

# ***Implementation of Internet of Things (IoT) System at PT XXX using the MQTT system***

**Katherine Febrianty Sumartono<sup>1</sup>, Yusran Timur Samuel<sup>2</sup>**

<sup>1,2</sup>Universitas Advent Indonesia; Jl. Kolonel Masturi No.288, Cihanjuang Rahayu, Kec. Parongpong, Kabupaten Bandung Barat, Jawa Barat 40559, (022) 2700163

<sup>1,2</sup>Fakultas Teknologi Informasi, Universitas Advent Indonesia  
e-mail: <sup>1</sup>2181007@unai.edu, <sup>2</sup>y.tarihoran@unai.edu

## ***Abstract***

*The Internet of Things (IoT) system has been implemented at PT XXX to improve operational efficiency, especially in monitoring the Ingersoll C1000 centrifugal compressor. This compressor produces high-pressure air that plays an important role in the nickel processing process. This system integrates the Message Queuing Telemetry Transport (MQTT) protocol, IoT sensors, Teltonika RUT951 gateway, and Mosquitto broker to send temperature data in JSON format in real-time. This data is then visualized using AVEVA PI Vision, which provides trend-based information to support analysis and decision making. Previously, monitoring was only carried out locally without historical data, making it difficult to detect potential damage. Operators also had to conduct direct inspections to the field to check the condition of the compressor, which was time-consuming and inefficient. With this IoT-based system, data can be monitored directly from the head office, saving travel time and increasing operational productivity. Trials in a real operational environment showed that data transmission via MQTT was stable even in a limited network. Visualization in AVEVA PI Vision provides a real-time picture that supports early detection of anomalies and data-driven decision making. The results of this study indicate that IoT technology with the MQTT protocol is able to improve monitoring efficiency, optimize system performance, and provide innovative solutions to connectivity challenges in the industry. This implementation also opens up new opportunities for IoT applications in supporting operational efficiency and sustainability.*

**Keywords:** *IoT, MQTT, Mosquitto, Teltonika RUT951, PI Vision*

## **Implementasi Sistem Internet of Things (IoT) pada PT XXX menggunakan sistem MQTT**

### **Abstrak**

Sistem Internet of Things (IoT) telah diimplementasikan di PT XXX untuk meningkatkan efisiensi operasional, khususnya dalam pemantauan kompresor sentrifugal Ingersoll C1000. Kompresor ini memproduksi udara bertekanan tinggi yang berperan penting dalam proses pengolahan nikel. Sistem ini mengintegrasikan protokol Message Queuing Telemetry Transport (MQTT), sensor IoT, gateway Teltonika RUT951, dan broker Mosquitto untuk mengirimkan data suhu dalam format JSON secara real-time. Data ini kemudian divisualisasikan menggunakan AVEVA PI Vision, yang menyediakan informasi berbasis tren untuk mendukung analisis dan pengambilan keputusan. Sebelumnya, pemantauan hanya dilakukan secara lokal tanpa data historis, sehingga sulit mendeteksi potensi kerusakan. Operator juga harus melakukan inspeksi langsung ke lapangan untuk memeriksa kondisi kompresor, yang memakan waktu dan kurang efisien. Dengan sistem berbasis IoT ini, data dapat dipantau langsung dari kantor pusat, menghemat waktu perjalanan dan meningkatkan produktivitas operasional. Uji coba pada lingkungan operasional nyata menunjukkan bahwa transmisi data melalui MQTT stabil meskipun dalam jaringan terbatas. Visualisasi di AVEVA PI Vision memberikan gambaran real-time yang mendukung deteksi dini anomali dan pengambilan

keputusan berbasis data. Hasil penelitian ini menunjukkan bahwa teknologi IoT dengan protokol MQTT mampu meningkatkan efisiensi pemantauan, mengoptimalkan kinerja sistem, dan memberikan solusi inovatif untuk tantangan konektivitas di industri. Implementasi ini juga membuka peluang baru bagi penerapan IoT dalam mendukung efisiensi dan keberlanjutan operasional.

