IoT real time data acquisition using MQTT protocol

2017 J. Phys.: Conf. Ser. 853 012003

(http://iopscience.iop.org/1742-6596/853/1/012003)

IOP Conf. Series: Journal of Physics: Conf. Series **853** (2017) 012003 doi:10.1088/1742-6596/853/1/012003

IoT real time data acquisition using MQTT protocol

R A Atmoko*, R Riantini, and M K Hasin

Marine Electrical Engineering, Shipbuilding Institute of Polytechnic Surabaya (SHIPS-PPNS), Indonesia

*Corresponding email address: ra.atmoko@ppns.ac.id

Abstract. The Internet of Things (IoT) provides ease to monitor and to gain sensor data through the Internet [1]. The need of high quality data is increasing to the extent that data monitoring and acquisition system in real time is required, such as smart city or telediagnostic in medical areas [2].Therefore, an appropriate communication protocol is required to resolve these problems. Lately, researchers have developed a lot of communication protocols for IoT, of which each has advantages and disadvantages. This study proposes the utilization of MQTT as a communication protocol, which is one of data communication protocols for IoT. This study used temperature and humidity sensors because the physical parameters are often needed as parameters of environment condition [3]. Data acquisition was done in real-time and stored in MySQL database. This study is also completed by interface web-based and mobile for online monitoring. This result of this study is the enhancement of data quality and reliability using MQTT protocol.

1. Introduction

The Internet of Things (IoT) is a concept that aims to extend the benefits of continuous internet connection for various things such as data sharing, remote control, and data monitoring. IoT is the idea of researchers who want to optimize equipment's such as sensor, radio frequency identification (RFID), wireless sensor network, and all equipment's connected to the Internet network to communicate with humans. IoT's challenge is to bridge the physical world and the information world such as the processing of data obtained from electronic equipment via an interface between the user and the equipment. Sensors collect physical data such as temperature and humidity, and then send it to the server to be stored in database or be displayed on the application interface [4].

IoT technology has started being developed in manifestations such as the Smart City, which has been developed in many metropolitan cities of the world, Smart Factory which is used for production optimization, Telediagnostic which facilitates the process of the patient's health monitoring, weather information system which is used for weather prediction, and so forth. Two of the processes that are often used in IoT applications are process control and monitoring (acquisition) of data. Communication protocols for acquisition of data that can be used are HTTP and MQTT protocol. HTTP protocol is used to web data exchange where it can usually be used for handling data acquisition of hardware, but the ability of HTTP is less suitable when being used on IoT-based applications.

HTTP protocol uses a request/response model, which is currently the most common message exchange protocol. MQTT uses a publish/subscribe pattern [5]. HTTP protocol is not designed for pervasive network and can cause a decrease in performance, especially in bandwidth usage and battery

durability. [6]. The protocol for IoT applications require less bandwidth, real-time response, and low energy use because it is commonly used for small appliances.

A study conducted by Vergara et.al showed the superiority of MQTT implemented on Android devices, which can decrease energy consumption compared to HTTP protocol [7]. MQTT and HTTP are both running over TCP. However, MQTT provides some advantages such as low energy, among others. [8]. Compared to IoT protocols such as COAP, MQTT also provides advantage to be used on client type of smartphone [9]. This study aims to implement the usability of MQTT protocol for temperature and humidity sensor data acquisition system interfaced on mobile-based and web-based applications**.**

2. HTTP protocol

Internet network is built to communicate via HTTP (Hyper Text Transfer Protocol). Various data, from images to texts, are sent over internet every day. HTTP is as a primary protocol interface to move a wide range of data quickly, easily, and stable from server to user devices such as browser. HTTP is built on TCP. HTTP ensures that data transmitted from one device to another will not corrupt so that the integrity of data transmitted is assured. HTTP is an open communication protocol that can be read by any devices that have been developed for HTTP protocol as browser or smartphone through browser application. An HTTP transaction consists of two parts: request command (request) sent from client to server, and response command (response) sent from server to client. The process of response and request is submitted using a data block with specific format known as HTTP Message. The messages are sent by HTTP which moves in one direction.

3. MQTT protocol

MQTT (Message Queuing Telemetry Transport) protocol is protocol specifically designed for "machine to machine" communication. MQTT protocol runs over TCP / IP and has a data packet size with low overhead minimum (> 2 bytes) so that consumption of the power supply is also small enough. This protocol is a data-agnostic protocol that can transmit data in various forms such as binary data, text, XML, or JSON and this protocol uses a publish/subscribe model rather than a client-server model.

