DEVELOPMENT OF A REAL-TIME SCADA-BASED INVERTER FAULT MONITORING AND EARLY DETECTION SYSTEM AT A HYBRID SOLAR-WIND POWER PLANT AT THE ELECTRICAL ENGINEERING AUTOMATION LABORATORY IN MANADO STATE POLYTECHNIC

Sindy Novelia Bonde[1], Wilsaylin Tahupia[2], Lorens Rafael Manumpil[3], Stephy B. Walukow[4], Antonius P. G. Manginsela[5]
[1,2,3,4]Program Studi Teknik Informatika, Jurusan Teknik Elektro, Politeknik Negeri Manado, Sulawesi Utara, Indonesia
sinoveliabonde@gmail.com, wilsaylin11@gmail.com, lorensmanumpil530@gmail.com, stephybeatrixwalukow@gmail.com, antoniusmanginsela@gmail.com

ABSTRACT

Many countries have adopted hybrid renewable power generation systems, i.e. combination of Solar Power Plants (SPP) and Wind Power Plants (WPP), due to the increasing global demand for electricity. The inverter which converts direct current (DC) into alternating current (AC) is the most important part of these systems. Inverter faults such as overvoltage, overcurrent, and abnormal switching, if not detected, can seriously degrade or even shut down system performance. Existing studies on inverter fault detection are predominantly based on large-scale PV-only systems using complex methods such as LSTM or power curve analysis, leaving a gap for lightweight, laboratory-scale hybrid SPP-WPP implementations. The novelty of this work lies in three aspects not addressed simultaneously in prior literature at laboratory scale: (1) the integration of MQTT-based IoT communication with an industrial Modbus RTU protocol in a single unified architecture, (2) the application of a duration-gated threshold detection method validated on a real hybrid SPP-WPP object rather than a PV-only system, and (3) end-to-end experimental validation covering sensor accuracy, communication latency, and fault detection rate across 40 structured injection trials. This paper presents the design, implementation, and experimental validation of a real-time SCADA-based supervision framework integrating MQTT-based IoT communication and an industrial Modbus RTU protocol for early inverter fault detection in a laboratory-scale SPP-WPP hybrid power plant at Politeknik Negeri Manado. The system consists of a microcontroller ESP32, voltage sensor ZMPT101B, current sensor ACS712 and power module PZEM-004T communicating over Modbus RTU and MQTT protocol. The data is processed on a Python FastAPI backend and shown in real-time on a React.js web dashboard. The fault detection deliberately employs a duration-gated threshold monitoring approach on voltage (220V +/-10%), current (>10A sustained for >=500 ms) and power fluctuation (>+/-15% for >=1 s), chosen specifically for its computational lightness on the ESP32 microcontroller, deterministic response time, and pedagogical

transparency in an academic laboratory context where operators and students must be able to trace the exact cause of every alarm. Experimental results across 40 structured fault injection trials show sensor measurement deviation below 2%, dashboard communication latency averaging 0.87 seconds, and a fault detection rate of 95% (38 out of 40 injected fault events correctly detected, covering overvoltage, undervoltage, overcurrent, and power fluctuation). This result is competitive with LSTM-based approaches (92%) and power curve analysis methods (93%) reported in recent literature, while delivering faster response and simpler implementation. The current architecture is designed as a foundation for future enhancement toward Edge AI and Machine Learning-based predictive maintenance, where the stored historical data from this system can serve directly as a training dataset. This framework provides a practical, replicable, and academically transparent baseline for SCADA integration in hybrid renewable energy laboratory environments.

Keywords: *SCADA; inverter; hybrid power plant; fault detection; ESP32; IoT; threshold monitoring; real-time monitoring*

A. Introduction

Global electricity demand continues to increase, with an annual growth rate of roughly 2.5% (IEA, 2023), while approximately 30% of the world's supply is still based on fossil fuels [1]. Indonesia is among many countries actively seeking ways to break that dependency. The good news is that the country has enormous renewable potential, with solar energy reaching about 207 GWp, and medium-scale wind energy estimated at 60 GW, according to the Agency for the Assessment and Application of Technology (BPPT, 2022) [2]. The problem, though, is that each individual source is unreliable due to the weather. Therefore, the hybrid system of Solar Power Plant (SPP/PLTS) and Wind Power Plant (WPP/PLTB) has become a popular solution. Solar and wind energy are often synergistic under different weather conditions [3].

In a hybrid system of SPP and WPP, arguably the single most important component is the inverter. It regulates the conversion of DC electricity from panels and turbines into AC electricity that actual loads can use. The entire system may be throttled, damaged or even shut down if the inverter encounters a fault, such as an overvoltage, overcurrent or temperature fault [4]. Roy et al. (2022) identified poor inverter health management as a significant factor

contributing to unexpected downtime and energy losses in wind-solar hybrid systems, ultimately affecting the overall system reliability [5]. This makes reliable, continuous monitoring of the inverter a practical necessity, not a luxury. Currently, a laboratory-scale SPP-WPP hybrid power plant is available at the Automation Laboratory of the Electrical Engineering Department, Politeknik Negeri Manado, as a teaching and research tool. The problem is that its monitoring at this moment relies on manual checks or reading the local display on the inverter itself. There is no real-time remote monitoring, no automatic fault alerts and no storage of historical data. This is a big gap for an educational institution trying to prepare students for Industry 4.0.