**Kata Kunci:** IoT, MQTT, Mosquitto, Teltonika RUT951, PI Vision

## 1. Introduction

In an increasingly connected digital era, the Internet of Things (IoT) has become an innovative solution to improve operational efficiency in various sectors, including industry. IoT enables small devices such as sensors and electronic devices to exchange information in real-time, creating a smarter and more integrated ecosystem [1], [2].

PT XXX involves several main stages that are structured and efficient. The process begins with the mining of nickel ore from an open pit, where abundant laterite nickel ore is extracted for further processing. After the nickel ore is mined, the next step is to transport the ore to a processing facility for further processing. At the facility, the transported nickel ore will be milled to reduce particle size and then dried to remove excess moisture [3]. The smelting process is carried out to convert nickel ore into nickel matte, which is an important step in converting raw ore into usable products. Nickel matte produced from the smelting process is then further processed to produce a final product that is ready to be exported to various countries, especially Japan. This ready nickel matte product is exported as the main raw material for the manufacture of stainless steel and other nickel-based products.

PT XXX also utilizes sophisticated equipment such as the Ingersoll C1000 centrifugal compressor, which functions to increase gas pressure through a centrifugal process. This equipment plays an important role in industrial applications that require high-efficiency gas compression and is able to reduce energy consumption in the compression process, which is very important in the mining industry. Despite having a structured work process and modern equipment, PT XXX still faces various business challenges. The remote location of operations causes challenges in terms of network connectivity and communication, so the use of technologies such as LTE is important even though it requires efficient costs and management.

PT XXX, a company engaged in nickel processing, faces challenges in maintaining smooth production processes and avoiding downtime due to equipment damage. To overcome this problem, the company implemented an IoT-based monitoring system with the Message Queuing Telemetry Transport (MQTT) protocol, which was chosen because of its ability to transmit data with low bandwidth and its reliability in unstable network environments [4], [5], [6].

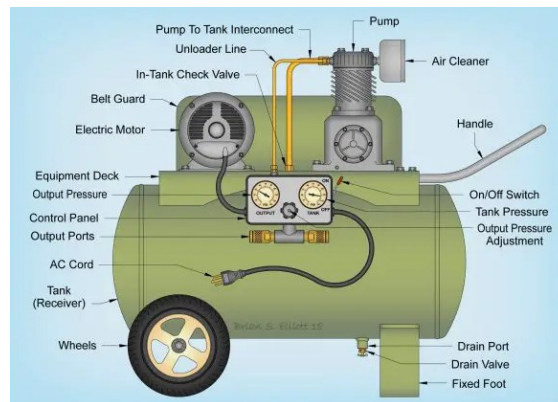
One of the critical equipment in PT XXX is the Ingersoll C1000 centrifugal thermal compressor, which functions to produce high-pressure air in the production process. Real-time monitoring of the compressor's condition is very important because compressor damage can cause major disruptions in the production process. By using the Mosquitto MQTT broker, data from the compressor's sensors is transmitted to the central system via the IoT Gateway, enabling efficient data collection and analysis [7]. This data is then stored and visualized via AVEVA PI Vision, which helps operators monitor compressor performance and make data-driven decisions [8].

Prior to the implementation of the Internet of Things (IoT), sensor data from the SAC/Centac controller on the centrifugal compressor was only obtained and controlled locally by the operator at PT XXX. This resulted in limitations in failure analysis, as there was no adequate sensor data recording for long-term evaluation. Therefore, the main objective of this study is data transmission to store historical data read from the Ingersoll C1000 centrifugal compressor. Through historical data storage, components that will experience wear or damage can be proactively predicted based on operational data analysis, so that repairs can be made before total damage occurs to critical components such as rotors, impellers, and bearings. With this approach, PT XXX can identify potential damage and make repairs before serious damage occurs, which can ultimately reduce downtime and increase production efficiency [9].