Stack TCP/IP is now widely supported by microcontroller likeSTM32Fx7 series, as well as common market device board such as Wemos and Raspberry Pi. There are so many options for implementing MQTT protocol on devices. A common system of MQTT requires two main software components:

- MQTT Client is to be installed on device. A web platform, which uses Javascript, can use Client PAHO library of Eclipse.
- MQTT Broker serves to handle publish and subscribe data. A Linux platform can use broker that is available free such as mosquitto, HiveMQ etc.

The advantage of publish/subscribe system is that the data sender (publisher) and the data receiver (client) do not know each other because there is a broker between both. In addition, there is time decoupling which makes publisher and client unable to be connected simultaneously so that client will stay to receive delayed data previously.

3.1. Control signal

MQTT has 14 types of control signal, namely:

- CONNECT—Client request to connect to Server
- CONNACK—Connection Acknowledgement
- PUBLISH—A message which represents a new/separate publish
- PUBACK—QoS 1 Response to a PUBLISH message
- PUBREC—First part of QoS 2 message flow
- PUBREL—Second part of QoS 2 message flow
- PUBCOMP—Last part of the QoS 2 message flow

- SUBSCRIBE—A message used by clients to subscribe to specific topics
- SUBACK—Acknowledgement of a SUBSCRIBE message
- UNSUBSCRIBE—A message used by clients to unsubscribe from specific topics
- UNSUBACK—Acknowledgement of an UNSUBSCRIBE
- message
- PINGREQ—Heartbeat message
- PINGRESP—Heartbeat message acknowledgement
- DISCONNECT—Graceful disconnect message sent by clients
- before disconnecting

From those signals, there are only four main signals which are used directly by the client,namely PUBLISH, SUBSCRIBE, UNSUBSCRIBE, CONNECT. Other signals are part of the publish/subscribe mechanism.

3.2. Topic & Quality of Service (QoS)

In MQTT, it is as known topic that serves as filter for broker in sending message to every client connected and subscribed. MQTT provides quality level of service, which is called QoS. This level guarantees the reliability of message delivery. Level 0 message is sent only once. Messages are sent depending on the existence of network, and there is no attempt to transmit a message back. Level 1 messages are sent at least once so that if the subscriber does not recognize (acknowledge) the message, then the broker will send a message to publisher to receive the message acknowledgment status from the client. Level 2 is to make sure that message was received. With this level, it can be ensured that the message is certainly conveyed and avoided from duplication of messages sent.

4. System implementation

To implement usage of MQTT protocol, this study requires the following equipments:

1. Wemos D1 Mini & ESP 8266 Wifi Module
2. DHT 11 Temperature & Humidity Sensor
3. PC Server as MQTT Broker.
4. Android Smartphone

Wemos D1 Mini is a minimum system of controller that is equipped with ESP 8266 seriesWi-Fi module. The Wemos has 9 Input/Output digital and 1 Input/Output analogue. Wemos serves as publisher to transmit data of DHT 11 sensor to MQTT broker. DHT 11 is type of digital sensor of temperature and humidity. DHT 11 can measure temperature of 0^0-50^0 Celsius and humidity of 20%-90% RH. PC server is as MQTT broker to facilitate exchange of data between publisher and subscriber, or among hardware, mobile application, and web application.

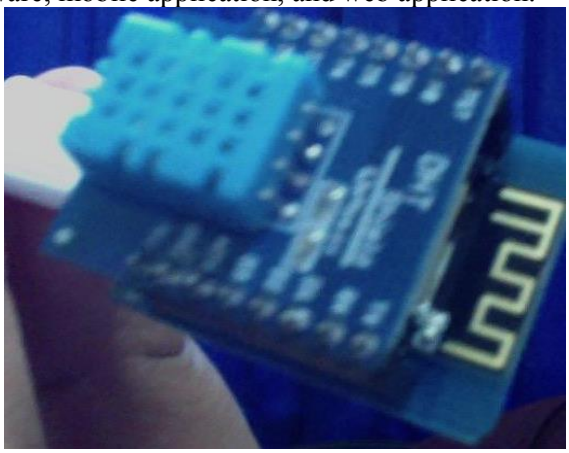

Figure 1.DHT11 Temperature & Humidity Sensor

A DHT11 sensor was connected to digital pin number 4 at Wemos. The sensor read temperature and humidity and then the information was sent (publish) to the broker. The MQTT broker used in this study was mosquitto that is an open source broker application of MQTT from Eclipse product. The data, which entered on broker then, was subscribed by client application. Subscribe application was built to use Javascript that can install a library MQTT named PAHO, an open source application of Eclipse product.

Table 1.MQTT Parameters.

