Journal of Electronics, Electromedical Engineering, and Medical Informatics (JEEEMI)
Multidisciplinary : Rapid Review : Open Access Journal Vol. 3, No. 2, July 2021, pp.64-71 ISSN: 2656-8632

RESEARCH ARTICLE OPEN ACCESS

Manuscript received June 5, 2021; revised July 2, 2021; accepted July 24, 2021; date of publication July 28, 2021
Digital Object Identifier (DOI): https://doi.org/10.35882/jeeemi.v3i2.1


# Performance Evaluation of IoT-based SpO2 Monitoring Systems for COVID-19 Patients

**Trie Maya Kadarina[1] and Rinto Priambodo[1]**
[1] Universitas Mercu Buana, Jl. Meruya Selatan No. 1 Kembangan. Jakarta Barat, Indonesia

Corresponding author: Trie Maya Kadarina (e-mail: trie.maya@mercubuana.ac.id).

**ABSTRACT** Internet of Things (IoT) applications can be used in healthcare services to monitor patients remotely. One implementation is that it is used to monitor COVID-19 patients. During the COVID-19 pandemic, people who are infected without symptoms must self-isolate so that the virus does not spread. Measurement of blood oxygen levels or SpO2 is one of the measurements that must be carried out in routine examination procedures for self-isolating patients for early detection of silent hypoxemia in COVID-19 patients. Previous research has developed an IoT-based health monitoring system with a Wireless Body Sensor Network (WBSN) and a gateway that can be used for data acquisition and transmission. The system uses a home pulse oximeter to measure SpO2 and heart rate and an Android application that functions as an IoT gateway to collect data from sensors and add location information before sending data to the server. The WBSN has been successfully integrated with two types of open source IoT platforms, namely ThingsBoard and Elasticsearch Logstash Kibana (ELK). However, it is necessary to carry out further studies on analytical and experimental performance tests of the two systems. Therefore, the purpose of this study is to develop a performance evaluation of the IoT-based SpO2 monitoring systems using the Thingsboard and ELK as IoT platforms. To evaluate the performance, we ran the monitoring system on both platforms using pulse oximeter and Android device as IoT gateway with HTTP and MQTT as a transport protocol for sending the data to the server. From this study, we found that the average time of message delivery in ELK compared to ThingsBoard using the same protocols was higher but stable.

**INDEX TERMS** IoT, SpO2, Monitoring System, COVID-19.

## I. INTRODUCTION

The World Health Organization (WHO) declared the COVID-19 virus to be a pandemic in March 2020, indicating that it has spread across the world. Additionally, the number of infected patients is continuously increasing. Many attempts have been made to identify COVID-19 patients by tracing contacts, which leads to patient isolation and thus slows the virus's spread [1].

Internet of Things (IoT) is a technology that can be used to treat COVID-19 patients as developed by [2] and [3]. The measurement of blood oxygen levels, or $SpO_2$, is one of the measurements that must be performed in regular assessment procedures for self-isolating COVID-19 patients. In COVID-19 patients, measuring $SpO_2$ at home will help diagnose silent hypoxemia early [4]. Measurement of $SpO_2$ is carried out regularly and continues to be monitored. If the value is less than 92%, then the patient must be immediately taken to the hospital [5]. Regular $SpO_2$ monitoring and hypoxemia detection can be done with a pulse oximeter device connected to the server via an Android smartphone [6]. In addition, the availability of location information will also make it easier for doctors and paramedics not only to determine the course of action but also to help patients perform actions independently if needed.

Wearable sensor devices, especially those based on IoT, can be used for observation and data recording at home, at work, or during travel for a longer duration when compared to observations made during laboratory visits [7]. This massive collection of data, when analyzed and presented in a visual that is easy for health workers to understand, has the potential to help improve the quality of health services and reduce costs [8]. In [8], also explains the architecture of a

remote health monitoring system that has a data acquisition component, a data transmission component, and a server that functions as data storage, analytics, and visualization. This component is available not only on existing IoT platforms (such as Thingsboard, Thingspeak Ubidots, Azure IoT, etc.) but also on other stack applications such as Elasticsearch Logstash Kibana (ELK). Although not specifically built as an IoT platform, ELK has an excellent reputation for the capabilities required, especially in health monitoring systems.

In previous research, two types of processing systems and presentation of patient location data along with the measurement results of the $SpO_2$ sensor were visualized on a server in one dashboard [9] [10]. The $SpO_2$ measurement results from previous studies came from direct measurements with a pulse oximeter as part of wireless body sensor networks (WBSN). Furthermore, Android devices with GPS and Location APIs are used to obtain the location data that is sent. Before sending the data to the server, the Android device that serves as a portal will add the patient ID. This patient ID is used to verify and distinguish between the ownership of incoming data from one patient to another.

