Jurnal Riset Rumpun Ilmu Teknik
Volume. 4 Nomor. 3 Desember 2025
e-ISSN : 2829-016X; p-ISSN : 2829-0178, Hal. 597-616
DOI: https://doi.org/10.55606/jurritek.v4i3.6828
Tersedia: https://prin.or.id/index.php/JURRITEK

Implementation of IoT in Water Quality Monitoring and Management System for Koi Fish Ponds Based on Web with RESTful API

Ida Bagus Gede Baskara[1*], Made Adi Paramartha Putra[2], Putu Trisna Hady Permana S[3], I Nyoman Yudi Anggara Wijaya[4], I Gede Juliana Eka Putra[5]
[1-5]Program Studi Informatika, Fakultas Teknologi Informasi dan Desain, Primakara University
[*]Author's correspondence: *gusbim226@gmail.com*[1]

Abstract: *Technological advancements, particularly in microcontrollers and the Internet of Things (IoT), have brought significant changes to various aspects of life. Microcontrollers such as the ESP32, DS18B20 sensor, water pH sensor, and other supporting sensors enable the development of efficient automation systems. This research addresses issues related to the design and development of IoT devices for water quality monitoring, as well as the integration of these devices with web-based systems. The objective is to develop a device capable of monitoring water quality parameters such as pH, temperature, turbidity, dissolved materials, salinity projection, and dissolved oxygen projection, while providing automatic notifications under certain conditions. The development process applies the Agile methodology, which allows for rapid and adaptive iterations. Theoretically, this study contributes to improving understanding of IoT implementation and information systems, while practically, it provides a technological solution to support Koi fish farmers in monitoring and managing pond water quality. The expected outcome is a web-based water quality monitoring and management device for Koi ponds that can deliver real-time information and condition notifications, thereby reducing aquaculture risks and improving productivity.*

Keywords: *Internet of Things (IoT); Koi Aquaculture; RESTful API; Water Quality Monitoring; Web-Based Information System*

1. INTRODUCTION

Technological advancements have significantly transformed various aspects of human life. Many organizations leverage technologies such as websites, microcontrollers, and other digital solutions to increase the efficiency and effectiveness of their activities. A microcontroller is a compact computer system consisting of a processor, memory, and input-output components (Chamim, 2010; Yanel, 2023). One of the most promising innovations in microcontroller development is the Internet of Things (IoT), which enables communication among devices and sensors through local networks or the internet to collect and share environmental data (Febrianti, Wibowo, & Vendyansyah, 2021). Current IoT implementations utilize devices such as the ESP32, DS18B20 sensors, water pH sensors, Total Dissolved Solids (TDS) sensors, servo, relay, and LCD, each serving as a controller, data collector, visualizer, or peripheral device.

Data obtained from IoT devices can be further optimized using information systems for visualization, either through mobile applications or websites. Websites serve as accessible information systems widely used across sectors such as business, education, politics, and others. They consist of interconnected text pages accessible via URLs (Dermawan, Wagiyati, Budilaksono, & Suwarno, 2018; Sanjaya & Hesinto, 2017). Before IoT data are visualized on

Naskah Masuk: 25 Agustus 2025; Revisi: 08 September 2025; Diterima: 23 September 2025; Tersedia: 30 September 2025

websites, they are stored in a real-time database such as Firebase Realtime Database (FRD), which supports multi-platform synchronization with integrated clients (Firman Maulana, 2020; Umami & Ningrum, 2020).

Integration between IoT devices and web-based information systems can be achieved using communication methods such as MQTT, GraphQL, and RESTful API. RESTful API allows communication across different systems using HTTP requests to access resources (Paramitha, Made Wiharta, & Suyadnya, 2022), while MQTT is a lightweight machine-to-machine connectivity protocol (Susanto, Atmadji, & Brenkman, 2018). Additional functionalities such as notifications can be implemented using If This Then That (IFTTT), which automates actions based on predefined conditions (Ningtyas, 2020).

In the context of Industry 4.0, IoT can automate tasks to enhance operational efficiency and productivity (Pandawana, Partama, Puspitawati, Dwipa, & Sari, 2023; Sitorus, Gifson, Mangapul, & Aziz, 2020). In Indonesia, one promising sector is ornamental fish aquaculture, particularly Koi fish (Cyprinus rubrofuscus), which is economically valuable and popular among hobbyists (Echo, 2022; Santanumurti, 2021). Koi farming is influenced by water quality parameters, including pH, temperature, and dissolved materials. Safe conditions for Koi farming are pH 6.5–8.0, temperature 24–28 °C, and total dissolved materials below 150 ppm (Ariyanto & Kusriyanto, 2023).