Parameter Used	
Broker	Mosquitto
Subscriber	PahoJavascript
Topic	teleppns/temphum
MQTT Port	1883
Websocket Port	9001

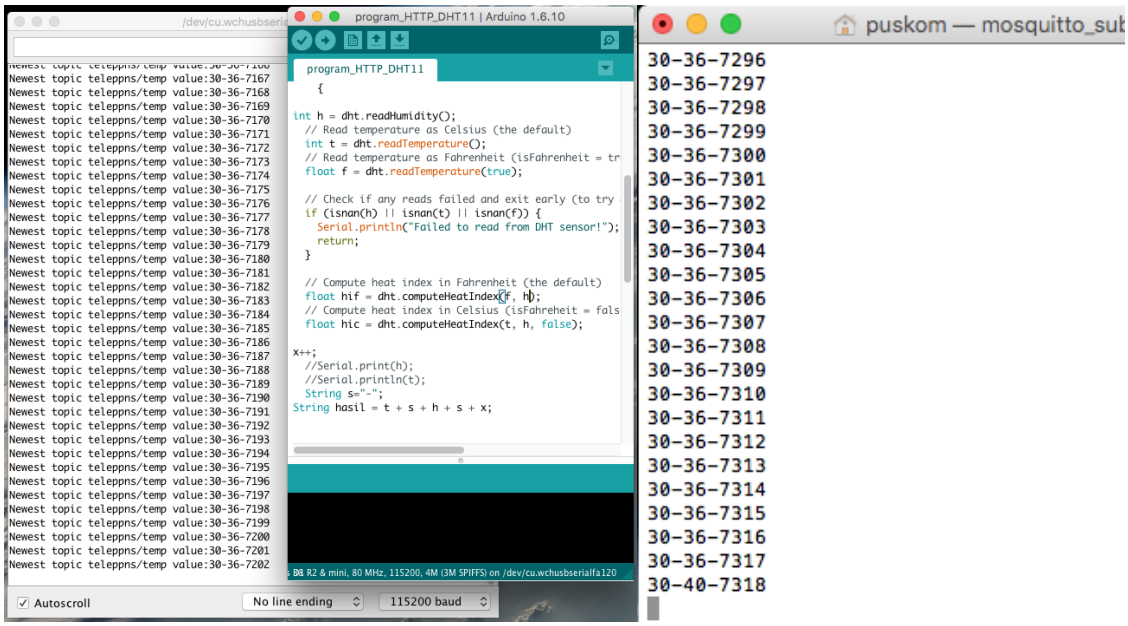

Figure 2.MQTT Broker &Arduino Serial Monitor.

The subscriber application for android platform was built with Phonegap using HTML and Javascript. The subscriber application of web platform was built using HTML and PHP. The subscriber application is added with a feature to save data to MySQL database. The subscriber application communicated to broker using Web Socket so that data exchange can happen in full duplex and real-time.

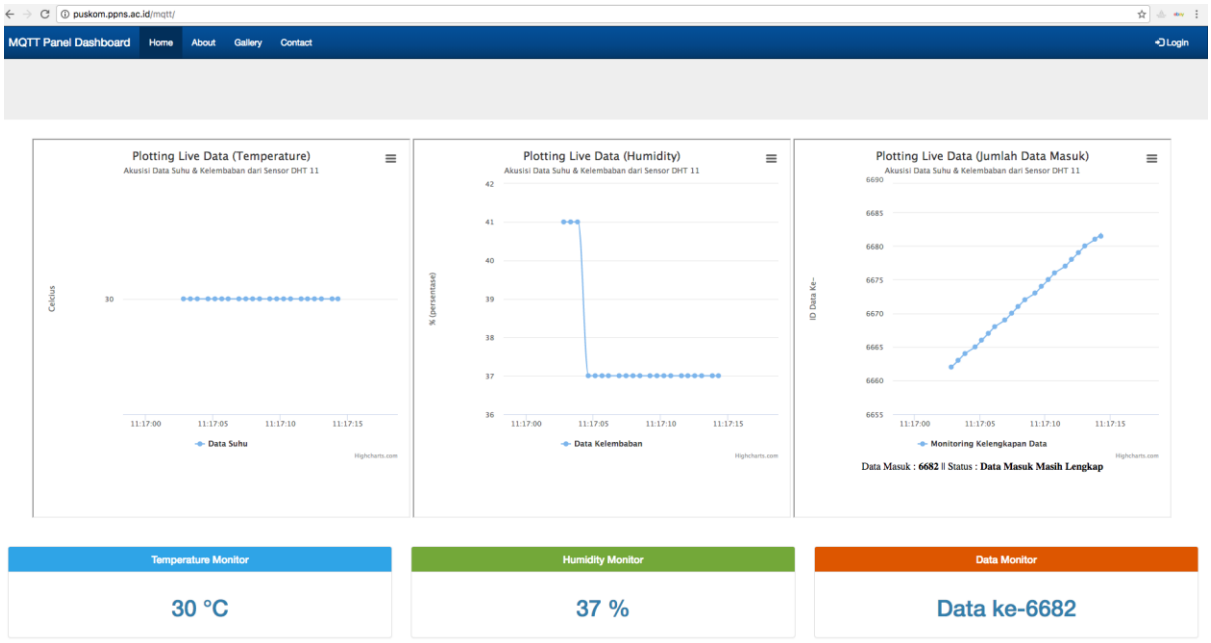

Figure 3. Web and Mobile Interfaces.