Previous research aims to develop a temperature and humidity monitoring system in a smart environment using the MQTT (Message Queuing Telemetry Transport) protocol. The method used includes setting up an ESP8266 NodeMCU as a publisher that collects data from the DHT11 sensor, and a Raspberry Pi 3 Model B as a subscriber that receives data through an MQTT broker. The received data is then visualized using Node-Red and ThingSpeak for real-time display. The results of the study showed that this system successfully monitored temperature and humidity accurately, with minimal reading differences (no more than 0.5 degrees Celsius) compared to other measuring instruments [5]. DHT11 has a data sending limit of every 2 seconds, and there is no data history storage setting. So this research is directly implemented on real industrial devices [10]. This study aims to evaluate the effectiveness of implementing an IoT-based monitoring system with the MQTT protocol at PT XXX, as well as providing valuable insights into the benefits and challenges of implementing IoT technology in industrial environments.

In this study, the main focus is the implementation of the MQTT protocol to support data transmission from IoT devices in a mining environment that has network limitations. This study also explores the integration of data from the MQTT broker into the AVEVA PI data management system to facilitate real-time data analysis and visualization. By adopting modern technologies such as MQTT, PI System, and Node.js, this study shows the potential for increasing operational efficiency and data-driven decision-making at PT XXX.

## 2. Research Methods

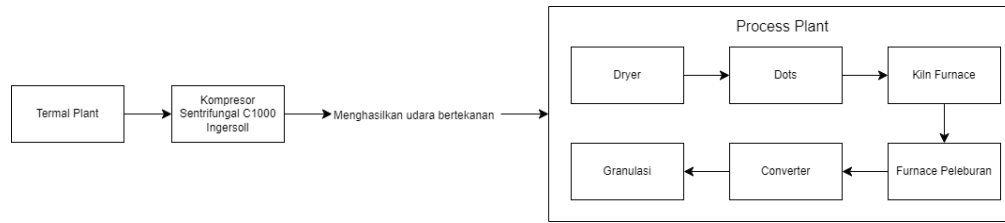


**Figure 1** Overview of Centrifugal Compressor Components

Figure 1 shows a picture of the components of a centrifugal compressor. The Ingersoll C1000 centrifugal compressor plays a role in producing high-pressure air used in various process stages, such as drying, kiln furnace, and converter. In the converter, high-pressure air is used in the oxidation process (blowing) which is an important part of nickel refining.

Sensor data from the compressor is sent in real-time in JSON format, including analog inputs such as temperature. Temperature monitoring is a critical aspect because overheating can degrade equipment performance or even cause damage. With the implementation of IoT, real-time data-driven monitoring offers a solution to improve the accuracy of decision-making and damage prediction.

To achieve this goal, the system uses an IoT Gateway that connects devices in the field to the central network using an LTE sim [11]. The collected data is forwarded via the MQTT protocol to the Mosquitto broker, which is then stored in the PI Database for further analysis. With this approach, PT XXX can perform more efficient maintenance and reduce the risk of downtime.



**Figure 2** Block Diagram Of Nickel Processing System

Figure 2 is a System Block Diagram, where from the Thermal Plant to the Ingersoll C1000 Centrifugal Compressor section then produces pressurized air which will be used in the Process Plant, where the Nickel processing process takes place: Dryer - Dots - Kiln Furnace - Smelting Furnace - Converter - Granulation [3].

IoT Integration for Monitoring and Failure Prediction The use of standard Ethernet cables at PT XXX allows real-time data delivery to the PI system via IoT Gateway and MQTT protocol. This supports data visualization in PI Vision and helps analysis and prediction for condition-based maintenance.



**Figure 3** Data Visualization In PI Vision

Figure 3 shows that data can be visualized in various types of visualizations. The graph consists of several types of data visualizations, including a Line Chart at the top that shows the trend of data changes over time, a Gauge Chart that is two circle graphs with a needle to display values in a certain color range such as safe or risky zones, a Bar Chart that displays data in the form of vertical or horizontal bars, and a table at the bottom with columns such as "Name", "Description", "Value", and "Status" for tabular data comparison [12]. Additionally, there is a Value Display that displays specific numbers for certain metric values.

