**TELKOMNIKA Telecommunication, Computing, Electronics and Control**
Vol. 19, No. 4, August 2021, pp. 1169~1176
ISSN: 1693-6930, accredited First Grade by Kemenristekdikti, Decree No: 21/E/KPT/2018
DOI: 10.12928/TELKOMNIKA.v19i4.18358 

# Data delivery performance on MongoDB generated by WSN over high data load

**Mochammad Hannats Hanafi Ichsan[1], Buce Trias Hanggara[2], Rakhmadhany Primananda[3]**
[1,3]Computer Engineering, Faculty of Computer Scujience, Brawijaya University
[2]Information System, Faculty of Computer Science, Brawijaya University

**Article Info**

***Article history:***

Received Jul 30, 2020
Revised Jan 22, 2021
Accepted Jan 31, 2021

***Keywords:***

Data delivery
High load data
MongoDB
MQTT
WSN

**ABSTRACT**

Wireless sensor networks (WSN) Environment is a collection of nodes that sensing and sending data to other nodes in a large number of nodes. One of the WSN concepts is continuous or periodic data collection, then the data obtained can be further processed. From this process, it is possible to produce large data, so the process of sending and storing data becomes a high load. Data obtained from WSN nodes vary greatly, which affects on sending and storing of data. Therefore, a study is needed to analyze WSN's ability to send and store large amounts of data. Message query telemetry protocol (MQTT) is used because it could save resources on communication and MongoDB used because it does not have the concept of tables and rows, this is very suitable for variations in data generated by WSN. In this study, it can be concluded that the performance of MongoDB on high data amount is acceptable, so MongoDB is highly recommended for WSN implementation.



***Corresponding Author:***

Mochammad Hannats Hanafi Ichsan
Department of Informatics Engineering, Faculty of Computer Science
University of Brawijaya
Veteran St. No. 8 Malang, East Java, 65145, Indonesia
Email: hanas.hanafi@ub.ac.id

## 1. INTRODUCTION

One of the emerging technologies that continuously improved on the internet of things (IoT) and machine to machine communication in wireless sensor network (WSN) [1]. WSN Environment is a collection of nodes composed of sensors, actuators, processing devices, and communication devices on each node [2], [3]. In this point of view billion of sensors, actuators, and everyday objects that connected into the internet exchange data between them and other nodes, cloud, or another machine [4]. Their interaction has several purposes, one of the purpose is how to minimize human intervention [5]. These interaction activities such as sensing, measuring, processing, organizing, and optimizing and how to control/operate/monitor their environment [6]. Based on WSN activities, it could be made many data transactions between nodes and other nodes, broker, or/and cloud [7]. The amount of WSN nodes as needed can reach hundreds to thousands of nodes. It is caused that data produces in a high amount of data too [8]. So it is possible occurs the data accumulation is large that affects failure in the process of sending data to other nodes and saving data into the database. In this case, the data delivery protocol and database is very influential on the success in sending and storing data. Therefore a study is needed to measure WSN's performance in sending and storing data, that produce feasible WSN device architecture over high load data.

In a very large number of nodes, nodes in WSN can produce very large data as well [9]. In other cases, the process of sensing and monitoring in a system is demanded in real-time. Therefore the data

generated will be very large in a short time [10]. One of the many communication protocols that can save resources is message query telemetry protocol (MQTT) [11], [12]. MQTT is a data-centric communication protocol type that utilizes a publish-subscribe mechanism using the TCP/IP protocol as the background of its communication [13]. The MQTT protocol, which is open and lightweight in publish/subscribe protocol rather than hypertext transfer protocol (HTTP) [14]. MQTT is specifically designed to be a machine-to-machine and mobile application. Because the high load data produced by sensor nodes is very large it is not recommended using the HTTP protocol. In another area, the MQTT processing system needs little resources than HTTP [15]. MQTT could produce and processed less sized data structure, so the size of the packets sent from MQTT could be reduced [16]. So using MQTT is more suitable than the other communication protocol on WSN.

