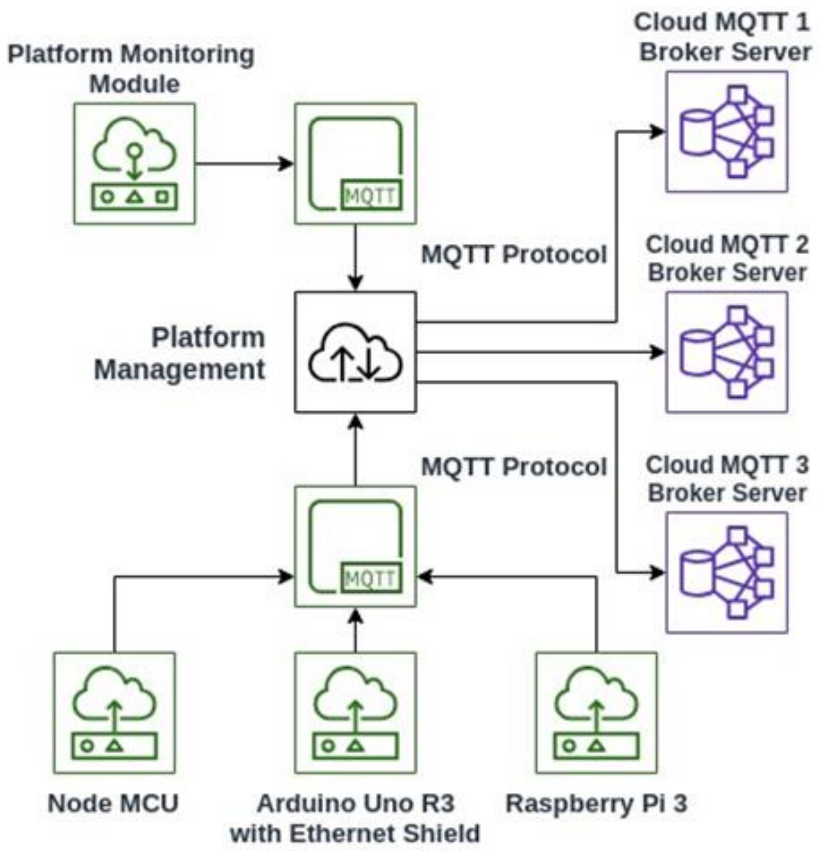

Platform Monitoring Module
MQTT
Cloud MQTT 1 Broker Server
MQTT Protocol
Cloud MQTT 2 Broker Server
Platform Management
MQTT Protocol
Cloud MQTT 3 Broker Server
MQTT
Node MCU
Arduino Uno R3 with Ethernet Shield
Raspberry Pi 3

Figure 3. Implementation Architecture

3. Result and Discussion

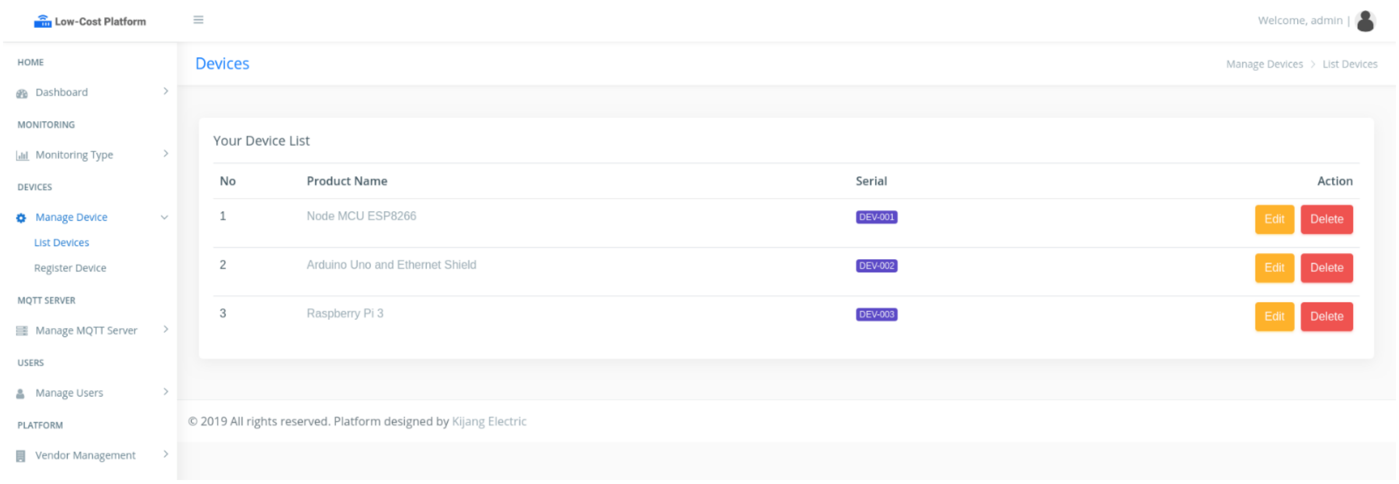
The testing for this study use usability testing. The usability testing measures the functionality from the device configuration process and MQTT broker configuration process. The usability testing also tests the configuration from 3 different vendors and 3 different devices is working properly or not.