### Functional Requirements

This study aims to display data received from the Ingersoll C1000 centrifugal thermal compressor through the IoT Gateway. The IoT Gateway used is the RUT951 router that sends data in real time to the PI system using the MQTT protocol. A JavaScript-based program runs as a service on the server, responsible for processing the data and sending it to the AVEVA PI database with relevant topics [13]. With this, system workflows can be automated to ensure seamless integration between field devices and data management systems.

### Non-functional Requirements

The system must be able to work efficiently with consistent data transmission frequency to PI Database via REST API. The operational success of the system depends on stable connectivity to PT XXX's internal network. The resulting graphs and visualizations must allow for further analysis and support failure

prediction using trend analysis. The hardware used includes IoT Gateway industrial router RUT951, Ingersoll C1000 centrifugal compressor, and microcontroller [7].

### Flowchart Of Data Flow In The System

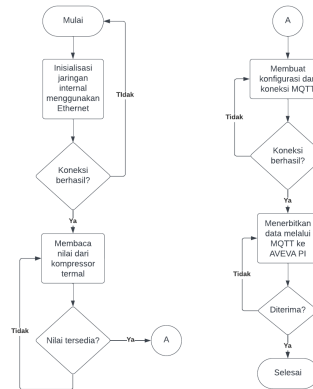


Figure 4 Flowchart Of Data Flow In The System

Figure 4 explains the flowchart of data flow in the system. The system starts by connecting the field device to the PT XXX internal network. Once connected, the IoT Gateway receives data from the compressor, which is then forwarded to the MQTT broker. The data published by the broker is forwarded to PI AVEVA to be visualized using PI Vision [14].

### IoT and MQTT System Topology

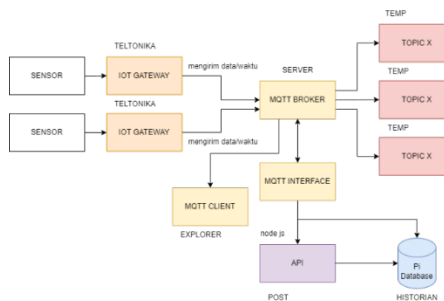


Figure 5 IoT System Topology

Figure 5 shows the topology of the implemented IoT system. The data flow starts from the sensor that reads the temperature parameters from the compressor. This data is sent to the Teltonika IoT Gateway, which acts as a bridge between the sensor devices and the central network.

#### Sensor

The sensor connected to the centrifugal thermal compressor is tasked with collecting operational data, the data taken for testing is temperature data. This data is forwarded to the IoT Gateway with TCP for further processing.

#### IoT Gateway Teltonika RUT951

Acts as a bridge between sensors and the network, collecting data from the thermal compressor and forwarding it to the MQTT Broker [13], [14]. Teltonika RUT951 plays a vital role in ensuring reliable connectivity between field devices and central systems.

#### MQTT Broker Mosquitto

Manage data distribution from sensors to clients.

#### MQTT Interface

The MQTT interface allows interaction between the broker and other systems. The interface application requests data from topics containing certain tags using the REST API, and the data is sent to the AVEVA PI.

### **Node.js**

Node.js is used to process the thermal compressor data received via MQTT Broker and forward it to the API. Node.js is suitable for server-side applications because it has high efficiency in data processing [15].

### **Server**

Server to run service call programs that run automatically.

### **API and PI Database**

The API acts as a link between the Node.js application and the PI Database to store the received data. The PI Database is used to store historical data, including data from the compressor, which can then be monitored via PI Vision.

### **Data Flow In The System**

Data from the thermal compressor is transmitted to the IoT Gateway in real time. The IoT Gateway then forwards the data to the MQTT Broker, which distributes messages to the MQTT interface and MQTT clients. Node.js processes the data received from the broker and forwards it to the API which then archives the data to the PI Database. With this approach, the system workflow can be automated to ensure seamless integration between field devices and data management systems [16].

### **MQTT Client**

MQTT Client acts as a publisher or subscriber, subscribing to a specific topic to receive messages from the broker, the MQTT client initiates a connection with the broker by sending a connection packet (connect packet), and because MQTT works on top of TCP/IP, the broker sends a connection acknowledgment (connack) [17]. MQTT Client, such as MQTT Explorer or Node.js, acts as a subscriber that receives data from the broker [18]. After receiving the acknowledgment, the client initiates data transmission to the broker and can publish messages based on the device's needs [19]. Clients can also unsubscribe and disconnect from the broker.

