Journal of Artificial Intelligence and Engineering Applications

Website: https://ioinformatic.org/

15th February 2026. Vol. 5. No. 2; e-ISSN: 2808-4519

Implementation of IoT and Machine Learning for Monitoring and Prediction of Tank Water Levels

Rizky Wahyudi[1*], Dedy Kiswanto[2], Windy Aulia[3], Selfi Audy Priscilia[4]

[1,2,3,4]*Program Studi Ilmu Komputer, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Negeri Medan*
rizkyw23@mhs.unimed.ac.id[1*], dedykiswanto@unimed.ac.id[2], windy.4231250021@mhs.unimed.ac.id[3], selfiaudy.4233250001@mhs.unimed.ac.id[4]

Abstract

The availability and quality of clean water in household storage tanks are essential yet often overlooked until problems such as depletion or contamination occur. Manual monitoring methods that rely on physical inspection tend to be inefficient, prone to delay, and unable to support predictive decision-making. This study proposes an automated monitoring solution by integrating Internet of Things (IoT) technology with Machine Learning-based analysis. The system is developed using an ESP32 microcontroller that continuously collects real-time data from an ultrasonic sensor to measure water level and a turbidity sensor to assess water clarity. The time-series data obtained is then analyzed using two algorithmic approaches. Linear Regression is employed to model the water depletion rate and generate predictions regarding the estimated remaining duration before the tank reaches an empty state. In parallel, Random Forest is applied as a comparative model to validate prediction accuracy under non-linear consumption patterns. Experimental results demonstrate that the combined IoT–Machine Learning framework provides accurate, timely, and informative insights for users. The proposed system improves water usage efficiency and strengthens early warning capabilities, making it a practical solution for supporting effective household water management.

Keywords: *Internet of Things (IoT), Machine Learning, Water Monitoring, Linear Regression, Random Forest*

1. Introduction

The availability of clean water is a vital necessity in daily life, both for domestic and industrial purposes. Water storage systems in tanks are often used as reserve storage solutions; however, most conventional systems still rely on manual inspection to monitor water level and quality. The absence of a real-time monitoring system poses a risk of sudden tank depletion, which can disrupt user activities. As explained by [1], manual monitoring methods are considered time-inefficient and prone to human error, necessitating Internet of Things (IoT)-based automation.

This problem becomes increasingly complex when users cannot estimate when the tank will be depleted. Previous studies indicate that there are still several significant limitations that remain unaddressed. The study conducted by [1] solely focused on monitoring water conditions such as level and turbidity in aquariums, without providing a prediction model capable of estimating future changes in water conditions. In another research regarding tank monitoring systems by [2], water level data is indeed displayed in real-time via ESP32-based IoT technology; however, the system lacks advanced analysis such as calculating the rate of water volume depletion or predicting the time of water depletion. An intelligent approach in water quality monitoring systems has also been implemented using the IoT-based Fuzzy Mamdani logic method, which is proven capable of providing fast response with high accuracy in determining river water pollution levels in real-time [3].

Meanwhile, research [4] incorporated machine learning-based prediction methods; however, the prediction was limited to irrigation water discharge, utilized the resource-intensive SVM algorithm, and was not designed to project tank depletion time. This is confirmed by [5], stating that the application of linear regression algorithms for water volume prediction is still rarely integrated directly into low-power IoT devices.

Based on these research perspectives, it can be concluded that no study has specifically integrated IoT-based water level and quality monitoring with a tank depletion time prediction model using linear regression. Therefore, this study aims to bridge this gap. [6] in their latest research emphasizes that for time-series data with clear trend patterns, the Linear Regression method often serves as a more efficient and robust benchmark compared to complex Deep Learning models. This forms the basis for the method selection in this study to ensure

the system can operate in real-time with minimal computational resources. To support reproducibility, the complete source code and hardware schematics for this project are openly available at: https://github.com/rizkywahyudiii/water-monitoring.

2. Related Work & System Theory

2.1 IoT Framework and Sensor Instrumentation

The integration of IoT in water management has evolved from simple monitoring to intelligent prediction. Previous studies, such as [1], [2] and [7], successfully implemented basic monitoring for aquariums and tanks but lacked predictive capabilities regarding resource availability. While complex logic like Fuzzy Mamdani [3] and Support Vector Machine (SVM) [4] have been applied, they often require significant computational resources. To address this, this study utilizes the ESP32 microcontroller. As highlighted by [8], ESP32 offers superior dual-core multitasking capabilities for handling sensor acquisition and server transmission simultaneously, supported by cloud integration [9] for scalable time-series analysis.

For level detection, the HC-SR04 ultrasonic sensor is selected as the primary input device, as shown in Fig. 1. Although a low-cost component, research by [10], [11], [12] confirms that with proper perpendicular calibration and signal processing, this sensor achieves industrial-grade accuracy with error margins below 2%.

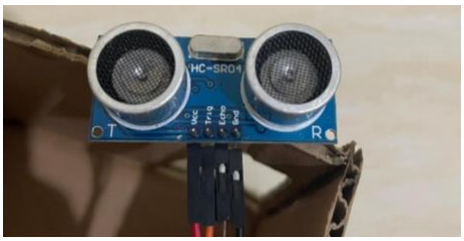

Fig. 1: HC-SR04 Ultrasonic Sensor

For water quality monitoring, the system employs an optical turbidity sensor (Fig. 2). Studies [13], [14] demonstrate that mapping voltage values from this sensor to specific turbidity thresholds (NTU) effectively classifies water clarity. Recent implementations [15] also prove that integrating these sensors with ESP32 provides stable real-time detection of suspended solids.