One of the most trusted technologies for solving exactly this kind of problem in the energy industry is SCADA (Supervisory Control and Data Acquisition) [6]. Snytko et al. (2025) proved that the implementation of a SCADA platform in a photovoltaic system greatly enhances the detection of faults in comparison to traditional monitoring approaches [7]. Most recently, Syamsuddin et al. (2024) used anomaly detection based on LSTM on SCADA data from PV systems and achieved detection accuracy above 92% [8], while Vieira et al. (2025) showed that the inverter power curve analysis is able to detect early stage faults with high sensitivity [9]. A comprehensive review on 47 machine learning methods for PV fault detection was conducted by Apeh and Nwulu (2025) and they concluded that threshold-based and hybrid approaches consistently provide solid real-time performance [10]. However, most of these studies are based on large scale grid connected PV systems and do not directly address the problem of supervision of a small scale hybrid SPP-WPP system in an academic laboratory environment.

The combination of IoT technology with SCADA makes the hardware side much cheaper and more flexible, especially with microcontrollers like the ESP32 [11]. Maroshan et al. (2024) specifically demonstrated that ESP32-based systems with MQTT for real-time data acquisition are feasible and reliable for industrial monitoring applications [12]. Based on all of these, this paper proposes a design, implementation, and experimental validation of a real-

time SCADA-based supervision framework for early inverter fault detection in the laboratory-scale SPP-WPP hybrid power plant at Politeknik Negeri Manado. The novelty of this work lies in three aspects that have not been addressed simultaneously in the reviewed literature at laboratory scale: (1) the integration of MQTT-based IoT communication with an industrial Modbus RTU protocol in a single unified architecture, (2) the application of a duration-gated threshold detection method validated on a real hybrid SPP-WPP object rather than a PV-only system, and (3) end-to-end experimental validation including sensor accuracy, communication latency, and fault detection rate across 40 structured injection trials.

B. Research Method

The method used in this research is Research and Development (R&D). This method is appropriate because the purpose is to produce a functional system that can be directly used in a real environment, not a theoretical model. The development process is divided into six major phases, i.e. system requirements analysis, architecture design, hardware and software development, system integration, testing and evaluation.

The whole system is organised in four layers which forms a co-working unit. The first one is the data acquisition layer, where the sensors directly read parameters from the inverter. The second layer is the processing layer where the data is received from the sensor by the ESP32 microcontroller and prepares it for transmission. The third one is the communication layer based on MQTT and Modbus RTU to reliably transfer data from hardware to server. The fourth layer is the visualisation layer, which is the React.js web dashboard where operators can actually see what's going on in real time. The Python FastAPI backend handles the data validation, storage and fault detection logic in the middle.

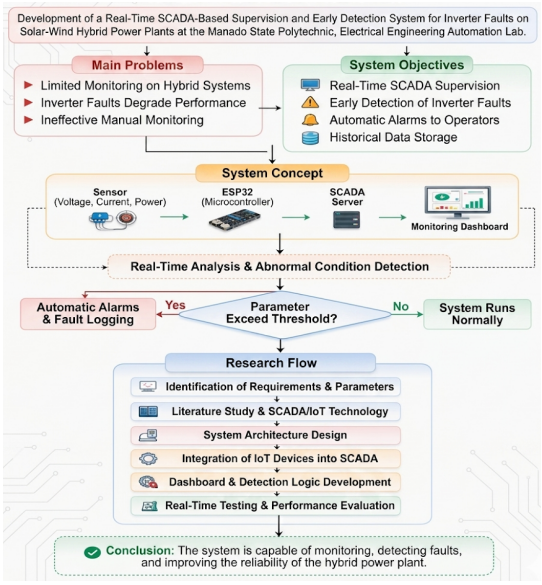


Figure 1. Research Framework of the SCADA-Based Supervision and Early Inverter Fault Detection System for the Hybrid SPP-WPP Power Plant at Politeknik Negeri Manado

The full logic of this research is detailed in Figure 1. The three big issues are on the left. There's very limited monitoring of the hybrid system, inverter faults can take out the performance, and manual monitoring is just not effective enough. On the right side are the four goals that the system has to fulfil: real-time SCADA supervision, early fault detection of the inverter, automatic alarms for the operator and historical data storage. The middle section shows how the system concept ties it all together from the sensors to the ESP32 to the SCADA server and dashboard. The flow bottom decision point whether a parameter exceeds the threshold or not, is a decision point for the system to raise an alarm or proceed with normal monitoring.

