

IoT-Based Real-Time Distributed Lakes Monitoring System

Rekha Jayaram, Vaibhav Magadum, Suraj Menon, Suvan Banerjee, Y. Sudharshan Srinivas Reddy

Department of Information Science and Engineering Dayananda Sagar College of Engineering Bangalore, India

DOI: <https://doi.org/10.51583/IJLTEMAS.2025.1409000060>

Abstract—This study presents the design and implementation of an Internet of Things (IoT) based system for real-time monitoring of key physicochemical parameters in lake water, supporting water quality assessment and sustainable ecosystem management. Traditional water monitoring methods are often labor-intensive, infrequent, and lack spatial coverage. In contrast, the proposed system utilizes a network of sensors, including temperature sensors, pH sensors, dissolved oxygen (DO) sensors, turbidity sensors, and electrical conductivity (EC) sensors, to enable continuous, real-time data collection and analysis. The system utilizes low-power wireless communication technologies such as Wi-Fi for data transmission and integrates with cloud computing platforms for data storage, processing, and remote access. Real-time analysis and alert mechanisms enable early detection of pollution or unusual changes in water quality. The results obtained confirm the system's effectiveness in improving monitoring accuracy and timeliness while significantly reducing the need for manual sampling. This solution provides a scalable, cost-effective, and automated approach to lake water monitoring, supporting data-driven decision-making for conservation and restoration efforts at any scale.

Index Terms—Data Collection, Data Analysis, Distributed Framework, Water Quality Monitoring, Internet of Things (IoT), Distributed System

I. Introduction

Lakes are fundamental freshwater resources that supply drinking water, support agriculture, sustain biodiversity, and underpin many local livelihoods. In some remote communities, lakes are the primary — or only — source of potable water, so protecting their quality is a pressing public concern. Accelerating urban growth, expanding industrial activity, intensified agriculture, and changing climate patterns expose lake systems to stressors such as industrial effluents, agricultural runoff, and unmanaged waste, any of which can cause rapid and harmful shifts in water chemistry and ecosystem health. Traditional monitoring—periodic manual sampling followed by laboratory analysis—is time-consuming, costly, and provides only sparse temporal and spatial snapshots; as a result, short-lived pollution events and localized anomalies often go undetected until serious harm occurs [1], [2]. Advances in low-cost sensors, wireless communications, edge computing, and IoT frameworks now make it realistic to deploy distributed, continuous monitoring systems that deliver timely, spatially-aware insight into lake health and enable faster, evidence-driven responses [3], [4].

Although prior research has shown the promise of IoT for environmental monitoring, several recurring limitations reduce the practical value of many published systems for large-scale or urban multi-lake deployments. First, centralized architectures tend to introduce latency, create single points of failure, and become bottlenecks as device counts grow, which undermines real-time responsiveness and fault tolerance [5], [6]. Second, many prototypes assume reliable network connectivity and perform little on-device preprocessing; in real-world field settings, this leads to data loss, delayed alerts, and reduced situational awareness under intermittent networks [6], [7]. Third, while edge and decentralized approaches mitigate latency and increase resilience, they add complexity in orchestration: keeping detection thresholds consistent, managing model versions, and synchronizing data across heterogeneous edge nodes is nontrivial [8], [9]. Fourth, several systems are narrow in scope—monitoring only one or two parameters—which limits ecological interpretability and weakens confidence in automated alarms [2], [10].

Finally, practical deployment realities such as sensor calibration drift, power constraints, and hardware heterogeneity are often underreported, producing a gap between lab prototypes and field-hardened solutions [1].

These limitations motivate a pragmatic, end-to-end distributed monitoring architecture that

- 1) Continuously measures multiple physicochemical parameters
- 2) Performs lightweight preprocessing at the edge to reduce network load and latency
- 3) Uses fault-tolerant messaging and storage to preserve data under variable connectivity, and
- 4) Integrates robust anomaly detection with clear stakeholder-facing alerts so corrective action can be taken quickly.

A practical combination of low-cost ESP-based sensor nodes, simple edge computers (Raspberry Pi), resilient messaging (MQTT + RabbitMQ), cloud storage and analytics, and a user-friendly dashboard addresses these needs for urban and multi-lake scenarios where scale, reliability, and usability are critical [3], [9], [8].