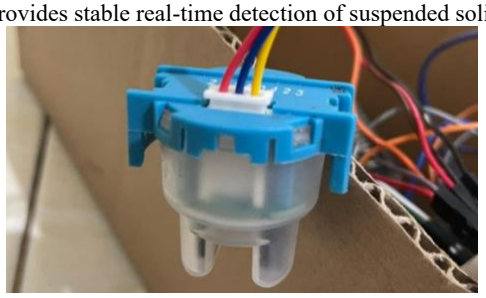

Fig. 2: Turbidity Sensor

To visualize this data locally on the hardware node, SSD1306 OLED displays (Fig. 3) are preferred over LCDs due to their energy efficiency and high contrast without backlight [16].

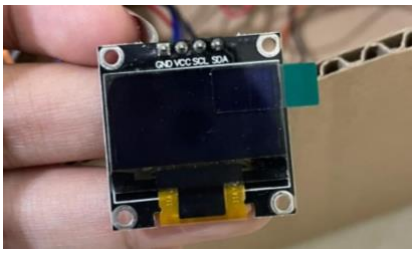

Fig. 3: SSD1306 OLED Display

Additionally, to provide immediate visual warnings without requiring the user to check the app, the system integrates an LED Traffic Light module (Fig. 4). This approach is validated by [17] as an effective method for real-time status alerts in IoT traffic systems.

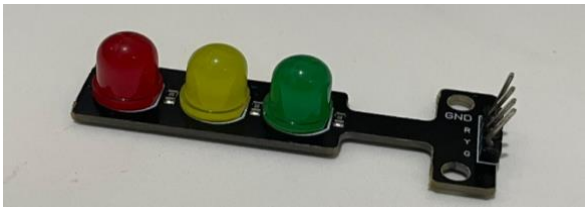

Fig. 4: LED Traffic Light Indicator

2.2 Predictive Modeling and Software Architecture

A key gap in existing research is the lack of lightweight predictive models on edge devices. While [5] noted that regression is rarely integrated into low-power IoT, recent findings by [6], [18] argue that for clear linear trends (such as constant tank draining), Linear Regression outperforms complex Deep Learning models like LSTM in terms of speed and efficiency. Consequently, this study adopts Linear Regression as the primary engine.

However, to mitigate the limitations of linear models in handling anomalies, Random Forest is employed as a validator. Research [19], [20] indicate that ensemble methods like Random Forest are robust against sensor noise and non-linear patterns. To maximize model performance, Feature Engineering is applied. As emphasized by [21], [22], transforming raw telemetry data (scaling and outlier removal) significantly enhances prediction stability compared to using raw sensor outputs.

On the software side, the system is built upon the Laravel framework. According to [23], modern web architectures requiring secure data transactions benefit significantly from Laravel's dependency management and security features (Zero Trust principles), ensuring that the high-frequency data transmission from the IoT nodes remains secure and consistent.

3. Research Method

3.1. Research Design

This study applies the Research and Development (R&D) method with an integrated system engineering approach. This method was chosen because the main objective of the research is not merely to test hypotheses, but to produce a tangible product in the form of a Smart Water Monitoring system that combines hardware (Internet of Things) and intelligent software (Machine Learning). The research procedure is adapted from the system development model which includes analysis, design, implementation, integration, and testing stages.

Fig. 5: Research Workflow

1) Requirements Analysis and Literature Study: The initial stage focuses on identifying problems related to the inefficiency of conventional water monitoring systems. A literature study was conducted to deepen the understanding of IoT communication protocols, Laravel-based web server architecture, as well as time-series prediction algorithms such as Linear Regression and Random Forest.
2) System Design: This stage encompasses the comprehensive designof the system architecture consisting of three main segments: a) Edge Segment: Schematic design of ESP32-based hardware and sensors. b) Server Segment: Design of database and back-end architecture using Laravel 12. c) Intelligence Segment: Logic flow design of Python services for data processing and automated prediction.
3) Implementation and Prototyping (Development): This stage realizes the system design into a tangible product. Activities include hardware assembly, API development on the server side, as well as the implementation of hybrid Machine Learning algorithms for real-time sensor data processing and model retraining.
4) System Integration: This stage connects all system components so that sensor data can be transmitted via REST API, stored in the database, and subsequently processed by the Python service running as a background service.
5) Testing and Evaluation: The final stage is conducted to validate system functionality and prediction model accuracy. Testing includes web dashboard functional tests, IoT connectivity tests, and prediction algorithm performance evaluation using MAE and RMSE metrics.

3.2. System Architecture

The system is designed using a centralized IoT architecture (server-centric architecture) that separates data acquisition, intelligent processing, and user visualization functions. Unlike conventional systems relying on third-party platforms (cloud-managed), this research develops a self-hosted server to ensure data management flexibility and Machine Learning model integration. Functionally, the system architecture is divided into three main layers (Three-Layer Architecture):

To avoid confusion, the family name must be written as the last part of each author name (e.g. John A.K. Smith).

Each affiliation must include, at the very least, the name of the company and the name of the country where the author is based (e.g. Causal Productions Pty Ltd, Australia). Email address is compulsory for the corresponding author.

