Internet of Things Gateway as a Cross-Platform Data Communication Service Based on Advanced Message Queueing Protocol

Yaddarabullah, Dewi Lestari, Ari Setijadi Prihatmanto, Reza Darmakusuma

Abstract— Nowadays there are many data communications systems used by the microcontroller in the delivery of data to the server. Some research on data communication on the internet of things applies data transmission directly from the microcontroller to the server using a single data communication path. There are two problems that occur. First, if the data communication system used by the microcontroller is different, the server must provide additional services in handling different data communications. Second, if there are many microcontrollers that send data to the server at the same time, causing high traffic connectivity and the server requires a lot of resources in the data processing. Therefore, in this research, the internet of things (IoT) gateway is developed to handle diverse data communications from different microcontroller platforms and reduce the server load in serving data from a large number of microcontrollers. The Advanced Message Queuing Protocol (AMQP) will be used on the IoT gateway as a service to handle diverse data communications from various microcontroller platforms. Data communication systems supported include MQTT, STOMP, REST and Socket connection. The functions provided in the gateway are receiving data from the microcontroller, storing it permanently, accessing data through the application, then periodically the data is sent to the server. AMQP test results show that with the amount of data served, as many as 7083 records require an average publish time of 0.20 ms and delivery time of 0.22 ms with a data size of 576 KB. While the CPU load is less than 10%, the average RAM memory usage is 550 MB and the bandwidth usage is less than 3 mbps.

Index Terms— Internet of Things; Gateway; Data Communication System; Advance Message Queueing Protocol.

1 Introduction

Internet of things (IoT) is a part of internet technology that is currently developing. The basic concept in the technology internet of things is to connect any devices to each other that can be accessed via the internet [1]. Technology Internet of things conceptually refers to three main elements, namely hardware that is equipped with a module internet of things called the Node Microcontroller Unit (MCU), an internet connection device such as a modem or wireless router, and a server as a place to store data and applications [2]. the IoT Analytics company noted that the use of the internet of things increased in 2016 which was implemented in various sectors and is predicted to increase in 2020 [3].

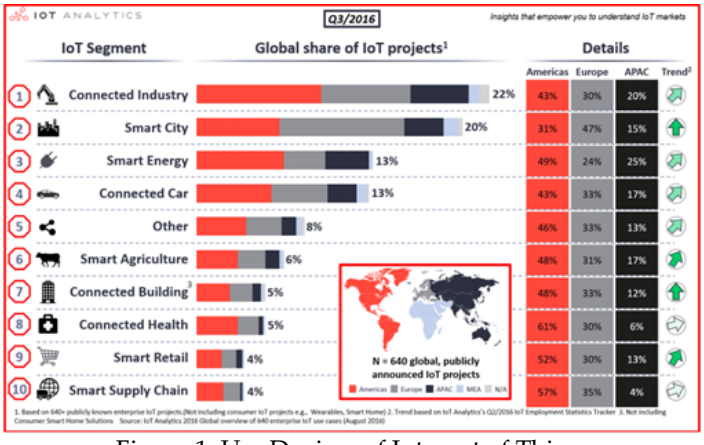

Figure 1. Use Devices of Internet of Things

In the last five years there have been many studies on the internet of things at both the national and international levels, including recording water meters using a mobile phone so that it can be monitored for the use of water liters and bills in real time [4]. The data communication system used between the Node MCUs to the REST-based server, the Node MCUs access address Application Program Interface the HTTP Protocol-based provided by the server. Components used in the Node MCU include Arduino UNO, SIM9100A GSM modem and water flow meter sensor. Data transmission is done directly from the Node MCU to the server. Other studies, namely the application of drip irrigation systems on plants that can be done remotely using a mobile phone making it easier for farmers to know the development of plants [5]. In this research, the component in the Node MCU used is Arduino UNO as a microcontroller and the data communication applied is REST. In a study entitled Implementation of Machine-to-Machine Solutions Using MQTT Protocol in Internet of Things (IoT) Environment to Improve Automation Process for Electrical Distribution Substations in Colombia, the data communication system between the Node MCU to the server uses MQTT, so only certain microcontrollers are used. can be used [6].