### **MQTT Broker**

Brokers are responsible for receiving messages from publishers, filtering them, and distributing them to subscribing clients [15], [19]. Brokers must be able to handle multiple devices and distribute data efficiently to multiple clients who need the information.

## **3. Result and Discussion**

### **Implementation Of The MQTT Protocol**

Figure 6 shows a diagram illustrating the communication path from the CENTAC control panel in the field to the central hub (distribution switch) in the thermal control room. This allows the sending of control signals and data to monitor and manage the thermal system.

The signal originates from the CENTAC control panel, which is responsible for controlling the thermal system in the field. After that, the signal passes through a Modbus RTU to a TCP converter that functions to convert the communication protocol from Modbus RTU, which is used in the field, to TCP/IP for network compatibility. The signal then travels through an Ethernet cable (CAT-5) that connects the converter to a network switch as a central hub that directs traffic to various destinations. For long-distance communication, the signal is forwarded through a fiber optic (FO) cable that connects the network switch in the field to the network switch in the thermal control room.



via REST API. The PI Database then stores the data historically for further analysis and real-time monitoring of the compressor's condition.

Data stored in the PI Database is visualized using AVEVA PI Vision, allowing operators to analyze data trends, detect anomalies, and make data-driven decisions. The system runs automatically as a Windows Service, ensuring continuity of data processing without manual intervention.

### **Teltonika RUT951 Device Configuration**

The first step in the implementation is to configure the Teltonika RUT951 device. After installing the SIM card and antenna, the user can access the firmware settings page via the router's default IP. The user must change the default username and password for security, and ensure that the SIM card is registered and connected properly. The "Data to Server" setting needs to be done so that the device can receive data from the sensor that is sent, followed by the configuration of the data to be captured and the MQTT Broker setting.

### **Mosquitto Broker Configuration**

After installing Mosquitto broker, the user must run it through Command Prompt as administrator. To set the password, you can use `-c passwordfile user`, and then the password filling and password refill will appear.

```
listener 1883 ipAddress      # Listener di port 1883 untuk IP tertentu
protocol mqtt                # Menggunakan protokol MQTT
allow_anonymous false       # Menonaktifkan akses anonim
password_file lokasiFilePassword # Lokasi file password untuk autentikasi
```

**Figure 8** mosquitto.conf Configuration

Figure 8 shows the mosquitto.conf configuration. The mosquitto.conf configuration file is set to specify the server IP, the port used, and security settings by setting allow anonymous to false.

### **Node.js Implementation For Data Delivery**

Node.js based program is designed to send data from Teltonika RUT951 using MQTT protocol through Mosquitto broker. Data is then sent to PI Database via REST API. This program sets up MQTT connection and manages periodic data delivery. By using functions to get the current time and send data, the system can monitor and send the latest values from sensors automatically.

The Node.js program used in the implementation of this IoT system has several important components. First, the program loads the necessary libraries such as MQTT and axios to enable communication with the MQTT broker and PI server. The main variables such as tagName, tagValue, and timeValue are declared to store data information that will be sent to the PI Database. The `getNow` function is used to get the current time in ISO format according to the needs of the PI server so that every data sent has an accurate timestamp.

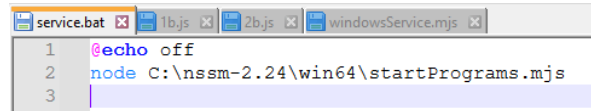
The `sendDataPI` function is responsible for sending data to the PI server using the HTTP POST method in JSON format. This data includes important information such as tag name, data value, and delivery time. To set the periodic delivery of data, a time interval function is used that calculates the interval until the next minute, ensuring that data is sent consistently at the beginning of each minute.

The program also creates a connection with the MQTT broker using the MQTT library, where the program subscribes to a specific topic to receive data from the IoT Gateway. The data received from the broker is then processed and sent to the PI server. This process is executed automatically using the `startSendingData` function, which starts the data-sending interval every one minute.

