Data Communication using MQTT for Black Soldier Fly Larvae Monitoring System

Sarah Alyaa Tsaabitah*‡, Muhammad Ogin Hasanuddin†‡,
Khairul Hadi Burhan§, Agus Dana Permana§, Wildan Trusaji¶
‡*School of Electrical Engineering and Informatics*
§*School of Life Sciences and Technology*
¶*Faculty of Industrial Technology*
Institut Teknologi Bandung
Bandung, Indonesia
*13218011@std.stei.itb.ac.id, †moginh@itb.ac.id

2022 8th International Conference on Wireless and Telematics (ICWT) | 978-1-6654-5122-2/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICWT55831.2022.9935472

***Abstract*—Black soldier fly larvae (BSFL) is an emerging insect-based waste management agent as it provides effective biowaste conversion. Though effective, it requires certain environmental conditions that needs to be monitored regularly to make sure the BSFL can process the waste effectively. But, BSFL farmers are not always on site and the human resource is limited. Thus, remote monitoring system is needed to ease the monitoring process. By using the Internet of Things (IoT) for environmental condition monitoring, data collection from the sensors needs to be obtained in real-time. To do so, a scalable IoT messaging protocol called Message Queuing Telemetry Tracking (MQTT) for the data communiction is implemented. As the data will continuously be sent, a cloud server using Amazon Web Service Elastic Computing 2 (AWS EC2) virtual instance will be used to build the MQTT Broker and to run the data processing program that will processed the data before it is saved to a MySQL database. Using this system, data from the sensors located on the farm is succesfully sent to the MQTT broker, and the saved data in database is showing the same result as the sent data. It was also found that the average delay for data transmission is 0.8125 seconds. Thus, the proposed system has succesfully perform real time data communication using the MQTT protocol for BSFL monitoring system.**

***Keywords*—Internet of Things, Black Soldier Fly, MQTT, Waste Management, AWS EC2.**

I. Introduction

Smart farming is a concept in which information technology and communication is implemented in agricultural sector as an effort to increase the efficiency of as agricultural process as well as its productivity [1]. With the increase of productivity, comes a possible consequence in the increase of waste produced as a byproduct of the process, which highlight the importance of waste management system.

Black soldier fly (BSF), *Hermetia illucens*, is an insect that has gained popularity among other insect-based biowaste treatments for its effectiveness to convert biowaste using its larvae. Biowaste biomass is consumed by the larvae, which then will be transformed into protein and fat of the larvae as well as its residue. The larvae is known to be having a high-quality protein suitable for feeding chicken and fish, and its residue contains nutrients and organic matter that helps reduce soil depletion. Additionally, its effectiveness in reducing the risk of bacterial transmission through biowaste makes BSF a safe option for biowaste treatment in farm level [2].

Though it is seen as a very effective method, insect-based waste management requires certain environmental condition for the insects to survive. BSF itself is known to be a very resilient insect that can survive in extreme conditions [2]. However, optimal environmental conditions have been found, in which without the proper conditions, the development and metabolism of BSF larvae was seen to be disturbed. These conditions includes the temperature, relative humidity level, moisture level of the substrate, as well as the pH level of the given substrate [3]. Thus, to obtain maximum result of using the BSF larvae for waste treatment, there is a need to monitor the environmental condition of the BSF larvae cultivation area.

The Internet of Things (IoT) has begun to emerge as it has helped human in controlling and monitoring essential conditions using devices which are able to capture, evaluate, and transmit information from the environment to the cloud, where the data will be stored. Environment monitoring system is one of the most important IoT systems which mainly includes data collections through sensors and data reviewing for short-term measure as well as remote management and observations [4].

By using the IoT system to maintain the environmental condition of BSF larvae on its optimum level, data collection from the sensors needs to be obtained in real time to make sure that the data is accurate and representing the exact information of the environmental condition, as well as to be saved automatically to the database for certain period. This is to ensure that the user will have the proper information to interpret the insect's environmental condition. Thus, data communication from the sensor to the database plays an important role in obtaining this need. This paper will focus on how the data transmission and storage are implemented in the IoT system to fulfill the requirements to monitor the environmental condition of the BSF larvae cultivation area.