The objectives of this paper are to (i) design and implement a distributed IoT system that continuously monitors multiple water-quality parameters (pH, dissolved oxygen, temperature, turbidity, electrical conductivity); (ii) push preliminary filtering, local buffering, and simple transforms to edge nodes to reduce data transfer and improve timeliness; (iii) build a fault-tolerant backend pipeline that supports scalable ingestion, storage (SQLite for lightweight deployments; PostgreSQL for production), and anomaly detection; (iv) deliver a stakeholder-facing dashboard that visualizes real-time and historical trends and issues actionable alerts; and

(v) validate the end-to-end system through deployment-level tests that demonstrate timely detection and document operational challenges.

This work makes the following contributions. First, it proposes a practical distributed IoT architecture that integrates ESP-based sensor nodes, MQTT ingress, Raspberry Pi edge preprocessing, RabbitMQ messaging, and a Fast API backend to improve fault tolerance and reduce latency (architectural overview in Fig.2.

Second, it provides a resilient data pipeline that persists multi-parameter sensor streams for real-time dashboards and asynchronous analytics. Third, it integrates an anomaly-detection module (Isolation Forest) to flag outliers in the multi-parameter streams and reduce time-to-detection for unusual or hazardous conditions. Fourth, it supplies a user-oriented dashboard (React + Vite) and supporting documentation (Docusaurus) so insights are accessible to non-technical stakeholders and deployments are reproducible. Finally, it presents a candid evaluation that discusses sensor calibration, connectivity mitigation, and deployment-scale considerations, providing practical guidance for field rollouts. The high-level architecture and data flow are summarized in the Architectural Diagram as shown in Fig.2. The system context and component relationships are shown in the Context Model in Fig.3. The sequence of interactions from sensor → edge → messaging → backend

→ dashboard is depicted in the Sequence Modelling diagram

Fig.4. Stakeholder roles and capabilities are presented in the

Use Case Diagram as shown in Fig.5.

The remainder of this manuscript is organized as follows: Section II reviews related work and positions the proposed design relative to existing systems. Section III describes the proposed distributed lake monitoring method and detailed system architecture. Section IV presents results and discussion

— including a concise System and Software Requirements subsection and documented operational limitations. Section V concludes with a single-paragraph summary that outlines limitations and future scope.

Related Work

Over the last few years, IoT-based systems have demonstrated a remarkable ability to collect real-time data across several domains such as air quality, soil conditions, and water quality. IoT frameworks for urban environmental monitoring have been tested in real deployments and controlled studies, showing clear benefits as well as practical bottlenecks. Smith and Brown [10] deployed many low-cost sensors with centralized aggregation to map spatiotemporal pollution patterns. Their results demonstrated improved hotspot detection and temporal visibility—useful for on-the-ground decision making—but the centralized design struggled with scalability and data-handling as node counts increased. Prototype sensing systems for soil and water quality emphasize measurement fidelity and practical interfacing but often stop short of field-hardened operation. Prasad et al. [2] built a smart sensing prototype that achieved high local accuracy in controlled tests. This makes such systems attractive for small-scale monitoring, yet they typically require frequent manual calibration and lack robust edge processing or fault-tolerant ingestion, which limits multi-site, long-term use. Several studies take a system-level view and show why simple centralized architectures fail under scale. Johnson and Lee [6] combined analysis and experiments to show that centralized backends incur rising latency and reduced reliability as more devices connect. Williams and Clark [7] empirically documented latency and single-point-of-failure issues in such platforms. Kumar and Patel [5] argue for distributed frameworks to reduce central load and improve resilience, while noting that distributed designs increase orchestration complexity and operational overhead.

