Design of Turbine Supervisory Instrument Prototype Based on Internet of Things

Nivika Tiffany Somantri [1,*,a], Yuda Bakti Zainal [2,b], Ghani Abdurahman [3,c], Ade Sena Permana [4,d], Atik Charisma [5,e], M.Reza Hidayat [6,f], Sofyan Basuki[7,g]

[1-7] *Universitas Jenderal Achmad Yani, Jl. Terusan Jenderal Sudirman,Cimahi, Jawa Barat, Indonesia*

[*,a] nivika.tiffany@lecture.unjani.ac.id, [b]zainalyuda20@gmail.com , [c]ghaniabdurahman@gmail.com, [d]adesenapermana@lecture.unjani.ac.id

***Abstract*— Internet of Things-based Turbine Monitoring System is composed of main components including sensors, controllers, alarms and interface applications for monitoring parameters. The sensors used are LM393 IR speed sensor, SW420 Vibration Sensor and DHT11 Temperature Sensor. While the controller used is Wemos D1. Output Alarm uses the Buzzer and the Thingspeak platform as a medium for monitoring the measurement results. The test results that the author can convey are that the Internet Of Things-based Turbine Supervisory Instrument Simulator has been functioning properly. For testing the speed and temperature sensors, they have been compared with standard measuring instruments. Through system testing in the time range of 16.05 to 16.11, we found speed measurement data from 0 to 4425 rpm, vibration measurement data from 0 to 734,401 microseconds or 0.7 seconds and temperatures from 28.6 degrees Celsius to 33 degrees Celsius. The alarm is activated according to the setpoint specified in the program, namely over speed alarm at 2000 rpm, over vibration alarm at 100,000 microseconds and over heat alarm at 38 degrees Celsius. As for the shortcomings that exist in order to be corrected in the development of the system in the future.**

Keywords*— *Instrument, Speed, Temperature, Turbine, Vibration.

I. Introduction

Electricity reliability is crucial in the market, so power generation systems require attention to both technological modernization and efficiency. The turbine is a key component of any power generation system. Turbine monitoring is a priority given the importance of measured parameters such as turbine vibration, turbine speed, and turbine area temperature. The Turbine Supervisory Instrument (TDI) monitors these parameters so that abnormal conditions can be addressed promptly and damage can be prevented or minimized.

In research [1] on instrument monitoring in wind turbines, this study utilized ultrasonic sensors, temperature and humidity sensors, pressure sensors, and other sensors. The controller used a National Instruments SCADA system, and the monitoring system utilized LabView, which features a GUI (Graphical User Interface). This study was quite compact because it used a SCADA system already commercially available in the industry. However, this study did not utilize the Internet of Things, so monitoring was conducted only in a local field control room.

Based on the explanations above, design and analysis can be carried out to find solutions to overcome existing obstacles. One of these is the design of an Internet-of-Things-based Turbine Supervisory Instrument Simulator. The prototype design of an Internet-of-Things-based turbine monitoring instrument aims to monitor turbine operational parameters in real time. The developed system consists of sensors to measure turbine speed, turbine vibration, and turbine area temperature, as well as an IoT platform that displays the measurement results in the form of a monitoring dashboard. Data obtained from the sensors will be processed by a microcontroller and sent over the internet, making it accessible to users via computers or smartphones.

Prototype development was carried out through several stages: hardware design, software design, IoT communication system integration, and instrument performance testing. Testing was conducted to evaluate the system's ability to read turbine parameters, transmit data to a server, and display information in real time. This research is expected to produce a simple, economical, and easy-to-implement turbine monitoring instrument. The research results not only provide a solution for real-time turbine condition monitoring but also serve as the first step in implementing the concept of smart monitoring and predictive maintenance in energy generation systems based on Internet of Things technology. Thus, this research contributes to supporting the development of a more effective, efficient, and integrated modern monitoring system.

II. Research Methodology

The research method used in this study is a system that provides turbine parameter data using speed sensors, vibration sensors, and temperature sensors. Data processing is then performed on a Wemos microcontroller, and the measured data can be received on the user's device via the Thingspeak application. The measured parameters are received on the user's device for remote monitoring. Initially, the sensor detects physical changes, in this case, changes in speed, vibration, and temperature.