Monitoring pond water quality conventionally is time-consuming and may lead to delayed interventions, risking fish health and mortality. To address these challenges, this study proposes the development of an IoT-based water quality monitoring and management system for Koi ponds integrated with a web-based platform using RESTful API. This system is designed to provide real-time data on pH, temperature, turbidity, TDS, salinity projection, and dissolved oxygen projection. It also includes automated management features such as dosing to adjust pH levels and notifications to alert farmers of critical conditions, ultimately reducing risks and improving productivity in Koi fish aquaculture.

2. RESEARCH METHODS

Data Collection and Processing

Data Collection Method

Data collection in this research was conducted using a combination of observation, interviews, and literature study. Observation was carried out by directly examining pond conditions, filter placement, and strategic locations for equipment installation over a period of approximately one week. Semi-structured interviews were conducted with Koi breeders and enthusiasts to gather insights from diverse perspectives, which helped enrich the research understanding and support system design. Additionally, a literature study was performed to collect relevant supporting information from various sources, including journals, books, and websites published between 2010 and 2024. This approach ensured that both practical observations and theoretical knowledge informed the development process.

Types of Data

The research utilized a combination of qualitative and quantitative data to obtain a comprehensive understanding of the problem. Qualitative data included descriptive information, such as user opinions regarding the monitoring system and interface, while quantitative data consisted of measurable parameters such as water pH, temperature, turbidity, and other water quality indicators. Collecting both types of data allowed for a more complete analysis and informed system design that met user needs and environmental conditions.

Data Sources

Data were obtained from both primary and secondary sources. Primary data were collected directly through interviews with Koi breeders and enthusiasts, as well as observations conducted at several Koi ponds to understand operational conditions and challenges. Secondary data were gathered through literature studies, including journals, books, and relevant websites, providing additional context and supporting information for system development and analysis.

Time and Place of Research

This research was conducted from March 2024 to August 2024. The experimental implementation and testing of the IoT-based water quality monitoring system were carried out in Koi ponds located in Cau Belayu Village, Marga District, Tabanan Regency, Bali, Indonesia. Meanwhile, the development of hardware, programming, and web integration processes was carried out at the Informatics Laboratory, Universitas Primakara.

System Development Methods

The Agile method is applied in developing the IoT device and web-based monitoring and management system for Koi Pond water quality. Agile emphasizes iterative development, user interaction, comprehensive documentation, and rapid adaptation to changes.

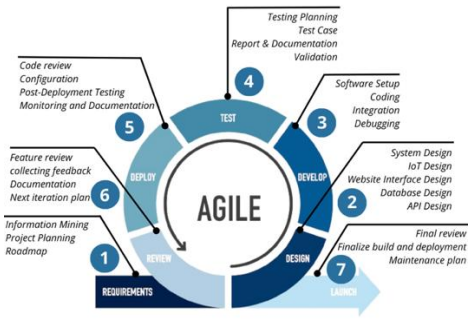

Figure 1. Stages of the Agile Method

Requirements/planning is collect supporting information regarding potential solutions, identify system requirements, plan the research, and create a project roadmap. Design system flows, hardware layout including microcontrollers, sensors, and actuators, as well as the website interface (UI/UX) with layout, navigation, typography, and icons. Develop, implement software setup, programming, interface integration, and debugging to ensure proper system functionality. Testing, conduct tests including planning objectives, scope, schedule, and success criteria using black box testing and smoke testing, test cases and validation. Deploy system to production including code review, configuration, smoke testing, monitoring, and documentation. Review, evaluate developed features, gather user feedback, document results, and prepare for the next iteration. Launching, conduct final review, deploy the system, verify features, and plan routine maintenance and future updates.

Planning and Design Stage

Instruments and Materials

The hardware used in this study includes a MacBook Pro 2015 (Intel i5, 8 GB RAM, macOS Big Sur) for software development and programming. The NodeMCU ESP32 microcontroller serves as the system's control unit, connected to sensors such as DS18B20 temperature, pH, TDS, and turbidity sensors for water quality monitoring. Supporting components include a 20x4 I2C LCD, relay, water pump, buzzer, LED indicators, step-down module, LiFePO4 battery, and solar panel, enabling autonomous and real-time operation in the koi pond environment.

The software comprises Arduino IDE 2.3.3 for microcontroller programming, Firebase Realtime Database for real-time data storage, and Visual Studio Code for code development

and integration. The web interface is built using Laravel and Node.js, with RESTful API for data communication. Additional tools include Fritzing for circuit design, Microsoft Word for documentation, and Google Chrome for accessing the web system.

Hardware Design

The wiring design between the sensors and actuators is carried out by connecting each component to the ESP32 microcontroller. Sensors are integrated through digital or analog input pins, while actuators such as the relay, buzzer, and LEDs are controlled via the microcontroller's output pins. The DS18B20 temperature sensor requires a pull-up resistor to function optimally. Additionally, power distribution is managed between the ESP32 supply and a step-down regulator to prevent system overload.