3.1 Platform Usability Testing

The usability testing scenario from this research focuses on device and MQTT broker server configuration process. The testing scenario will implement until the data can send from the device to the MQTT server. The first usability testing is registering 3 different devices profile to the platform system to creating configuration data from the devices. The device configuration process starts with register the device profile. The User will input the device vendor name (e.g Arduino, NodeMCU, or Raspberry) and device ID to the platform.

The result of this testing can see in Figure 4. The platform can register different devices profile, also the device's profile that already insert can edit or delete from the platform system. After configuration is already input, the device must program using the same as the configuration in the platform.

The second usability testing is registering 3 different MQTT broker server profile to the platform system to create configuration data from the MQTT broker server. The MQTT broker server configuration process starts with register the MQTT broker server profile. The user will input MQTT Host, MQTT Username, MQTT Password, MQTT Port, MQTT Websocket Port, MQTT TLS, MQTT Topic, MQTT Vendor and Serial Device.



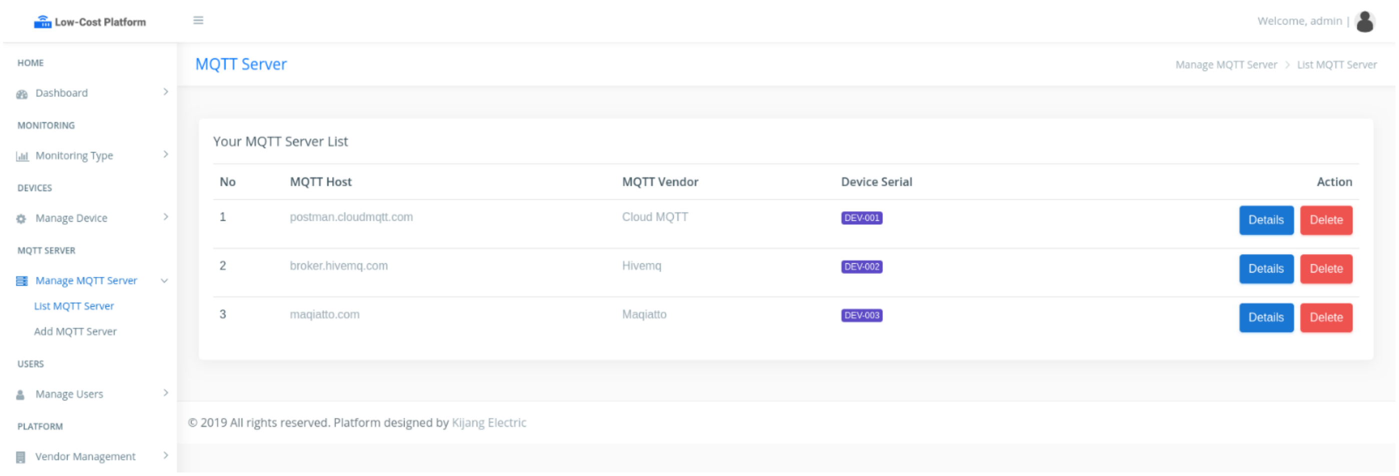
Devices

Your Device List

No	Product Name	Serial	Action
1	Node MCU ESP8266	DEV-001	Edit Delete
2	Arduino Uno and Ethernet Shield	DEV-002	Edit Delete
3	Raspberry Pi 3	DEV-003	Edit Delete