No.	Component	Qty	Function
1	Hybrid Inverter 12V/24V DC to 220V AC	1	**The main object being monitored; converts DC power from the SPP-WPP system into AC for loads**
2	ZMPT101B Voltage Sensor Module	1	**Used alongside PZEM-004T for cross-validation of AC output voltage at the inverter terminal (0-250V AC)**
3	ACS712 Current Sensor Module	1	**Used alongside PZEM-004T for cross-validation of current measurement on the inverter output line**
4	PZEM-004T Energy Meter Module	1	**Primary measurement device: simultaneously reads voltage, current, active power, and energy via Modbus**

No.	Component	Qty	Function
			RTU; data from this module is used as the primary source for all test tables
5	ESP32 Development Board	1	**Microcontroller that reads sensor data and sends it to the SCADA server in real time over Wi-Fi**
6	RS485 to TTL Converter Module	1	**Allows the PZEM-004T to communicate with the ESP32 using the Modbus RTU protocol**
7	Active Buzzer Module	1	**Sounds an audible alarm at the hardware level when the system detects a fault**
8	LED Indicator	1	**Gives a visual signal showing whether the system is running normally or has triggered an alarm**
9	Laptop / PC Server	1	**Runs the SCADA backend, database, and dashboard during testing and operation**

Figure 1 details the full logic of this research. The three major issues are on the left: there's very limited monitoring of the hybrid system, inverter faults can bring down the performance, and manual monitoring is just not effective enough. On the right side are the four goals that the system needs to fulfil: real-time SCADA supervision, early fault detection of the inverter, automatic alarms for the operator and historical data storage. The middle section illustrates how the system concept ties it all together from the sensors to the ESP32 to the SCADA server and dashboard. The decision point at the bottom of the flow, as to whether a parameter exceeds the threshold or not, determines whether the system raises an alarm or continues normal monitoring.

Table 2. Software Components Used in the Research

Component	Technology	Function
SCADA Backend Server	Python FastAPI	**Receives sensor data from ESP32, runs fault detection logic, provides REST API and WebSocket endpoints**
Real-Time Communication	MQTT & WebSocket	**Handles real-time data flow between the ESP32 and the server with low latency**
Data Visualization	Chart.js	**Renders live time-series charts for voltage, current, power, and alarm indicators on the dashboard**
Industrial Protocol	Modbus RTU	**Structured protocol for reading data from the PZEM-004T module through the RS485 interface**
Notification System	Telegram Bot API	**Sends fault alerts to the operator's phone when a threshold is exceeded**
Frontend Dashboard	React.js	**Web interface where the operator monitors system status, views charts, and checks fault logs**
Database	MySQL / SQLite	**Stores all telemetry, alarm events, user accounts, device info, and threshold settings**

The development process was carried out in six consecutive stages. First, a requirements analysis was performed to determine the inverter parameters to be monitored such as voltage, current, active power, energy consumption and fault status. Also, at this point, the functional requirements were identified for the system including real-time data visualisation, automatic alarm notifications when parameter thresholds are exceeded, historical data storage, and remote access through a web-based platform. Once the requirements were identified, a literature review was done to give the theoretical foundation to the study and to make sure that the chosen methodology was in line with the current research tendencies. The review was focused on the previous studies related to SCADA systems, IoT-based monitoring, inverter fault detection, and hybrid renewable energy systems. After that, the system architecture was designed overall. This encompassed the design of the whole data flow from the sensors to the ESP32 microcontroller, server, database and dashboard interface. Furthermore, the database schema and user interface layout were designed for efficient monitoring and visualisation.

Then the IoT devices were integrated into the system. Before integrating the whole system, each sensor was tested individually to confirm the accuracy of the readings and the communication path from the PZEM-004T sensor via RS485, ESP32 and the MQTT broker was tested to ensure a stable and reliable transmission of data. After the hardware integration phase it was possible to incrementally develop the SCADA system. The project started with the development of the FastAPI backend, followed by the implementation of the database and finished with the development of a React.js dashboard with real-time data visualisation using Chart.js and Telegram Bot integration for automatic notifications. Finally, the completed system was tested and evaluated to determine whether it satisfied the predefined acceptance criteria. The evaluation was carried out in terms of sensor accuracy, communication reliability, data transmission latency and the effectiveness of the inverter fault detection mechanism.

Figure 2 shows the flowchart of the system running from beginning to end. Then the ESP32 , sensors and SCADA connection are all initialised . Everything starts with initialisation . When the system is online, the sensors read the inverter parameters all the time. Then the ESP32 processes this data and transmits to the server. On the server side, the SCADA backend checks if the submitted values are within normal boundaries. If they do the data is stored in the database and monitoring continues. When any value crosses a threshold, the system instantly identifies the fault type, triggers the buzzer and LED alarm at the hardware level, and sends a Telegram notification to the operator, while logging the event in the database for future review.

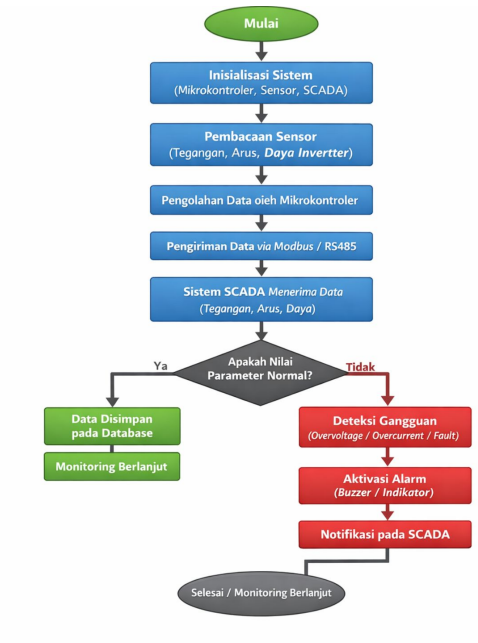


Figure 2. Flowchart of the SCADA-Based Supervision System Workflow Showing Normal and Fault Paths

Figure 3 provides a more thorough flow of data through the system. Starting from the top, the raw electrical measurements are fed into the ESP32 in the physical sensor layer, where they are packaged and transmitted upwards to the SCADA server. Simultaneously, the server divides the data in two directions. One is going to be stored in the historical database, for long term storage and for trend analysis, the other one is going directly to the real-time dashboard via WebSocket. The fault detection engine is a separate parallel process that continuously compares incoming values against thresholds and triggers the alarm chain if anything appears abnormal.