978-1-6654-5122-2/22/$31.00 ©2022 IEEE

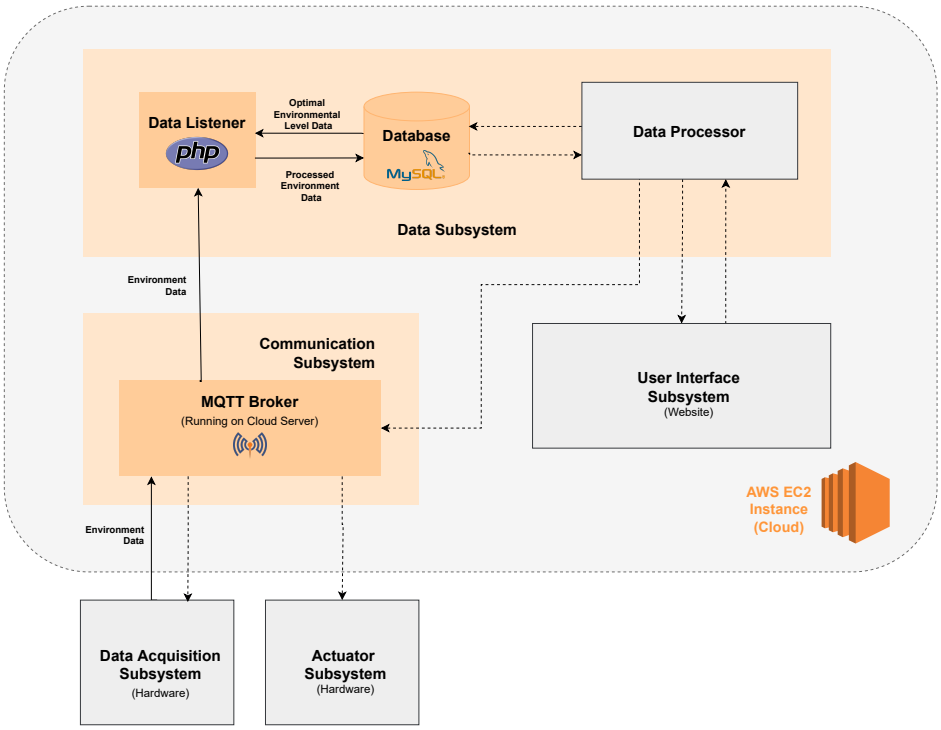

Fig. 1. Architecture of the Proposed BSFL Monitoring System Data Communication (Orange box, solid arrows)

II. Related Work

In IoT implementations, there are several IoT communication protocols that are commonly used for different applications. A study has been done for analyzing the messaging protocols commonly used in IoT system, including MQTT, CoAP, AMQT, and HTTP [5]. Based on the study, MQTT is chosen to fit the implementation of the intended system better as it provides better reliability to ensure successful data transmission, as well as its wider usage which provides ease in the development process and support from various organizations such as IBM, Amazon Web Service, Google Cloud Platform, and Red Hat.

Message Queue Telemetry Transport or MQTT is a lightweight and scalable IoT messaging protocol that implements a publish/subscribe model, which is more suitable for machine to machine communication, compared to the traditional request/response model [6]. It is commonly used for system with constrained networks and small devices which requires efficient bandwidth and battery use such as sensors [7]. It requires a central server called broker that act as a central device that connects clients that are connected through topic. There are 2 types of clients: Publisher, which will send data to a topic, and Subscriber, which will receive any data from certain topic. With this system, multiple devices can send data to a certain device, and multiple devices can receive data from the same device, if they are publishing and/or subscribing to the same topic [8].

Several implementations of MQTT have been done in IoT-based monitoring system. One of which is for temperature monitoring using MQTT-Dashboard which utilizes the HiveMQ MQTT Broker. The implementation resulting in a successful data transmission for its use case by using MQTT protocol for data transmission [9]. Another use case using ESP32 and MQTT protocol for solar power plant monitoring has also successfully been implemented by using MQTT Broker provided by Google Cloud Platform. ESP32 act as the publisher for the data which then will send the data to the broker placed in the cloud. The platform also provides the database system in which the data that has been processed upon subscribing will be stored [10].

III. Proposed System

This paper presents the communication and data subsystem as a part of the IoT-based Black Soldier Fly Monitoring System. Each subsystem consecutively will deal with the transmission of data obtained from the sensors that read the environmental data of the BSF Larvae cultivation area and how the transmitted data are processed until they are saved in the database. For a data to reach the end user, there are several layers that it must go through. The communication subsystem, that will be placed in a cloud server, consists of an MQTT broker that will connect the thing, which are the sensors, to the internet, which is the data subsystem, through MQTT protocol.

Since the MQTT runs on TCP/IP connection, the device will be connected to the internet through a WiFi connection. Before passing the data through the MQTT Broker, a registered username-password pair must be assigned to each client for it to initiate a connection with the broker.