The miniature turbine is mounted on the same axis as the speed measurement object. This measurement object is detected through the LM393 IR sensor gap. When something enters the "opto-interruptor" gap, the DO output will produce a logic HIGH or 5V signal. Under default conditions, the DO output will produce a logic LOW signal (the green DO LED is on). The LM393 IC functions as a Schmitt trigger to stabilize the output. This HIGH condition is programmed to activate the Interrupt command in the microcontroller.

The SW420 Vibration Sensor is a switch that operates by opening and closing electrical contacts. By default, the vibration sensor is in the closed state. When no vibration is sensed, it remains closed, or in the conducting state. As soon as a vibration is detected by the sensor, the contacts open and the resistance increases. This generates a pulse, triggering the circuit. This is passed to the LM393 voltage comparator IC, which digitizes the signal and then makes it available on the module's digital output pin. A HIGH state on the DO is programmed to activate the command to calculate the vibration time.

The DHT11 sensor consists of a capacitive humidity sensor element and a thermistor for temperature detection. The humidity-sensing capacitor has two electrodes with a moisture-retaining substrate acting as a dielectric between them. The capacitance value changes with changes in humidity levels. The IC measures and processes this resistance value, converting it to digital form.

To send data from the microcontroller, use the void sendSensorDataToThingspeak command by including the Field Chart addresses for Speed, Vibration, and Temperature. Thus, in addition to parameters that can be viewed from the serial monitor on the Arduino, they can also be viewed on the Thingspeak Web. However, to make the interface seem more attractive, a special application was created for this research so that data from the Thingspeak Web is sent to the monitoring application so that monitoring will be easier by simply opening the application on the gadget available in the user's gadget menu display.

A. System Block Diagram

Sometimes, research methodology is demonstrated in form of diagrams or tables. Author(s) should ensure that those diagrams and tables are clearly readable if placed in one column, however, if the diagrams or tables require more spaces, the diagrams or tables can be placed in two columns, but it should still confirmed in balanced format.

Figure 1. shows a block diagram of an Internet of Things-based turbine supervisory instrument system. A system consists of input, processor, and output. In this system, the input components are the LM393 sensor that detects speed, the SW420 sensor that detects vibration, and the DHT11 sensor that detects temperature. The processor is a Wemos ESP8266 microcontroller, and the output is a parameter that is measured remotely wirelessly via the Thingspeak application. There is also an additional output in the form of a buzzer alarm indicator when the system experiences overspeed, over vibration, or overheating.

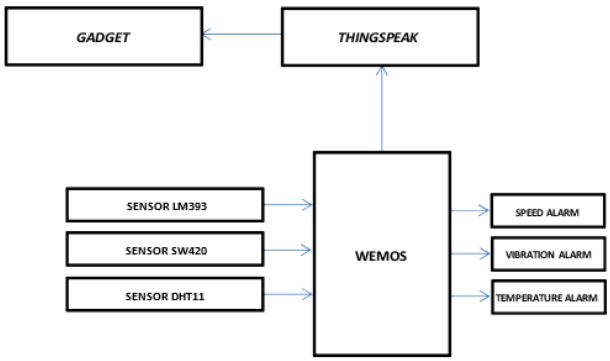

Figure 1. System Blok Diagram

When the system is operated by starting up the stirling engine, the LM393 Infrared speed sensor begins to detect changes in speed, the SW420 vibration sensor begins to

detect changes in vibration and the DHT11 temperature sensor begins to detect changes in temperature. The detection results from each of these sensors are sent to the wemos D1 esp8266 microcontroller for data acquisition according to the specified program. Furthermore, the data is transmitted via the esp8266 wifi module that is included in the wemos d1 microcontroller via the internet to a web-based monitoring system called Thingspeak. Then the data from Thingspeak is sent to a specially created android application. Then the system output in the form of a buzzer will be actuated at a certain speed, vibration and temperature that has been set which indicates the occurrence of an overspeed, overvibration and overtemperature alarm so that a shutdown is necessary to continue the troubleshooting process.