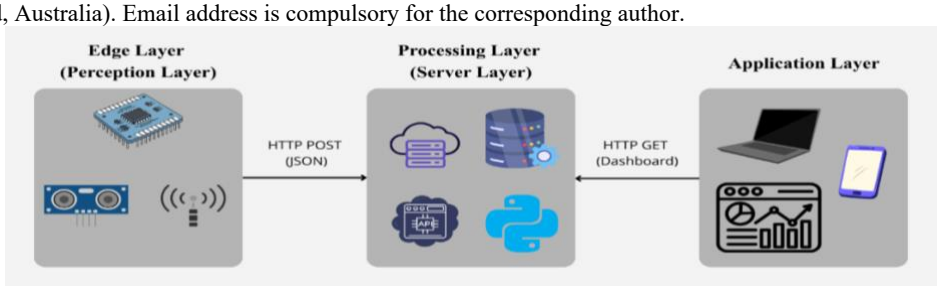

Fig. 6: System Architecture

a) Perception Layer (Edge Layer): This layer consists of an ESP32 microcontroller connected to an HC-SR04 ultrasonic sensor and a turbidity sensor. The main function of this layer is to acquire physical water data and transmit it to the server via the HTTP REST API protocol. No heavy processing is performed on this side to maintain hardware power efficiency.
b) Server and Processing Layer (Processing Layer): This layer is the core of the system hosted in a server environment using aaPanel. This layer has three vital components: a) Web Server (Laravel 12): Functions as the API endpoint provider (SensorController) to receive incoming data from ESP32, as well as managing application business logic. b) Database Server (MySQL): Functions as a centralized storage medium for raw sensor data, activity logs, and prediction results. c) AI Automation Service (Python Daemon): A background service managed by the Process Control System (Supervisor). This service runs the Hybrid Machine Learning algorithm continuously to monitor new data trends in the database, perform Linear Regression predictions, and trigger automatic Random Forest model retraining.
c) Application Layer: This layer is a web-based interface built using Laravel. Users can monitor water status in real-time, view tank depletion time estimates, and access data history via a responsive dashboard.

3.3. Software Design

Software design in this system is focused on developing a web-based interface and relational database management capable of handling high-speed IoT data transactions. Software implementation is divided into three main technical aspects: Back-End development, database design, and server environment configuration.

a) Framework and API Development: The development of this web-based system utilizes the Laravel 12 framework built upon the Model-View-Controller (MVC) architecture concept. The selection of this framework is based on its efficient dependency management capabilities and modern security features [23]. In this structure, the SensorController holds a vital role as the API (Application Programming Interface) gateway. This component is tasked with validating the JSON data format received via HTTP POST requests from the ESP32 device before saving it into the database.
b) Server Environment Management: To ensure high service availability (uptime), the system is hosted using the aaPanel server management panel. One crucial feature implemented is the use of Supervisor as the Process Control System. Supervisor is configured to execute the Python script (ml_auto_service.py) as a daemon process. This ensures that the Machine Learning algorithm continues to run in the background, monitors incoming data in real-time, and automatically restarts in the event of system failure, thereby maintaining prediction continuity without manual intervention.
c) Data storage utilizes the MySQL database management system. The database structure is designed to accommodate the needs of recording historical sensor data and model evaluation results in a multi-user environment. Based on the requirements analysis, there are four main interconnected table entities. The users table serves as the parent entity storing authentication data while also marking data ownership for each user. The sensor_data table functions to hold raw telemetry data such as turbidity and water levels, directly linked to the associated user ID (user_id).
Furthermore, the predictions table stores the computation results of the Linear Regression algorithm, covering tank depletion time estimation and R^2 accuracy, which has a one-to-one relationship with sensor data. Finally, the model_evaluations table records the performance track of the Random Forest model (such as MAE and RMSE) conducted by a specific user during the retraining process. This relational structure ensures data integrity between users remains maintained, organized, and ready to support advanced analysis. Below is the definition of the relationship structure between tables (ERD) used in the system:

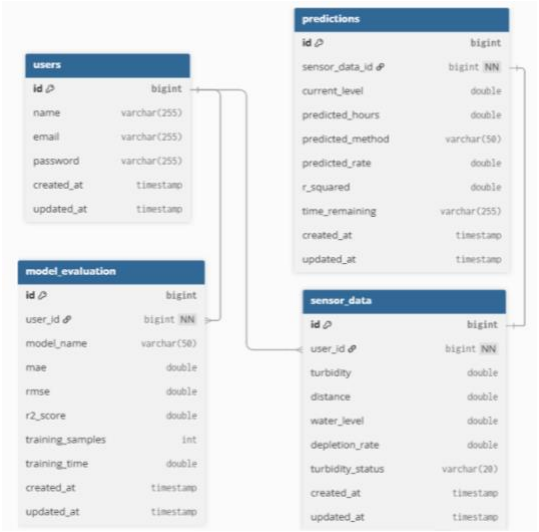

Fig. 7: System Database ERD

3.4 Implementation of Hybrid Machine Learning Model

To address the limitations of conventional prediction methods, which are often inaccurate in handling dynamic water fluctuations, this study develops a Hybrid Machine Learning algorithm. This algorithm runs automatically on the server side using a Python service. The hybrid approach combines two different models to handle two prediction aspects:

3.4.1 Real-Time Trend Prediction (Linear Regression with Rolling Window)

This is used to provide instant tank depletion time estimation when new data is received from sensors. Since the water surface is often undulating (noise), the system applies Pre-processing and Windowing techniques as follows:

a) Noise Reduction Filter: Water height data from the ultrasonic sensor often experiences minor fluctuations due to water ripples. The system applies the Moving Average algorithm with a window size of the last 5 data points to obtain a more stable trend value before processing.
b) Sliding Window: The system retrieves the last 60 data points (window size) from the database to form a regression line.
c) Linear Regression Calculation: The algorithm calculates the slope of the line from the data within that window.