In both previous studies, open-source IoT platforms were used, namely Thingsboard and ELK (Elasticsearch, Logstash, and Kibana) [9] [10]. Thingsboard is an open-source IoT platform application that emerged in 2016. Thingsboard is a dependable and stable platform. In addition, the technology used is very general. Java 8 and its databases support MySQL and Postgresql, both of which are widely used databases. RESTful HTTP or MQTT can be used to connect to computers, and both libraries are commonly available and supported on the Arduino and Raspberry Pi platforms. Thingsboard helps us to track and manage IoT devices, collect and visualize data from IoT devices, analyze data and cause alerts with complex event processing, send device data to other systems, create workflows based on device life-cycle events, REST API events, RPC requests, and more, as well as develop custom features based on the use-case. (thingsboard.io).

Elasticsearch, Logstash, and Kibana (ELK) is a technology that excels at gathering vast volumes of log data and other data from a variety of sources and presenting it in graphs and charts. ELK is a collection of three applications that, although they can be used separately, are most often used as part of an integrated solution known as Elastic Stack [11]. Elasticsearch provides functions for storing, indexing, and querying large amounts of data in real-time, as needed by monitoring and analytics applications [12]. The Logstash framework can collect data from multiple sources in real-time and send it to Elasticsearch for storage and indexing. Kibana, a data visualization tool, can be used to view Elasticsearch data that has been indexed and stored. ELK supports vertical and horizontal scale growth, which is needed in IoT development. As a result, an IoT device based on ELK can be built quickly and cheaply [11] .

Both IoT-based applications using the open-source Thingsboard [9] and ELK [10] have been successfully realized and have been able to display patient condition data and the location where the patient is. The data collected is obtained automatically by integrating a pulse oximeter with an IoT gateway. However, it is necessary to carry out further studies on analytical and experimental performance tests of the two systems. Studies on the performance evaluation of various types of IoT platforms have also been conducted by [13] using MQTT and HTTP protocols where ThingsBoard performed better than SiteWhere. Performance evaluation includes scalability (throughput and average response time) and stability (in resource utilization and resilience) [13]. Also, in [14], characteristics of Cloud-IoT Platform services are also being compared, including the use of MQTT and HTTP protocols. A study on the evaluation of the performance of the IoT system for health applications has also been conducted by [15]. They conducted a comparative analysis study of conventional cloud computing models and fog computing concepts in scenarios for health monitoring applications. In order to determine which platform is more appropriate for implementation, this study will compare the performance of IoT-based $SpO_2$ monitoring systems built on the Thingsboard and ELK platforms. It is hoped that after conducting this comparative study, it can be concluded that the right platform to be implemented in a health monitoring system is in accordance with the criteria needed for handling COVID-19 patients in Indonesia.

## II. MATERIALS AND METHODS

One of the tests that must be performed in regular assessment procedures for COVID-19 self-isolating patients is blood oxygen levels or $SpO_2$. The $SpO_2$ measurements data is obtained from direct measurements with a home pulse oximeter linked to Android as part of the wireless body sensor networks (WBSN), as shown in FIGURE 1.


**FIGURE 1. WBSN (Home Pulse Oximeter with Android)**

Android devices with GPS and Location APIs are used to obtain the location data that is sent. Before sending the data to the server, the Android device that serves as a portal will add the patient ID. This patient ID is used to verify and distinguish between the ownership of incoming data from one patient to another. The pulse oximeter used in this system can send data via BLE or Bluetooth Low Energy. Unlike traditional Bluetooth, BLE has many benefits, including lower energy consumption and the ability to remain connected even when not transmitting data. BLE-enabled devices can connect to several connections at the same time, so choosing a number of devices that can

communicate via BLE is a good way to create a wireless network, as suggested.

A sensor device may send a large amount of data as an array of bytes in a specific order, allowing a large amount of data to be transmitted at once to other devices or applications. A BLE connectivity, as shown in FIGURE 2, differs from traditional Bluetooth in that a system that wants to exchange data through BLE can activate one or more services, each of which can contain multiple data streams (characteristic). Applications that want to get data from sensor devices using BLE (peripheral devices) should register first, and when the data is available, it will be sent directly to the data subscriber or the central device, similar to the publish-subscribe principle in the message broker framework. An Android-based application that functions as an IoT gateway is another module in the wireless body sensor network. This application is critical to the monitoring process because, in addition to incorporating location information collected from the Android device's GPS sensor, it also adds user information stored on the device. As a result, consumers will be able to manage the details more easily in the future.