B. Hardware Design

The Input Block contains a Speed Sensor, a Vibration Sensor, and a Temperature Sensor. This block provides input to the microcontroller to be processed according to the instructions programmed into the microcontroller.
For the IR LM393 speed sensor output, it is connected to I/O pin D5 on the WeMos, and VCC to the WeMos VCC 5V, while the sensor GND is connected to the WeMos GND.

For the SW420 vibration sensor output, it is connected to the I/O pin D3 on the wemos and VCC to the VCC of the wemos 5V, while the GND of the sensor is connected to the GND on the wemos.

For the DHT11 temperature sensor output, it is connected to the I/O pin D4 on the wemos and VCC to VCC wemos 5V while the GND of the sensor is connected to GND on the wemos.

The Output Block contains three buzzers that indicate excess speed, excess vibration, and excess temperature above the system's measurement limits. This block is the microcontroller's output, following the instructions programmed into the microcontroller.

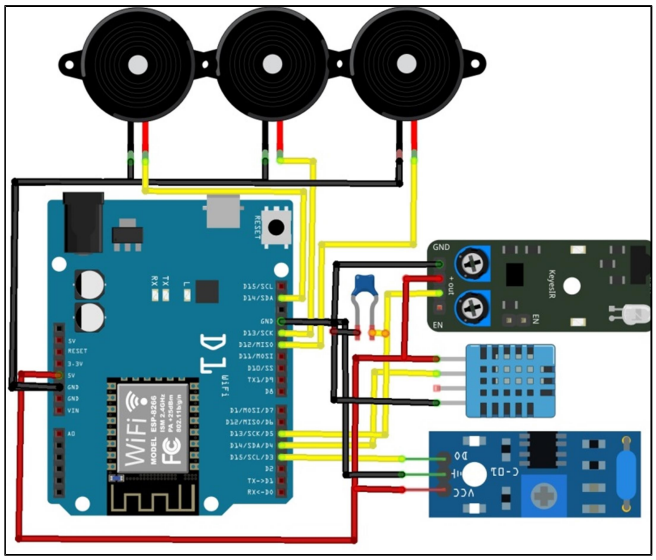

Figure 2. Wiring Diagram Hardware Design

C. Software Design

a. Thingspeak Monitoring Application Design

The first step requires a Thingspeak account by signing up. If you already have one, you can sign in to the Thingspeak website and then create a new channel. Once you have a new channel, you can set it up on the dashboard and fill in the chart fields as desired.

In the Thingspeak channel settings, there's a menu for viewing the API. An API is a programming interface used to access data on the platform. Write API keys are used to write data to the channel. Using the API makes it easier for users to access data on the platform, without having to write programs.

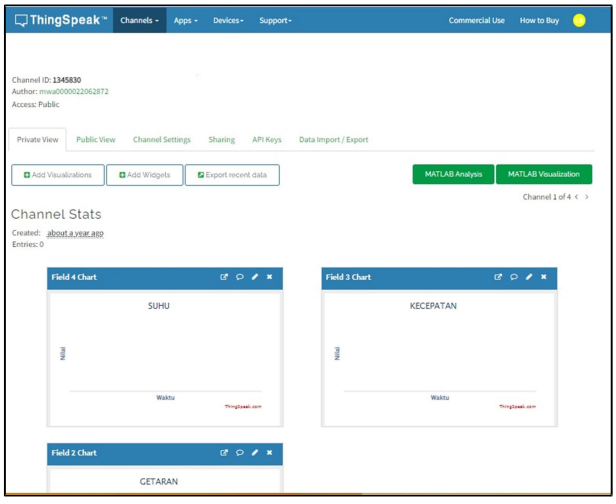

Figure 3. Thingspeak Monitoring View

b. Design of Kodular Application

The Kodular application is used to create software. In this case, software is being created to access Thingspeak so that monitoring can be viewed in a different view through an application created in Kodular. This application created from Kodular will serve as a dedicated interface so that monitoring is not performed directly on the Thingspeak website but can be more easily accessed through the Android app located in the user's device's display menu.