The water level change slope value is utilized as the primary reference in determining the tank condition. A significant negative slope indicates a decrease in water volume; thus, the system calculates the estimated depletion time based on the ratio between the current water level and its depletion rate. If the slope is positive, the system recognizes that the tank is in the filling process, while a slope approaching zero indicates a relatively stable water level condition. This approach allows the system to provide tank condition monitoring that is more adaptive and precise.

3.4.2 Pattern Recognition and Evaluation (Random Forest Regressor)

The second model, Random Forest, works as a system performance validator in the long term. Unlike linear regression, which only views straight lines, Random Forest is trained using time features (hours, minutes) and turbidity to learn more complex (non-linear) water usage patterns. This process operates with an Auto-Retraining mechanism:

1) The system monitors the amount of new data entering the database.
2) Whenever new data reaches a certain threshold (e.g., 10-100 data points), the Python service will trigger the Random Forest model retraining.
3) Model evaluation results (MAE, RMSE, and R^2 Score) are saved to the model_evaluation table. This data serves as an indicator for system administrators to determine whether water usage patterns have drastically changed or if sensors are experiencing reduced accuracy.

3.4.3 Processing Pipeline

All processes above are executed in an automated cycle managed by a Python script (ml_auto_service.py) with the workflow:

Fig. 8: ML Workflow in Python Script

1) Fetch: Retrieve the latest sensor data from MySQL.
2) Process: Calculate linear regression trend on the last 60 data points.
3) Predict: Determine status (Filling/Draining) and estimated time.
4) Store: Save prediction results to the predictions table.

5) Evaluate: Check data count to trigger Random Forest retraining if necessary.

3.5 System Testing and Evaluation

The final stage of the methodology is system testing to validate whether the built solution meets functional and non-functional requirement specifications. Testing is conducted through three main testing scenarios:

3.5.1 Functional Testing (Black Box Testing)

System testing is conducted using the black-box testing approach, a method focusing on checking software logic functions without regarding the internal code structure. Testing is applied to the Laravel-based Web Dashboard interface to ensure all features run according to needs. Aspects tested include the authentication process, namely administrator login mechanism validation and page access protection via route protection.

Furthermore, API endpoint testing is performed to verify that sensor data sent by the ESP32 via HTTP POST is correctly received in JSON format and stored according to the structure in the MySQL database. Additionally, data visualization features are also tested to ensure graphs and status indicators on the dashboard display real-time changes following the latest data updates. This approach ensures the system functions consistently from the user perspective and is ready for use in an operational environment.

3.5.2 Model Performance Evaluation

To assess the accuracy level of the Hybrid Machine Learning algorithm in predicting tank depletion time, the system applies statistical evaluation methods processed automatically via the Python service. Three main parameters are used in model performance measurement. First, Mean Absolute Error (MAE) is used to calculate the average absolute difference between the predicted depletion time and the actual depletion time, thereby providing a direct overview of the prediction error magnitude. Second, Root Mean Squared Error (RMSE) is used to measure prediction error by penalizing larger errors more heavily, making this metric sensitive to deviations far from the true value. Third, R-Squared (R^2 Score) serves as an indicator of how well the sensor data can be explained by the regression model with a value range of 0 to 1, where a higher value indicates better model fit. All evaluation results are automatically recorded into the model_evaluations table every time the retraining process is completed, allowing model performance to be monitored continuously.

3.5.3 Hardware Integration Test

Hardware integration testing is conducted to ensure that all sensor components work accurately and consistently when connected to the microcontroller system. This process includes calibrating the HC-SR04 distance sensor by comparing its reading results against manual measurements using a measuring ruler under several water height conditions: 0%, 50%, and 100%. This testing aims to validate the sensor's precision in detecting water level changes. Additionally, turbidity sensor validation is performed by measuring voltage response in both clear and turbid water conditions. Test results are used to determine threshold values representing water quality status accurately. Thus, all sensors are ensured to work according to specifications before being used in the IoT-based water monitoring system.

4. Results and Discussion

4.2 System Implementation

The development of the Smart Water Monitoring system has been successfully implemented end-to-end, integrating ESP32-based hardware with a self-hosted web-based interface system. Unlike the initial design which planned to use a third-party platform (Blynk), the final implementation is focused entirely on developing a dedicated web dashboard to ensure flexibility in data management and visualization of more complex predictions.

4.2.1 Hardware Implementation

The hardware is packaged in the form of a prototype installed on a miniature water tank with a height of 14 cm. The sensor node consisting of the ESP32, HC-SR04 ultrasonic sensor, and turbidity sensor is placed in a static position at the top of the container to ensure data reading stability.