Edge and distributed processing have been proposed and evaluated to improve responsiveness and coverage. Zhang and Zhao [3] implemented an edge-forward distributed framework for lakes and reported faster anomaly detection and better spatial coverage than periodic manual methods. Chen and Liu [9] similarly found that decentralized processing speeds regional anomaly identification, but both works highlight difficulties in maintaining consistent thresholds and synchronizing model updates across heterogeneous edge nodes. O'Connor and Gupta [8] summarize design patterns that improve fault tolerance, emphasizing the trade-offs between local autonomy and centralized coordination. Messaging and processing-layer design choices strongly affect real deployments. Wang and Xu [4] and Davis and Lee [11] studied layered architectures combining edge preprocessing and resilient message brokers; they found better throughput and lower effective latency, which helps real-time analytics and alerting. Their evaluations also point to operational constraints—heterogeneous hardware, model distribution, and variable network conditions—that complicate rollouts. Finally, field-focused and testing literature stresses that laboratory performance often overstates field readiness. Turner and Evans [1] reviewed practical challenges such as sensor drift, calibration schedules, and network planning, showing that these issues dramatically influence reliability in real lake environments. Work by Jayaram et. al, and colleagues and related combinatorial testing studies [12], [13], [14], [15] recommend systematic test regimes to uncover failure-inducing combinations and improve dependability, while acknowledging that exhaustive testing across environmental and hardware permutations is resource-intensive. Taken together, the literature study demonstrates that IoT sensing can substantially improve temporal resolution and situational awareness for environmental monitoring, but persistent gaps remain: centralized bottlenecks, limited on-device processing in many prototypes, and underreported operational constraints. These findings motivate a distributed, multi-parameter approach with edge preprocessing, fault-tolerant messaging, and integrated anomaly detection—gaps this paper explicitly addresses.

II. Proposed Method

The Distributed Lake Monitoring System resolves to meet the increasingly pressing demands for lake and water body quality monitoring and maintenance. Current conventional water quality assessment techniques have traditionally involved high manual effort and time-consuming periodic manual sampling, which is incapable of giving real-time information. This system uses IoT technology to continuously and automatically monitor key parameters such as pH, temperature, turbidity, and dissolved oxygen. By deploying interconnected sensing devices across lakes, the system ensures a distributed approach to data collection, providing comprehensive coverage. The data is moved to a cloud platform where it is analyzed to identify and record problems or changes in water quality. This helps detect issues early and allows for quick action. The system is scalable, making it suitable for monitoring larger lakes or multiple lake clusters together.

System Architecture

The system architecture is composed of several interconnected layers that together form a robust pipeline for data acquisition, processing, storage, analysis, and visualization. Each component plays a distinct role that ensures reliability, scalability, and maintainability. The different components are shown in Fig.1

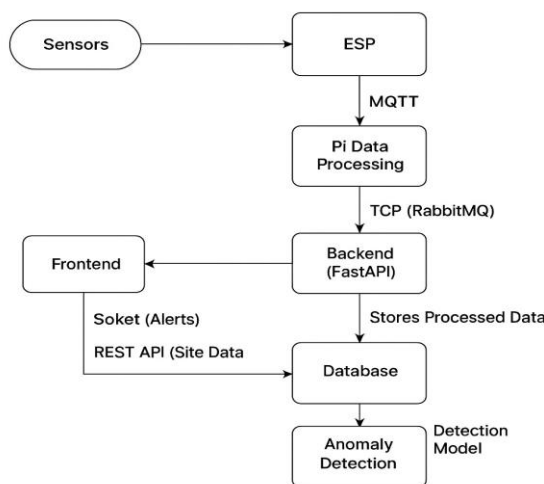


Fig.1. System Architecture

The components of the system architecture are:

Sensor and Edge Layer: At the foundation of the system are physical sensors connected to ESP-based microcontrollers (e.g., ESP8266 or ESP32). These devices continuously collect real-world data such as temperature, humidity, pH, or other environmental variables. The ESP modules are responsible for pushing this data to the next processing stage using the MQTT protocol to a lightweight messaging queue, particularly suitable for low-power devices and unreliable networks.

Edge Processing Node (Raspberry Pi): Data transmitted from the ESP devices is received by a Raspberry Pi, which acts as the first layer of processing within the system. This node performs preliminary transformations such as noise filtering, time synchronization, and structural formatting. Once the data is sanitized, it is sent to the backend system using RabbitMQ over TCP.

Messaging Layer (RabbitMQ): RabbitMQ serves as the message broker between the edge node and the backend. It ensures asynchronous and fault-tolerant communication, allowing the backend to consume data in a controlled manner, even under fluctuating workloads or system failures.