The data that is sent through the MQTT Broker from the device will then be received by the data subsystem, which includes a data listener block that runs in the cloud server. This block will receive all the data from the broker that is sent to a topic where the device sends its data. Data will be received through MQTT protocol by subscribing the topic. The data will then be processed by this block to analyze whether it is inside, above, or below the optimum environmental condition level range based on the environmental condition data level that is inputted by user. The result will then be assigned as the flag for each data with every condition will be represented by 0, 1, and -1, consecutively. This flag will be a trigger for the notification that will be represented in the user interface subsystem. The data that has been processed will then be inputted to a MySQL database that is also located in the same cloud server. The visualization of the proposed system can be seen in Figure 1, with the discussed systems marked with orange boxes and solid line arrows.

IV. Implementation

The implementation of the data communication part for the system designed as in Figure 1 consists of Mosquitto as the MQTT broker, data listener program written in PHP, and database using MySQL. This part is built in a cloud server using Amazon Web Service (AWS) Elastic Computing 2 (EC2) Instance. As for the connection, the publish/subscribe relation in betweenn the system that uses the MQTT protocol are shown in Figure 2.

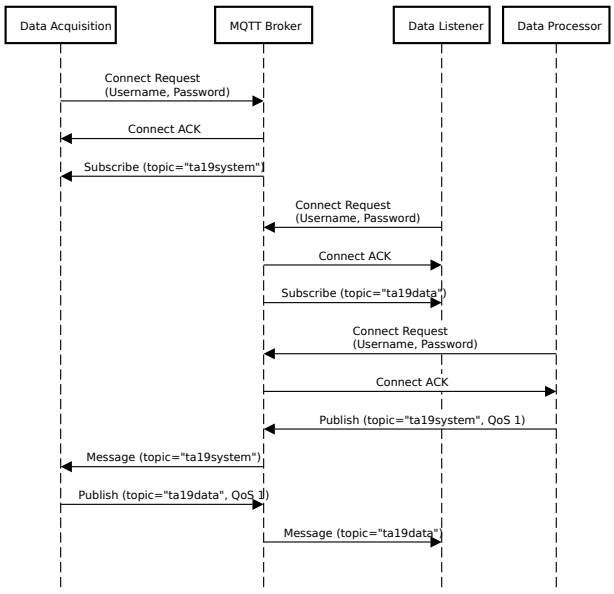

Fig. 2. Sequence Diagram of MQTT Connection

A. Implementation of MQTT Broker on Cloud Server

The cloud server that will be used to build the system is an AWS EC2 instance with Ubuntu 20.04 operating system (OS). The choice of using this is because of its various libraries and tutorials which will ease the development process. In port settings, specifically for MQTT Broker, port 1883 will be used with a route address of 0.0.0.0 which means this port can be accessed by anyone connected to the Public IP Address of the instance used. After the setup of the server is done and the instance has been launched, a .pem file will be generated containing the keys and certificates that will be needed to access the instance through the local computer that is used.

The broker that will be used is Mosquitto, a popular lightweight open source broker written in C. Its installation will be done through shell access using ssh. After the installation is done, username and password need to be generated and saved in the password file inside the Mosquitto configuration file. By adding this, only clients assigned with the registered username and password are eligible to acces the MQTT Broker.

B. Implementation of Data Transmission from Data Acquisition Subsystem (Hardware) to Broker

In this section, the communication between the data acquisition subsystem using a microcontroller and the broker that has been installed on the cloud server will be implemented. Data received from the sensors will be published to the broker by the microcontroller using the MQTT protocol. To successfully send the data, the publisher needs to send data on the same topic as the topic that is subscribed by the client, or in this case is the data listener block.

In this use case, the microcontroller that will be used is an ESP32, which is equipped with WiFi module. For ESP32 to be able to communicate data with the MQTT protocol, it is necessary to use the following libraries:

- WiFi.h
  This library is used to communicate with the WiFi layer on the ESP32 so that the ESP32 can connect to the internet network (TCP/IP).
- PubSubClient.h
  This library is used for ESP32 to communicate using the MQTT protocol as a client (subscriber or publisher). This library will help to recap messages into a standard packet structure of sending data using MQTT.

In ESP32, there is a radio module with a receiver and transmitter on the 2.4 GHz frequency. Therefore, it is necessary to ensure that the WiFi used has the same frequency as the ESP32. A connection between ESP32 and WiFi is required to communicate using the MQTT protocol because this protocol works on top of the TCP/IP protocol, so a network with the internet is required and can be provided using WiFi.

C. Implementation of Data Listener using PHP on Cloud Server

For the data to be saved to the database, there will be another client that will subscribe to the topic where the data acquisition

subsystem is publishing. This client is written in PHP and will run in the background of the cloud server, meaning that this data listener will continuously listening to the topic. There are several steps included in the data listener part.