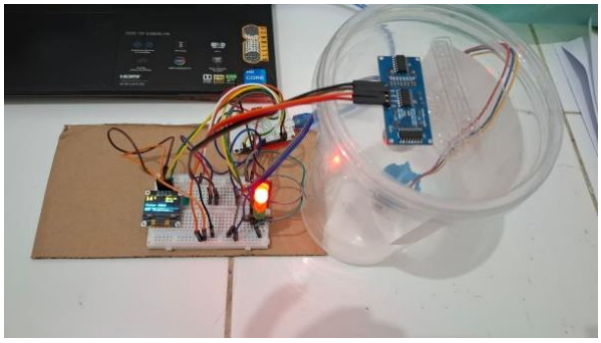

Fig. 9: Hardware Implementation of Sensor Node on Tank Prototype

4.2.2 Web Interface Implementation

The user interface was developed using the Laravel 12 framework, presenting telemetry data in real-time. The main dashboard page (Figure 10) is designed to provide comprehensive water status information to the user. Key features successfully implemented on the dashboard include:

a) Water Level Indicator & Prediction: Displays the water height percentage and estimated time remaining until the tank is empty based on Linear Regression algorithm calculations.
b) 3-State Water Quality Monitoring: The system is capable of classifying and displaying water clarity status into three visual categories based on sensor voltage: "Clear," "Slightly Turbid," and "Turbid." This classification provides more specific information compared to a mere binary status (clean/dirty).
c) Real-time Trend Graph: Visualizes the movement of water level depletion and the rate of change per hour, allowing users to monitor water consumption patterns directly.

4.2.3 Deployment Infrastructure (Cloud Server)

Unlike the initial development phase which generally ran in a local environment (localhost), this system has been successfully deployed to a production environment using a Cloud Virtual Private Server (VPS). Server management is conducted using the aaPanel control panel configured with a shared hosting scheme, allowing for optimal application resource isolation.

This architecture enables the system to be accessed globally via the public internet and ensures the background service (Python Daemon) can run 24 hours non-stop without relying on a local computer device that must remain on. The use of a VPS also proves system stability in handling continuous HTTP requests from IoT devices via the real internet network.

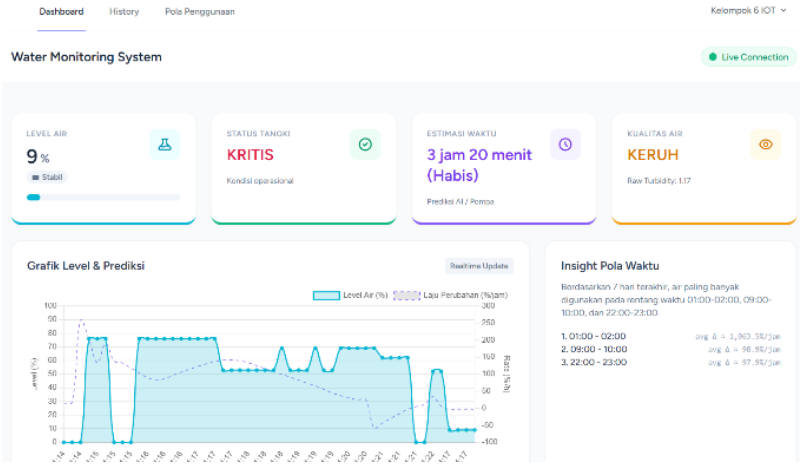

Fig. 10: Real-Time Monitoring Dashboard Display Showing Water Level and Quality Status

4.3 Hardware and Functional Testing Results

This testing aims to validate sensor reading accuracy and data transmission reliability to the Cloud VPS server. Testing is conducted in two stages: sensor calibration and system connectivity testing.

4.3.1 Ultrasonic Sensor Calibration

Testing of the HC-SR04 distance sensor was performed by comparing sensor reading values (measured value) against manual measurements using a measuring ruler (actual value) on a tank prototype with a maximum height of 14 cm. Testing was conducted at various water height points to observe reading consistency.

Table 1. Ultrasonic Sensor Calibration Results Against Manual Measurement

No	Actual Distance (cm)	Sensor Reading (cm)	Error (%)	Status
1	2.0	2.04	2.0%	Valid
2	5.0	4.96	0.8%	Valid
3	10.0	10.05	0.5%	Valid
4	14.0	13.98	0.41%	Valid
Average Error			**0.86%**	

The results in Table 1 show that the sensor has a very low average error rate of 0.86%. This level of accuracy is highly adequate to be used as the main input variable in volume and water depletion rate calculations by the Linear Regression algorithm.

4.3.2 Turbidity Sensor Logic Calibration

Based on initial testing, the analog turbidity sensor used possesses high sensitivity characteristics, where the resulting voltage change is relatively small between clean water and dirty water. To address this hardware limitation, the study applies a software calibration approach by establishing multi-level thresholding logic.

The system no longer uses a single limit value but rather a range of values to classify water conditions into three statuses. The implementation of this calibration logic is presented in Table 2.

Table 2. Logic for Determining Water Quality Status Based on Voltage

Voltage Range (ADC Value)	Status Classification	Condition Description
>1800	CLEAR	Clean water, suitable for use.
1751-1800	SLIGHTLY TURBID	Light particles detected, monitoring needed.
≤ 1750	TURBID	High particle content, unsuitable for use.

By applying this logic on the server side, the system is capable of translating sensor voltage fluctuations into stable status information that is easily understood by users on the dashboard.

4.3.3 Connectivity and Cloud Integration Test

Connectivity and REST API integration testing indicate that two-way communication between the ESP32 and the server runs stably via the HTTP protocol. The device sends sensor data at intervals of approximately 10 seconds, consisting of turbidity, distance, and water_level values in JSON format. As seen in Figure 11, all requests sent successfully received a response from the server with HTTP status code 200. The JSON response—including predicted_hours and time—was processed well by the ESP32 and displayed back on the Serial Monitor, confirming that the prediction computation process has been fully offloaded to the server.