In the research that has been mentioned there are some vulnerabilities, among others, if there are a lot of additional Node MCUs, there will be an increase in connectivity and traffic to the server. The increase in traffic causes the data processing on the server will require more time, RAM memory usage will be large and the CPU capacity is high. Another weakness is that if the water meter recording system or drip irrigation device is developed using different platforms micro controller such as ATMega32, OrangePi, Intel Galileo with different data communication systems such as Socket Connection, STOMP, MQTT, CoAP and XMPP, the

server must provide services which can handle a variety of data communication systems. This causes there will be many services on the server and RAM memory usage becomes high. Another problem is that if the connection between the Node MCU and the server is lost, the user cannot access the latest data until the connection is re-connected, causing the information generated to be inaccurate and unreliable.

Based on the problems above, in this study a device that functions as developed, that gateway is a bridge between the Node MCU and the server. The functions of gateway this include receiving and storing data from various Node MCU platforms with different data communication systems. Data stored on the gateway will be sent periodically to Virtual Private Server. The purpose of developing this IoT gateway is to provide flexibility to developers in developing Node MCUs with a variety of microcontroller platforms, ease the burden on the server and facilitate users in accessing information offline by connecting to the gateway and can also access information online by accessing the address of the server. The IoT gateway was developed based on the Advance Message Queueing Protocol (AMQP). By using the AMQP protocol, it can overcome different data communication systems from the microcontroller platform and be able to handle Node MCUs in large numbers [7].

2 METHODS

2.1 Review of Literature

The first stage in this research is to study the literature study on the workings of the method Advance Message Queueing Protocol (AMQP). Some articles that are used as a reference include a study entitled Distributing Messages Using Rabbitmq with Advanced Message Exchanges [8] which discusses the use of RabbitMQ in handling large amounts of data and has services to serve different data communication systems, State-of-the-Art of Messaging for Distributed Computing Systems [9] which discusses the latest techniques in handling the distribution of data sourced from platforms different and A Multi-Protocol IoT Platform Based on Open-Source Frameworks [7] which discusses the workings of using different data communication protocols in one device. AMQP is an opensource service that consists of two main parts, namely exchange and queue. Exchange functions to provide services in receiving data from various communication systems, including STOMP, MQTT, REST and Socket Connection. Then map the data received from the Node MCU according to its type which is then forwarded to the queue. Queue is a temporary storage place for data at AMQP. One of the tools in the AMQP protocol, MQTT, is generally implemented on Cloud Server as one of the services used to receive data transmission from various microcontroller platforms [10], such as Bluemix developed by IBM and Amazone Web Services. In this study the AMQP protocol will be implemented into the IoT gateway. There are several tools used to implement the AMQP protocol, including ActiveMQ, QPID and RabbitMQ [11]. In this study tools AMQP protocol that will be used is RabbitMQ. Other functions that will be available at the gateway include permanent data storage, data monitoring offline, services remote access to the gateway and periodic data transmission to the server. The IoT gateway developed will use Raspberry Pi B + R3, in which is installed and configured RabbitMQ, MariaDB, SSH, XRDP Server, Wi-fi Hotspot and Apache2 Web Server.

2.2 System Architecture

Communication architecture of gateway the designed consists of Node MCUs, devices gateway, multi data communication systems, Virtual Private Server and REST data communication.

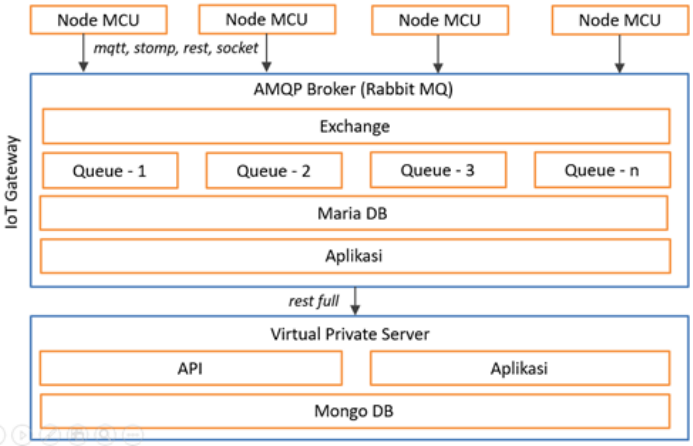

Figure 2. IoT Gateway Data Communication Architecture

The workings of the architecture are started from sensor data from various Node MCUs with different platforms microcontroller, then the data is sent to the gateway through supported protocols namely MQTT, STOMP, REST and Socket. Sending data from the Node MCU to the gateway can use Wi-fi media or a wired LAN network. The service at the gateway consists of RabbitMQ as AMQP Broker which consists of three components, namely: (1) Exchange, (2) Queue, (3) Maria DB, which functions as a database to store data from the Node MCU, (4) Application functions to display data that has been stored. Data on RabbitMQ that has been copied into MariaDB will automatically be deleted, so the RAM memory usage by RabbitMQ will decrease and the capacity of the queue will return empty. The application was developed on gateway web-based and can be accessed through a Wi-fi network provided by the IoT gateway. The service periodically gateway will send data stored on MariaDB to a Virtual Private Server (VPS) using the REST-Full communication [12] i.e accessing the API address provided by VPS. In VPS there are several components, namely: (1) Application Program Interface (API) which has the function to receive data sent from the gateway, (2) MongoDB as a database that functions to store these data, (3) Web-based applications that function to display data that has been stored in VPS.