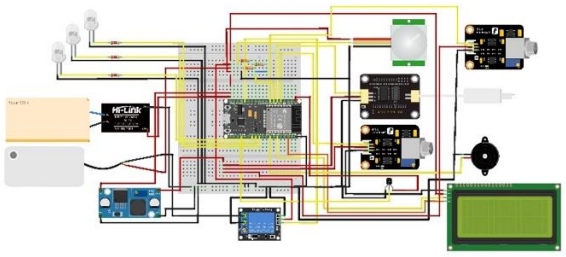

Figure 2. Hardware Wiring Design

Software Architecture Design

In this system, an MQTT broker is used to transmit data from the ESP32 microcontroller directly to the Firebase Realtime Database via the MQTT broker.

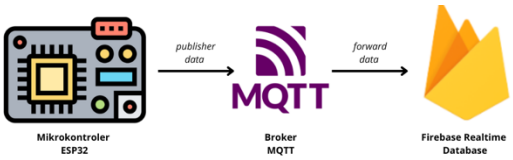

Figure 3. MQTT Implementation Diagram

The RESTful API is used to connect the data stored in Firebase with the Laravel-based website interface.

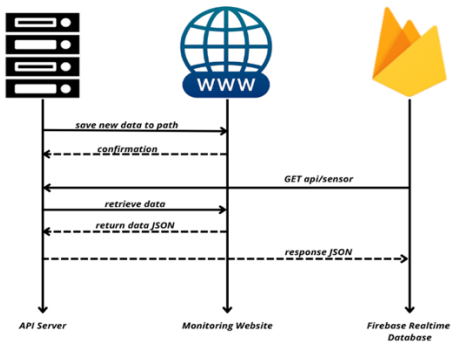

Figure 4. RESTful API Implementation Diagram

User Interface Design

The website is designed to visualize data obtained from the IoT sensors, including water pH, temperature, turbidity, TDS, estimated salinity, and estimated dissolved oxygen levels.

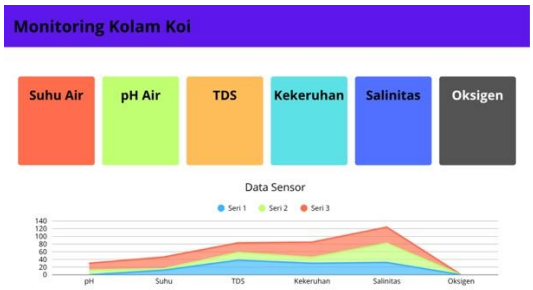

Figure 5. Website UI Design

Sensor Calibration Procedure

Sensor calibration is carried out to ensure accurate readings. The process involves comparing the sensor outputs with a reference device and adjusting the programming formula to minimize discrepancies. The calibration method applied in this study is as follows:

Table 1. Calibration Procedure

Sensor	Comparison tool	Calibration Method
DS18B20	Termometer Digital	Comparing the sensor temperature with a thermometer in water at various temperatures
pH sensors	Digital pH Meter + Buffer	Calibration using buffer solutions of pH 4, 7 and 10, with a linear regression curve
TDS sensors	Digital TDS Meter	Measure the 500ppm solution, then adjust the coefficient
Turbidity Sensors	Reference Sample	Measure the output on several different water samples, then create an equation relating the ADC to the NTU value.

System Testing Design

System testing is carried out to ensure that all functions within the Koi pond water quality monitoring system operate properly and in accordance with the design. The testing covers aspects of system functionality and initial feasibility. Two main methods are used.

Black-box testing

This testing is performed from the user's perspective without considering the internal structure or program code, focusing on the interaction between system inputs and outputs. The objective is to verify that the system responds correctly according to predefined scenarios.

Smoke testing

This testing is conducted to ensure that the system can run as a whole without errors during its initial activation, prior to more in-depth functional testing. It is performed immediately after the system is uploaded and powered on.

3. RESULTS AND DISCUSSION

System Development Stage

Preparation of the Development Environment

To support project programming and integration with Firebase, several libraries need to be added through the Library Manager in the Arduino IDE. To install a library, open the Sketch menu in the Arduino IDE, select Include Library, and then choose Manage Libraries.

Table 2. Library Project

No.	Library	Function
1	WiFi.h	Connecting the ESP32 to a Wi-Fi network.
2	FirebaseESP32.h	Connecting ESP32 with Firebase.
3	OneWire.h	Supports communication with DS18B20 temperature sensor.
4	DallasTemperature.h	Read and manage temperature data from the DS18B20 sensor.
5	LiquidCrystal_I2C.h	Displays data on an I2C-based LCD.
6	EEPROM.h	Store and read data from the ESP32's internal EEPROM.
7	GravityTDS.h	Reading TDS sensor from DFRobot.
8	UniversalTelegramBot.h	Communication with Telegram Bot API.
9	PubSubClient.h	Supports MQTT communication.
10	ArduinoJson.h	Parsing and creating JSON data in ESP32/Arduino.