With so many nodes in WSN, the IoT concept fully exploits the economic model offered by the latest technology from cloud computing, to improve the quality of services delivered to users and help them facilitate problem-solving [17] and low power consumption [18]. However, related IoT services require large-scale data collection that results from the large number of sensors using a special gateway and stored in a database system so that a large amount of data will be a problem in sending and storing data. To overcome this problem, database technology with the concept of Not Only SQL (NoSQL) has been developed such as HBase, CouchDB, MongoDB, and Cassandra [19]. MongoDB is a document-oriented database system that uses the C++ programming language [20]. Data objects generated by the sensor will be stored in the form of BSON data. In MongoDB is different from the SQL concept that has several characteristics such as, every single data object is stored in a separate document, data storage is a graph. In MongoDB Data objects do not have the same table or column structure, so the storage process in MongoDB is easier and more effective for data storage because it is more flexible [21]. MongoDB is used because it can accommodate structured, semi-structured, and unstructured data efficiently on a large scale (big data/cloud). So MongoDB is very suitable for storing a large amount of data and the data with a different structure. Because of the WSN implementation, the data structure of each node can be different.

To measure how feasible the WSN architecture, required testing process based on data load that called load testing scenario. Load testing scenario is the process of running an application by imitating an actual user with the burden of sending large amounts of data [22]. Because sending large amounts of data caused a bottleneck on the system. The main purpose of this load testing scenario is to measure scalability, availability, and performance aspect from hardware and software used on the WSN architecture used [23]. In the data sensing environment in a node, the sensor only reads the data and then sent to the broker and then the broker sent into subscriber which is represent the cloud/database. The sensor reads real-time data about environmental conditions but does not change or delete stored data. Another thing that causes problems is the process of sending data, so it would be only to perform the insert query. Data generated by WSN nodes must not be lost, in the sense that the data generated by sensors even though there are problems on the network must not be lost. The data will be accumulated and sent to the sensor node so that the stack of data at the sensor node will be sent and query that is executed in large numbers.

In this research, data from the sensor node will be produced in such a way as to represent the process of collecting large amounts of data. Large amounts of data are generated from several sensor nodes/publishers and then sent to brokers with the MQTT protocol. After that, the data from the broker sent to the end node/subscriber. MongoDB installed on subscriber so insert query executed by subscriber. From here the feasibility analysis of the WSN architecture can be done by using MQTT and MongoDB for a large amount of data could be measured.

## 2. RESEARCH METHOD

The research that analyzes database insert performance using WSN over high data load is: the first is managing IoT data using JavaScript object notation (JSON) [24], this research measure insert performance into the JSON table, but not in MongoDB. The first research result is inserting 50.000-row data into the JSON table around 10.08 millisecond, but the data have not been stored into the database. That results in around 3.02 milliseconds for 6000 data. The second research is to measure database management system (DBMS) performance in MariaDB and JSON files [25]. The result of insert data for 20.000 data is 132.555 milliseconds around 148.437 milliseconds for 6000-row data, and for JSON files is 142 milliseconds or around 47.3 milliseconds. The third research compares HBase and Casandra [26], the research result around 1200 data row for Hbase is 1172.704 milliseconds and Cassandra 412 millisecond. The third research used fewer row data than the other. So, this research was conducted to meet the suitability and acceptability WSN requirement using MongoDB over 6000 data.

The state of the art of this research will be conducted by designing input, process, and output. Input on client/publisher that have done by NodeMCU to perform create and sending data into MongoDB using a

JSON data format that is formatted by broker. The data that was formatted would be inserted into MongoDB. This research will be conducted by designing input, process, and output. Input on client/subscriber is done by NodeMCU. NodeMCU used not only could process data but also send data at once process [27]. In this study, the client is not directly sensing data, data is made in the form of random data based on sensor representation. One of NodeMCU type for prototyping is ESP8266 [28], [29].

Then the data obtained from publisher will be sent to MQTT broker that uses mosquito then sent to publisher/host located on the PC. Mosquito is used because it does not need additional programs to be made and does not require special operations [30]. The broker only holds data from the publisher and is compiled to be sent to the subscriber. The block diagram of this research can be seen in Figure 1.

The flowchart of WSN architecture design in Figure 1 is illustrated in more detail in Figure 2. The process starts from the publisher node representing the sensor to acquire data by generating random data according to sensor data, for example, temperature and/or humidity data. To analyze with large amounts of data, publisher performs random data generation with the stack process. So the data generated accumulates in large numbers with the specified length.