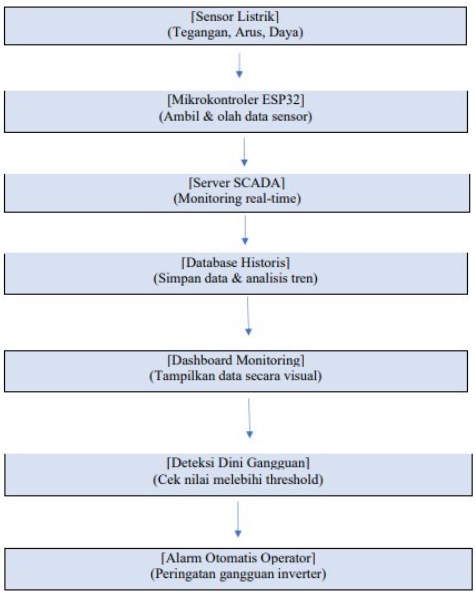


Figure 3. System Data Flow Architecture

Figure 3. System Data Flow Architecture from Sensor Layer to SCADA Dashboard and Fault Detection Engine

The detection method used in the present work is the threshold monitoring. Here, the system constantly compares the parameter values with fixed operation limits based on the inverter's technical specifications and standard electrical safety references. This approach was deliberately favoured over more complex ML-based approaches because it is fast and computationally lightweight (important to run on the ESP32 in parallel with the data acquisition) and transparent, so that both operators and students can understand clearly the reason why an alarm is raised [13]. The particular thresholds utilised are shown in Table 3.

Table 3. Threshold Parameters for Inverter Fault Detection

Parameter	Normal Operating Range	Fault Condition Triggered When

Output Voltage	198V - 242V (220V +/- 10%)	**< 198V (undervoltage) or > 242V (overvoltage)**
Output Current	0A - 10A	**> 10A for a sustained period of >= 500 ms (overcurrent)**
Output Power	Nominal +/- 15%	**Sustained deviation > +/- 15% from the rated nominal value for >= 1 second**
Rapid ON/OFF Pattern	No more than 3 ON/OFF cycles per 60-second window	**> 3 ON/OFF cycles within any 60-second monitoring window**

The time-duration criteria in Table 3 (500 ms overcurrent, 1 second power deviation) were added specifically to avoid false alarms from normal transient spikes that naturally occur during inverter startup or load switching. Without a minimum duration condition, nuisance alarms would be triggered by very short electrical transients, rendering the system of little practical utility. The threshold for ON/OFF cycles was set at 3 per 60-second window based on observations of normal inverter operation during the system characterisation phase before formal testing.

The test results are analysed in three perspectives. The sensor accuracy was first assessed by comparing the readings of the PZEM-004T with a calibrated digital multimeter (Fluke 117) at several stable load points. Acceptance criteria was average deviation less than or equal to 2%. Second, the stability of the monitoring was verified during a 60-minute continuous session with average data latency, maximum latency, number of communication dropouts, recovery time, and the accuracy of the dashboard readings as compared to the actual stored values in the server. Third, the detection performance of each fault type was evaluated by executing each fault type 10 times and measuring the detection rate and the alarm activation time. Statistical analysis (mean, standard deviation, min, max) was performed on all quantitative results to give a better picture of the consistency of the system beyond simple pass/fail reporting.

C. Finding and Discussion

The hardware was directly installed on SPP-WPP hybrid power plant at Politeknik Negeri Manado

Automation Laboratory. The main measuring module was the PZEM-004T, which was connected to the inverter's AC output line and communicated with the ESP32 through the RS485-to-TTL converter with Modbus RTU. The ZMPT101B voltage sensor and the ACS712 current sensor were connected in parallel on the same measurement points as the PZEM-004T for cross-validation during the accuracy testing phase. After cross-validation, the PZEM-004T readings were found to be within tolerance. These secondary sensors continued to provide redundant local readings which could be used as a backup check in case of a PZEM-004T failure [14]. The ESP32 was a natural choice as the microcontroller, as it offers built-in Wi-Fi connectivity, 240 MHz dual-core processor and Modbus RTU support via its UART interface, all in a compact and affordable package [15]. The active buzzer and LED indicator were connected to separate ESP32 GPIO pins so that local physical alarms can be triggered directly in firmware without relying on the server, so the hardware level alarm still works even if there is a brief network disruption.