Hardware Implementation and Programming

The programming implementation is designed to control how the ESP32 microcontroller reads, processes, and transmits data obtained from the sensors. In addition, it manages the actuators used in the developed device circuit.

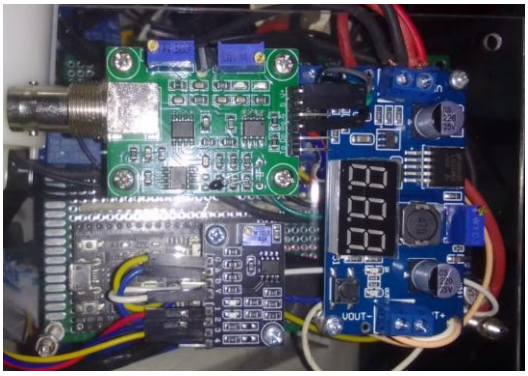

Figure 6. Hardware Implementation

Figure 7. Hardware Program View

The main functions implemented in the code are as follows:

a. Wi-Fi Connection

The connect To WiFi () function establishes a connection to a Wi-Fi network using WiFi.begin (ssid, password) and monitors the connection status up to 20 times at 500 ms intervals until successfully connected.

b. pH Sensor

The pH sensor reads an analog signal, converts it to voltage, and calculates the pH value using a linear calibration formula.

c. Temperature Sensor

The DS18B20 sensor periodically measures the water temperature, which is then adjusted using a calibration offset to improve measurement accuracy.

d. TDS Sensor

Analog readings are averaged, compensated for temperature, and converted into Total Dissolved Solids (ppm) using a polynomial equation with calibration factors.

e. Turbidity Sensor

Multiple analog samples are averaged, converted to voltage, and then transformed into NTU values by comparing with a clear water reference voltage.

f. Salinity Calculation

Salinity (ppt) is estimated from TDS values using an empirical conversion factor: Salinity = TDS × 0.0005.

g. Dissolved Oxygen Estimation

The maximum dissolved oxygen concentration (mg/L) is calculated from water temperature using a thermodynamic equation that incorporates four empirical coefficients.

Website Implementation and Programming

The Koi pond monitoring website was developed using the Laravel framework to provide a modular and scalable platform. It visualizes real-time data from IoT sensors (pH, temperature, turbidity, and TDS) collected via ESP32 and transmitted through MQTT and RESTful API to Firebase. The web interface also displays projected salinity and dissolved oxygen levels, offering a user-friendly dashboard for efficient water quality monitoring across devices.

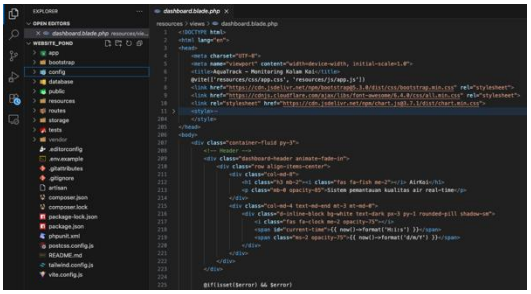

Figure 8. Website Program View

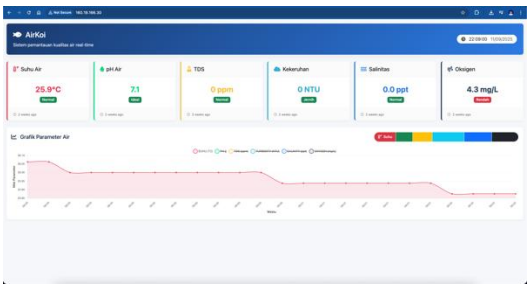

Figure 9. Website Dashboard View

System integration

System integration ensures efficient communication and data exchange between IoT devices, cloud services, and web applications. In this study, the integration involves connecting ESP32-based sensors to Firebase for real-time data storage, implementing MQTT for secure and real-time data transmission, and using a RESTful API to connect the database to a Laravel-based web application. This integration facilitates effective monitoring, control, and alerts for the Koi pond water quality system.

a. Firebase Integration

The integration is carried out to utilize the services provided by Firebase. The process begins with creating a new project on Firebase targeting a web-based application. Once the project is created, services such as Authentication and Realtime Database need to be enabled and configured according to the requirements.

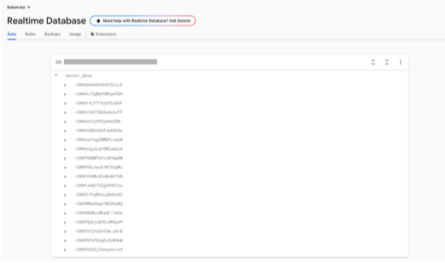

Figure 10. Realtime Database View

b. MQTT Integration

MQTT configuration is carried out to establish a connection between the IoT devices and the application server using the EMQX Cloud broker. The broker is deployed dedicatedly on Google Cloud Platform using the MQTT over TLS protocol on port 8883 to ensure secure data transmission.