2.3 System Integration

In this study, six Node MCUs were developed, among others, for measuring temperature, humidity, air pressure, light intensity, wind speed and soil moisture. The six Node MCUs were developed with different microcontroller

platforms and different data communication systems. After the six Node MCUs are assembled, they are integrated with the IoT gateway. At this stage, integration between the Node MCU and the IoT gateway is done for one week to see the incoming traffic to the IoT gateway and the operating system performance at the gateway in handling data. Connectivity between Node MCUs and devices gateway using Wifi and LAN network.

2.4 Testing

The final stage in this research is testing the IoT gateway. There are four tests to be carried out. First test the data communication system, which is to see and measure the ability of the gateway to handle data from different platforms microcontroller. The second tests the ability of data logging at the gateway to store data from the Node MCU. Third, test data connectivity traffic, which is to find out the bandwidth, the amount of incoming and outgoing data that can be handled by the gateway. Fourth, test the operating system performance on the IoT gateway, which is to find out the CPU's ability to handle data processing and RAM memory usage.

3. Result and Discussion

The results of this study are a IoT gateway that can be installed at mini computer like Raspberry Pi or Orange Pi and has functions to control information from the connected Node MCU. Following scheme are the application of IoT Gateway.

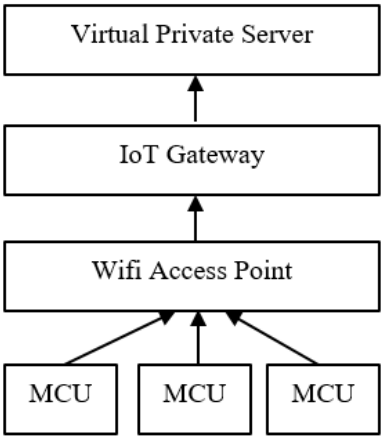

Figure 3. Application of IoT Gateway Via Wifi Access Point

Figure 3 shows the Node MCU connected to IoT via a Wifi network. Node MCU consists of a microcontroller which is equipped with an Wifi module like ESP8266 or Wifi Ethernet Shield for Arduino. In this condition the IoT gateway connected to the Wifi Access Point using a UTP cable or connected through a Wifi network. The advantage of this first scheme is that the number of IoT gateways is sufficient so that it can reduce costs. If there are many Node MCUs, then it is needed to add the number of Wifi Access Points. This scheme is suitable for cases that require a lot of Node MCUs. The second scheme is the IoT gateway is configured to be a hostpot, then the Node MCU is connected directly to the IoT gateway as shown below.