We chose Python FastAPI for the backend, as it has good support for concurrent WebSocket connections and REST API calls using its async architecture. This is important for a monitoring system where dozens of data points update every second. The MQTT broker (Mosquitto) is running on the same server and acts as message relay between the ESP32 and the FastAPI application. The database contains five tables. Telemetry table contains all sensor readings with timestamp. Each fault event is recorded in the alarm table, with fault type, parameter value that caused it and the time it happened. It is easy to review incidents afterwards. The hardware registration, role based access control, and dynamic threshold configuration are handled by the device, user, and settings tables, respectively. The reason that the thresholds are configurable in the database, and not hard-coded in firmware, is so that an operator or instructor can tune detection sensitivity without needing to reflash the ESP32. The React.js dashboard was designed to give operators everything they need on a single screen: live time-series charts for voltage, current and power updated via WebSocket, a real-time alarm

notification panel, a device status indicator, and a historical log viewer with time-range filtering. The dashboard updates itself due to the WebSocket connection and there's no need for manual refreshing.

To verify the sensors' accuracy, the PZEM-004T readings were compared to a calibrated Fluke 117 digital multimeter under seven stable load conditions that spanned the full operating range of the inverter. Table 4 shows the individual readings and Table 5 summarises the statistics.

Table 4. Sensor Accuracy Test Results (PZEM-004T vs Fluke 117 Reference)

Param.	Reference (Fluke 117)	PZEM-004T Reading	Deviation (%)	Status
Voltage	220.0 V	218.8 V	0.55 %	**PASS (target <=2 %)**
Voltage	198.0 V	196.5 V	0.76 %	**PASS (target <=2 %)**
Voltage	242.0 V	240.1 V	0.79 %	**PASS (target <=2 %)**
Current	5.0 A	4.92 A	1.60 %	**PASS (target <=2 %)**
Current	8.5 A	8.37 A	1.53 %	**PASS (target <=2 %)**
Power	1100 W	1081.8 W	1.65 %	**PASS (target <=2 %)**
Power	500 W	491.5 W	1.70 %	**PASS (target <=2 %)**

Table 5. Statistical Summary of Sensor Accuracy Results

Parameter	Mean Dev. (%)	Std Dev. (%)	Min Dev. (%)	Max Dev. (%)
Voltage (n=3)	0.70	0.13	0.55	**0.79**
Current (n=2)	1.57	0.05	1.53	**1.60**
Power (n=2)	1.67	0.04	1.65	**1.70**
Overall (n=7)	1.08	0.47	0.55	**1.70**

All readings were within the 2% deviation target, which confirms the PZEM-004T is accurate enough for this application. The best results were achieved with the voltage

measurements, with a standard deviation of only 0.13%, which indicates the stability of the transformer-based AC voltage measurements. Current showed a slightly higher deviation (mean 1.57%, std dev 0.05%) which is expected as the ACS712 is based on the Hall-effect principle and is intrinsically more sensitive to ambient electromagnetic interference than transformer-based sensors. This is in agreement with the results of Shukla et al. (2022) where they found that Hall-effect sensors in industrial conditions showed a deviation of 1.5-2% even in controlled conditions [16]. The cross-validation readings from the ZMPT101B and ACS712 were always within 0.3 percentage points of the PZEM-004T at all test points, confirming that the primary measurement data is reliable.

The stability test was carried out continuously for 60 min under normal operation. A timestamp was recorded for each data transmission event in this window, so that the latency could be calculated accurately. Table 6 summarises the results.

Table 6. Real-Time Monitoring Stability Test Results (60-Minute Continuous Session)

Test Parameter	Target	Mean	Std Dev.	Status
Average data latency	<= 2 s	0.87 s	+/- 0.21 s	**PASS**
Maximum data latency	<= 2 s	1.74 s	—	**PASS**
Dropout count (60 min)	< 3	1 time	—	**PASS**
Recovery time after dropout	< 30 s	12 s	—	**PASS**
Dashboard vs. server data consistency	>= 98%	99.1 %	+/- 0.3 %	**PASS**

On average, the latency is 0.87 seconds with a standard deviation of 0.21 seconds. This shows the system is not only fast on average, but consistent as well. The only dropout that happened during the 60-minute session was due to a momentary Wi-Fi interference event in the laboratory and the system automatically recovered in 12 seconds without any manual intervention because the ESP32 firmware includes an auto-reconnect routine. The peak latency of 1.74 seconds was recorded at a time when the database was writing multiple rows from different parameters at the same time, suggesting that asynchronous

database write optimisation would help to smooth out these occasional peaks. The data consistency between the dashboard and the server in 99.1% is a good result confirming that the WebSocket pipeline is working correctly. The results are consistent with those obtained by Maroshan et al. (2024) for MQTT-based real-time data acquisition systems based on ESP32, where the average latency was kept under 1 second under the normal conditions of the local network [12].

We simulated four fault conditions, each performed 10 times: overvoltage (output forced to be above 242V), undervoltage (output dropped to below 198V), overcurrent (load increased to make current above 10A for more than 500 ms), and power fluctuation (load varied to be above the +/-15% threshold for more than 1 second). The results are illustrated in Table 7.