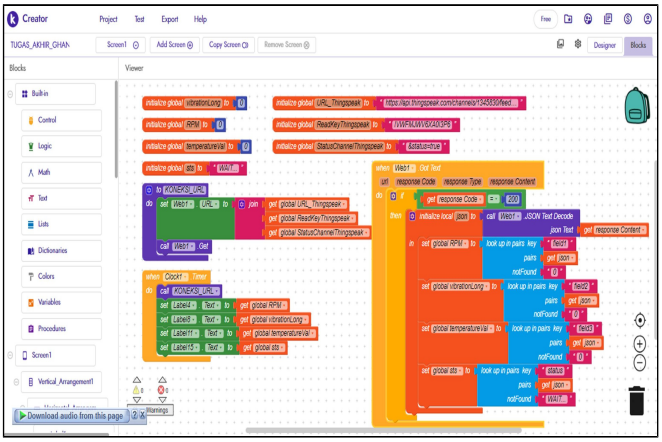

Figure 4. Program View on Kodular

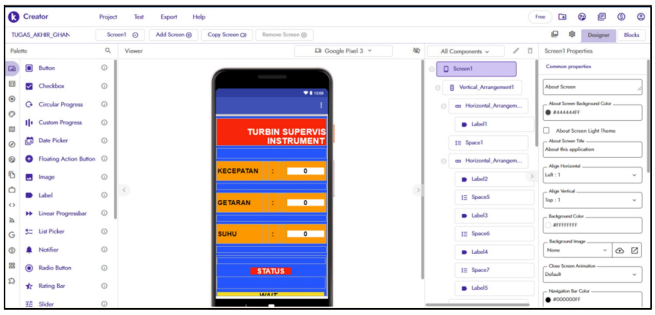

Figure 5. Monitoring Display Design

III. Results and Discussion

The testing in this study consists of two parts: the first part is hardware testing and the second part is software testing. Hardware design involves arranging electronic components to create a system that works as desired. This testing is conducted to ensure all components function according to their intended purpose. Meanwhile, software testing is conducted to ensure the coding programmed into the microcontroller runs as desired and that the interface used for monitoring functions is functioning properly.

The overall system testing is performed when each component of the input block, microcontroller block, and output block has been confirmed to be functioning properly through a single test. The results of the overall parameter testing are listed in Table 1 and 2 below:

Table 1. System Testing

Waktu	Serial Monitor IDE			Thingspeak		
	Speed (rpm)	Vibrasi (u sekon)	Temp (°C)	Speed (rpm)	Vibrasi (u sekon)	Temp (°C)
16.05	0	0	28.6	0	0	28.6
16.08	2775	0	30.7	2775	0	30.7
16.10	3885	734401	31.7	3885	734401	31.7
16.11	4425	3032	33	4425	3032	32.9

Table 2. System Testing

Aplikasi			Buzzer		
Speed (rpm)	Vibrasi (u sekon)	Temp (°C)	S	V	T
0	0	28.6	0	0	0
2775	0	30.7	1	0	0
3885	734401	31.7	1	1	0
4425	3032	33	1	0	0

The graph shown in Figure 6. below shows the measured speed variables in the system summarized in Table 1 and 2 above, compared to the sampling time during testing. The result is that the linear speed measurement is based on time.

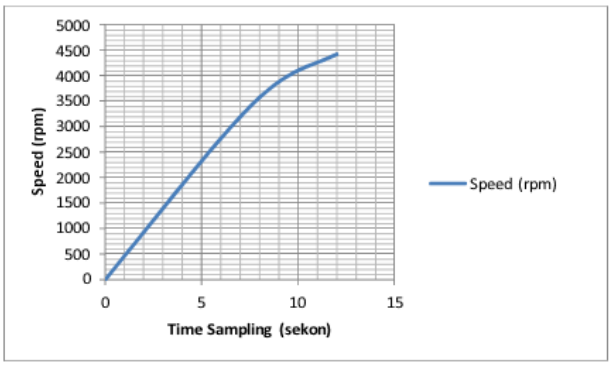

Figure 6. Speed Measurement Graph

The graph shown in Figure 7 below shows the measured vibration duration variable in the system, summarized in Table 1 and 2 above, compared to the sampling time during testing. The results indicate that the vibration measurements change or oscillate. Therefore, the test revealed that the measured vibration is not linear, but rather varies with changes in speed.