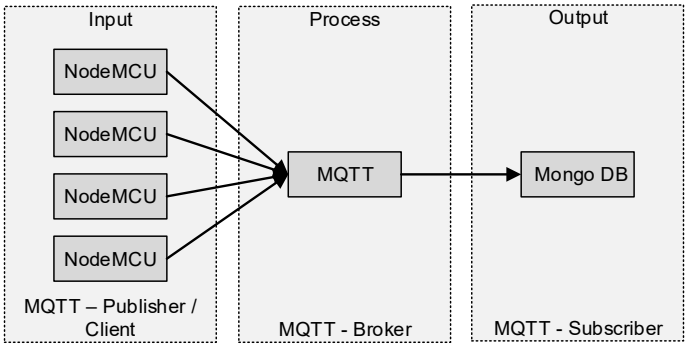


Figure 1. WSN architecture block diagram

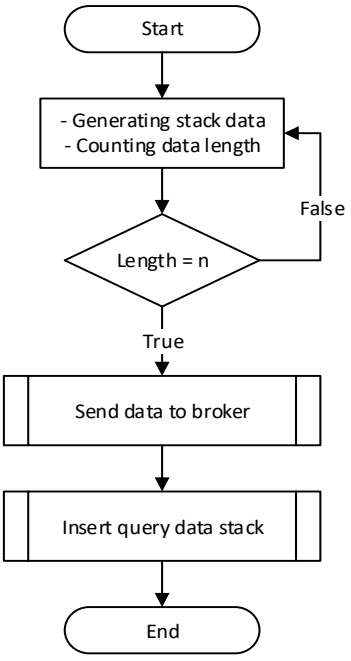


Figure 2. System flowchart design

After the data is collected according to the specified N-data, the data will be sent by the entire node to the broker. Then the data from the broker will be processed with a query which is then entered into the publisher that is installed by MongoDB. So that all data from the sensor node will simultaneously be entered

into the MongoDB query. From this, it can be justified the feasibility of MongoDB in receiving queries in a very large number of data rows. Queries on MongoDB are textual based so that data is represented in the form of rows of data not the size of the data.

Based on the flowchart design in Figure 2, the overall system design is as follows;

– Designing MQTT client/publisher

Client/subscriber software has functional needs, which are related to conducting random data gathering and creating a data stack. MQTT publisher is made in C on Arduino with Arduino IDE. The library used is the MQTT client library to create a data stack process to the process of sending data to the broker. Data sent to brokers in the JSON data format. When the initial run, the publisher will do the discovery process so that the publisher can be recognized by the broker. After the publisher is connected to the broker, then the new publisher will generate random and data stack. After that, the data created will be sent to the broker.

– Designing MQTT broker

MQTT broker designed that cover software functional requirement for writing and compiling programs. MQTT broker programmed by Phyton and the library used is PyMongo which enabling Phyton and MongoDB communication function. MQTT broker started from connecting broker with subscriber using MQTT. When the program is connected, it would be sent data stacked from publisher.

– Designing MongoDB/subscriber

MongoDB uses different names with the SQL database system. Tables in MongoDB are called collections, lines in MongoDB are called documents, columns in MongoDB are called Fields, and combinations of MongoDB are called Embedding & Linking. MongoDB ERD for database design is shown in Figure 3, there is a database named "SensorData" and has a collection of "NodeData", "ArduinoTimes", and "Times". "NodeData" contains information about sensor reading data. "ArduinoTimes" has information about the recording time, when the stack program inside the broker runs. "Times" has information about recording the time when data sent by MQTT-broker has been stacked into the database.

– Implementation

The MQTT block diagram as shown in Figure 1 consists of the publisher, broker, and MQTT subscriber. The flowchart thus explains how each component of the MQTT works. First, the broker program is turned on as a mediator of the discovery process, before running the publisher and customer program for data transmission. MQTT-Publisher publishes data to the broker, but the publisher does not store any data but only ensures the communication process is running. After done activating publisher to do the discovery process with the broker. This discovery process is the process of finding a broker to be able to communicate well. After all, publishers do a discovery, they can do the process of sending data.