© 2019 All rights reserved. Platform designed by Kijang Electric

Figure 4. Registered Devices



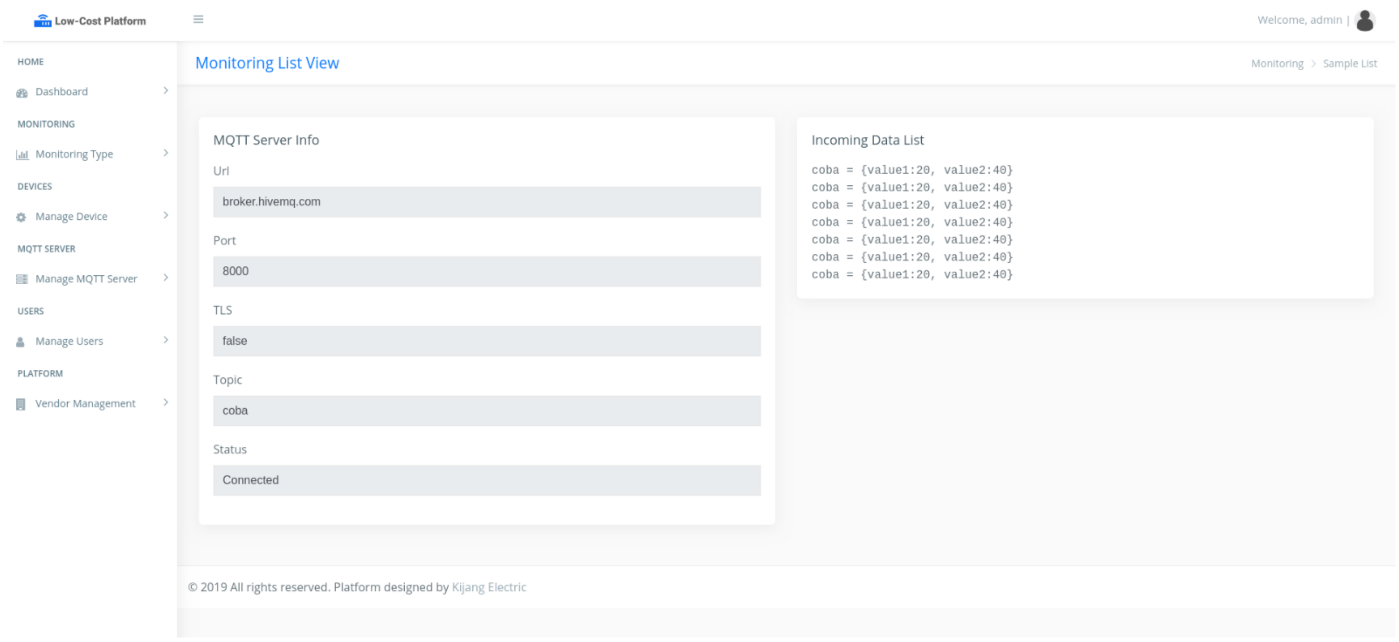
MQTT Server

Your MQTT Server List

No	MQTT Host	MQTT Vendor	Device Serial	Action
1	postman.cloudmqtt.com	Cloud MQTT	DEV-001	Details Delete
2	broker.hivemq.com	Hivemq	DEV-002	Details Delete
3	maqiatto.com	Maqiatto	DEV-003	Details Delete

© 2019 All rights reserved. Platform designed by Kijang Electric

Figure 5. Registered MQTT Broker Server and Topic



Monitoring List View

MQTT Server Info

Url	broker.hivemq.com
Port	8000
TLS	false
Topic	coba
Status	Connected

Incoming Data List

```
coba = {value1:20, value2:40}
coba = {value1:20, value2:40}
coba = {value1:20, value2:40}
coba = {value1:20, value2:40}
coba = {value1:20, value2:40}
coba = {value1:20, value2:40}
coba = {value1:20, value2:40}
```

© 2019 All rights reserved. Platform designed by Kijang Electric

Figure 6. Subscribe data from MQTT Broker Server

The result of this testing can see in Figure 5. The platform can register different MQTT broker server, also the MQTT broker server profile data that already insert can view, edit or delete from the platform system. After configuring and determining the topic on the MQTT broker server for each device, the device will be programmed according to the specified configuration. The result is a device programmed according to the configuration can send data to the MQTT broker server.Communication on each MQTT broker server must have a username, password, TLS configuration, port, and address. Therefore, this platform can accept all configurations of each MQTT broker server that is a different vendor. MQTT broker server that does not have a username and password, or does not have Transport Layer Security (TLS) configuration can also be configured on this platform system.