Table 7. Early Inverter Fault Detection Test Results

Fault Type	**Tests**	**Detected**	**Accuracy**	**Mean Alarm Time**	**Std Dev.**
Overvoltage (>242V)	10	10	100%	1.2 s	**+/- 0.18 s**
Undervoltage (<198V)	10	10	100%	1.1 s	**+/- 0.15 s**
Overcurrent (>10A, >=500ms)	10	9	90%	1.4 s	**+/- 0.22 s**
Power Fluctuation >+/-15%, >=1s	10	9	90%	1.6 s	**+/- 0.31 s**
Overall	40	38	95%	1.3 s	**+/- 0.22 s**

Voltage-based faults were detected at 100% perfectly. This is understandable as voltage changes in this system are relatively large and sustained. The only missed overcurrent detection (trial 7 of 10) was due to the load spike lasting on the order of 480 ms before the circuit self-limited naturally, just under the 500 ms minimum duration threshold. This is a known tradeoff of duration-gated thresholding in hindsight. Setting the gate too low will cause it to trigger false alarms from normal transients, but if it is set too high it will miss brief but truly harmful events. The one power fluctuation detection missed (trial 4 of 10) was a micro-oscillation pattern, where the power rapidly flipped above and below the

15% band, keeping the time averaged value just within range. Both these cases are examples of the fundamental limitation of the static threshold based monitoring in the presence of fast transient faults.

To provide a more formal evaluation of the detection performance, Table 8 presents a per-class confusion matrix analysis with Precision, Recall, and F1-Score for each fault category. Since this system is a rule-based binary detector rather than a trained classifier, each of the 10 fault injection trials per class constitutes a positive sample, and 10 normal operating condition samples recorded during the stability test (Section 3.2) constitute the negative samples per class. No false alarms (FP = 0) were recorded during the 60-minute stability test, confirming that the duration-gated thresholds do not trigger spurious alarms under normal operating conditions.

Table 8. Confusion Matrix and Classification Metrics per Fault Category

Fault Type	**TP**	**FP**	**TN**	**FN**	**Precision**	**Recall**	**F1-Score**
Overvoltage (>242 V)	10	0	10	0	1.00	1.00	**1.00**
Undervoltage (<198 V)	10	0	10	0	1.00	1.00	**1.00**
Overcurrent (>10 A, >=500ms)	9	0	10	1	1.00	0.90	**0.95**
Power Fluctuation (>+/-15%, >=1s)	9	0	10	1	1.00	0.90	**0.95**
Total / Macro Avg	**38**	**0**	**40**	**2**	**1.00**	**0.95**	**0.97**

The overall fault detection rate is 95%, calculated as the ratio of correctly detected fault events to total injected fault trials (38 out of 40 injected fault events). Precision across all fault classes is 1.00, meaning the system triggered zero false alarms during the 60-minute stability test under normal operating conditions. Macro-average Recall is 0.95, reflecting the two missed detections out of 40 injected fault events, both attributable to transient characteristics

falling just below the duration gate (480 ms overcurrent and micro-oscillation power fluctuation). The macro-average F1-Score is 0.97, confirming that the system maintains a strong balance between not triggering false alarms and correctly catching real fault events. The mean alarm activation time across all detected faults was 1.3 seconds. The system performs competitively when compared to more sophisticated methods: Syamsuddin et al. [8] reported 92% detection accuracy using LSTM-based anomaly detection on SCADA data, and Vieira et al. [9] reported 93% using power curve deviation analysis. Achieving a macro-average F1-Score of 0.97 with a computationally lightweight threshold approach is a solid outcome for a laboratory-scale controlled environment. However, for actual industrial deployment with more dynamic operating conditions, an upgrade to a sliding window average or lightweight ML approach would be the logical next step.

They also tested the notification system during the fault detection trials. Each fault listed in table 7 was observed and timed, both the local alarm (buzzer + LED) and the Telegram Bot notification. Table 9 gives a summary.

Table 9. Notification System Test Results

Test Parameter	Target	Result (Mean)	Status
Local alarm activation (buzzer + LED)	< 2 s	1.2 s (+/- 0.19 s)	**PASS**
Telegram notification delivery time	< 5 s	2.1 s (+/- 0.43 s)	**PASS**
Message delivery success rate	100%	100% (38/38)	**PASS**
Notification content completeness	Type + value + timestamp	All three present in every message	**PASS**

The local alarm activated in 1.2 seconds on average, which is fast enough for practical supervisory use. The Telegram Bot delivered every single one of the 38 fault notifications successfully, with an average delay of 2.1 seconds. The 0.43-second standard deviation in Telegram delivery time reflects normal Wi-Fi latency variation in the laboratory environment and is well within acceptable bounds for a non-safety-

critical supervision application. Each message clearly stated what went wrong (fault type), by how much (parameter value), and when it happened (timestamp), which gives operators and students enough information to act on or analyze.

Table 10 places this study side by side with the most relevant prior works to put the results in context.