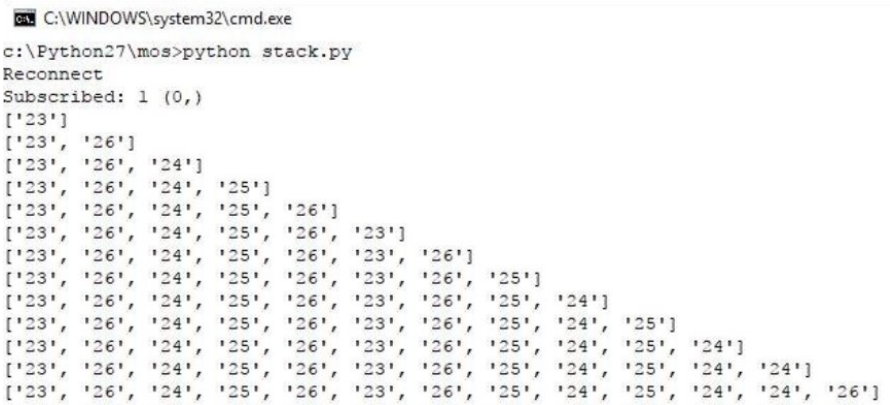


Figure 3. Data stack on publisher

## 3. RESULTS AND ANALYSIS

Results that have been obtained after implementation of the MQTT client/publisher hardware, the implementation consists of a NodeMCU random-generated data input module connected to the broker. MQTT-broker runs through the command prompt as a data transmission media broker. Data sent by publisher will be stored up to N-data following the program that has been designed. As shown in Figure 3, the "reconnect" record shows that the broker is in the process of connecting to the MQTT system, "subscribed" shows the number of NodeMCU/publishers that are subscribed to the broker.

Figure 3 explanations are, number "zero/0" after "Subscribed" indicates the MQTT protocol with QoS 0. After that the next line there are numbers 23 to 26 as data that has been sent by the publisher and received by MQTT-broker. It can be shown that the broker can receive the data stack properly. For the testing process, 1 node will generate random data from 50 to 1500 data lines. It is estimated that the process of 4 nodes will be carried out by producing 6000 data rows.

The test has been divided into three stages to determine the feasibility of the architecture that has been designed. The first test is to determine whether the publisher can be recognized by the broker. The second test is sending data from publisher to broker. The third test is testing the insert query from broker to MongoDB/subscriber. In this section, it is explained the results of the research and at the same time was given a comprehensive discussion. Results can be presented in figures, graphs, tables, and others that make the reader understands easily [2], [5]. The discussion can be made in several sub-chapters.

### 3.1. Discovery testing from publisher to broker

The first test carried out was testing the connectivity between publisher and broker. This test is carried out to test whether the publisher can connect to the broker at a certain distance. The distance is calculated when the publisher broadcasts to find the broker, then after the broker receives the broadcast message he will return the package related to the publisher's recognition.

Table 1 shows that 4 publisher one by one tested from 1 to 4 nodes. Each test is carried out 10 times, then the average is displayed in the table. The test takes into account the distance for discovery from 8, 12, 16, and 20 meters. At 8 to 16 meters can still be connected even though relatively long (up to more than 3 seconds) but can still be connected. When at 20 meters, the entire node cannot be connected to the broker. So that with many nodes, publisher and broker with NodeMCU can be connected with a maximum distance of 16 meters.

Table 1. Average discovery testing scenario

| Delay | 8 Metre (Second) | 12 Metre (Second) | 16 Metre (Second) | 20 Metre (Second) |
|---|---|---|---|---|
| 1 node | 0.0234 | 2.0156 | 2.0173 | Fail |
| 2 node | 0.2521 | 2.1214 | 2.0021 | Fail |
| 3 node | 0.2976 | 2.8735 | 2.0003 | Fail |
| 4 node | 0.4652 | 3.1947 | 3.3229 | Fail |

### 3.2. Data delivery testing from publisher to broker

This first test is the discovery protocol process, by activating the broker first. When the broker is active, then NodeMCU/publisher is activated. When publisher is activated, it will immediately conduct a broker search by publish/subscribe according to the MQTT protocol. This stage is the initial stage to determine the success of the overall research.