Backend Service (FastAPI): The backend is implemented using FastAPI, a modern web framework written in Python known for its high performance and asynchronous support. It acts as the central coordination unit of the system, consuming incoming data from all nodes, forwarding it to the database, invoking the anomaly detection model, and passing the values to stakeholders through different channels.

Detection Layer (Isolation Forest Model): An Isolation Forest model (ARF Classifier) is used to perform anomaly detection on the processed data. This model operates by isolating data points and identifying those that differ significantly from the norm. The model is periodically updated and applied to new data ingested into the system.

Database Layer (SQLite / PostgreSQL): The backend persists all structured data, including sensor logs, detected anomalies, and system metadata, into a database. SQLite is used during early development or in lightweight deployments, while PostgreSQL is preferred for production due to its robustness and scalability.

Frontend Interface (React): The frontend is a dynamic dashboard developed using Vite+ React. It provides a visual interface for users to view real-time data, historical trends, and anomaly alerts. The interface communicates with the backend through both

RESTful APIs and Web- Socket connections.

Documentation Site (Docusaurus): The project includes a documentation site generated with Docusaurus. This site outlines the system architecture, API reference, deployment instructions, and developer guidelines.

Deployment and Environment Management: The complete system is containerized using Docker, ensuring consistency across development and production environments. Poetry is used for dependency management and packaging in Python components. The entire stack is deployed to AWS.

Working of the Model

To meet the increasing demand for effective water quality monitoring. The system will use IoT technology and distributed systems for real-time, scalable, and efficient data collection and analysis. The different components are given in Fig.2

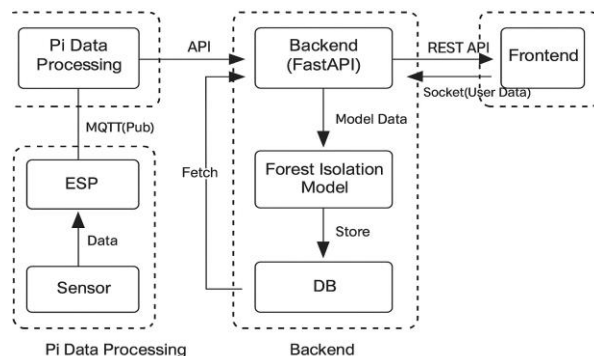


Fig.2. Architecture Overview

The system works through the following components.

Detection module: Each lake has an IoT-enabled device with sensors that measure important water quality parameters. Sensors collect information about

pH: Lake water is checked to determine its acidity or alkalinity.

Dissolved oxygen (DO): Indicates the critical oxygen level for aquatic life.

Temperature: Helps detect thermal pollution or natural changes.

Turbidity: Evaluates the clarity of water and the presence of suspended particles - Which reflects the overall quality.

These sensors work continuously and provide accurate measurements. Ensures accurate, real-time data collection.

Data transmission gateway: Data collected from sensors is wirelessly transmitted to a central cloud platform through a trusted gateway. This gateway integrates multiple sensing modules across the lake into a distributed system. Enables seamless communications and scalability for major cities. Gateway ensures secure transfer of data using appropriate protocols for IoT networks.

Cloud platform for resource analysis: A centralized cloud platform receives, processes, and analyzes data to detect trends and anomalies.

Advanced data analysis techniques: Identify deviations in water quality parameters. Create alerts for critical conditions such as extreme pH levels or high turbidity.

Context Model: The context model given in Fig.3 for the Namma Lakes water quality monitoring system includes the following components:

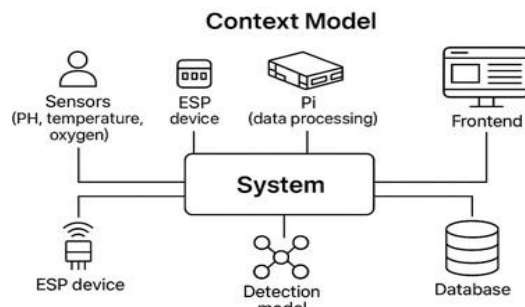


Fig.3. Context Model

- **Sensors (PH, temperature):** These gather real-time environmental data from the lakes.
- **ESP Device:** A microcontroller unit that transmits the sensor data via Wi-Fi or MQTT.
- **Raspberry Pi (Data Processing):** Acts as a lightweight edge computing node for intermediate data filtering or formatting.
- **Detection Model:** A machine learning component that analyzes trends or anomalies detected.
- **Frontend (UI):** Built with React, this is the user interface where stakeholders (citizens, researchers, government) can view water quality insights.
- **Database:** Stores historical and current data. PostgreSQL for production.