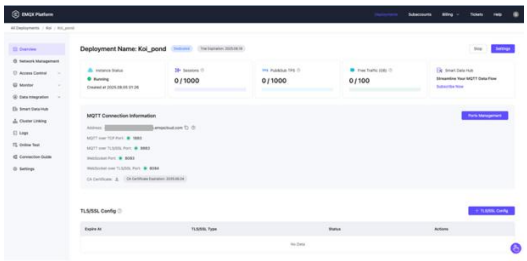

Figure 11. EMQX Broker Dashboard View

c. Server Code

The server code is implemented using Node.js and Express to handle communication between the IoT devices and the web application. It initializes and configures the MQTT client to connect securely to the EMQX Cloud broker, subscribes to relevant topics, and listens for incoming sensor data from the ESP32. Incoming messages are validated, formatted, and stored in Firebase Realtime Database for real-time access. Additionally, a RESTful API endpoint is provided to retrieve the latest sensor readings, enabling the Laravel-based web application to display the data and support monitoring, analysis, and visualization of water quality parameters in the Koi pond.

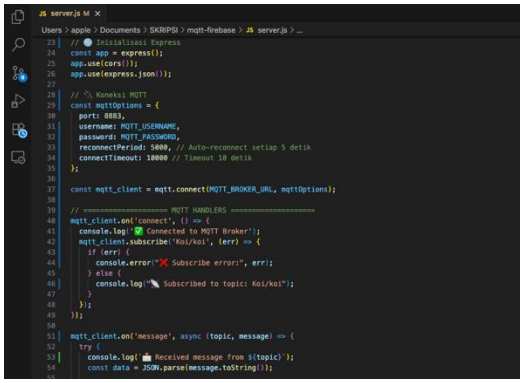

Figure 12. Server Program View

Calibration Results

The sensor calibration process was carried out to ensure the accuracy and reliability of the data generated by the water quality monitoring system.

a. pH Sensor

Calibration was carried out using three 250 ml solutions prepared with pH buffer powder. The pH sensor readings were compared with a digital pH meter.

Table 3. pH Sensor Calibration Results

pH buffers	pH sensor	pH meter	Error(%)
4.00	4.01	4.00	0.25
	3.95	4.00	1.25
	4.02	4.00	0.5
	4.01	4.00	0.25
6.86	6.87	6.86	0.15
	6.85	6.86	0.15
	6.85	6.86	0.15
	6.85	6.86	0.15
9.18	9.17	9.18	0.10
	9.11	9.18	0.76
	9.12	9.18	0.65
	9.18	9.18	0
Average Error			**0.32**

Based on the table above, it can be seen that the pH sensor readings have an average error of 0.32%.

b. Temperature Sensor

The DS18B20 sensor calibration was carried out using three separate water conditions, each in its own container. The DS18B20 sensor readings were compared with a digital thermometer.

Table 4. Temperature Sensor Calibration Results

Water Condition	Temperature Sensor	Thermometer	Error(%)
Cold	10.24	10.24	0.25
	10.26	10.24	1.25
	10.25	10.24	0.5
	10.23	10.24	0.25
Room Temperature	27.48	27.49	0.15
	27.50	27.49	0.15
	27.48	27.49	0.15
	27.48	27.49	0.15
Warm	50.12	50.14	0.10
	50.14	50.14	0.76
	50.15	50.14	0.65
	50.14	50.14	0
Average Error			**0.06**

Based on the table above, it can be seen that the DS18B20 sensor readings have an average difference of 0.01 and an error of 0.06%.

c. TDS Sensors

The calibration of the TDS sensor was carried out using a 500 ppm standard solution, where the TDS sensor readings were compared with a TDS meter.

Table 5. TDS Sensor Calibration Results

TDS sensor	TDS meter	Error(%)
499.7	500	0.06
500	500	0
500	500	0
499.6	500	0.08
500	500	0
500	500	0
499.7	500	0.06
500	500	0
500	500	0
500	500	0
500	500	0
500	500	0
Average Error		**0.06**

Based on the table above, it can be seen that the TDS sensor readings have an average difference of 0.33 and an error of 0.06%.

d. Turbidity Sensor

The turbidity sensor calibration will be carried out using three reference solutions with different turbidity levels.

Table 6. Turbidity Sensor Calibration Results

Reference Fluid	Turbidity Sensor	Results
clear	Clear Water	valid
clear	Clear Water	valid
clear	Clear Water	valid
clear	Clear Water	valid
cloudy	Murky Water	valid
cloudy	Murky Water	valid
cloudy	Murky Water	valid
cloudy	Murky Water	valid
very cloudy	Very Turbid Water	valid
very cloudy	Very Turbid Water	valid
very cloudy	Very Turbid Water	valid
very cloudy	Very Turbid Water	valid

Based on the table above, it can be seen that the turbidity sensor readings are overall valid when compared with the reference solutions.