Initial testing is done by calculating the distance of delay in sending data between publisher and broker after it is connected. The overall results of the test are summarized in Figure 4. Where "testing number" 1-9 is for sending with 1 node, 10-18 for sending from 2 nodes, 19-27 for 3 nodes, and 28-36 for 4 nodes. Each node sends 50-1500 data rows of data in each test process. From this test scenario we get that all shipments with 50 data lines get a delay of 2.96 seconds, and a maximum delay of 123.9982 for the amount of data 1500 per node (6000 data for 4 nodes). Sending data obtained a stable increase of 1 to 4 nodes each, sending from 50 to 6000 data rows.

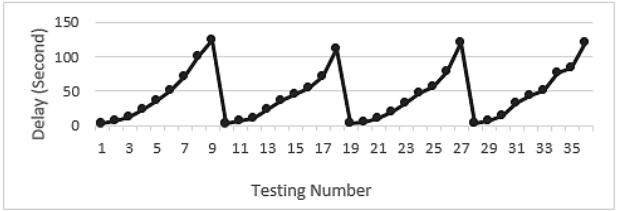


Figure 4. Discovery testing result

### 3.3. MongoDB insert data testing scenario

In the second test, the measurement of the success of the broker is to input the data query to MongoDB. This test is divided into two processes, the first process is to input query data with the database provided. The second process is the input data query without any database provided. This testing process is

carried out because the concept of WSN, the data obtained from the sensor is very convergent so that it is possible to change data from the sensor node/publisher. Same as the test in section 3.2 overall results of the test are summarized in Figure 5. Where "testing number" 1-9 is testing is sending 1 node, 10-18 for sending 2 nodes, 19-27 sending 3 nodes, 28-36 for sending 4 nodes. Each node sends data 50-1500 data row every testing process, so if all of four nodes sending data together total data is 6000 data row.

In Figure 5 this testing without database provided, it can be concluded that insert query requires unstable time, but increases with the number of rows that are input. Without database delay the minimum times are 0.095 seconds and the maximum times are 0.443 seconds in the insert query process. Furthermore, in Figure 6. related to testing with the database provided, it can be concluded that insert query requires unstable time, but increases with the number of rows that are input. With database delay the minimum times are 0.006 seconds and the maximum times are 0.164 seconds in the insert query process.

Based on Figure 5 and Figure 6, testing by providing a database and without a database has an average for without database testing of 0.2109 seconds and with a database of 0.0753. The difference between the two is 0.1356 second, this can happen because if without a database, MongoDB will make the database creation process first when the data is sent by the broker.

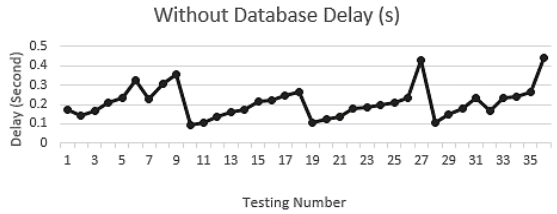


Figure 5. Publisher to broker testing without existing database result

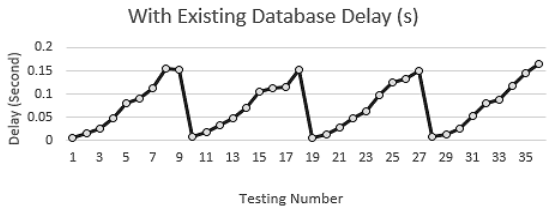


Figure 6. Broker to MongoDB with existing database result

## 4. CONCLUSION AND FUTURE WORKS

The WSN architecture design using the MongoDB database system on the MQTT protocol can be designed by determining what will be the MQTT component first. Components in the MQTT protocol are MQTT-publisher, MQTT-subscriber, and MQTT-broker. MQTT-subscriber uses a NodeMCU microcontroller to send data to the broker, MQTT-broker designed with Mosquitto software, MQTT-subscriber uses the MongoDB NoSQL database system where it functions as a receiver and storage sensor that is transmitted by IoT data devices.