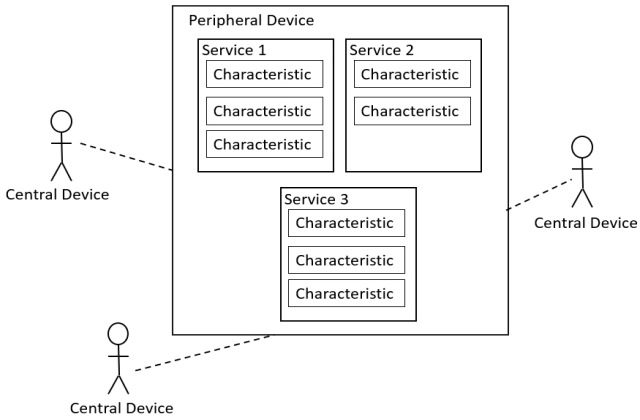



**FIGURE 2. BLE connection concept**

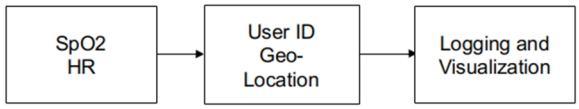



**FIGURE 3. Data collection up to visualization**

FIGURE 3 shows the data flow starting from data collection on the wireless body sensor network to visualization on the server. Sensors collect data from $Sp0_2$ and heart rate measurements and send them to the gateway to get a user ID and geolocation information attached to the data. Afterward, these data packages are sent to a server for logging and displayed to end-users when needed.

ThingsBoard is built as an IoT platform with some key components. ThingsBoard Transports works as a transport layer that receives data from devices. This transport layer provides MQTT, HTTP, and CoAP based API so that we can choose which protocol works best with the devices that will be used in the project. ThingsBoard also has Core components that are responsible to manage the API calls, subscriptions, devices, and tenants so we can organize multiple devices by assigning them into a group of tenants or customers. Messages coming from devices will be processed by ThingsBoard Rule Engine, where data validation, data processing, or data processing can take place to trigger actions based on an event. The last major component is the web user interface for the users to manage the devices and for visualizing the data stored in a database. This web communicates with ThingsBoard Core using REST API to get data from the database. ThingsBoard uses PostgreSQL as the database and optional Cassandra or TimescaleDB to store the time series data. FIGURE 4 shows the monolithic architecture of ThingsBoard with components used in this experiment.

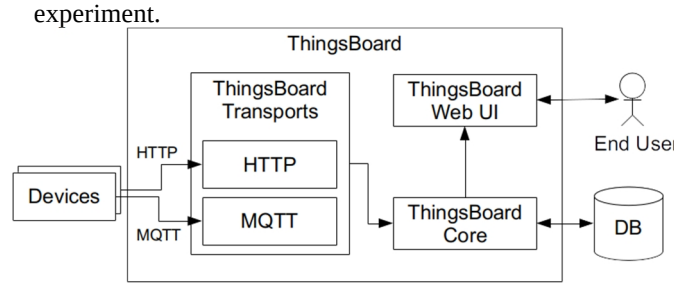



**FIGURE 4. The monolithic architecture of ThingsBoard**

ThingsBoard architecture brings all components in a single Java Virtual Machine (JVM) while running. While it is easier to launch and minimize the use of hardware resources for all of the components, each component in monolithic mode has to share memories so that when one of the components needs to work on a lot of processes, it may impact other components. Therefore, in this scenario where only one IoT device involved the monolithic mode is feasible to use.

On the contrary ELK stack is only available in a microservices architecture. FIGURE 5 illustrates service components in ELK that were used in this experiment. Each component works independently and communicates to each other using HTTP(S) or TCP protocol. This gives flexibility for deployment to meet the scale of the devices and processes that will be involved in the ecosystem. While not designed specifically as an IoT platform, ELK stack provides similar capabilities with ThingsBoard or any other IoT platform. It has Beats and Logstash that will ingest monitoring data from many locations and many formats. It also has the capabilities to process, validate or transform the data before inserting it to a database. Elasticsearch is designed to store very large data with distributed architecture and optimized for searching, and Kibana has a user interface to navigate through the data and visualize them. Both ThingsBoard web user interface and Kibana can display time-series data in many forms like charts, tables, and maps. With the capability of monitoring logs and activities of any kind of application, the ELK stack can also be used in a similar use case like monitoring and visualizing data of IoT devices. Research in [10] has shown that the ELK stack can be used for

monitoring patients wearing sensors and sending data continuously.

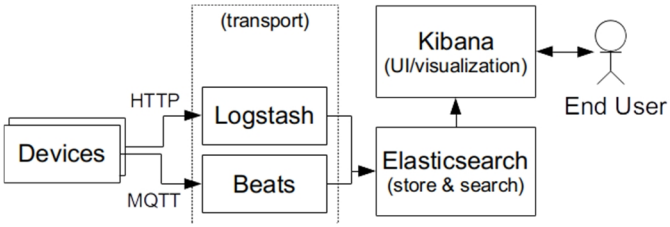



**FIGURE 5. The microservice architecture of the ELK stack**

In this experiment, we ran two scenarios of tests on both platforms using a different communication protocol between the Android device and the server. The flow diagram of the process running each platform can be seen in FIGURE 6.