After all configuration setup, we will program the device with the same configuration as in the platform. In this research, we will configure Arduino Uno R3 with the ethernet shield, NodeMCU ESP8266, and Raspberry Pi 3 to connect in MQTT broker server with the same configuration as in the platform. Arduino Uno R3 with ethernet shield and NodeMCU ESP8266 programmed using Arduino IDE with C language, then the Raspberry Pi 3 programmed using python. The result of this testing can see in Figure 6. The platform can subscribe to data from the device. If the configuration of the device is not the same with the platform, so the platform can not subscribe to the data from the device.

All usability testing from the platform module can see in Table 3. The usability testing in every module have the scenario, and the result from the scenario will prove that the module is working properly or not.

Based on the testing result in Table 3, the platform module and feature are working properly. The platform can properly manage devices and MQTT broker servers. The platform not only has features for managing brokers and devices but also has other features related to business processes such as user management and vendor management.

The second test is the quality of service from the platform. One measurement that indicates the quality of service from the website is response time. The testing will measure the response time from the platform. This testing used response website tools from https://www.webpagetest.org/. The test location was located in Jakarta and used chrome browser to open the web page. The connection from the internet that used for the test is Cable (5/1 Mbps 28ms RTT). The result of the testing shows in Figure 7, Figure 8, and Figure 9.

This test will measure website response time from device manager module, MQTT broker manager module, and subscriber data page. The focus of the testing in those 3 modules because the user will interact more often with this module when using the platform.

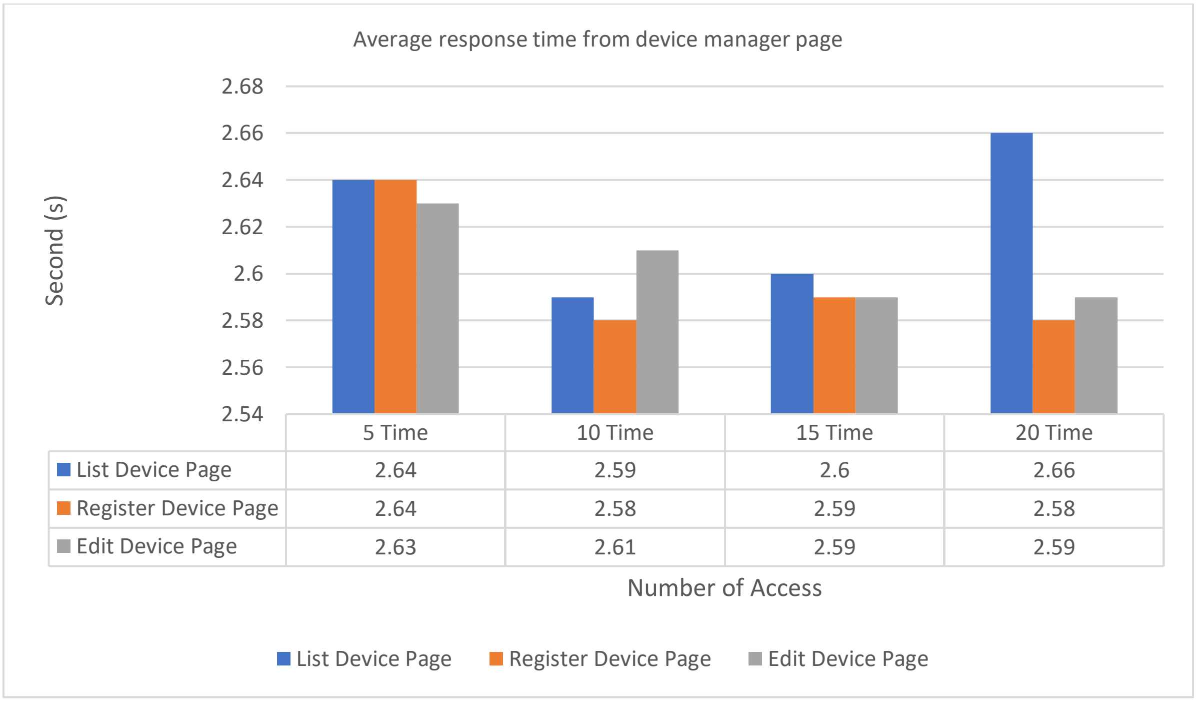
Based on Google data, average response time from the website is not more than 10 second. When the website has response time more than 10 seconds, it will make users are frustrated and are likely to abandon tasks. They may or may not come back later [31]. The performance measurement of device manager module in Figure 7 and the performance measurement of the MQTT broker manager module in Figure 8 is not more than 10 second.