In addition, the program supports running multiple processes in parallel by utilizing the `spawn` function. This allows separate program files to be run simultaneously to support multiple MQTT topics at once, with



output logs or error messages displayed for debugging. To ensure the program runs automatically every time the Windows system starts, the Non-Sucking Service Manager (NSSM) is used which configures the program as a Windows service. The service.bat batch file is used to easily manage this service through the Windows Service Manager, ensuring operational continuity without manual intervention.



```
1 @echo off
2 node C:\nssm-2.24\win64\startPrograms.mjs
3
```

**Figure 9** service.bat Program

Figure 9 shows the service.bat program. To run this program as an automatic service on Windows, use the Non-Sucking Service Manager (NSSM) to configure the Node.js application as a Windows service. After going to the nssm location directory `cd C:\nssm.2.24\win64>nssm install windowsService` then the message will appear; Service "windowsService" installed successfully!. Next, `nssm start windowsService` then the message will appear, windowsService: START: The operation completed successfully. Then a new tab will appear for the Path, Startup directory, and Arguments settings. After success, you can open the Windows service to see if the service is running properly.

### **Message MQTT**

Here is an example of an MQTT message used

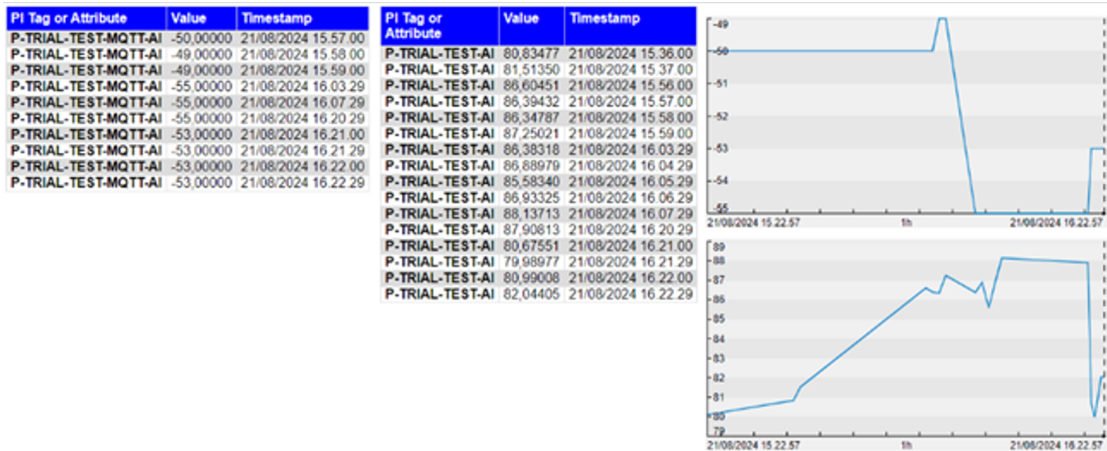
```
{
  "tagName": "P-TRIAL-TEST-AI",
  "tagValue": 90,
  "timeValue": "2024-05-15 18:59"
}
```

This text shows that there are unused name variables, values , and time information [20].

### **Visualization Of Result In Pi Vision**

AVEVA PI Vision is a data visualization application that allows users to monitor and analyze operational data in real time. With PI Vision, operational teams can create dashboards that display data from various sources, including IoT sensors, in the form of graphs, tables, and trends. The application is designed to provide the necessary information quickly and intuitively, enabling better decision-making. Key features of PI Vision include the ability to provide notifications or alarms when data indicates abnormal conditions, as well as the ability to perform trend analysis of historical data stored in the PI Data Archive.

Data successfully sent to the PI Database can be visualized via AVEVA PI Vision, allowing for further monitoring and analysis [21]. This visualization provides a clear picture of compressor performance and supports data-driven decision-making.



**Figure 10** Display Result In AVEVA PI Vision

Figure 10 shows the results of data visualization in AVEVA PI Vision, where data can be presented in the form of a line graph (right in Figure 10) and data history is displayed (left in Figure 10). The history or history of the data received can make it easier to do further analysis in the future while the data presented in the form of a graph makes it easier to see anomalies in data movement.