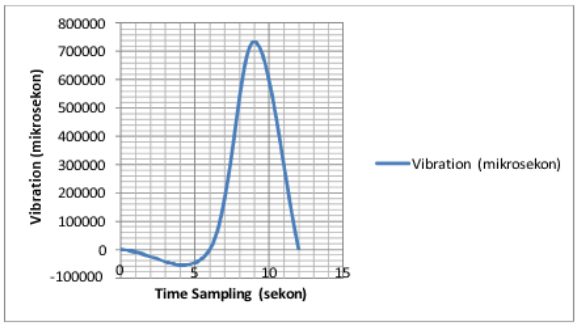

Figure 7. Vibration Measurement Graph

The graph shown in Figure 8 below shows the measured temperature variables in the system, summarized in Table 1 and 2 above, compared to the sampling time during testing. The results show that the temperature measurements are linear based on time.

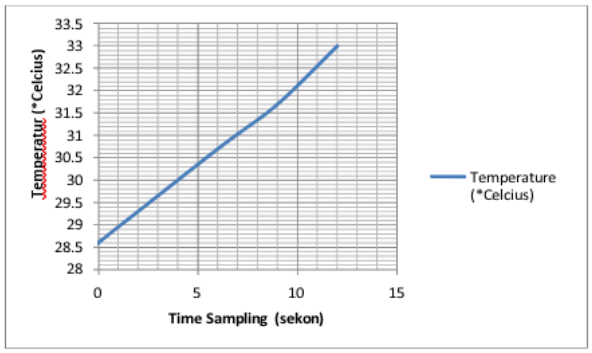

Figure 8. Temperature Measurement Graph

During the first measurement at 4:05 PM, the turbine was still off, so the speed and vibration were at 0, and the temperature measured room temperature. In this first measurement, there was no difference between the parameters measured in the serial monitor, Thingspeak, and the application.

Furthermore, there were no alarms while the turbine was off. The speed alarm was set to 1 when the speed reached 2000 rpm, the vibration alarm to 1 when the vibration reached 100,000 rpm, and the temperature alarm to 1 when the temperature reached 38 degrees Celsius.

The results of the first data collection are demonstrated through the parameters measured in the Arduino IDE, Thingspeak, and the application, as shown in Figure 9 below.

Figure 9. 1st data collection at 16.05

In the second measurement, at 4:08 PM, the turbine was started and the speed change was measured at 2775 rpm. The speed alarm was activated, causing the overspeed buzzer to sound logic 1 (sounding) because the speed was above 2000 rpm.

At the turbine speed of 2775, no vibration was detected in the system, so the vibration remained at 0. The temperature also increased from 28.6 degrees Celsius to 30.7 degrees Celsius.

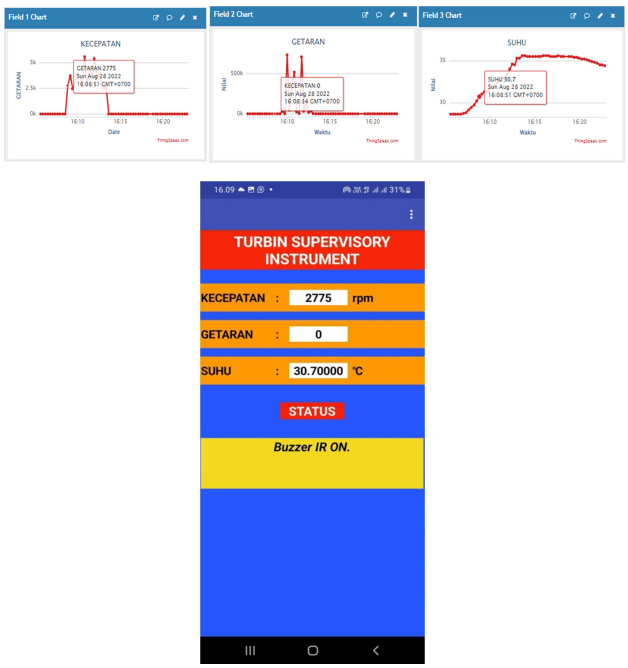

Figure 10. 2nd data collection on 16.08

The results of the second data collection are demonstrated through the parameters measured in the Arduino IDE, Thingspeak, and the application, as shown in Figure 10. below.