Figure 9 shows that the average response time from subscriber data page is not more than 10 second. The device manager module, MQTT broker manager module, and subscriber data page response time is acceptable to access by multiple users.

Table 3. Usability Testing for Device and MQTT Broker Configuration

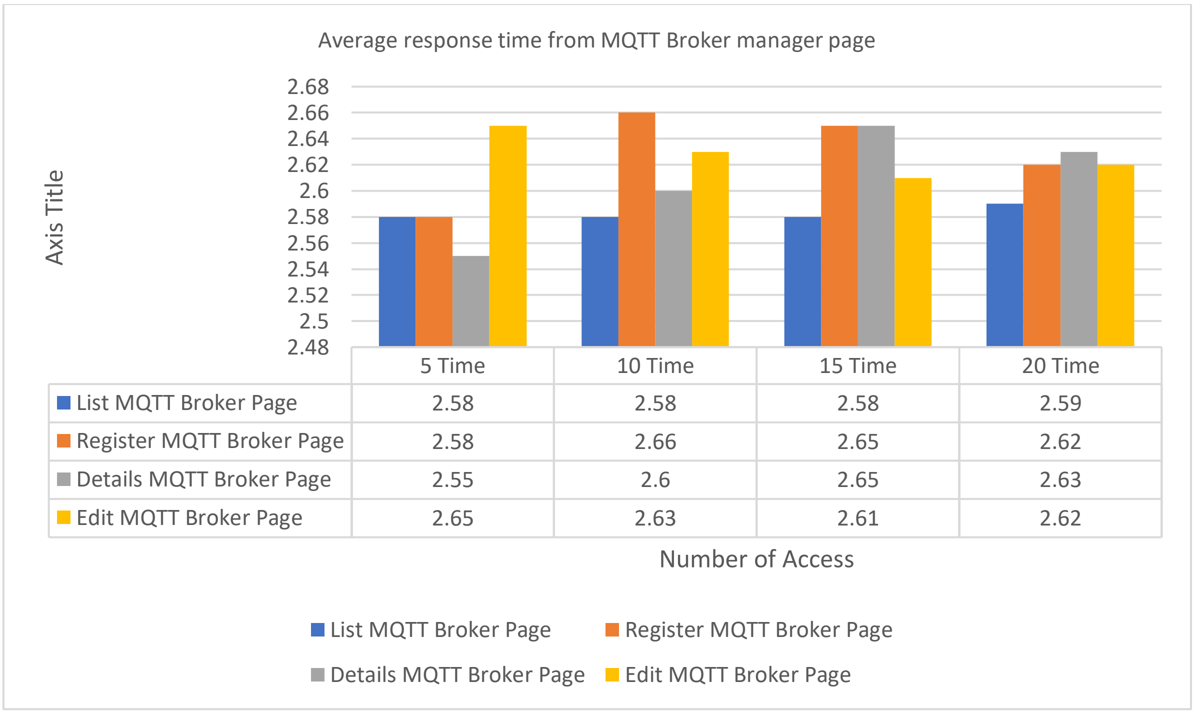
No	Module Name	Testing Scenarion	Result
1.	User Manage	View, Input, Update, and Delete user profile data in User Manage module system.	The platform can perform View, Input, Update, and Delete user profile data in User Manage module system.
2.	Monitoring Data View	View monitoring data from publisher. The publisher (device) configuration same as in the platform system.	The platform can perform View, Input, Update, and Delete user profile data in User Manage module system.
3.	Vendor Manager	View, Input, Update, and Delete vendor profile data in Vendor Manager module system.	The platform can perform View, Input, Update, and Delete vendor profile data in Vendor Manager module system.
4.	Device Manager	View, Input, Update, and Delete device profile data in Device Manager module system.	The platform can perform View, Input, Update, and Delete device profile data in Device Manage module system.