### Result Testing Criteria

The research results testing criteria cover several important aspects to ensure the effectiveness of system implementation. First, testing is carried out to verify the reliability of data transmission from IoT devices to MQTT brokers, including testing under unstable network conditions. Second, integration testing between the MQTT broker and PI System is carried out to ensure that the data received matches what sent and can be accessed correctly in PI Vision. Finally, testing also includes an analysis of the overall system performance, including data delivery speed. The system trial showed a significant increase in efficiency. Operators no longer need to go to the field to check the condition of the compressor, because data can now be accessed in real-time from the office. This not only reduces the time required for checking but also minimizes operational travel distances, allowing the team to focus more on decision-making and proactive maintenance. By meeting these criteria, it is expected that the system can operate optimally and provide significant benefits to PT XXX.

## 4. Conclusion

The implementation of MQTT-based IoT architecture at PT XXX has successfully enabled real-time transmission, monitoring, and analysis of thermal compressor data. By using IoT Gateway (Teltonika RUT951) and MQTT broker (Mosquitto), the system ensures data is delivered accurately and visualized through PI Vision. Operators can now monitor compressor conditions from the head office without the need to conduct direct inspections in the field. This saves time, reduces travel distance, and speeds up response to potential problems, thereby increasing human resource efficiency in the monitoring and maintenance process.

The system also supports predictive maintenance by detecting abnormal flow patterns and preventing potential mechanical failures. The real-time data collected helps identify problems before serious failures occur, accelerates data-driven decision-making, and provides greater flexibility in operational management. With effective monitoring and appropriate preventive measures, PT XXX has successfully improved overall efficiency and productivity.