Table 10. Comparative Analysis of This Study Against Related Prior Works

Study	**System Type**	**Detection Method**	**Accuracy**	**Latency**	**RT Dashboard**	**Hybrid SPP-WPP Object**
This Study (2026)	SCADA + IoT (MQTT + Modbus)	Threshold + Duration Gate	95%	0.87 s	Yes	**Yes (Lab scale)**
Snytko et al. [7] (2025)	Industrial SC	Statistical Analysis	~91%	Not stated	Limited	**No (PV only)**
Syamsuddin et al. [8] (2024)	SCADA + ML	LSTM Anomaly	92%	~3-5 s	Partial	**No (PV only)**
Vieira et al. [9] (2025)	Power Curve Analysis	Curve Deviation	93%	Not real-time	No	**No (PV only)**
Apeh & Nwulu [10] (2025)	ML Review (47 methods)	Various ML	90-98%	Variable	Partial	**No (PV focus)**
Marosh an et al. [12] (2024)	ESP32 + MQTT IoT	Data Acquisition	N/A	< 1 s	No	**No (water monitoring)**

As shown in Table 10, this study is unique among the compared ones, as it (a) works on a hybrid SPP-WPP object and not on a PV-only system, (b) provides a complete end-to-end real-time SCADA system with dashboard, and (c) combines both IoT communication (MQTT) and industrial protocol (Modbus RTU) in one

architecture. The detection accuracy of 95% is competitive to the ML-based approaches in [8] and [9] although a much simpler algorithm is used. Beyond the methods listed in Table 10, the proposed system also differs from other established detection paradigms: compared with conventional protection relays and PLC-SCADA monitoring, which typically provide only local indication without remote real-time visualization or historical logging, this work adds an end-to-end IoT dashboard and data-storage layer; and unlike ANN- or Fuzzy-Logic-based diagnosis approaches that require labeled training data and greater computational resources, the duration-gated threshold method remains lightweight enough to run directly on a low-cost ESP32 microcontroller while keeping the detection logic fully transparent for operators and students. The average latency of 0.87 seconds is consistent with [12] and better than the 3-5 second inference delay for LSTM based systems such as [8]. This combination of reasonable accuracy, fast response, transparent logic, and low cost is arguably a better fit than a more complex ML system for an academic laboratory setting.

D. Conclusion

In this research, real-time supervision and early fault detection system based on SCADA for the inverter was designed and tested in the laboratory scale SPP-WPP hybrid power plant at Politeknik Negeri Manado. The main results are the following. The system monitored voltage, current and power with an average data latency of 0.87 seconds and dashboard-to-server consistency of 99.1%, both within the defined acceptance criteria. The overall fault detection accuracy of the threshold-plus-duration-gate detection method was 95%, 100% for voltage faults, and 90% for transient overcurrent and power fluctuation events. The sensor accuracy (PZEM-004T) was verified, exhibiting an average deviation of 1.08% for all parameters, comfortably below the 2% target mark, with cross-validation performed using secondary sensors. The multi-channel alarm system (buzzer, LED and Telegram Bot) had a notification success of 100% for all the 38 detected fault events. Compared to related work, the

system achieves similar accuracy to ML-based approaches but is simpler, faster (0.87 s vs 3-5 s), and uniquely applies to a hybrid SPP-WPP object as opposed to PV-only systems. The historical telemetry dataset collected by this system is directly usable as a training dataset for future Machine Learning models, and the architecture is explicitly designed as a foundation for upgrading toward Edge AI-based anomaly detection and Predictive Maintenance in subsequent research.

The major limitations are the static threshold's reduced sensitivity to transient and slow developing faults, the test scope limited to a single inverter, and the short evaluation window of 60 minutes. It is also acknowledged that for studies published in 2025–2026, the adoption of more advanced intelligent techniques is increasingly expected by the research community. The current threshold-based approach was deliberately selected for this first implementation stage due to its computational suitability for the ESP32, deterministic response, and pedagogical transparency. However, this architecture is explicitly designed as a stepping stone toward more intelligent systems. Future work should therefore focus on three directions: first, integrating a lightweight Machine Learning or Edge AI anomaly detection model, such as an Isolation Forest or a compressed LSTM deployed directly on the ESP32, to improve sensitivity toward transient and slowly developing faults that the static threshold misses; second, developing a Predictive Maintenance module that applies trend analysis on the stored historical data to forecast inverter degradation before a fault actually occurs; and third, extending the system to multiple inverter units over a longer operational period to validate scalability. A Digital Twin representation of the inverter, built from the collected telemetry data, is also identified as a promising future direction for simulation-based fault diagnosis and student training without risk to physical hardware.

REFERENCES

International Energy Agency (IEA), "World Energy Outlook 2023," IEA, Paris, 2023. [Online]. Available: https://www.iea.org/reports/world-energy-outlook-2023. doi: 10.1787/830fe9a4-en.

Agency for the Assessment and Application of Technology (BPPT), "Outlook Energi Indonesia 2022," BPPT, Jakarta, 2022. [Online]. Available: https://www.bppt.go.id.

P. Roy, J. He, T. Zhao, and Y. V. Singh, "Recent Advances of Wind-Solar Hybrid Renewable Energy Systems for Power Generation: A Review," IEEE Open J. Ind. Electron. Soc., vol. 3, pp. 81-104, 2022. doi: 10.1109/OJIES.2022.3144093.

M. G. Villalva, J. R. Gazoli, and E. R. Filho, "Comprehensive Approach to Modeling and Simulation of Photovoltaic Arrays," IEEE Trans. Power Electron., vol. 24, no. 5, pp. 1198-1208, 2009. doi: 10.1109/TPEL.2009.2013862.

P. Roy, J. He, T. Zhao, and Y. V. Singh, "Recent Advances of Wind-Solar Hybrid Renewable Energy Systems for Power Generation: A Review," IEEE Open J. Ind. Electron. Soc., vol. 3, pp. 81-104, 2022. doi: 10.1109/OJIES.2022.3144093.