5.	MQTT Broker Server Manager	View, Input, Update, and Delete MQTT broker cerver profile data in Topic Manager and Broker Manager module system.	The platform can perform View, Input, Update, and Delete MQTT broker server profile data in Topic Manager and Broker Manager module system.



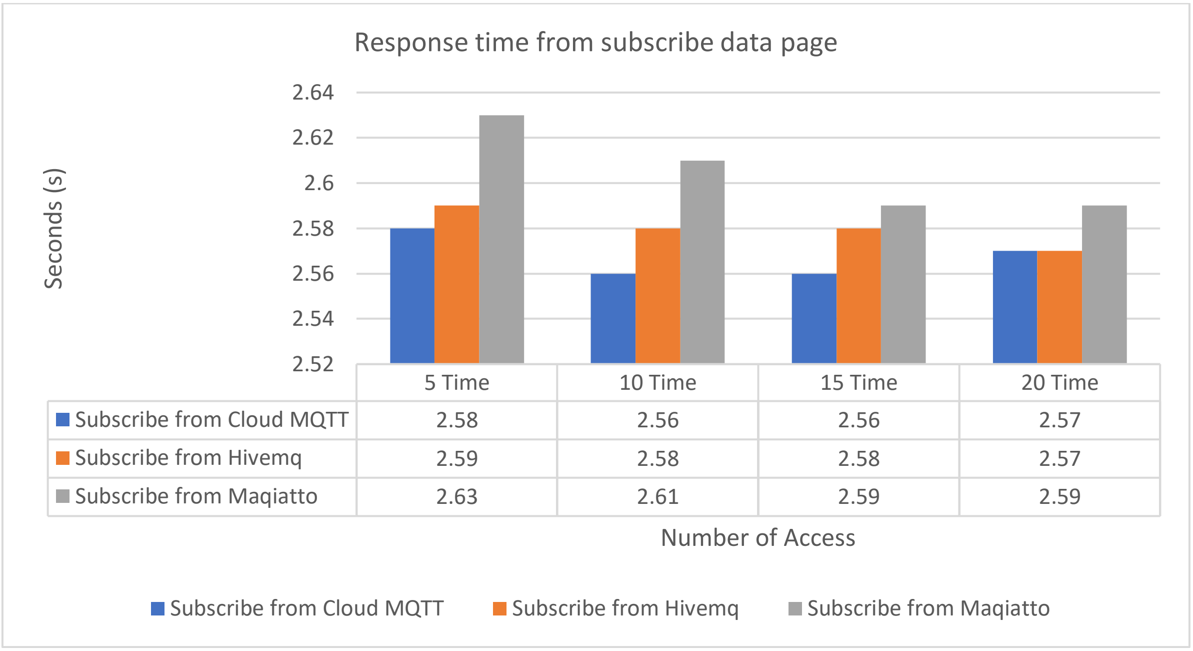
	5 Time	10 Time	15 Time	20 Time
🟦 List Device Page	2.64	2.59	2.6	2.66
🟧 Register Device Page	2.64	2.58	2.59	2.58
⬜ Edit Device Page	2.63	2.61	2.59	2.59

Figure 7. Response Time From The Device Manager Module



	5 Time	10 Time	15 Time	20 Time
🟦 List MQTT Broker Page	2.58	2.58	2.58	2.59
🟧 Register MQTT Broker Page	2.58	2.66	2.65	2.62
⬜ Details MQTT Broker Page	2.55	2.6	2.65	2.63
🟨 Edit MQTT Broker Page	2.65	2.63	2.61	2.62

Figure 8. Response Time From The MQTT Broker Manager



	5 Time	10 Time	15 Time	20 Time
🟦 Subscribe from Cloud MQTT	2.58	2.56	2.56	2.57
🟧 Subscribe from Hivemq	2.59	2.58	2.58	2.57
⬜ Subscribe from Maqiatto	2.63	2.61	2.59	2.59

Figure 9. Response Time From The Subscriber Data Page

4. Conclusion

This study proposed a prototype Internet of Things platform to manage multiple MQTT broker server. The platform purpose is helping the user easier to manage multiple MQTT broker server and device in the Internet of Things system. After usability testing, the platform is functional to use. The platform is able to manage multiple devices and multiple MQTT broker server. This study also measures the response time from the website page. Based on response time testing from MQTT broker manager module, device manager module, and subscriber data page, the response time is not more than 10 seconds.