Blackbox Testing Results

Blackbox testing was conducted to verify the system's functionality from the user's perspective. This testing focuses on the system's inputs and outputs, ensuring that all features operate according to user requirements. The following table presents the results of the Blackbox testing performed.

Table 7. Blackbox Testing Results

No.		Testing	Result
1	Features/Modules	DS18B20 sensor reading	
	Test Scenarios	The system reads the water temperature from the DS18B20 sensor.	
	Input	Water temperature: 25°C	Valid
	Expected Output	The temperature value is displayed on the LCD and the website with an accuracy of ±0.5°C.	
2	Features/Modules	pH sensor reading	
	Test Scenarios	The system reads the water pH from the pH sensor.	
	Input	Buffer solution: pH 6.86	Valid
	Expected Output	The pH value is displayed on the LCD and the website with an accuracy of ±1.5%.	
3	Features/Modules	TDS sensor reading	
	Test Scenarios	The system reads the water TDS level from the TDS sensor.	
	Input	Standard solution: 500 ppm	Valid
	Expected Output	The TDS value is displayed on the LCD and the website with an accuracy of ±10%.	
4	Features/Modules	Turbidity sensor reading	Valid

No.		Testing	Result
	Test Scenarios	The system reads the water turbidity level from the turbidity sensor.	
	Input	Clear water and turbid water	
	Expected Output	The turbidity description and value (NTU) are displayed on the LCD and the website.	
5	Features/Modules	Automatic Pump Control	
	Test Scenarios	The system activates the pump to lower the pH if the pH > 8.0.	
	Input	The water pH reaches 9.18.	Valid
	Expected Output	The water pump runs for 10 seconds to dispense the pH-lowering solution.	
6	Features/Modules	Website Data Visualization	
	Test Scenarios	The user opens the dashboard page to view real-time data.	
	Input	Accessed via a browser.	Valid
	Expected Output	Temperature, pH, TDS, turbidity, salinity, and oxygen data are displayed on the website along with graphical representations.	
7	Features/Modules	Firebase Integration	
	Test Scenarios	Sensor data is sent to Firebase and stored.	
	Input	All sensor readings are recorded.	Valid
	Expected Output	The data is saved in the Firebase Realtime Database and can be accessed by the website.	
8	Features/Modules	Website Responsiveness	
	Test Scenarios	Users access the website through mobile devices.	Valid
	Input	Access via smartphone.	
	Expected Output	The website layout adapts to the mobile screen without any issues.	

System Features Review and feedback

After completing the Blackbox testing process, a discussion was held with the koi fish farmers regarding the developed project. Based on the discussion, the farmers expressed the need for the device to be portable and capable of operating on alternative power sources other than household electricity. In addition, for notifications and direct interaction with the system, they requested integration with a mobile application such as Telegram. Therefore, an iteration of system development is required to incorporate this feedback.

First Iteration Phase

This iteration phase was carried out as a response to the feedback provided by koi fish farmer after the initial system testing. The main objectives of this stage are to enhance the

portability of the device by allowing it to be easily relocated, to integrate an alternative power source beyond household electricity, and to improve user interaction through mobile-based notifications and control, specifically by utilizing the Telegram application. These adjustments mark the beginning of the system refinement process to ensure better usability and practicality in real-world conditions.

a. Iteration 1 Design

In this iteration, a solar panel, solar charge controller, and battery are added as the system's power source. A step-down module is also included to regulate the output voltage.

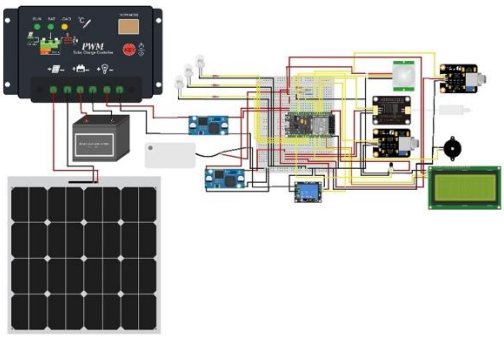

Figure 13. Hardware Wiring Design Iteration-1

In the system service block diagram, a Telegram bot function is integrated to provide notification features and enable direct interaction between users and the system.

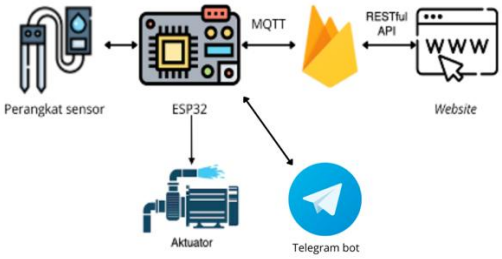

Figure 14. Service Block Diagram Design Iteration-1

b. Iteration 1 Development

In the first development iteration, the system was enhanced with a Telegram bot to provide both automatic notifications and direct user interaction. The implementation included three main functions: handleTelegramMessages(), which processes basic commands such as /start, /status, and /help; sendAlert(), which delivers automatic warnings when water quality parameters exceed defined thresholds (e.g., pH > 8.0 or temperature > 30 °C); and handleRelayCommand(), which allows remote control of the relay through commands like /relay_on, /relay_off, and /relay_status. These enhancements

enabled the system to operate not only as a monitoring tool but also as an interactive control platform, effectively meeting the operational needs of koi fish farmers.

c. Iteration 1 Testing

After the development process, Blackbox testing was carried out to verify the newly added functions.