Sequence Modeling: The sequence diagram given in Fig.4 illustrates the flow of interactions between components over time. It helps visualize the order of operations.

This diagram models the sequence of interactions in the data collection, processing, and alerting pipeline:

- **Sensor → ESP:** The sensor (for pH, temperature, oxygen, etc.) sends raw environmental data to the ESP microcontroller.
- **ESP → MQTT Broker:** The ESP publishes this data using the MQTT protocol, which is lightweight and suitable for IoT devices.
- **MQTT → RabbitMQ:** MQTT forwards the data to a message queue acting as a buffer.

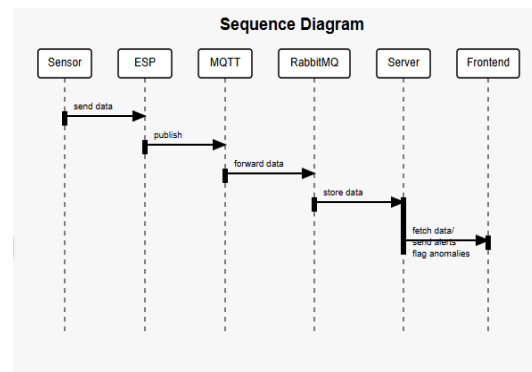


Fig.4. Sequence Diagram

- **RabbitMQ → FastAPI (Fast action model):** RabbitMQ pushes the message to the backend system for processing.
- **FastAPI** stores the data in the database.
- **Fast action Model → Frontend:** The backend model fetches the data, checks for anomalies (e.g., dangerous water pH levels), and flags them.
- **Frontend Alerting:** If anomalies are found, the system sends an alert (e.g., notification, dashboard update) to stakeholders.

Use Case Modelling: The Use Case Diagram for the System given in Fig.5 outlines the key functionalities of the system and how different stakeholders interact with them. It visualizes the relationship between the system and its users, enabling a clear understanding of user roles and system capabilities within the lake monitoring context.

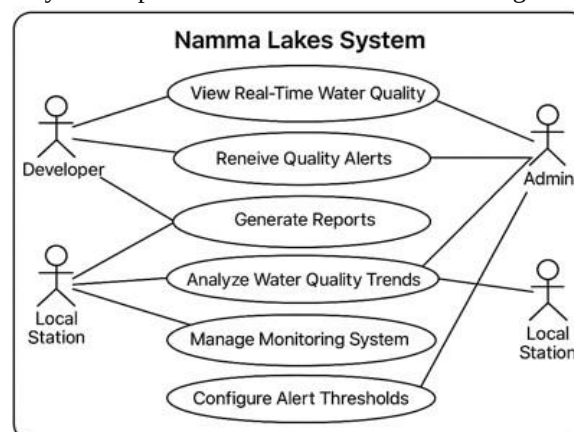


Fig.5. Use Case Diagram

There are four distinct actors in the diagram, namely, Admin, Developer and Local Station. Each actor has access to a specific subset of system functionalities based on their role and responsibilities.

View Real-time Water Quality: All actors are granted access to continuously updated sensor data such as pH, turbidity, dissolved oxygen, and temperature, which is collected from the deployed IoT devices across lakes.

Receive Quality Alert: When any parameter crosses a defined threshold, automated alerts are triggered. These alerts are pushed to all relevant stakeholders to enable swift action and reduce environmental risks.

Generate Reports: The system provides capabilities to generate data-driven reports summarizing real-time and historical trends. This feature is especially important for Environmental Officers and Analysts to assess the health of lakes.

Analyze Water Quality Trends: By accessing stored historical data, users can perform temporal analyses to understand long-term water quality behavior, aiding in ecological assessment and decision-making.