	id_data	date1_data	date_data	suhu_data	kelembaban_data	nomer_data
Edit Copy Delete	11401	2016-10-19 18:09:46	09:46.237677097321	35	39	3496914
Edit Copy Delete	11402	2016-10-19 18:09:46	09:46.244874000549	35	39	3496916
Edit Copy Delete	11403	2016-10-19 18:09:46	09:46.244784116745	35	39	3496917
Edit Copy Delete	11404	2016-10-19 18:09:46	09:46.248136997223	35	39	3496915
Edit Copy Delete	11405	2016-10-19 18:09:46	09:46.374195098877	35	39	3496918
Edit Copy Delete	11406	2016-10-19 18:09:46	09:46.576997995377	35	39	3496919
Edit Copy Delete	11407	2016-10-19 18:09:46	09:46.578825950623	35	39	3496920
Edit Copy Delete	11408	2016-10-19 18:09:46	09:46.594352006912	35	39	3496922
Edit Copy Delete	11409	2016-10-19 18:09:46	09:46.594355106354	35	39	3496921
Edit Copy Delete	11410	2016-10-19 18:09:46	09:46.681370019913	35	39	3496923

Figure 4. Data Storage MYSQL.

5. Results and discussion

This study also reports the performance test of HTTP and MQTT protocol. The test was performed by taking data for 60 seconds. The following table shows the capability of each protocol to transfer data from hardware to server and store it into a MySQL database.

Table 2. Comparison HTTP & MQTT.

Data aqcuiring order	Amount of Success Data transfer via HTTP Protocol	Amount of Success Data transfer via MQTT Protocol
1	938 data	6560 data

Data aqcuiring order	Amount of Success Data transfer via HTTP Protocol	Amount of Success Data transfer via MQTT Protocol
2	930 data	6530 data
3	935 data	6501 data
4	931 data	6505 data
5	938 data	6505 data
Mean	934.4 data	6520.2 data

The performance test was conducted by calculating the successfully sent data from hardware to server and also the successfully inserted data into MYSQL database for 60 seconds. Each transmission from hardware has a sequential ID, so that when data has been entered into database, it can be checked for its completeness of data to determine whether a loss occurs. The test results show that the ability of MQTT is better in transferring data than HTTP. The above table shows that the MQTT has the ability to send data up to six times faster than HTTP. This is because MQTT architecture is designed to provide transfer speed by small data size that can reduce data transmission time. MQTT has a header data size smaller than HTTP.

6. Conclusion

This study has implemented the use of MQTT protocol to build data acquisition application of temperature and humidity sensor with a mobile interface, which is android and web-based. The test result indicates that MQTT protocol has the ability of transfer data faster than HTTP protocol, which can transfer amount of data 6 times of HTTP capability. MQTT usage can be an option for the hardware data acquisition real-time application based on the Internet of Things.

References

[1] Andrea Z, Nicola B, Angelo C, Lorenzo V, Michele Z 2014 *Internet of Things for Smart Cities*, *IEEE Internet of Things J.* **1** (1)

[2] Mitsa T 2003 An Evolvable Software Framework for an Internet-based Telediagnostic System *Information Technology Applications in Biomedicine 4th International IEEE EMBS Special Topic Conf.*

[3] Atmoko R A 2013 Sistem Monitoring dan Pengendalian Suhu dan Kelembaban Ruang pada Rumah Walet Berbasis Android, Web, dan SMS *Semantik* **3** (1) pp. 283-290 ISSN 979-26-02666

[4] Suresh P, Daniel J V, Aswathy R H 2014 A state of the art review on the Internet of Things *(IoT) History, Technology and fields of deployment*

[5] Lampkin V *et al*. 2012 Building smarter planet solutions with MQTT and IBM WebSphere MQ telemetry *IBM, ITSO*

[6] Colitti W, Steenhaut K, De Caro N 2011 Integrating Wireless Sensor Networks with the Web *Proc. IP+SN* Chicago, USA

[7] Vergara E J, Prihodko M, Nadjm-Tehrani S 2013 Mobile Location Sharing: An Energy Consumption Study *e-Energy* pp 289-290

[8] De Caro N, Colitti W, Steenhaut K, Mangino G, Reali G 2013 Comparison of Two Lightweight Protocols for Smartphone-based Sensing *IEEE 20th Symposium on Communications and Vehicular Technology in the Benelux (SCVT) Namur* pp. 1-6.