In further research, the platform can be developed to do auto routing against the connection traffic that exists on the MQTT broker server. In addition, the configuration of the device can be generated automatically to make it easier during the deployment process. The other feature to support the interfacing for mobile application or another system, this platform will provide web API for universal communication to another system.

5. Acknowledgement

Thank you to the Internet of Things Studio, Telkom University which has become a place for researchers to develop this journal research. Hopefully, this research can make a major contribution to the advancement of technology in Indonesia.

References

[1] M. Chernyshev, Z. Baig, O. Bello, and S. Zeadally, "Internet of Things (IoT): Research, Simulators, and Testbeds," *IEEE Internet of Things Journal*., Vol. 5, No. 3, Pp. 1637–1647, 2018. https://doi.org/10.1109/JIOT.2017.2786639
[2] J. Rui and S. Danpeng, "Architecture Design of the Internet of Things Based on Cloud Computing," in *Seventh International Conference on Measuring Technology and Mechatronics Automation*, Pp. 206–209, 2015. https://doi.org/10.1109/ICMTMA.2015.57
[3] P. Lea, *Internet of Things for Architects*. Packt Publishing, 2018.
[4] H. C. Hwang, J. Park, and J. G. Shon, "Design and Implementation of a Reliable Message Transmission System Based on MQTT Protocol in IoT," Wireless Pers Commun., Springer, 2016. https://doi.org/10.1007/s11277-016-3398-2
[5] P. Waher, *Learning Internet of Things*. Packt Publishing, 2015.
[6] W. Pipatsakulroj, V. Visoottiviseth, and R. Takano, "muMQ: A Lightweight and Scalable MQTT Broker," in *IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*, 2017. https://doi.org/10.1109/LANMAN.2017.7972165
[7] A. Zabasta, *et. al,* "MQTT Service Broker for Enabling the Interoperability of Smart City Systems," in *IEEE Energy and Sustainability for Small Developing Economies (ES2DE)*, 2018. https://doi.org/10.1109/ES2DE.2018.8494341
[8] P. Jutadhamakorn, *et. al,* "A Scalable and Low-Cost MQTT Broker Clustering System," in *2nd International Conference on Information Technology (INCIT)*, 2017. https://doi.org/10.1109/INCIT.2017.8257870
[9] S. Sen and A. Balasubramanian, "A Highly Resilient and Scalable Broker Architecture for IoT Applications," in *10th International Conference on Communication Systems & Networks (COMSNETS) - Proceeding*, Pp. 336–341, 2018. https://doi.org/10.1109/COMSNETS.2018.8328216
[10] Y. Xu, V. Mahendran, and S. Radhakrishnan, "Towards SDN-Based Fog Computing: MQTT Broker Virtualization for Effective and Reliable Delivery," in *Workshop on Wild and Crazy Ideas on the interplay between IoT and Big Data*, 2016. https://doi.org/10.1109/COMSNETS.2016.7439974
[11] K. Mekki, *et. al,* "A Comparative Study of LPWAN Technologies for Large-Scale IoT Deployment," *ICT Express.*, Elsevier, Vol. 5, 2019. https://doi.org/10.1016/j.icte.2017.12.005

[12] A. C. F. da Silva, *et. al,* “OpenTOSCA for IoT: Automating the Deployment of IoT Applications Based on the Mosquitto Message Broker,” in *ACM 6th International Conference on the Internet of Things*, 2016. https://doi.org/10.1145/2991561.2998464

[13] W. Li, *et. al,* “Performance Comparison of Cognitive Radio Sensor Networks for Industrial IoT With Different Deployment Patterns,” in *IEEE Systems Journal*, 2014. https://doi.org/10.1109/JSYST.2015.2500518

[14] J. Huang, *et. al,* “A Novel Deployment Scheme for Green Internet of Things,” in *IEEE Internet of Things Journal*, 2014. https://doi.org/10.1109/JIOT.2014.2301819