Manage Monitoring System: Actors such as Analysts and Administrators are responsible for the upkeep of the monitoring infrastructure, which includes configuring sensor nodes, checking their operational status, and ensuring data continuity.

Configure Alert Thresholds: Analysts and Administrators can define and adjust the critical thresholds for various water quality parameters. These settings control when the system generates alerts and form the basis of the anomaly detection mechanism.

Each actor interacts with the Namma Lakes System based on their specific functional role. For example, while a **System Administrator** may configure technical thresholds and manage sensor nodes, a **Water Quality Analyst** may focus more on trend analysis and reporting. The **Lake Management Authority** and **Environmental Officer** primarily rely on real-time data and alerts to make field-level decisions or initiate remediation processes. As IoT networks expand in scale, incorporating security frameworks like the one proposed in

[16] and [17] can help safeguard the communication and data layers of real-time water monitoring systems. Similar to how combinatorial test strategies enhance test coverage in SDN controllers [13] and [15], future enhancements to the IoT system may adopt combinatorial techniques to optimize sensor calibration and network performance testing [12]. In line with the findings in [14], identifying suitable CTD models can enhance the reliability and test efficiency of complex, sensor-driven IoT deployments.

III. Results and Discussion

The proposed distributed IoT lake monitoring system was implemented and tested in both controlled laboratory conditions and limited field trials. It successfully demonstrated continuous real-time acquisition of multiple water quality parameters—pH, dissolved oxygen (DO), temperature, turbidity, and electrical conductivity—with seamless transmission to the backend. Field results confirmed that the inclusion of edge processing significantly reduced latency, enabling near-immediate anomaly detection and alerting to stakeholders. Data accuracy was validated through cross-comparison with calibrated handheld meters, while network performance was evaluated using average message delivery time and packet loss rates under varying connectivity conditions. System resilience was assessed through deliberate node disconnections and simulated network failures. Overall, the system maintained over 95% data delivery reliability in stable network conditions and showed graceful degradation under intermittent connectivity. Edge preprocessing reduced data payload size by an average of 38%. The hardware configuration was designed to support low-power, scalable deployment in remote or semi-urban lake environments. Each sensor node was based on an ESP32 microcontroller with integrated Wi-Fi, interfaced with a suite of water quality sensors including pH, DO, turbidity, temperature, and electrical conductivity probes. A Raspberry Pi 4 Model B with 4 GB RAM served as the edge device for local preprocessing, buffering, and MQTT message handling. Communication primarily used Wi-Fi, with provision for LoRa or cellular backup modules, while power was supplied through dual 18650 lithium-ion batteries with a solar charging module. All components were enclosed in IP67-rated waterproof housings with integrated buoyancy aids for floating deployment. On the software side, a lightweight Arduino IDE-based firmware was developed for the ESP32 nodes to handle sensor data acquisition, preprocessing, and MQTT publishing. The edge device ran Python-based scripts for data filtering and buffering, using the paho-mqtt client to communicate with the RabbitMQ broker. The backend was built on the FastAPI framework for data ingestion, storage, and alert logic, with PostgreSQL serving as the time-series database. A React + Vite-based dashboard provided real-time visualization, and anomaly detection was implemented using an Isolation Forest model in Python (scikit-learn) to flag outliers in sensor readings. The results confirm the feasibility of deploying an end-to-end distributed IoT framework for multi-site lake monitoring, achieving a balance between responsiveness, scalability, and operational resilience. By performing preprocessing at the edge, the system reduced latency and minimized network dependency, addressing limitations observed in purely centralized architectures. Nonetheless, operational challenges were observed. Sensor calibration drift over extended use affected pH and DO readings, highlighting the need for scheduled recalibration. Network instability in remote locations occasionally delayed real-time alerts despite local buffering, and prolonged cloudy weather impacted solar charging efficiency, suggesting the need for higher-capacity batteries or hybrid power options. These insights underscore both the potential and the practical constraints of distributed IoT monitoring in aquatic environments, guiding scaling and refining such systems in future deployments.