This research produced results that prove MongoDB insert performance over high data load successfully done. Based on literature provided on over 6000 data rows as JSON performed on 1.08 milliseconds and MariaDB performed on 148 milliseconds. HBase on 412 milliseconds and Cassandra 1172 millisecond only on below 1200 data. MongoDB has a better result produced 0.43 millisecond over 6000 data rows. The discovery process was very good up to 16 meters, but the NodeMCU device was unable to broadcast at a distance of 20 meters. The next test result is the process of storing data into MongoDB carried out through the MQTT protocol. MQTT-broker can send 50 to 6000 data rows. The data storage speed for storing all data into an empty database takes less than 0.5 seconds, while for an existing database it takes less than 0.2 seconds. Based on testing that has been done it can be concluded that all components can work well. publishers can generate data as a representation of sensor data, then brokers can also receive all data sent from the publisher to 6000 data rows. MongoDB's performance is also very good, overall for 6000 data rows, it takes under 1 second to process the sensor data query over high data load. So, the MQTT-subscriber, broker, and publisher that are designed on this research are suitable and acceptable using this architecture and design.

For future work, there are many opportunities to develop this work, some of the directions are: 1) using the MQTT Protocol from QoS 0 to 2 for testing and implementation to get higher accuracy values; 2) using parameters other than Load Testing for the further testing depth of testing its database system, such as the stress testing database; 3) using MongoDB implemented in the form of a shaded cluster for maximum performance results, or implementing it on a cluster computer; 4) using synchronized time sending data from the NodeMCU microcontroller to MQTT to avoid data packet collisions or disposal of data packages; and 5) performing the same test in the same environment but using another database, e.g. Hbase, Cassandra and CouchDB to get maximum results and compare for the best DB.

**ACKNOWLEDGEMENTS**

Thanks to Robotics and Embedded System Laboratory, Faculty of Computer Science, Brawijaya University. This research has been developed and tested on Robotics and Embedded System Laboratory. This research dedicated to Icha, Zio, and Ezra who give their encouragement while working on research and in pandemic conditions.

**REFERENCES**

[1] V. Reddy and G. P., "Integration of internet of things with wireless sensor network," *International Journal of Electrical and Computer Engineering (IJECE),* vol. 9, no. 1, pp. 439~444, 2019, doi: 10.11591/ijece.v9i1.pp439-444.

[2] A. R. Jaladi, K. Khithani, P. Pawar, K. Malvi and G. Sahoo, "Environmental Monitoring Using Wireless Sensor Networks(WSN) based on IOT," *International Research Journal of Engineering and Technology (IRJET),* vol. 4, no. 1, pp. 1371-1378, 2017.

[3] P. M. Mahajan, "WSN: Infrastructure and Applications," *International Journal of Scientific Research in Network Security and Communication (IJSRNSC),* vol. 6, no. 1, pp. 6-10, 2018.

[4] L. Atzori, A. Iera and G. Morabito, "Understanding the Internet of Things: definition, potentials, and societal role of a fast evolving paradigm," *Elsevier : Ad Hoc Networks,* vol. 56, no. 1, pp. 122-140, 2017, doi: 10.1016/j.adhoc.2016.12.004.

[5] T. Issac, S. Silas and E. B. Rajsingh, "Investigations on System Modeling Simulations for Solving Heterogeneous WSN Task Assignment Problem using Multilayer Feed Forward Neural Networks," in *International Conference on Advances in Material Science & Nanotechnology*, 2020, doi: 10.1016/j.matpr.2020.03.774.

[6] S. Abdollahzadeh and N. J. Navimipour, "Deployment strategies in the wireless sensor network: A comprehensive review," *Elsevier : Computer Communications,* vol. 91, pp. 1-16, 2016, doi: 10.1016/j.comcom.2016.06.003.

[7] A. Mahboub, E. M. En-Naimi, M. Arioua, H. Barkouk, Y. E. Assari and A. E. Oualkadi, "An energy-efficient clustering protocol using fuzzy logic and network segmentation for heterogeneous WSN," *International Journal of Electrical and Computer Engineering (IJECE),* vol. 9, no. 5, pp. 4192-4203, 2019, doi: 10.11591/ijece.v9i5.pp4192-4203.

[8] G. A. Mutiara, N. Suryana and O. Mohd, "WSN nodes power consumption using multihop routing protocol for illegal cutting forest," *TELKOMNIKA Telecommunication, Computing, Electronics and Control,* vol. 18, no. 3, pp. 1529-1537, 2020, doi: 10.12928/TELKOMNIKA.v18i3.14844.