1) Subscribe to MQTT
   To subscribe to the related topic, the program needs to establish a connection with the MQTT broker. To do so, an open source library called php-mqtt for PHP will be used. The connection settings that need to be configured consists of Username, Password, and Keep alive interval. A client configuration will also be made by assigning the server, port, and client ID. In this case, the server is the IP address where the broker is located, the port will be 1883 as assigned for the broker in the server, and the client ID can be anything, but we need to make sure that this client ID is unique because there will not be any other connection available using the same ID. Using the connection setting and the client setting, a connection will be made using the connect() function from the php-mqtt library. After that, the subscribe() function will be called with the assign topic and function callback whenever new data is received.
2) Data Processing
   The data processing part is the callback function that will be run whenever a new data is received via the subscription method. The process includes 2 functions:
   - Check flag
     This function perform to compare the data received with the optimum condition level data that is already in the database. There are 3 flag that will be assigned to the data:
     – Flag 1: When the data is greater than the maximum optimal condition
     – Flag -1: When the data is less than the minimum optimal condition
     – Flag 0: otherwise
   - Save to database
     This function will process all the data that is received from the broker until it is saved to the database. The data received will be in the form of JSON string and it will be parsed to array so that it can be flagged using the check flag function and can be saved to the MySQL database.
3) Running the PHP on the Server Background
   As mentioned before, this program will run in loop. To do so, this program will be run in the background of the cloud instance using the nohup command in Ubuntu. To stop this process, the program needs to be killed manually from the server.

V. Result and Discussion

This part will be divided into 3 parts, including publishing data from data acquisition to broker, storing received data to the database, and data transmission and processing latency.

A. Publishing data from data acquisition subsystem to MQTT Broker

The data that has been received from sensors will be processed by the microprocessor, in which ESP32 will be used, and then will be sent to the broker through the topic "ta19data". A client is assigned in the cloud server to subscribe to the topic to check whether the data has been sent properly. The data that is sent by the ESP32 is shown in Figure 3 and the received data from the subscriber can be seen in Figure 4. From Figure 3, it shows the process that the ESP32 need to

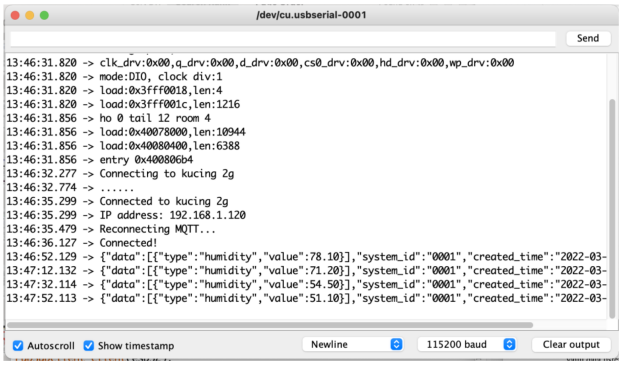

Fig. 3. Data published by ESP32 to MQTT Broker

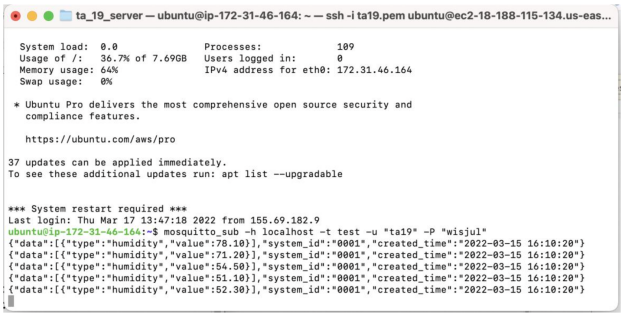

Fig. 4. Data received by subscriber on server from MQTT Broker

do before finally sending the data. Since MQTT is running on TCP/IP, a connection with the WiFi is necessary before it can be connected to the broker to then publish the data to the broker. When the data is sent to the topic, the client that subscribes to that topic in Figure 4 receives the data with the same message as what it is sent, which contains the environmental condition data, system id, and the created time.

B. Storing received data to MySQL Database

On this section, the data that is received from the MQTT broker will be processed by the data listener block. This program will subscribe to the topic "ta19data". The result of the received data can be seen in Figure 5. When data is received by the program through the topic, it will proceed to parsing the data and flagging the data. The process of parsing and flagging would be done individually for each type, which are pH, soil moisture, humidity, and temperature data. When

the flagging process has been done, the data would be saved do the MySQL database and the process of flagging the next data would be continued. The result of the saved data and flagging can be seen in Figure 6.