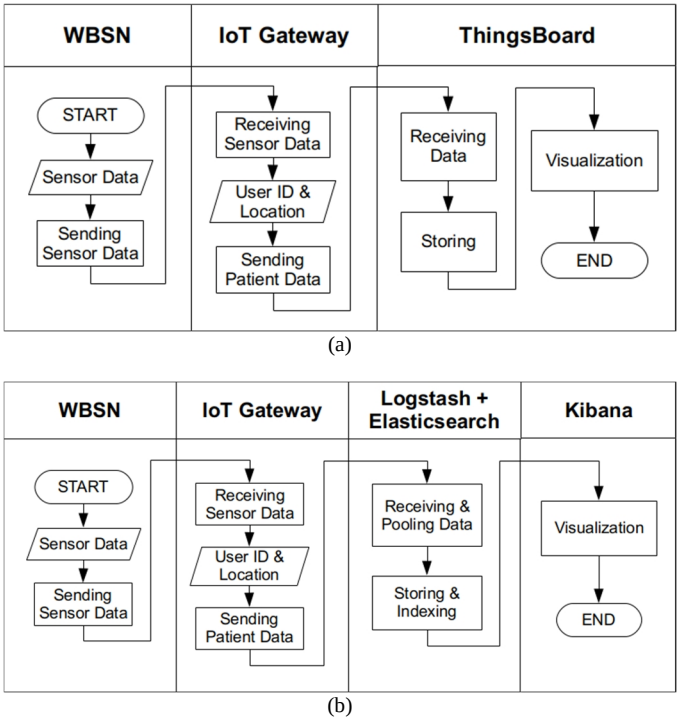



**FIGURE 6. Flow diagram of the monitoring system (a) using ThingsBoard (b) using ELK stack**

In FIGURE 6, WBSN uses a BLE connection to send data to the IoT gateway. In addition, the IoT gateway collects sensor data as well as patient location information and transmits it to the server. ThingsBoard or Elasticsearch and Kibana will receive the information and display it on its dashboard.

The first scenario in this experiment used the MQTT protocol, and the second one used HTTP. Both the ThingsBoard platform and ELK stack support MQTT and HTTP protocol to communicate with IoT devices. The testing environment consisted of a pulse oximeter sensor for measuring $SPO_2$ and heart rate. This pulse oximeter sensor was connected to an android device using Bluetooth Low Energy (BLE). An application in the Android device would establish a persistent connection with the sensor and subscribe to a service to start receiving the data from the measurement. As an IoT gateway, the Android application would forward the data to the server. However, before sending them the application, they would try to acquire the location information of the device. In order to get a location information update, an Android application needs to register a request to a Location Service in the operating system.

In this experiment, we set the interval to receive the update every 5 seconds and the 1-second interval as the fastest. It means that the IoT gateway application will receive a new location information every 5 seconds or if any other application running in the same device request location updates in less than 5 seconds, then the IoT gateway will also receive that update if the interval is more than 1 second since the last update. Since the aim of this system is location-based monitoring, the Android application was set to send the sensor data to the server every time it gets location information updates from the Location Service. Although the IoT gateway received sensor data every 1 or 2 seconds, it would only send the message to the server every 5 seconds or less when it received location information from the service.

Assuming that the Android is a personal device, a unique ID of the device can be considered to represent the patient and would be sent together with the coordinate of the device location and the sensor data to the server. With a patient ID attached to every sensor value sent to the server, the data can be visualized and analyzed by the location of the patient where the data was coming from. FIGURE 7 shows an example of a message payload sent from an Android device and received by the server.

{"id":1622071247194,"userid":123456,"msg":
1622071247194,"characteristic":"cdeacb81-
5235-4c07-8846-93a37ee6b86d",
"sensorvalue1":71,
"sensorvalue2":96,"sensorvalue3":56,"lat":-
6.2892532, "lon":106.698929, "location":"-
6.2892532,106.698929"}

**FIGURE 7. Example of the message payload**

Message payload, as shown in FIGURE 7, is JSON formatted containing ten key-value pairs of message attributes. The format of the message complies with ThingsBoard MQTT and HTTP API, and since it is a standard JSON format, it is possible to configure the receiver or the transport layer of ELK to be able to parse the data and transform them into a more suitable structure for Elasticsearch API. This way, and also by using the same topic for MQTT publish/subscribe in ELK, will minimize effort in configuration. TABLE 1 shows the description of each attribute.

Timestamp information is created and stored in milliseconds value and is represented in 13 digits of

numbers. A characteristic is a 16-bit UUID or unique identifier of a BLE device given by the manufacturer of the device. This identifier can be used to identify which device was the sensor data coming from. There are three attributes that can be used to represent measurement values. Android application must be configured to map each value given by the sensor to those attributes. In this experiment, we only used $SpO_2$ and heart rate values from the sensor; therefore, only two of three attributes were available that would be processed in the server. ThingsBoard web user interface and Kibana both have the capability to show data on a map. A map widget in the ThingsBoard dashboard will require the latitude and longitude of a location in separate attributes, although Kibana needs both values provided in a single attribute to match the geo_point data type. For this reason, the message payload has three attributes to transport the location coordinate. This way transport layer in ELK does not have to do the transformation for those fields.