In the third measurement, at 16:08, the speed increased to 3885 rpm, and vibration was detected for 734,401 microseconds, or 0.7 seconds. This resulted in the vibration alarm being set to logic 1 because the vibration was above 100,000 microseconds.

The temperature alarm was not yet set to logic 1 because the temperature value was still below the safe range.

The results of the third data collection are verified through the parameters measured in the Arduino IDE, Thingspeak, and the application, as shown in Figure 11 below.

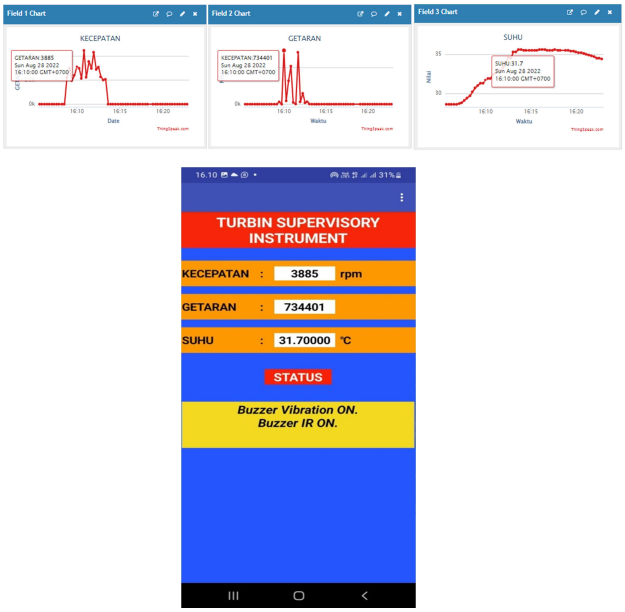

Figure 11. 3rd data collection on 16.10

In the fourth measurement at 16:11, the speed increased again to 4425 rpm, and vibration was detected for 3032 microseconds, or 0.003 seconds. The over-vibration alarm did not sound at this time because it was still below the specified setpoint. The temperature alarm also did not reach logic 1 because the temperature value was still below the safe range. The overheat alarm will be activated when the temperature reaches 38 degrees Celsius.

The results of the fourth data collection are proven through the parameters measured in the Arduino IDE software, Thingspeak, and the application, as shown in Figure 12 below.

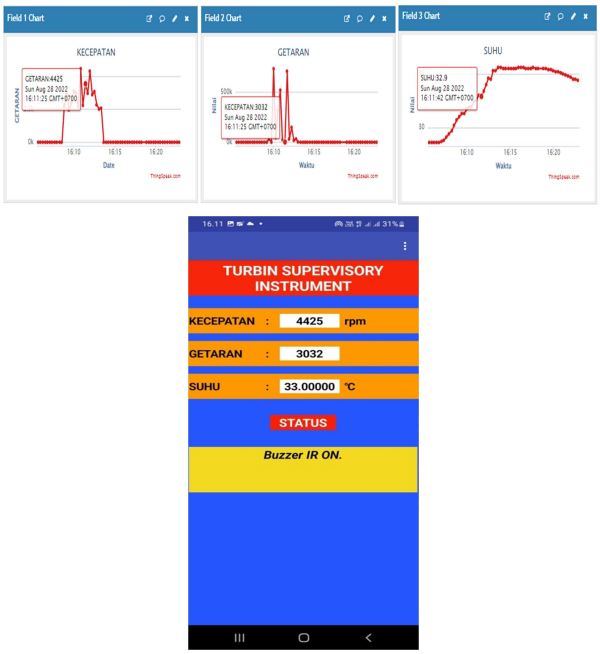

Figure 12. 4th data collection on 16.11

IV. Conclusion

Based on the test and analysis results, the following conclusions can be drawn:

1. Turbine instrument parameters, such as speed, vibration, and temperature, can be monitored remotely via an interface application on the user's device with fairly good accuracy. There was no difference in values between the serial monitor, Thingspeak, and the interface application.