Table 8. Blackbox Testing Results Iteration-1

No		Testing	Result
1	Features/Modules	Solar panel performance	Valid
	Test Scenarios	The solar panel generates power under sunlight.	
	Input	The panel is placed under direct sunlight, and the output voltage is measured.	
	Expected Output	The output voltage meets the specifications	
2	Features/Modules	Power supply to IoT system	Valid
	Test Scenarios	The IoT system operates using battery power.	
	Input	The system is powered on using the battery.	
	Expected Output	Sensors and the microcontroller are active, and data is successfully transmitted to Firebase.	
3	Features/Modules	Telegram bot connection	Valid
	Test Scenarios	The bot responds to basic commands (/start, /help).	
	Input	Commands are sent to the Telegram Bot via the application.	
	Expected Output	The bot replies with a welcome message or help menu.	
4	Features/Modules	Automatic notification	Valid
	Test Scenarios	The bot sends a notification when the water pH is abnormal.	
	Input	Set the pH value outside the safe range (e.g., pH > 8.0).	
	Expected Output	A notification appears in Telegram with detailed parameters.	

d. System Review and Deployment

After completing the first development iteration, a review was conducted to gather feedback from koi fish farmer. Based on the discussion, the current system was found to be satisfactory, and no additional feedback for further feature development was provided.

Smoke testing results

Smoke testing was conducted to ensure that the system's core features function properly in the implementation environment. This testing covered the system's basic operations, including device connectivity, data transmission, and user interface display.

Table 9. Smoke Testing Results

No		Testing	Result
1	Features/Modules	ESP32 Wi-Fi Connection	Valid
	Test Scenarios	Verify the ESP32 can successfully connect to the pre-configured Wi-Fi network.	
	Input	Power on the device.	
	Expected Output	The LCD shows a sequence of messages ending with a successful connection status (e.g., WiFi Connected or IP: 192.168.1.X). No error messages are displayed.	
2	Features/Modules	Data transmission to Firebase	Valid
	Test Scenarios	Sensor data is sent to Firebase via MQTT.	
	Input	Check the Firebase Realtime Database.	
	Expected Output	Sensor data appears in Firebase with the latest timestamp.	
3	Features/Modules	LCD Display	Valid
	Test Scenarios	The LCD displays sensor data.	
	Input	Check the LCD screen.	
	Expected Output	Sensor values are clearly shown on the LCD.	
4	Features/Modules	Website Access	Valid
	Test Scenarios	The user opens the website dashboard.	
	Input	Open the website URL in a browser.	
	Expected Output	The dashboard loads and displays the sensor data.	

4. CONCLUSION

Based on the results of the research and testing, it can be concluded that the IoT-based water quality monitoring system developed in this study was successfully built and functioned properly. The device, which consists of a NodeMCU ESP32 and several sensors including the DS18B20 for temperature, pH sensor, TDS sensor, and turbidity sensor, was able to measure pond water quality parameters in real time with high accuracy. Calibration results showed a very low average error, indicating that the collected data is reliable to support aquaculture activities. The system was also able to integrate hardware, software, data communication via

MQTT and RESTful API, cloud-based storage (Firebase Realtime Database), and user interfaces through a website dashboard and LCD display. In addition, the automatic notification feature via Telegram Bot worked effectively in providing alerts when water quality parameters exceeded safe thresholds. Testing using Blackbox and Smoke methods confirmed that all main functions operated according to the design, while feedback from koi fish farmers indicated that the system provided tangible benefits by improving monitoring efficiency, enabling faster responses to water quality changes, and reducing fish mortality risks. Furthermore, the use of the Agile development method proved to be effective, as it allowed feature adjustments based on user needs, including the addition of solar panels, batteries, and Telegram Bot integration in subsequent iterations.

ACKNOWLEDGEMENT

Thank you to the supervisors, colleagues, and Koi fish farmers in Cau Belayu Village who have helped through direction, input, and valuable information in the observation process. Thank you also to family and friends for the prayers, motivation, and moral support provided, so that this research can be completed properly.