[15] M. B. Yassein, *et. al,* “Internet of Things: Survey and Open Issues of MQTT Protocol,” in *International Conference on Engineering & MIS (ICEMIS)*, 2017. https://doi.org/10.1109/ICEMIS.2017.8273112

[16] J. Yun, I. Ahn, N. Sung, and J. Kim, “A Device Software Platform for Consumer Electronics Based on the Internet of Things,” *IEEE Transactions on Consumer Electronics*., Vol. 61, No. 4, Pp. 564–570, 2015. https://doi.org/10.1109/TCE.2015.7389813

[17] R. Banno, *et. al,* “Dissemination of Edge-Heavy Data on Heterogeneous MQTT Brokers,” in *IEEE 6th International Conference on Cloud Networking (CloudNet)*, 2017. https://doi.org/10.1109/CloudNet.2017.8071523

[18] P. S. Rompas, A. A. Wardana, and Albarda, “Robust Flood Monitoring Platform using Message Queueing Telemetry Transport Protocol,” in *International Conference on Information Technology Systems and Innovation (ICITSI)*, 2017. https://doi.org/10.1109/ICITSI.2017.8267949

[19] N. Shofa, A. Rakhmatsyah, S. A. Karimah, “Infusion Monitoring Using WiFi (802.11) through MQTT Protocol,” in *5th International Conference on Information and Communication Technology (ICoIC7)*, 2017. https://doi.org/10.1109/ICoICT.2017.8074693

[20] M. A. A. D. Cruz, *et. al*, “A Reference Model for Internet of Things Middleware,” *IEEE Internet of Things Journal*., Vol. 5, No. 2, Pp. 871–883, 2018. https://doi.org/10.1109/JIOT.2018.2796561

[21] M. T. Lazarescu, “Design of a WSN Platform for Long-Term Environmental Monitoring for IoT Applications,” *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*., Vol. 3, No. 1, Pp. 45–54, 2013. https://doi.org/10.1109/JETCAS.2013.2243032

[22] Y. J. Heo, *et. al,* “A Lightweight Platform Implementation for Internet of Things,” in *3rd International Conference on Future Internet of Things and Cloud*, pp. 526–531, 2015. https://doi.org/10.1109/FiCloud.2015.29

[23] I. Culic and A. Radovici, “Development Platform for Building Advanced Internet of Things Systems,” in *16th RoEduNet Conference: Networking in Education and Research (RoEduNet)*, 2017. https://doi.org/10.1109/ROEDUNET.2017.8123761

[24] R. Elghondakly, S. Moussa, and N. Badr, “Waterfall and Agile Requirements-Based Model for Automated Test Cases Generation,” in *IEEE Seventh International Conference on Intelligent Computing and Information Systems (ICICIS)*, 2015. https://doi.org/10.1109/IntelCIS.2015.7397285

[25] M. Kuhrmann, *et. al,* “Hybrid software and System Development in Practice: Waterfall, Scrum, and Beyond,” in *ACM International Conference on Software and System Process*, 2017. https://doi.org/10.1145/3084100.3084104

[26] L. D. Vito, *et. al,* “An IoT-enabled Multi-sensor Multi-user System for Human Motion Measurements,” in *IEEE International Symposium on Medical Measurements and Applications (MeMeA)*, 2017. https://doi.org/10.1109/MeMeA.2017.7985877

[27] E. Bocchi, L. D. Cicco, and D. Rossi, “Measuring the Quality of Experience of Web users,” in *ACM SIGCOMM Computer Communication Review*, 2016. https://doi.org/10.1145/3027947.3027949

[28] L. Nguyen, *et. al,* “Modelling of Quality of Experience for Web Traffic,” in *IEEE Second International Conference on Network Applications, Protocols and Services*, 2010. https://doi.org/10.1109/NETAPPS.2010.22

[29] M. Varvello, *et. al,* “EYEORG: A Platform for Crowdsourcing Web Quality Of Experience Measurements,” in *ACM Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies*, 2016. https://doi.org/10.1145/2999572.2999590

[30] A. E. Minarno, “Web Reporting Service dengan Database Terdistribusi untuk Monitoring Transaksi berbasis Point of Sales,” in *Universitas Muhammadiyah Malang*, 2009.

[31] Google, “Measure performance with The RAIL Model”