2. Turbine speed, vibration, and temperature can be monitored on the Thingspeak dashboard. Turbine speed, vibration, and temperature parameters are measured graphically, and there are also overspeed, overvibration, and overheat alarm indicators on the dashboard if the logic is HIGH.

3. An alarm indicates an abnormal condition in the turbine, such as the detection of a possible shaft shift that has become unbalanced due to dynamic turbine movement. Therefore, further troubleshooting is necessary.

Acknowledgement

We would like to express our deepest gratitude to Department of Informatics and Computer Engineering at the State Polytechnic of Ujung Pandang for their valuable support.

References

[1] B.F.Mahendra, S.Mulyanto, Prihastono, “Sistem Monitoring Prototype Energi Listrik Hybrid menggunakan Panel Surya, Turbin Angin & Piezoelektrik Diatas Kapal”, Publikasi Ilmu Teknik, Teknologi Kebumian dan Ilmu Perkapalan, vol.3 no.4, 2025.

[2] M. Zaini, S. Safrudin, and M. Bachrudin, "Perancangan sistem monitoring tegangan, arus dan frekuensi pada pembangkit listrik tenaga mikrohidro berbasis IoT," TESLA, vol. 22, no. 2, pp. 139-150, Nov. 2020.

[3] A. Akhwan, B. Gunari, S. Sunardi, and W. A. Wirawan, "Rancang bangun pembangkit listrik tenaga mikrohidro (PLTMH) Politeknik Perkeretaapian Indonesia Madiun," Eksergi: Jurnal Teknik Energi, vol. 17, no. 1, pp. 15-24, 2021. S.

[4] Aida, S. Sahrul, T. Lety, and T. Tahdid, "Prototipe pembangkit listrik tenaga mikro hidro (PLTMH) turbin Pelton kapasitas 300 watt kajian debit dan arah aliran pada alat," Prosiding SENIATI, vol. 5, no. 4, pp. 118-122, 2019.

[5] H. Hamira, A. I. Pawelloi, R. Rudy, and M. I. Halim, "Pengaruh jumlah sudu terhadap tegangan output pada pengujian turbin Pelton prototipe PLTMH," in Seminar Nasional Teknik Elektro dan Informatika (SNTEI), vol. 9, no. 1, pp. 53-58, 2023.

[6] I. P. A. Wiranata, I. G. N. Janardana, and I. W. A. Wijaya, "Rancang bangun prototype pembangkit listrik tenaga mikro hidro menggunakan turbin cross-flow," Spektrum, vol. 7, no. 4, 2020.

[7] G. N. Saputra, L. Jasa, and I. W. A. Wijaya, "Pengaruh jumlah sudu pada prototype PLTMH dengan menggunakan turbin Pelton terhadap efisiensi yang dihasilkan," Jurnal SPEKTRUM, vol. 7, no. 4, 2020.

[8] J. Lambert, R. Monahan, and K. Casey, "Power consumption profiling of a lightweight development board: Sensing with the INA219 and Teensy 4.0 microcontroller," Electronics, vol. 10, no. 7, p. 775, 2021.

[9] Z. B. Abilovani, W. Yahya, and F. A. Bakhtiar, "Implementasi protokol MQTT untuk sistem monitoring perangkat IoT," Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, vol. 2, no. 12, pp. 7521-7527, 2018.

[10] S.Salsabila, M.R.Zulkarnain, L.O.Musa, dan M.Y.Yunus, “Sistem Pengaturan Beban dan Monitoring Berbasis IoT pada Pembangkit Listrik Tenaga Mikrohidro Jenis Turbin Pelton”, Vol. 23 No. 2 (2025): Oktober 2025.