REFERENCES

Ariyanto, D., & Kusriyanto, M. (2023). Sistem Pemantau Kualitas Air Kolam Ikan Koi Berbasis IoT. *Technologia*, *14*(1). Retrieved from https://ojs.uniska-bjm.ac.id/index.php/JIT. https://doi.org/10.31602/tji.v14i1.9199

Chamim, A. N. N. (2010). Penggunaan Microcontroller Sebagai Pendeteksi Posisi Dengan Menggunakan Sinyal GSM. *Jurnal Informatika*, *1*, 431–432. Retrieved from https://media.neliti.com/media/publications/102952-ID-penggunaan-microcontroller-sebagai-pende.pdf

Dermawan, I., Wagiyati, S., Budilaksono, S., & Suwarno, M. A. (2018). Pengembangan Web Semnas Ikra-Ith Dengan Metode WDLC (Web Development Life Cycle). *Jurnal Komputer Dan Informasi*, *3*(2). Retrieved from https://journals.upi-yai.ac.id/index.php/ikraith-informatika/article/view/317/208

Echo, P. (2022, September 30). Ikan Hias Indonesia Semakin Laku di Dunia. Retrieved 19 April 2024, from https://fpp.umko.ac.id/2022/09/30/ikan-hias-indonesia-semakin-laku-di-dunia/

Febrianti, F., Wibowo, S. A., & Vendyansyah, N. (2021). Implementasi IoT (Internet Of Things) Monitoring Kualitas Air dan Sistem Administrasi Pada Pengelola Air Bersih

Skala Kecil. *Jurnal Mahasiswa Teknik Informatika)*, *5*(1). https://doi.org/10.36040/jati.v5i1.3249

Firman Maulana, I. (2020). Penerapan Firebase Realtime Database pada Aplikasi E-Tilang Smartphone berbasis Mobile Android. *Jurnal Rekayasa Sistem Dan Teknologi Informasi*, *4*(5), 854–863. Retrieved from https://www.jurnal.iaii.or.id/index.php/RESTI/article/view/2232/315 https://doi.org/10.29207/resti.v4i5.2232

Ningtyas, S. (2020, May 17). Panduan Lengkap dan Mudah Cara Menggunakan IFTTT. Retrieved 19 May 2024, from https://www.niagahoster.co.id/blog/cara-menggunakan-ifttt/

Pandawana, I. D. G. A., Partama, I. G. Y., Puspitawati, N. M. D., Dwipa, I. M. S., & Sari, P. A. P. (2023). Penerapan Teknologi Smart Jair Berbasis IoT dalam Meningkatkan Produktivitas Budidaya Ikan Mujair dan Mitigasi Kematian Ikan Massal Di KJA Danau Batur-Kintamani. *Communnity Development Journal*, *4*(6), 13013–13018. Retrieved from https://journal.universitaspahlawan.ac.id/index.php/cdj/article/view/23522/16555

Paramitha, I. A. K. P., Made Wiharta, D., & Suyadnya, I. M. A. (2022). Perancangan dan Implementasi RESTful API Pada Sistem Informasi Manajemen Dosen Universitas Udayana. *Jurnal SPEKTRUM*, *9*(3), 15. Retrieved from https://ojs.unud.ac.id/index.php/spektrum/article/download/92900/46519

Sanjaya, R., & Hesinto, S. (2017). Rancang Bangun Website Profil Hotel Agung Prabumulih Menggunakan Framework Bootstrap. *Jurnal Teknologi Dan Informasi*, *7*(2). Retrieved from https://ojs.unikom.ac.id/index.php/jati/article/view/758. https://doi.org/10.34010/jati.v7i2.758

Santanumurti, M. B. (2021, March 31). Ikan Koi dan Potensinya di Indonesia. Retrieved 19 April 2024, from https://unair.ac.id/ikan-koi-dan-potensinya-di-indonesia/

Sitorus, M. B., Gifson, A., Mangapul, J., & Aziz, H. (2020). Pelatihan Mikrokontroler Dalam Pengenalan Robotika Sebagai Respon Revolusi Industri 4.0 Di SMK Media Informatika Dasana Indah Tengerang. *Jurnal Pengabdian Pada Masyarakat Menerangi Negeri*, *2*(2), 144–150. https://doi.org/10.33322/terang.v2i2.989

Susanto, B. M., Atmadji, E. S. J., & Brenkman, W. L. (2018). Implementasi MQTT Protocol pada Smart Home Security Berbasis Web, *4*(3). Retrieved from https://media.neliti.com/media/publications/266688-implementasi-mqtt-protocol-pada-smart-ho-afd9be49.pdf. https://doi.org/10.33795/jip.v4i3.207

Umami, Z., & Ningrum, N. K. (2020). Pengujian Implementasi REST API Pada Website Sistem Pencarian Informasi Produk Fashion di Shopee. *Jurnal Sistem Informasi Dan Teknologi*, *3*(2). Retrieved from http://www.jurnal.umk.ac.id/sitech. https://doi.org/10.24176/sitech.v3i2.5671

Yanel, K. (2023). Alat Pengusir Burung. *Jurnal Teknologi Manufaktur*, *1*. Retrieved from https://ejournal.polman-babel.ac.id/index.php/manutech/article/view/287/206. https://doi.org/10.33504/manutech.v15i01.287