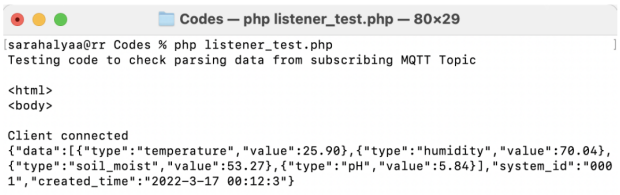

Fig. 5. Data received by data listener block

id ▾ 1	type	value	flag	created_time	system_id
3245	pH	5.84	-1	2022-03-18 08:56:48	0001
3244	soil_moist	53.27	0	2022-03-18 08:56:47	0001
3243	humidity	70.04	0	2022-03-18 08:56:47	0001
3242	temperature	25.9	0	2022-03-18 08:56:46	0001

Fig. 6. Processed data that has been stored to the database

C. Data Transmission and Processing Latency

The latency of the data transmission and processing will be calculated by comparing the time when the data is sent from the ESP32 until the data is saved to the database. This is done by saving the data of the sent time from the ESP32 to the database and using the current timestamp method from MySQL to capture the time when the data is created in the database. The test is done by using 2-minute period between sent data, and the result can be seen in Table I.

TABLE I
AVERAGE DELAY OF DATA COMMUNICATION AND PROCESSING

No	Data Type	Average Delay (sec)
1	Temperature	0.775
2	Humidity	0.85
3	Substrate Moisture	0.775
4	Substrate pH	0.85

From the table above, we can get that the average latency of the data transmission is 0.8125 s. This delay is received through the data processing which includes parsing and flagging that is done before the data is stored in the database. The flagging process requires query to the database to achieve the latest optimal environmental condition inputted by user. But, because the process is not complex, the delay observed is not significant.

VI. Conclusion

The proposed data communication system has been able to send data appropriately using the MQTT protocol. The MQTT broker that is built in the cloud server using Amazon Web Service EC2 virtual instance has been performing as expected, connecting the data that is sent from the topic properly to each subscriber. The data that is sent by the data acquisition subsystem are received by the data listener block, where each data type is assigned with the proper flag based on its position relative to the optimum environmental condition range. The processed data then are successfully stored in MySQL database located in the server, which are accessible by other parts of the system. Through this, it has been proven that the MQTT protocol can be used for data transmission between hardware with accessible WiFi connection and a client located anywhere, with a broker installed in the cloud server using an AWS EC2 instance.

References

[1] M. J. O'Grady and G. M. O'Hare, "Modelling the smart farm," *Information Processing in Agriculture*, vol. 4, no. 3, pp. 179–187, 2017.
[2] B. Dortmans, S. Diener, B. Verstappen, and C. Zurbrügg, *Black Soldier Fly Biowaste Processing - A Step-by-Step Guide*. Eawag - Swiss Federal Institute of Aquatic Science and Technology, 2017.
[3] C.-H. Kim, J. Ryu, J. Lee, K. Ko, J. Lee, K. Y. Park, and H. Chung, "Use of black soldier fly larvae for food waste treatment and energy production in asian countries: A review," *Processes*, vol. 9, p. 161, 2021.
[4] V. Mane, "Environmental monitoring using internet of things," *International Journal of Electrical and Computer Engineering*, vol. 11, p. 149, 01 2022.
[5] N. Naik, "Choice of effective messaging protocols for iot systems: Mqtt, coap, amqp and http," in *2017 IEEE international systems engineering symposium (ISSE)*. IEEE, 2017, pp. 1–7.
[6] M. O. Al Enany, H. M. Harb, and G. Attiya, "A comparative analysis of mqtt and iot application protocols," in *2021 International Conference on Electronic Engineering (ICEEM)*, 2021, pp. 1–6.
[7] R. Vahidnia and F. J. Dian, "Cellular internet of things for practitioners," Jan 2021. [Online]. Available: https://pressbooks.bccampus.ca/cellulariot/
[8] N. Nikolov, "Research of mqtt, coap, http and xmpp iot communication protocols for embedded systems," in *2020 XXIX International Scientific Conference Electronics (ET)*, 2020, pp. 1–4.
[9] A. F. Oklilas, R. Zulfahmi, Ermatita, and A. P. Jaya, "Temperature monitoring system based on protocol message queue telemetry transport (mqtt)," in *2019 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS)*, 2019, pp. 61–66.
[10] J. M. Ramadhan, R. Mardiati, and I. N. Haq, "Iot monitoring system for solar power plant based on mqtt publisher / subscriber protocol," in *2021 7th International Conference on Wireless and Telematics (ICWT)*, 2021, pp. 1–6.