## 5. Daftar Pustaka

- [1] "Performance Monitoring of MQTT-based Messaging Server and System," *J. Logist. Inform. Serv. Sci.*, Jan. 2022, doi: 10.33168/LISS.2022.0107.
- [2] B. Mishra, B. Mishra, and A. Kertesz, "Stress-Testing MQTT Brokers: A Comparative Analysis of Performance Measurements," *Energies*, vol. 14, no. 18, p. 5817, Sep. 2021, doi: 10.3390/en14185817.
- [3] F. Bahfie, A. Manaf, W. Astuti, F. Nurjaman, and U. Herlina, "TINJAUAN TEKNOLOGI PROSES EKSTRAKSI BIJIH NIKEL LATERIT," *J. Teknol. Miner. Dan Batubara*, vol. 17, no. 3, Art. no. 3, Sep. 2021, doi: 10.30556/jtmb.Vol17.No3.2021.1156.
- [4] E. Bertrand-Martinez, P. Dias Feio, V. D. Brito Nascimento, F. Kon, and A. Abelém, "Classification and evaluation of IoT brokers: A methodology," *Int. J. Netw. Manag.*, vol. 31, no. 3, p. e2115, May 2021, doi: 10.1002/nem.2115.
- [5] "IoT Monitoring System Based on MQTT Publisher/Subscriber Protocol," *Iraqi J. Comput. Commun. Control Syst. Eng.*, pp. 75–83, Jul. 2020, doi: 10.33103/uot.ijccce.20.3.7.
- [6] B. Mishra and A. Kertesz, "The Use of MQTT in M2M and IoT Systems: A Survey," *IEEE Access*, vol. 8, pp. 201071–201086, 2020, doi: 10.1109/ACCESS.2020.3035849.
- [7] M. Saiqul Umam, S. Adi Wibowo, and Y. Agus Pranoto, "IMPLEMENTASI PROTOKOL MQTT PADA APLIKASI SMART GARDEN BERBASIS IOT (INTERNET OF THINGS)," *JATI J. Mhs. Tek. Inform.*, vol. 7, no. 1, pp. 899–906, Jun. 2023, doi: 10.36040/jati.v7i1.6131.
- [8] M. Bender, E. Kirdan, M.-O. Pahl, and G. Carle, "Open-Source MQTT Evaluation," in *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*, Las Vegas, NV, USA: IEEE, Jan. 2021, pp. 1–4. doi: 10.1109/CCNC49032.2021.9369499.
- [9] B. B. Darujati, D. K. Silalahi, and A. Z. Fuadi, "Perancangan Perangkat Deteksi Anomali Pada Kipas Saluran".
- [10] H. Hindy *et al.*, "A Taxonomy of Network Threats and the Effect of Current Datasets on Intrusion Detection Systems," *IEEE Access*, vol. 8, pp. 104650–104675, 2020, doi: 10.1109/ACCESS.2020.3000179.
- [11] F. Susanto, N. K. Prasiani, and P. Darmawan, "IMPLEMENTASI INTERNET OF THINGS DALAM KEHIDUPAN SEHARI-HARI," *J. Imagine*, vol. 2, no. 1, pp. 35–40, Apr. 2022, doi: 10.35886/imagine.v2i1.329.
- [12] V. Borisov, T. Leemann, K. Seßler, J. Haug, M. Pawelczyk, and G. Kasneci, "Deep Neural Networks and Tabular Data: A Survey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 6, pp. 7499–7519, Jun. 2024, doi: 10.1109/TNNLS.2022.3229161.
- [13] I. Vaccari, G. Chiola, M. Aiello, M. Mongelli, and E. Cambiaso, "MQTTset, a New Dataset for Machine Learning Techniques on MQTT," *Sensors*, vol. 20, no. 22, p. 6578, Nov. 2020, doi: 10.3390/s20226578.
- [14] R. F. Pratama, R. S. R. Wicaksono, and A. N. Pramudhita, "PERANCANGAN DAN IMPLEMENTASI PROTOKOL MQTT PADA SISTEM PARKIR CERDAS BERBASIS IOT," *J. Inform. Dan Tek. Elektro Terap.*, vol. 11, no. 3, Aug. 2023, doi: 10.23960/jitet.v11i3.3191.
- [15] A. Turnip, F. R. Pebriansyah, T. Simarmata, P. Sihombing, and E. Joelianto, "Design of smart farming communication and web interface using MQTT and Node.js," *Open Agric.*, vol. 8, no. 1, p. 20220159, Oct. 2023, doi: 10.1515/opag-2022-0159.
- [16] S. Pawar, N. Panigrahi, J. A.P, M. Lokhande, D. Godse, and J. D B, "Evaluation of Delay Parameter of MQTT Protocol," *Int. J. Eng. Trends Technol.*, vol. 71, no. 3, pp. 227–235, Mar. 2022, doi: 10.14445/22315381/IJETT-V71I3P223.

- [17]E. Di Paolo, E. Bassetti, and A. Spognardi, "Security assessment of common open source MQTT brokers and clients," Sep. 07, 2023, *arXiv*: arXiv:2309.03547. doi: 10.48550/arXiv.2309.03547.
- [18]U. Achlison, Khoirur Rozikin, and Fujjama Diapoldo, "Analisis Implementasi Temperature Screening Contactless berbasis Internet Of Things (IOT) Menggunakan Protokol Message Queue Telemetry Transport (MQTT)," *Pixel J. Ilm. Komput. Graf.*, vol. 14, no. 2, pp. 315–322, Dec. 2021, doi: 10.51903/pixel.v14i2.617.
- [19]M. A. Khan *et al.*, "A Deep Learning-Based Intrusion Detection System for MQTT Enabled IoT," *Sensors*, vol. 21, no. 21, p. 7016, Oct. 2021, doi: 10.3390/s21217016.
- [20]E. B. Sanjuan, I. A. Cardiel, J. A. Cerrada, and C. Cerrada, "Message Queuing Telemetry Transport (MQTT) Security: A Cryptographic Smart Card Approach," *IEEE Access*, vol. 8, pp. 115051–115062, 2020, doi: 10.1109/ACCESS.2020.3003998.
- [21]"IOT BASED HEALTH MONITORING SYSTEM," *J. Crit. Rev.*, vol. 7, no. 04, Feb. 2020, doi: 10.31838/jcr.07.04.137.