TABLE 1
ATTRIBUTES OF MESSAGE

| Attribute | Description |
|---|---|
| id | Message-id or a unique identifier for each message sent from IoT gateway. |
| userid | User-id or a unique identifier stored in Android device to represent a patient. |
| msg | Contain timestamp of the system when the message is going to be sent. |
| characteristic | UUID of a BLE service from a specific device or sensor. |
| sensorvalue1 | Measurement result from sensor |
| sensorvalue2 | Measurement result from sensor |
| sensorvalue3 | Measurement result from sensor |
| lat | Latitude of location coordinate provided by Android device |
| lon | Longitude of location coordinate provided by Android device |
| location | Latitude and longitude as one attribute. |

The testing environment for this experiment consists of a pulse oximeter sensor, an Android device, and a laptop PC running the ThingsBoard platform or ELK stack. Android devise and laptop PC are both connected to the same local network using WIFI and ethernet cable. Consequently, the risk of the intermittent network caused by the quality of the internet service can be ignored. TABLE 2 shows the specification of the hardware used in this experiment.

There are two measurement points used for data collection. First was the message origin or the Android IoT gateway. Android application printing out data required for analysis to Logcat and read from the PC via USB debugging. This data is then exported to a text file for future analysis. The second was the message destination or the server. Telemetry data was retrieved from each platform's data storage. ThingsBoard provides a special API for recalling historical telemetry data from a database. Using this HTTP-based API, we can define the attributes and time range of the data we want to get. Meanwhile, using the Discover feature in Kibana, we can retrieve monitoring data from a specific range of arrival times and export them to a CSV file. From those points of measurement, we compare the first and the second timestamp to calculate the time difference. We also compare the number of packets sent from the gateway and the number of packets received in the server as well as calculate the average message payload during testing. Each scenario of the test was running for about 10 minutes resulting in 100-200 rows of test data. The pulse oximeter and Android device were less than 2 meters in the distance or similar to the distance between a patient and a bedside monitoring device.

TABLE 2
HARDWARE SPECIFICATION

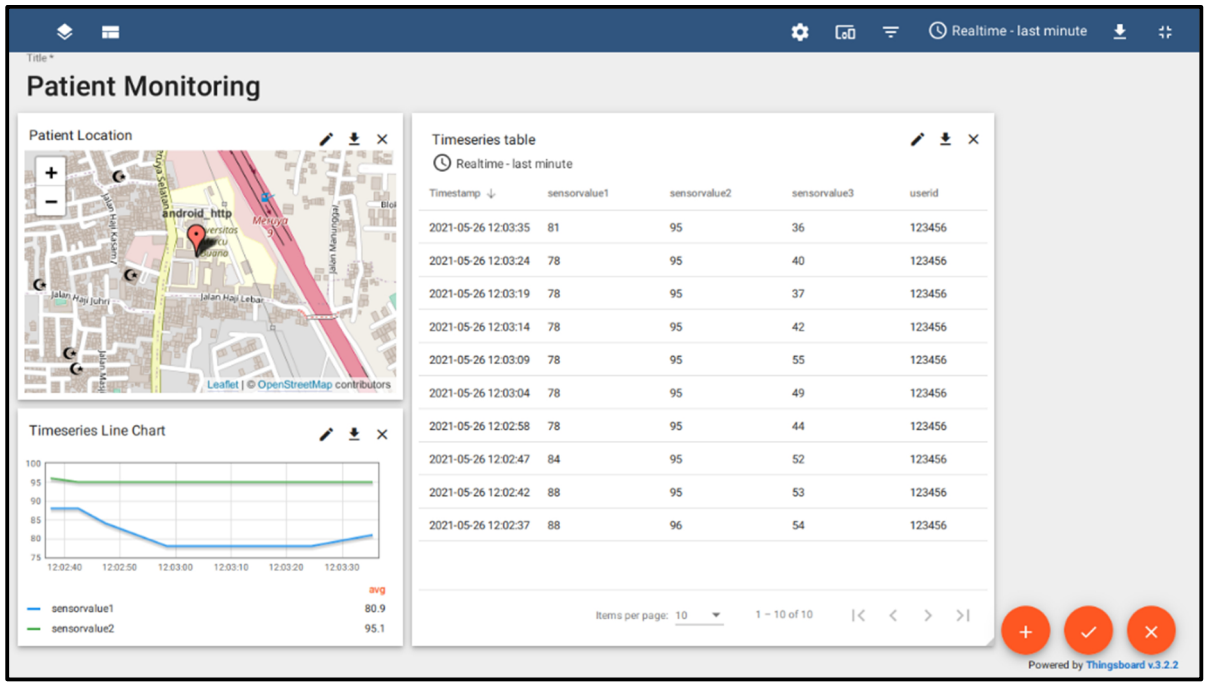
| Hardware | Specification |
|---|---|
| Pulse Oximeter | Model: JPD-500G<br>$SpO_2$ measurement: 70%-99% ±2%<br>Pulse Rate measurement: 35bpm-250bpm ±2bpm |
| IoT Gateway | OS: Android OS version 5.1.1 (Lollipop)<br>API/Service: Network, Bluetooth Low Energy (BLE), Location Service |
| Server | OS: Linux (Ubuntu 18.04)<br>Elasticsearch: 7.7.0<br>Logstash/Filebeat: 7.7.0<br>Kibana: 7.7.0<br>JDK: OpenJDK 14<br>ThingsBoard: 3.2.2<br>Database: PostgreSQL 12.7 |