Fig. 11: Screenshot of REST API communication on ESP32 Serial Monitor

Furthermore, proof of data recording in Figure 12 shows that every data packet sent by the device was successfully stored into the MySQL database table sequentially and without data loss. Recorded latency was in the range of 2–5 seconds, which is still within tolerance limits for a water monitoring system that does not require hard real-time response. These results reinforce that the integration between the IoT device, server API, and database runs well, stably, and is ready for use in water monitoring system implementation.

	id	turbidity	distance	water_level	depletion_rate	turbidity_status	timestamp	created_at	updated_at
Edit Copy Delete	1328	1.22	12.5	11	0	KERUH	2025-12-07 14:57:38	2025-12-07 14:57:38	2025-12-07 14:57:38
Edit Copy Delete	1327	1.2	12.5	11	0	KERUH	2025-12-07 14:57:29	2025-12-07 14:57:29	2025-12-07 14:57:29
Edit Copy Delete	1326	1.28	12.5	11	0	KERUH	2025-12-07 14:57:19	2025-12-07 14:57:19	2025-12-07 14:57:19
Edit Copy Delete	1325	1.2	12.5	11	0	KERUH	2025-12-07 14:57:09	2025-12-07 14:57:09	2025-12-07 14:57:09
Edit Copy Delete	1324	1.27	12.5	11	0	KERUH	2025-12-07 14:56:59	2025-12-07 14:56:59	2025-12-07 14:56:59
Edit Copy Delete	1323	1.26	12.5	11	0	KERUH	2025-12-07 14:56:49	2025-12-07 14:56:49	2025-12-07 14:56:49
Edit Copy Delete	1322	1.22	12.5	11	0	KERUH	2025-12-07 14:56:39	2025-12-07 14:56:39	2025-12-07 14:56:39
Edit Copy Delete	1321	1.12	12.5	11	0	KERUH	2025-12-07 14:56:28	2025-12-07 14:56:28	2025-12-07 14:56:28
Edit Copy Delete	1320	1.27	12.5	11	0	KERUH	2025-12-07 14:56:18	2025-12-07 14:56:18	2025-12-07 14:56:18
Edit Copy Delete	1319	1.26	12.5	11	0	KERUH	2025-12-07 14:56:08	2025-12-07 14:56:08	2025-12-07 14:56:08
Edit Copy Delete	1318	1.26	12.5	11	0	KERUH	2025-12-07 14:55:58	2025-12-07 14:55:58	2025-12-07 14:55:58
Edit Copy Delete	1317	1.26	12.5	11	0	KERUH	2025-12-07 14:55:48	2025-12-07 14:55:48	2025-12-07 14:55:48
Edit Copy Delete	1316	1.31	12.4	11	0	KERUH	2025-12-07 14:55:39	2025-12-07 14:55:39	2025-12-07 14:55:39
Edit Copy Delete	1315	1.32	12.4	11	0	KERUH	2025-12-07 14:55:29	2025-12-07 14:55:29	2025-12-07 14:55:29

Fig. 12: Proof of sensor data storage in MySQL database table

4.3.4 Functional Testing (Black Box)

System testing was conducted using the black-box approach to validate that the Web Dashboard and API endpoints function according to the requirements without inspecting the internal code structure. The testing scenarios and results are presented in Table 3.

Table 3. Black Box Testing Results

No	Test Scenario	Test Case / Action	Expected Result	Actual Result	Status
1	User Authentication	Access login page and input valid admin credentials.	System validates credentials and redirects user to the main Dashboard.	Successful redirection to Dashboard.	Valid
2	API Data Ingestion	ESP32 sends a JSON payload (turbidity, distance, water_level) via HTTP POST.	Server returns HTTP 200 OK and data is stored in the MySQL database.	Data recorded in database and HTTP 200 received.	Valid
3	Real-Time Visualization	Observe dashboard charts when new sensor data enters the database.	The Level and Quality charts update automatically without page refresh.	Charts updated in real-time (approx. 2-5s delay).	Valid
4	Status Logic (Turbidity)	Simulate sensor voltage input ≤ 1750 (High Turbidity).	Dashboard status displays "KERUH" (Turbid) and indicator turns red.	Status displayed as "KERUH".	Valid
5	Status Logic (Level)	Simulate water level input $< 20\%$ capacity.	Dashboard status displays "KRITIS" (Critical) and "Filling Needed".	Status displayed as "KRITIS".	Valid
6	Prediction Display	Provide time-series data with a negative slope (draining trend).	System calculates and displays "Time Remaining" in Hour:Minute format.	Prediction time displayed correctly based on slope.	Valid

4.4 Machine Learning Model Performance Evaluation

Model performance evaluation aims to measure the accuracy level and computational efficiency of the applied algorithms. The system is designed with a Hybrid mechanism, where Linear Regression is used as the primary prediction engine for fast response, while Random Forest is run periodically as a comparator (validator) to detect complex non-linear patterns.

Evaluation data was taken from automated training (auto-training) logs in the database with a sample of 1,106 data points. Metrics used include Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), R-Squared (R^2), and training time.