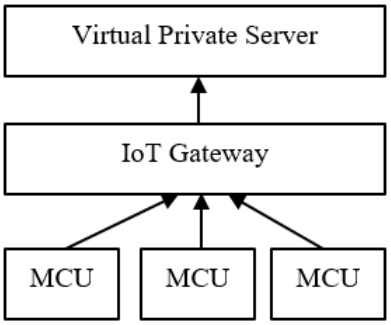

Figure 4. Application of IoT Gateway As Hotspot

In this case, IoT gateway configured as hotspot and the Node MCU connected directly to the IoT gateway via the Wifi network. This scheme is suitable for cases that use few Node MCUs. The disadvantage of this scheme is that if there are many Node MCUs installed over long distances then the number of IoT gateways must also be increased and it cause cost to increase. Here is the algorithm on the IoT gateway that handles the process of receiving data from node MCU with RabbitMQ, save locally to the database MariaDB then periodically send data to the VPS.

```
IoT Gateway ALGORITHM
{Used to handle various data communications on the
IoT gateway}

DECLARATION
 E: exchange
 Q: Queue
 D, S: float
 T: time
 Addr, Port, API: string
 SC: connection
 P: protocol

DESCRIPTION
 // 1. Read from Node MCU and save to RabbitMQ
 P = MQTT // protocol MQTT / STOMP / Socket / REST
 Set Protrot (P)
 Read (D) from the microcontroller
 Store (D, E) // save from D to E
 Bind (E, Q) // route from E to Q
 StoreTemporary (Q)

 // 2. Move from RabbitMQ to MariaDB
 While Q is not EMPTY
 Do
   GetData (Q, S) // move from Q to S
   MariaDBInput (S) // input S to MariaDB
   SetStatus (S, 0) // set status of S
   EmptyQueue (D)
 EndWhile

 // 3. Send from Gateway to VPS
```

```
Addr = http://31.220.48.158/
Port = 27160
API = Gateway01-Data Temperature
While Status of S is 0
Do
  MariaDBGetData (D)
  SC = SetConServer (Addr, Port , API)
  SendData (SC, D)
  SetStatus (S, 1) // set status of S
EndWhile
```

The algorithm in the gateway consists of three parts. The first stage starts from reading the data that enters RabbitMQ. Node MCU periodically publish data from sensor to IoT gateway through MQTT/STOMP/REST/Socket Connection. The data will be temporarily stored in the exchange, then will be forwarded to the queue accordance with the type of sensor that has been determined. Data stored in the queue is only temporary.

The second stage of the system reads the data in the queue. If data is available in the queue it will be moved to MariaDB, then given a status of 0 to the data that has been stored in MariaDB. Data that has been moved into MariaDB then RabbitMQ will empty the data in the queue.

The third step is to configure the address, port and API to be addressed to the VPS, then send the data. VPS provide API for receive the data that send by IoT gateway according to type of sensor. After that the data that has been sent to the VPS will be updated to 1. Web applications on the IoT gateway can be used through the Wi-fi network provided by the gateway, then by using a web browser accessed through the address http://192.168.10.1/spotnet/. Following figure 5 is a display of a web application on the IoT gateway.

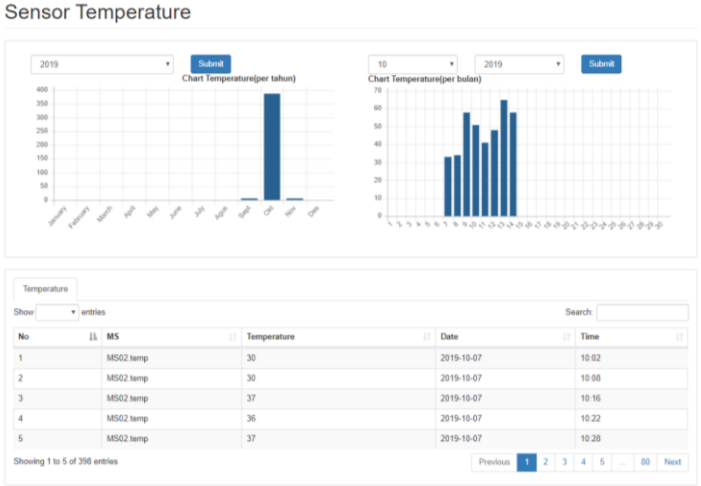

Figure 5. Data Monitoring Web Application at Gateway

Besides being accessed offline, a web application for monitoring data can also be opened online by accessing the VPS address at http://31.220.48.158/spotnet/. The appearance and menu on the web application are the online same as the offline one. This research implements 6 node MCU, which are temperature, air humidity, light intensity, pressure sensor, wind speed and soil moisture sensor. The location of the test was done by placing the node MCU and the IoT gateway in the Trilogi University garden for one week. The placement of the Node MCUs is separated by a gateway with a distance of two meters. The following picture 6 is a photo of a testing site of IoT gateway.

Figure 6. Testing Location of Node MCU and IoT Gateway

After one week, then a recapitulation of data communication is stored in RabbitMQ. Data transmission from the Node MCU to the IoT gateway is done every hour starting at 09:00 until 15:50. Specifically at the Node MCU the temperature sensor uses three types of data communication, while the other five Node MCUs use two data communications. The amount of data sent by all Node MCUs and stored in the IoT gateway is 7083 records. The following table 1 is the average test results of data communication system values from the six Node MCUs with different data communication protocols.