## III. RESULTS

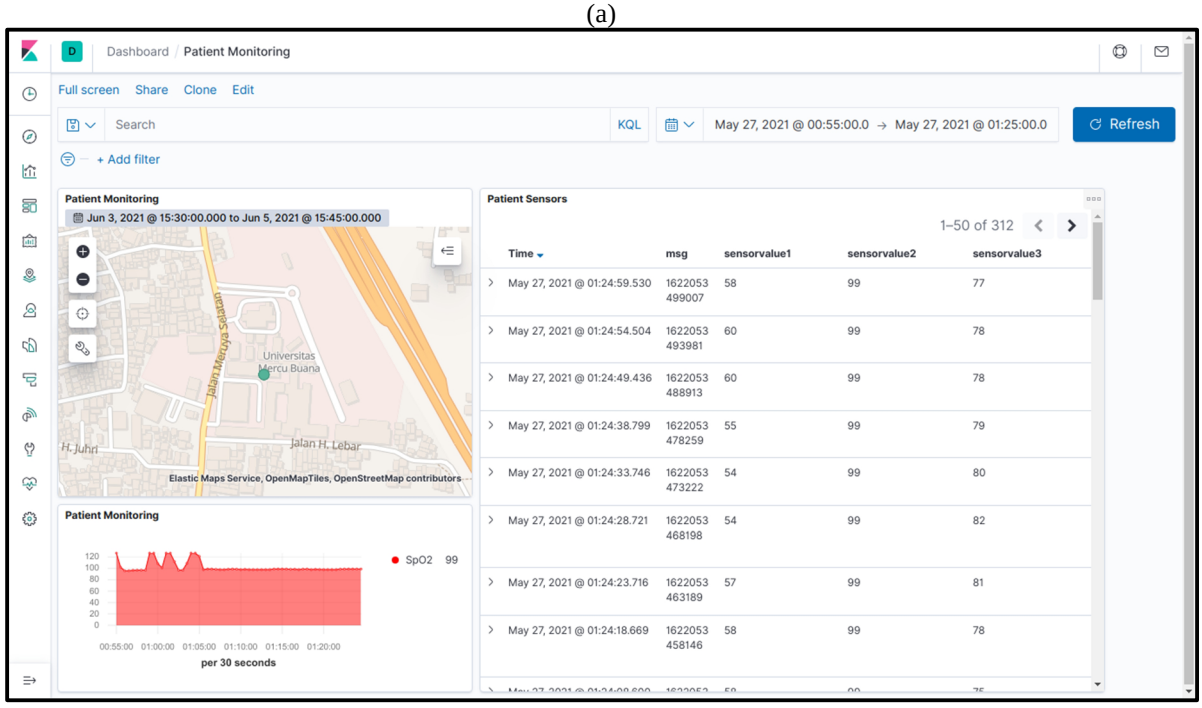
The implementation of the monitoring system of COVID-19 using ThingsBoard and ELK stack had resulted in a dashboard showing the measurement data coming from the $SpO_2$ sensor and IoT gateway. FIGURE 8 shows the sample of the dashboard captured from the experiment. Both in ThingsBoard web UI and Kibana, the data were displayed in three different ways. First, the sensor data were laid out on a map based on the location information attached. For this reason, the medical staff can monitor the location of the patients doing self-isolation. Secondly, the data were displayed as time series to watch the change time after time. Either in ThingsBoard web UI and Kibana, we can easily change the time period of the displayed data so we can check out historical data in a few steps only. While the map and time-series chart displayed data visually, we also configured the dashboard to display the data in a browseable table so that the examiner could check over the period of time easier. After running all test scenarios, we found that the average interval of a message received in the IoT gateway from the sensor was 1.33 seconds. However, since the configuration in IoT gateway to receive location updates every 5 seconds and not sooner than 1 second, we measured the average interval of the message sent from IoT gateway to server was

5626.67 milliseconds. Nevertheless, this average interval can be lower in a real situation depending on how many other application requests for location updates and their requested interval are.

Since we used the same message format for both platforms and protocols, we measured the same payload size between 235-237 Bytes or 236.6 Bytes in average. Meanwhile, the delivery time which were calculated from the interval of the transmitting and receiving time of the packet, showed a big difference between the ThingsBoard and the ELK platforms.