Table 4: Performance Comparison of Linear Regression vs Random Forest Models

Prediction Model	MAE (Hour)	RMSE	R^2 Score	Training Time (seconds)
Linear Regression	1.06	1.76	0.9928	0.009
Random Forest	0.60	1.40	0.9954	0.209

Source: Backend System Log (Timestamp: 2025-12-04)

Based on Table 3, both models show excellent prediction performance with R^2 values above 0.99, indicating that more than 99% of water level depletion variation can be explained by the models. However, Random Forest provides more accurate results with a MAE of 0.60 hours, significantly lower than Linear Regression which recorded a MAE of 1.06 hours. This means the average prediction error of Random Forest is only about 36 minutes, whereas Linear Regression misses by about 1 hour. This advantage arises because Random Forest is capable of capturing non-linear patterns and small fluctuations that cannot be handled by the linear regression model. On the other hand, Linear Regression remains superior in terms of computational efficiency, with a training time of only 0.009 seconds, which is approximately 23 times faster than Random Forest. These results indicate a clear trade-off between accuracy and computational speed.

4.5 Discussion

The selection of the prediction model demonstrates a clear trade-off between accuracy and computational speed. Linear Regression was chosen as the primary model because it offers significantly faster training time, making it more suitable for an IoT system that exchanges data every 10 seconds. Although its accuracy is slightly lower compared to Random Forest (a MAE difference of 0.46 hours), this deviation is still within tolerance limits for tank condition early warning needs. Thus, priority is given to server processing efficiency without significantly sacrificing accuracy.

The reliability of the system architecture became increasingly apparent after migration from the local environment to the Cloud VPS, which provided better stability and scalability. The applied Edge-to-Cloud architecture allows the ESP32 device, which has limited processing capacity, to still display prediction results in real-time through a server-side processing mechanism. Two-way communication via REST API ensures the device remains lightweight, while the server handles the predictive computational load consistently and efficiently.

Sensor anomaly handling is also an important aspect successfully addressed through the application of multi-level thresholding and sensor reading filtering. The turbidity sensor, which tends to produce fluctuating values, can be stabilized using clear classification thresholds, so that water quality status remains consistent even though the sensor is a low-cost device. Similarly, noise filtering on the ultrasonic sensor improves distance reading accuracy, enabling the system to provide reliable water condition information to the user.

5. Conclusion

This study has successfully designed an IoT-based Smart Water Monitoring system integrated with Cloud VPS and a Laravel web interface, thereby capable of replacing manual monitoring through the presentation of real-time water level and quality data. The implementation of the Linear Regression algorithm as the primary prediction model is effective in estimating tank depletion time through water depletion rate modeling, with excellent performance (R^2 0.9928; MAE 1.06 hours). Although Random Forest recorded a lower MAE (0.60 hours), Linear Regression was selected due to significantly higher computational efficiency and an execution time that is 23 times faster, thus being more suitable for the responsiveness needs of an early warning system. This study still has limitations regarding the assumption that the tank possesses a uniform cross-section; the model has not accounted for volume correction for tanks with irregular shapes, so the relationship between water height and volume has not been fully accommodated. Future work aims to integrate geometric volume correction algorithms to handle irregular tank shapes and explore the implementation of TinyML to run more complex models directly on edge devices.

Acknowledgement

The authors would like to thank the Department of Computer Science, Universitas Negeri Medan, for providing the facilities and support required to conduct this research. We also express our gratitude to Mr. Dedy Kiswanto, S.Kom., M.Kom. for his valuable guidance and constructive suggestions during the development of this system.

References

[1] I. B. Prasetiyo, A. A. Riadi, and A. A. Chamid, "Perancangan Smart Aquarium Menggunakan Sensor Turbidity Dan Sensor Ultrasonik Pada Akuarium Ikan Air Tawar Berbasis Arduino Uno," *J. Teknol. Univ. Muhammadiyah Jakarta*, vol. 13, no. 2, pp. 193–200, 2021, doi: 10.24853/jurtek.13.2.193-200.

[2] H. Darmanto, L. Lamsadi, and H. Asrul, "Monitoring Ketinggian Air Tandon Berbasis IoT Dengan ESP32 Melalui Website," *JUSTER J. Sains dan Terap.*, vol. 4, no. 2, 2025, doi: 10.57218/juster.v4i2.1507.

[3] M. N. Novian, M. Walid, and M. Makruf, "Implementasi Logika Fuzzy untuk Monitoring Tingkat Pencemaran Air Sungai berbasis Internet Of Things," *J. Sist. Komput. dan Kecerdasan Buatan*, vol. 7, no. 3, pp. 248–289, 2024, doi: 10.47970/siskom-kb.v7i3.687.

[4] T. A. Cinderatama, R. Zulmy Alhamri, Y. Yunhasnawa, F. Sofian Efendi, and R. Ariyanto, "Sistem Monitoring Irigasi dan Prediksi Debit Air Berbasis IoT Dan Support Vector Machine (SVM)," *J. Inform. Polinema*, vol. 11, no. 2 SE-Articles, pp. 171–184, Feb. 2025, doi: 10.33795/jip.v11i2.6812.

[5] Nurdin, N. Suarna, and W. Prihartono, "ALGORITMA REGRESI LINIER SEDERHANA UNTUK PREDIKSI PENGGUNAAN VOLUME AIR BERDASARKAN JENIS PELANGGAN PDAM," *J. Kecerdasan Buatan dan Teknol. Inf.*, vol. 4, no. 1 SE-Articles, pp. 43–52, Jan. 2025, doi: 10.69916/jkbti.v4i1.187.