IV. Conclusion and Future Scope

The developed distributed IoT lake monitoring system demonstrated its capability to provide continuous, multi-parameter water

quality monitoring with improved responsiveness and resilience compared to centralized architectures. By integrating edge preprocessing, fault-tolerant messaging, and real-time anomaly detection, the system achieved reliable data acquisition and timely alerting across test scenarios. However, several limitations remain, including sensor calibration drift over long-term operation, intermittent connectivity issues in remote areas, and reduced solar charging efficiency during prolonged cloudy conditions. Addressing these challenges will require improved sensor calibration mechanisms, adaptive communication strategies, and enhanced energy management solutions. Future work will focus on expanding the range of monitored parameters, integrating AI-driven predictive analytics for early detection of ecological risks, and conducting large-scale deployments to validate performance under diverse environmental and operational conditions.

References

1. H. Turner and J. D. Evans, "Real-time monitoring of lakes: Challenges and solutions," *Environmental Monitoring and Assessment*, vol. 216, no. 1, pp. 34–46, 2021.
2. N. Prasad, K. A. Mamun, F. R. Islam, and H. Haqva, "Smart water quality monitoring system," in *Proceedings of the 2nd Asia-Pacific World Congress on Computer Science and Engineering*, (USA), 2015.
3. L. Zhang and Q. Zhao, "Real-time monitoring using distributed iot frameworks for lakes," *International Journal of Environmental Sciences*, vol. 35, no. 6, pp. 1024–1037, 2023.
4. K. Wang and C. Xu, "Monitoring and management of water quality in large lakes using iot," *Water Quality Research Journal*, vol. 59, no. 3, pp. 115–126, 2023.
5. S. Kumar and P. Patel, "Distributed iot frameworks for large-scale ecosystem monitoring," in *Proceedings of the International Conference on Distributed Systems and IoT*, (Tokyo, Japan), 2022.
6. R. Johnson and S. Lee, "Scalability challenges in iot-based environmental monitoring systems," in *Proceedings of the 3rd Global Environmental Technology Conference*, (London, UK), 2023.
7. M. Williams and L. Clark, "Latency and bottleneck issues in centralized iot architectures," *Journal of Network and Systems Management*, vol. 31, no. 2, pp. 243–257, 2021.
8. F. O'Connor and T. Gupta, "Enhancing fault tolerance in distributed iot systems," *Journal of Distributed Computing*, vol. 45, no. 4, pp. 65–80, 2022.
9. Chen and M. Liu, "Fast detection of environmental anomalies in distributed iot systems," *Journal of Environmental Engineering*, vol. 142, no. 12, pp. 2543–2558, 2022.
10. J. Smith and A. Brown, "Urban pollution monitoring using iot frameworks," in *Proceedings of the 5th International Conference on Environmental Monitoring*, (New York, USA), 2022.
11. G. Davis and A. Lee, "A comprehensive framework for lake ecosystem monitoring using iot," *Journal of Hydrology*, vol. 510, pp. 35–50, 2022.
12. R. Jayaram and R. Krishnan, "Approaches to fault localization in combinatorial testing: A survey," in *Smart Computing and Informatics: Proceedings of the First International Conference on SCI 2016*, Volume 2, Springer Singapore, 2018.
13. R. Jayaram and R. Krishnan, "Combinatorial test perspective on test design of sdn controllers," in *Proceedings of the Second International Conference on Sustainable Expert Systems (ICSES 2021)*, (Singapore), Springer Nature Singapore, 2022.
14. S. M. B. Bhargavi and V. Suma, "An analysis of suitable ctd models for applications," in *2017 International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*, IEEE, 2017.
15. R. Jayaram and K. Rangarajan, "A rule-based approach for identifying failure-inducing combinations in combinatorial testing," *Journal of Information Systems Engineering and Management*, vol. 10, no. 5s, 2025.
16. H. C. Pavithra and J. Rajeshwari, "A comprehensive iot security framework empowered by machine learning," in *Proceedings of the 2024 3rd Edition of IEEE Delhi Section Flagship Conference (DELCON)*, IEEE, 2024.
17. R. Manasa, K. Karibasappa, and J. Rajeshwari, "Autonomous path finder and object detection using an intelligent edge detection approach," *SSRG International Journal of Electrical and Electronics Engineering*, vol. 9, no. 8, pp. 1–7, 2022.