[9] A. Djedouboum, A. A. Ari., A. Gueroui, A. Mohamadou and Z. Aliouat, "Big Data Collection in Large-Scale Wireless Sensor," *Sensors,* vol. 18, no. 1, p. 4474, 2018, doi: 10.3390/s18124474.

[10] D. Pal, T. Triyason and P. Padungweang, " Big Data in Smart-Cities: Current Research and Challenges," *Indonesian Journal of Electrical Engineering and Informatics,* vol. 6, no. 4, pp. 351-360, 2018, doi: 10.11591/ijeei.v6i4.543.

[11] V. Gazis, M. Görtz, M. Huber, A. Leonardi, K. Mathioudakis, A. Wiesmaier, F. Zeiger and E. Vasilomanolakis, "A survey of technologies for the internet of things," in *2015 International Wireless Communications and Mobile Computing Conference (IWCMC)*, Dubrovnik, Croatia, 2015, doi: 10.1109/TENCON.2018.8650550.

[12] J. Dizdarević, F. Carpio, A. Jukan and X. Masip-Bruin, "A Survey of Communication Protocols for Internet of Things and Related Challenges of Fog and Cloud Computing Integration," *ACM Computing Surveys,* vol. 51, no. 6, pp. 116:1-116:29, 2019, doi: 10.1145/3292674.

[13] A. Rodríguez, L. M. Kristensen and A. Rutle, "Formal Modelling and Incremental Verification of the MQTT IoT Protocol," Transactions on Petri Nets and Other Models of Concurrency XIV, Springer, Berlin, Heidelberg, 2019, doi: 10.1007/978-3-662-60651-3_5.

[14] A. Shaout and B. Crispin, "Using the MQTT Protocol in Real Time for Synchronizing IoT Device State," *The International Arab Journal of Information Technology,* vol. 15, no. 3A, pp. 515-521, 2018.

[15] S. Xuan and D. Kim, "Development of Cloud of Things Based on Proxy Using OCF IoTivity and MQTT for P2P Internetworking," *Peer-to-Peer Networking and Applications,* vol. 13, no. 1, pp. 729-741, 2020, doi: 10.1007/s12083-020-00891-9.

[16] J.-H. Park, H.-S. Kim and W.-T. Kim, "DM-MQTT: An Efficient MQTT Based on SDN Multicast for Massive IoT Communications," *Sensors,* vol. 18, no. 1, p. 3071, 2018.

[17] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum and N. Ghani, "Demystifying IoT Security: An Exhaustive Survey on IoT Vulnerabilities and a First Empirical Look on Internet-Scale IoT Exploitations," *IEEE Communications Surveys & Tutorials,* vol. 21, no. 3, pp. 2702-2733, 2019, doi: 10.3390/s18093071.

[18] D. Imededdin, A. Salih and H. Medkour, "Design and implementation of low power consumption wireless sensor node," *TELKOMNIKA Telecommunication, Computing, Electronics and Control,* vol. 17, no. 6, pp. 2729-2734, 2019, doi: 10.12928/TELKOMNIKA.v17i6.12047.

[19] R. Mahima, K. S. K. Ruba and A. Pushpalatha, "Energy Optimized Data Routing Using PSO in WSN," *IJETIE,* vol. 6, no. 3, 2020.

[20] F. Lu, T. Fang, Z. Zhang, S. Li, J. Chen, H. An and W. Han, "Improving the Performance of MongoDB with RDMA," in *2019 IEEE 21st International Conference on High Performance Computing and Communications*, Zhangjiajie, China, 2019, doi: 10.1109/HPCC/SmartCity/DSS.2019.00144.

[21] J. Kachaoui and A. Belangour, "MQL2SQL: A Proposal Data Transformation Algorithm from MongoDB to RDBMS," *International Journal of Advanced Trends in Computer Science and Engineering,* vol. 9, no. 2, pp. 2457-2463, 2020, doi: 10.30534/ijatcse/2020/234922020.