(a)




(b)

**FIGURE 8. Screen capture of the dashboard interface in (a) ThingsBoard web UI and (b) Kibana**

The average time of delivery when using ThingsBoard and MQTT protocol was 177 ms and 221 ms for HTTP protocol. On the other hand, the average time of delivery when using MQTT in the ELK stack was 541.94 ms and 568 ms when using HTTP. From this measurement, we can see that it took more than two times longer in the ELK platform than in ThingsBoard to bring the data from the gateway into the storage. TABLE 3 shows the complete result of the test calculated from 100 samples taken in almost 10 minutes.

TABLE 3
DELIVERY TIME

| | Delivery Time (ms) | | | |
|---|---|---|---|---|
| | Average | Max | Min | Std. Dev |
| **ThingsBoard** | | | | |
| MQTT | 177 | 185 | 170 | 2.82 |
| HTTP | 221 | 280 | 194 | 16.27 |
| **ELK** | | | | |
| MQTT | 541.94 | 883 | 516 | 78.40 |
| HTTP | 568 | 870 | 545 | 37.5 |

From the sample of the measurement data, we can also see the difference between the results from both platforms. FIGURE 9 shows time-series data when using the ThingsBoard platform.

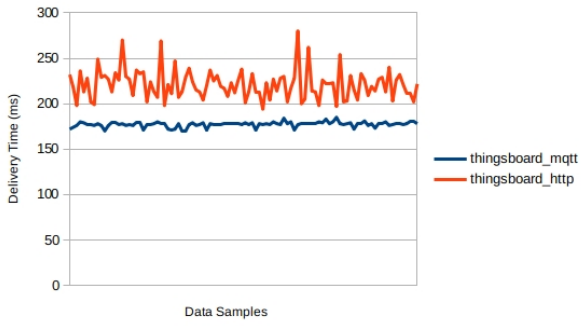



FIGURE 9. Delivery time in ThingsBoard is shown as time series

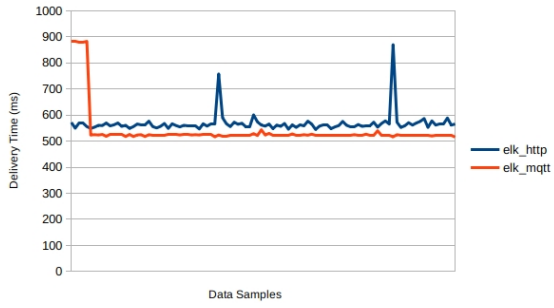



FIGURE 10. Delivery time in ELK is shown as time series

The data samples are placed on the horizontal axis with delivery time in milliseconds as a vertical axis. With a standard deviation of only 2.82 ms, as shown in TABLE 3, delivery time using MQTT looks more stable than HTTP. On the other hand, delivery time in ELK when using MQTT also looks stable despite the fact that the average time was higher than in ThingsBoard. Nevertheless, using HTTP in ELK had an even higher delivery time. It showed that ELK, with its microservice architecture, could manage the load although running on a laptop PC. Time-series data of delivery time in ELK can be seen in FIGURE 10. The data samples are placed on the horizontal axis with delivery time in milliseconds as a vertical axis.

## IV. DISCUSSION

In conducting the performance test of the two $SpO_2$ monitoring systems, namely systems with Thingsboard IoT platforms and ELK in the server, the same WBSN device is used which is a home pulse oximeter and Android device as an IoT gateway. In this design, the pulse oximeter is responsible for gathering data from the sensor and forwarding the data to an Android device where patient identifier and location information will be added before sending them to the server. Wherefore, ELK stack or ThingsBoard will receive the data and keep them in each store. In this experiment, both systems have successfully received and display sensor data and patient location on the dashboard. Sensor data with location information and patient identifier can be displayed not only as a list of measurement values and GPS coordinates like in our previous study using ThingsBoard but also in a map. As a result, medical staff can quickly identify the location of a patient that is being monitored.

While we can configure the time frame of the displayed data, it is also important to provide the monitoring data to the user as soon as possible. Consequently, a shorter time of message delivery will become a major consideration for choosing the more suitable platform to implement the system. In this experiment, we found that the two platforms had a significant difference for the result of the measurement of the delivery time, where the average time of delivery in ThingsBoard was less than half of the average time when using ELK. Compared to our previous research with a similar ELK stack, the experiment with ThingsBoard still has a lower average delivery time. In other words, delivering monitoring data using ThingsBoard in this experiment is faster than ELK stack.

In this experiment, we have deployed ThingsBoard as a standalone application since our scenario only involved traffic far below the minimum estimated number of messages for single-server deployment based on deployment guidelines provided by ThingsBoard on its website. Same with ELK stack, while it is designed for deployment in multiple machines with its microservice architecture, in this experiment, we decided to run the set of applications in a single PC due to the limitation of available infrastructure for this research. However, with adequate RAM capacity, the application could manage the load and keep the performance stable even though the delivery time became slower than its comparator. Accordingly, the performance is expected to be much better in a more suitable hardware configuration.