Table 1. Test Results Communication Data

No	Node MCU	Test Unit	Protocol	Remarks
1	Temperature Sensor	Periodic Data Transmission to IoT Gateway	MQTT, STOMP, Socket	Period: 5 second Size: 1 KB Publish: 0.22 ms Deliver: 0.24 ms RAM: 6.1 MB Queue: 3
2	Air Humidity Sensor Module	Periodic Data Transmission to IoT Gateway	MQTT, STOMP	Period: 5 second Size: 1 KB Publish: 0.19 ms Deliver: 0.21 ms RAM: 6.1 MB Queue: 2
3	Light Intensity Sensor Module	Periodic Data Transmission to IoT	MQTT, STOMP	Period: 5 second Size: 1 KB Publish: 0.22

		Gateway		ms Deliver: 0.24 ms RAM: 6.1 MB Queue: 3
4	Pressure Sensor Module	PeriodicData Transmission to IoT Gateway	MQTT, STOMP	Period: 5 second Size: 1 KB Publish: 0.18 ms Deliver: 0.22 ms RAM Usage: 6.1 MB Queue: 4
5	Wind Speed Sensor Module	PeriodicTransmission Datato IoT Gateway	MQTT, STOMP	Period: 5 second Size: 1 KB Publish: 0.21 ms Deliver: 0.22 ms RAM U sage: 6 MB Queue: 5
6	Soil Moisture Sensor Module	Periodic Data Transmission to IoT Gateway	MQTT, STOMP	Period: 5 second Size: 1 KB Publish: 0.19 ms Deliver: 0.22 ms RAM Usage: 6 MB Queue : 6

Based on test results from data communication from the six Node MCUs to the device, it gateway can run well and be successful. The average of the delivery time is 5 seconds with a size of 1 KB, the average time publish is 0.20 ms and time delivery of 0.22 ms. The results of this test indicate that the IoT gateway can handle data communication with a relatively fast time with a small file size. The following figure 7 is a graph of the results of testing data communication at the gateway.

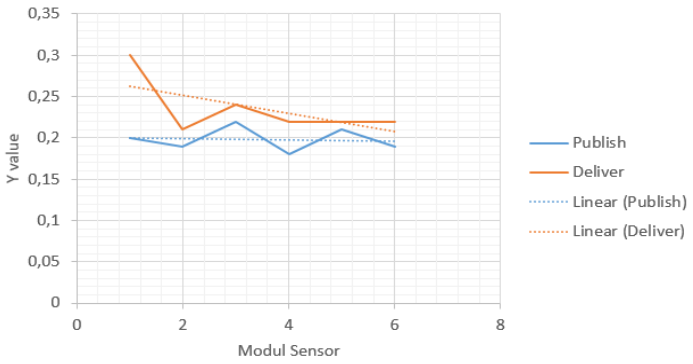

Figure 7. Testing Data Gateway

Testing The second test is a measurement of data logging on advice gateway that is used as data storage offline. This function is useful for users in accessing data through web applications provided by the gateway when the internet connection is lost. Data logging on gateway this uses the MariaDB database. The following table 2 is the result of data testing logging on the gateway.

Table 2. Test Results Communication Data

No	Table Data	Number of Data	Size (KB)	Overhead
1	MS01_ldr	1527	112	0
2	MS02_hum	924	96	0
3	MS02_temp	1296	96	0
4	MS04_ketan	851	96	0
5	MS05_tekud	2025	144	0
6	MS06_rpm	229	16	0
7	MS06_ws	231	16	0

Type the engine used in data logging above is InnoDB. The results of the test of logging data can be concluded that with the use oftype engine the InnoDB database overhead, low, relatively small data size and a large amount of data are obtained. The third test is network traffic which aims to determine the ability of the IoT gateway to handle incoming data from the sensor module device. The following figure 8 is a graph of the results of testing of network traffic to the gateway.

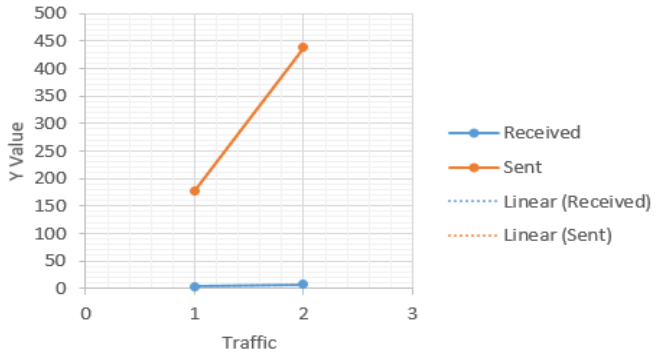

Figure 8. Graph of Traffic Testing Gateway

Based on the above test results it can be concluded that the average amount of data received by the data is 7.5 data per hour with an average size of 3 MB, while the average amount of data sent back is 437, 9 per hour with an average size of 176.8MB. The next test is to measure the performance of the operating system on the IoT gateway that aims to determine the level of resilience in the IoT gateway in handling processes, memory usage, CPU and bandwidth. The average CPU load is less than 10%, the average bandwidth usage is less than 3 mbps, and the average RAM memory usage is 550MB. Based on the results of tests on the performance of the

operating system on the IoT gateway, it can be concluded that the performance of the operating system is quite reliable in handling incoming data and functions carried out by the IoT gateway.