[22] R. R.-Zurrunero, R. Utrilla, E. Romero and A. Araujo, "An Adaptive Scheduler for Real-Time Operating Systems to Extend WSN Nodes Lifetime," *Advanced Wireless Communications and Mobile Computing Technologies for the Internet of Things,* vol. 2018, no. 1, pp. 1-10, 2018, doi: 10.1155/2018/4185650.
[23] P. O'Donovan, C. Gallagher, K. Leahy and D. T.J.O'Sullivan, "A comparison of fog and cloud computing cyber-physical interfaces for Industry 4.0 real-time embedded machine learning engineering applications," *Elsevier : Computers in Industry,* vol. 110, no. 1, pp. 12-35, 2019, doi: 10.1016/j.compind.2019.04.016.
[24] C. Asiminidis, G. Kokkoni and S. Kontogiannis, "Managing IoT data using relational schema and JSON fields, a comparative study," *IOSR Journal of Computer Engineering (IOSR-JCE) ,* vol. 20, no. 6, pp. 46-52, 2018, doi: 10.9790/0661-2006034652.
[25] I. Lazarev and I. Gosudarev, "DBMS Performance Issues on a Single-Board Computer Raspberry Pi 3 Model B," ITMO University, St. Petersburg, Russia, 2019.
[26] P. Jakkula, "HBase or Cassandra? A Comparative study of NoSQL Database Performance," *International Journal of Scientific and Research Publications,* vol. 10, no. 3, pp. 808-820, 2020.
[27] A. C. Bento, "IoT: NodeMCU 12e X Arduino Uno, Results of an experimental and comparative survey," *International Journal of Advance Research in Computer Science and Management Studies,* vol. 6, no. 1, pp. 46-56, 2018.
[28] R. Firmansyah, A. Widodo, A. D. Romadhon, M. S. Hudha, P. P. S. Saputra and N. A. Lestari, "The prototype of infant incubator monitoring system based on the internet of things using NodeMCU ESP8266," in *IOP Conf. Series: Journal of Physics: Conf. Series 1171*, 2018, doi: 10.1088/1742-6596/1171/1/012015.
[29] D. A. Aziz, "Webserver Based Smart Monitoring System Using ESP8266 Node MCU Module," *International Journal Of Scientific & Engineering Research,* vol. 9, no. 6, pp. 801-807, 2018.
[30] R. Chiwariro and S. Rajendran, "Intra-Optimised Lightweight Enciphering Algorithm based on MQTT Protocol for Internet of Things Secure Application," *International Journal of Pure and Applied Mathematics,* vol. 120, no. 6, pp. 2629-2644, 2018.

## BIOGRAPHIES OF AUTHORS

**Mochammad Hannats Hanafi** Ichsan received Bachelor Degree at Informatics – Electronics Engineering Polytechnic Institute of Surabaya (EEPIS), Surabaya, East Java, Indonesia on 2010, completed master degree at Electrical Engineering - Brawijaya University, Malang, East Java, Indonesia on 2013. From 2014 until present as Lecturer at Computer Engineering Study Program and member on Robotics and Embedded System laboratory, Informatics Department Faculty of Computer Science – Brawijaya University. Research interest Computer System Engineering, Intelligence System and Optimization.

**Buce Trias Hanggara** received Bachelor Degree at Informatic Engineering in Brawijaya University, Malang, East Java, Indonesia on 2012. Completed Master Degree at Information System in Institute Teknologi Sepuluh Nopember (ITS), Surabaya, East Java, Indonesia on 2016. Presently Lecturer at Information Technology Study Program and member of Multimedia Game and Mobile Applications Research Team, Faculty of Computer Science in Brawijaya University. Research interest in multimedia, integrarion technology, and IT Governance & Management.

**Rakhmadhany Primananda** received Bachelor Degree at Department of Electrical Engineering - Brawijaya University, Malang, East Java, Indonesia in 2010. Then, completed Master Degree at Department of Informatics - Sepuluh Nopember Institute of Technology, Surabaya, East Java, Indonesia in 2014. Currently working as a lecturer at Computer Engineering Study Program and as a member at Information Based Network laboratory-Department of Informatics, Faculty of Computer Science - Brawijaya University. Research interest include Wireless Sensor Network, Internet of Things, Fault Tolerant Computer System.