## V. CONCLUSION

There are many platforms available to use as IoT infrastructure. Some of them are built specifically for IoT like ThingsBoard, and some others are built for many purposes but are equipped with features required in an IoT-based monitoring system. The Elastic Stack, which has transport modules that support MQTT and HTTP, was proven to be able to work as an IoT platform for monitoring COVID-19 patients doing self-isolation at home. Both platforms were able to deliver measurement data from a pulse oximeter and visualize the data to the end-user as a map or time-series chart.

The purpose of this study is to develop a performance evaluation of the IoT-based SpO2 monitoring systems using the Thingsboard and ELK as IoT platforms. In this research, we found that compared to ThingsBoard using the same protocols showed that delivery time when using ELK was higher but stable. The measurement result showed that the average time of message delivery in ELK was more than twice as in ThingsBoard. While ELK was built for higher load and a higher specification of hardware, this could mean that running them on a machine with lower specification had caused the application to run at a lesser speed to keep the reliability. Therefore, since ThingsBoard was also designed for a larger scale, in the future, it would be interesting to observe the performance of each platform with different architecture and configuration, using better hardware and a larger number of devices.

## REFERENCES

[1] C. Sandeepa, C. Moremada, N. Dissanayaka, T. Gamage, and M. Liyanage, "Social Interaction Tracking and Patient Prediction System for Potential COVID-19 Patients," 2020, doi: 10.1109/5GWF49715.2020.9221268.

[2] R. P. Singh, M. Javaid, A. Haleem, and R. Suman, "Internet of things (IoT) applications to fight against COVID-19 pandemic," *Diabetes Metab. Syndr. Clin. Res. Rev.*, vol. 14, no. 4, pp. 521–524, Jul. 2020, doi: 10.1016/j.dsx.2020.04.041.

[3] M. Otoom, N. Otoum, M. A. Alzubaidi, Y. Etoom, and R. Banihani, "An IoT-based framework for early identification and monitoring of COVID-19 cases," *Biomed. Signal Process. Control*, 2020, doi: 10.1016/j.bspc.2020.102149.

[4] R. Jouffroy, D. Jost, and B. Prunet, "Prehospital pulse oximetry: A red flag for early detection of silent hypoxemia in COVID-19 patients," *Critical Care*. 2020, doi: 10.1186/s13054-020-03036-9.

[5] S. Shah *et al.*, "Novel Use of Home Pulse Oximetry Monitoring in COVID-19 Patients Discharged From the Emergency Department Identifies Need for Hospitalization," *Acad. Emerg. Med.*, 2020, doi: 10.1111/acem.14053.

[6] J. Teo, "Early Detection of Silent Hypoxia in Covid-19 Pneumonia Using Smartphone Pulse Oximetry," *Journal of Medical Systems*. 2020, doi: 10.1007/s10916-020-01587-6.

[7] A. Adriansyah and A. G. Amrullah, "Design of Health Monitoring Framework Model using oneM2M Standard," *Int. J. Electr. Energy Power Syst. Eng.*, vol. 4, no. 1, pp. 107–112, Feb. 2021, doi: 10.31258/ijeepse.4.1.107-112.

[8] M. Hassanalieragh *et al.*, "Health Monitoring and Management Using Internet-of-Things (IoT) Sensing with Cloud-Based Processing: Opportunities and Challenges," 2015, doi: 10.1109/SCC.2015.47.

[9] T. M. Kadarina and R. Priambodo, "Monitoring heart rate and SpO2 using Thingsboard IoT platform for mother and child preventive healthcare," 2018, doi: 10.1088/1757-899X/453/1/012028.

[10] R. Priambodo and T. M. Kadarina, "Monitoring Self-isolation Patient of COVID-19 with Internet of Things," 2020, doi: 10.1109/Comnetsat50391.2020.9328953.

[11] M. Bajer, "Building an IoT data hub with elasticsearch, Logstash and Kibana," 2017, doi: 10.1109/FiCloudW.2017.101.

[12] N. Shah, D. Willick, and V. Mago, "A framework for social media data analytics using Elasticsearch and Kibana," *Wirel. Networks*, 2018, doi: 10.1007/s11276-018-01896-2.

[13] A. A. Ismail, H. S. Hamza, and A. M. Kotb, "Performance Evaluation of Open Source IoT Platforms," 2019, doi: 10.1109/GCIoT.2018.8620130.

[14] P. Pierleoni, R. Concetti, A. Belli, and L. Palma, "Amazon, Google and Microsoft Solutions for IoT: Architectures and a Performance Comparison," *IEEE Access*, 2020, doi: 10.1109/ACCESS.2019.2961511.

[15] P. H. Vilela, J. J. P. C. Rodrigues, P. Solic, K. Saleem, and V. Furtado, "Performance evaluation of a Fog-assisted IoT solution for e-Health applications," *Futur. Gener. Comput. Syst.*, 2019, doi: 10.1016/j.future.2019.02.055.