[6] W. Nie *et al.*, "Rethinking deep learning: linear regression remains a key benchmark in predicting terrestrial water storage," 2025. doi: 10.48550/arXiv.2510.10799.
[7] L. V. Q. Danh, D. V. M. Dung, T. H. Danh, and N. C. Ngôn, "Design and Deployment of an IoT-Based Water Quality Monitoring System for Aquaculture in Mekong Delta," 2020. doi: 10.18178/ijmerr.9.8.1170-1175.
[8] R. G. Riyansyah, D. Wahiddin, and D. S. Kusumaningrum, "Smart Monitoring Alat Infus Pasien Berbasis Internet Of Things (IoT) Menggunakan Mikrokontroler ESP32," *Sci. Student J. Information, Technol. Sci.*, vol. 2, no. 1, pp. 142–148, 2021.
[9] M. Ansari and M. Alam, "An Intelligent IoT-Cloud-Based Air Pollution Forecasting Model Using Univariate Time-Series Analysis.," *Arab. J. Sci. Eng.*, pp. 1–28, May 2023, doi: 10.1007/s13369-023-07876-9.
[10] I. M. S. Wibawa and I. K. Putra, "Design and Manufacture of Air Quality Measurements Based on Arduino ATmega 2560 Using Dust ZH03A Laser Sensor," *Int. J. Phys. Sci. Eng.*, vol. 5, no. 1, pp. 8–15, 2021, doi: 10.29332/ijpse.v5n1.800.
[11] T. S. Pereira, T. P. de Carvalho, T. A. Mendes, and K. T. Formiga, "Evaluation of Water Level in Flowing Channels Using Ultrasonic Sensors," 2022. doi: 10.3390/su14095512.
[12] W. Gao, W. Liu, F. Li, and Y. Hu, "Analysis and Validation of Ultrasonic Probes in Liquid Level Monitoring Systems," 2021. doi: 10.3390/s21041320.
[13] M. A. Delwizar, A. Arsenly, H. Irawan, M. Jodiansyah, and R. M. Utomo, "Perancangan Prototipe Sistem Monitoring Kejernihan Air Dengan Sensor Turbidity Pada Tandon Berbasis IoT," *J. Teknol. Elektro*, vol. 12, no. 3, Oct. 2021, doi: 10.22441/jte.2021.v12i3.002.
[14] T. D. Hendrawati and R. M. Hibban, "Pengembangan Sistem IoT untuk Pemantauan Kualitas Air Kolam Koi Berbasis Sensor," *JTERA (Jurnal Teknol. Rekayasa)*, vol. 9, no. 2, pp. 113–120, 2024, doi: 10.31544/jtera.v9.i1.2024.113-120.
[15] M. Nurul and D. Maizana, "Rancang Bangun Sistem Monitoring Kekeruhan Air Pada Mesin Rotary Drum Filter Berbasis Sensor TSS Dan Mikrokontroler ESP32," *Riau J. Tek. Inform.*, vol. 4, no. 1 SE-Articles, pp. 138–141, Mar. 2025, doi: 10.30606/rjti.v4i1.3305.
[16] S. M. Safrilly and R. Badarudin, "Sistem Monitoring Suhu secara Real-Time berbasis Arduino Uno untuk Pemantauan Lingkungan," *ELECTROPS J. Ilm. Tek. Elektro*, vol. 3, no. 2, pp. 1–7, 2024, doi: 10.30872/electrops.v3i1.15789.
[17] A. I. Asry, L. Lutfi, A. Umar, and R. Ahmad, "Monitoring System for Traffic Light Lamp Damage using BLYNK Application Based on IOT ESP32," *J. Teknol. Transp. dan Logistik*, vol. 5, no. 1 SE-, pp. 59–66, Jun. 2024, doi: 10.52920/jttl.v5i1.230.
[18] A. C. P Fernandes, A. R Fonseca, F. A. L. Pacheco, and L. F. Sanches Fernandes, "Water quality predictions through linear regression - A brute force algorithm approach.," *MethodsX*, vol. 10, p. 102153, 2023, doi: 10.1016/j.mex.2023.102153.
[19] J. Xu *et al.*, "Assessing and predicting water quality index with key water parameters by machine learning models in coastal cities, China," *Heliyon*, vol. 10, no. 13, Jul. 2024, doi: 10.1016/j.heliyon.2024.e33695.
[20] Y.-C. Liang, Y. Maimury, A. H. Chen, and J. R. Juarez, "Machine Learning-Based Prediction of Air Quality," 2020. doi: 10.3390/app10249151.
[21] J. V. Filho, A. Scortegagna, A. P. de S. D. Vieira, and P. A. Jaskowiak, "Machine learning for water demand forecasting: Case study in a Brazilian coastal city," *Water Pract. Technol.*, vol. 19, no. 5, pp. 1586–1602, Apr. 2024, doi: 10.2166/wpt.2024.096.
[22] L. Santoso and P. Priyadi, "Comparative Study of Feature Engineering Techniques for Predictive Data Analytics," *J. Technol. Informatics Eng.*, vol. 3, no. 2 SE-Articles, pp. 417–435, 2024, doi: 10.51903/jtie.v3i2.225.
[23] O. Alvansyah, D. Kiswanto, A. N. Nasution, and M. F. Zulfi, "IMPLEMENTASI ZERO TRUST ARCHITECTURE DENGAN MULTI-FACTOR AUTHENTICATION DAN CONTINUOUS VERIFICATION PADA SISTEM LOGIN," *J. Manaj. Inform. Sist. Inf. dan Teknol. Komput.*, vol. 4, no. 2, pp. 617–623, 2025, doi: 10.70247/jumistik.v4i2.220.