4. Conclusion

The application of the Advanced Message Queueing Protocol (AMQP) to the IoT gateway can overcome various data communication problems from the microcontroller. The test results show that the use of gateways can also minimize the burden of traffic entering the server, because sensor data are collected in the gateway first. The reliability of the tool RabbitMQ can also be seen when handling incoming traffic with stable performance and relatively low resource requirements such as CPU usage and RAM memory. In further research it is recommended to analyze the comparison between several tools from AMQP such as RabbitMQ with ActiveMQ and QPID. In addition, further research is about how handle data in the form of images or sounds from the node MCU to the IoT gateway.

Acknowledgment

Thank you to the Directorate of Research and Community Service, Ministry of Technology Research - National Innovation Research Agency which has provided funding through collaborative research schemes between universities. This research was conducted jointly between Universitas Trilogi and Pusat Penelitian Teknologi Informasi dan Komunikasi Institut Teknologi Bandung.

References

[1] S. M. P. P. S. A. P. Keyur K Patel, "Internet of Things-IOT Definition articl," *Int. J. Eng. Sci. Comput.*, vol. 6, no. 5, pp. 6122–6131, 2016.

[2] J. Gubbi, R. Buyya, and S. Marusic, "Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions," in *International Conference on I-SMAC*, 2017, pp. 1–19.

[3] N. Angelova, G. Kiryakova, and L. Yordanova, "The Great Impact of Internet of Things on Business," *Trakia J. Sci.*, vol. 15, no. Suppl.1, pp. 406–412, 2017.

[4] L. ;Yaddarabullah Y. Dewi, "Perancangan Alat Pembacaan Meter Air PDAM Menggunakan Arduino Uno," *Al-Fiziya J. Mater. Sci. Geophys. Instrum. Theor. Phys.*, vol. 1, no. 2, pp. 36–41, 2018.

[5] D. Kurniawan, Y. Yaddarabullah, and G. Suprayitno, "Implementasi Internet of Things pada Sistem Irigasi Tetes dalam Membantu Pemanfaatan Urban Farming," *Proceeding of The URECOL*, pp. 106–117, 2018.

[6] H. Eslava, L. A. Rojas, and R. Pereira, "Implementation of Machine-to-Machine Solutions Using MQTT Protocol in Internet of Things (IoT) Environment to Improve Automation Process for Electrical Distribution Substations in Colombia," *J. Power Energy Eng.*, vol. 03, no. 04, pp. 92–96, 2015.

[7] C. Akasiadis, V. Pitsilis, and C. D. Spyropoulos, "A multi-protocol IoT platform based on open-source frameworks," *Sensors (Switzerland)*, vol. 19, no. 19, pp. 1–25, 2019.

[8] M. P. Madhu and S. Dixit, "Distributing Messages Using Rabbitmq with Advanced Message Exchanges," *Int. J. Res. Stud. Comput. Sci. Eng.*, vol. 6, no. 2, pp. 24–28, 2019.

[9] C. Stipe, M. Eugen, and S. Zeljko, "STATE-OF-THE-ART OF MESSAGING FOR DISTRIBUTED COMPUTING SYSTEMS," *Int. J. Vallis Aurea*, vol. 3, no. 2, pp. 6–18, 2017.

[10] A. Shaout and B. Crispin, "Using the MQTT protocol in real time for synchronizing IoT device state," *Int. Arab J. Inf. Technol.*, vol. 15, no. 3A Special Issue, pp. 515–521, 2018.

[11] M. S. Balamurugan and R. Manojkumar, "Internet of things based gateways: Applications and challenges," *Int. J. Innov. Technol. Explor. Eng.*, vol. 8, no. 6, pp. 1816–1819, 2019.

[12] Yaddarabullah, M. F. Muttaqin, and M. Rafiansyah, "Service-Oriented Architecture for E-Marketplace Model Based on Multi-Platform Distributed System," *{IOP} Conf. Ser. Mater. Sci. Eng.*, vol. 662, p. 42028, Nov. 2019.