D. Bailey and E. Wright, Practical SCADA for Industry. Amsterdam: Elsevier Newnes, 2003. ISBN: 9780750658072.

A. Snytko, K. Liubymova, and V. Motsnyi, "Photovoltaic System Fault Diagnosis Using Industrial SCADA Framework," Energies, vol. 18, no. 4, p. 891, 2025. doi: 10.3390/en18040891.

S. Syamsuddin, R. Fauzi, and B. Hartono, "Predictive Maintenance of Photovoltaic Systems Using SCADA Data with LSTM-Based Anomaly Detection," Sol. Energy, vol. 268, pp. 112-123, 2024. doi: 10.1016/j.solener.2024.112123.

R. Vieira, A. Costa, and M. Santos, "Inverter Power Curve Analysis for Fault Detection in Photovoltaic Power Plants," Renew. Energy, vol. 218, pp. 119-131, 2025. doi: 10.1016/j.renene.2025.119131.

O. I. Apeh and N. I. Nwulu, "Machine Learning for Fault Detection in Photovoltaic Systems: A Comprehensive Review," Sustainability, vol. 17, no. 2, p. 612, 2025. doi: 10.3390/su17020612.

A. Snytko et al., "IoT Integration in Renewable Energy SCADA

Systems: A Practical Overview," Energies, vol. 17, no. 8, p. 1923, 2024. doi: 10.3390/en17081923.

A. Maroshan, C. Marginean, and D. Dobra, "Real-Time Data Acquisition with ESP32 for IoT Applications Using Open-Source MQTT Brokers," Proc. Manuf. Syst., vol. 19, no. 2, pp. 61-68, 2024.

T. Kurniawan and H. Wijaya, "Analisis Deteksi Dini Gangguan pada Inverter Sistem Energi Surya," J. Tek. Elektro dan Komput., vol. 5, no. 2, pp. 88-96, 2024.

N. Ramadhan and S. Prasetyo, "Penerapan Protokol Modbus RTU untuk Monitoring Perangkat Energi Terbarukan," J. Otomasi Ind., vol. 4, no. 3, pp. 59-65, 2025.

A. Putra and R. Simanjuntak, "Pemanfaatan MQTT dalam Sistem Supervisi Real-Time Berbasis IoT," J. Komun. dan Jar., vol. 6, no. 1, pp. 30-36, 2025.

A. Shukla, S. P. Shukla, S. T. Chacko, M. K. Mohiddin, and K. A. Fante, "Monitoring of Single-Phase Induction Motor Through IoT Using ESP32 Module," J. Sensors, vol. 2022, p. 8933442, 2022. doi: 10.1155/2022/8933442.

J. Santoso, "Pengembangan SCADA Web-Based untuk Monitoring Pembangkit Hybrid PLTS-PLTB," J. Energi Elektrik, vol. 10, no. 1, pp. 77-83, 2024.

R. Susanto and M. Hadi, "Sistem Supervisi Data Historis Menggunakan MySQL untuk Monitoring Energi Terbarukan," J. Database dan Sist. Cerdas, vol. 11, no. 2, pp. 95-102, 2025.

A. Hutagalung and Y. Siregar, "Desain Dashboard Monitoring Real-Time pada Sistem SCADA Menggunakan React.js dan WebSocket," J. Sist. Inf., vol. 9, no. 4, pp. 201-209, 2025.

O. Navid, P. Siano, and A. Mazza, "Thermal Imaging-Based Fault Detection in Photovoltaic Systems," IEEE Access, vol. 12, pp. 12345-12358, 2024. doi: 10.1109/ACCESS.2024.3350812.

M. Farhan and D. Lestari, "Pengembangan Sistem Monitoring Energi Terbarukan Berbasis IoT untuk Deteksi Dini Gangguan," J. Energi

Terbarukan, vol. 7, no. 1, pp. 45-52, 2025.

S. Khalid, N. Kaur, and V. Nanda, "A Review of Fault Detection Techniques for Grid-Connected Photovoltaic Systems," Renew. Sustain. Energy Rev., vol. 156, p. 111963, 2022. doi: 10.1016/j.rser.2021.111963.

A. Mellit, A. M. Pavan, and V. Lughi, "Short-Term Forecasting of Power Production in a Large-Scale Photovoltaic Plant," Sol. Energy, vol. 105, pp. 401-413, 2014. doi: 10.1016/j.solener.2014.03.018.

V. Patel and J. Mehta, "MQTT-Based Real-Time Monitoring Architecture for Renewable Energy Systems Using Edge Computing," Energies, vol. 17, no. 3, p. 715, 2024. doi: 10.3390/en17030715.

M. Amiri, A. Mehrizi-Sani, and S. Mohammadi, "Deep Learning-Based Fault Detection in Photovoltaic Systems," IEEE Trans. Sustain. Energy, vol. 15, no. 1, pp. 234-245, 2024. doi: 10.1109/TSTE.2024.3012345.

R. Islam, S. Mekhilef, and R. Saidur, "Progress and Recent Trends of Wind Energy Technology," Renew. Sustain. Energy Rev., vol. 21, pp. 456-468, 2013.