WebLog Journal of Computer Science and Technology
Review Article
Published: 17 Jun, 2026

Development of a Scalable Micro-Services IoT Irrigation System (SMIIS) Application for the Growth of *Talinum Triangulare*

O. Ipinnimo[1], I. J. Munezero[2], E. E. Atojunere[3] and U. C. Ogochukwu[1]*

[1]Department of Systems Engineering, University of Lagos, Akoka, Nigeria

[2]Department of Computer and Software Engineering, University of Rwanda, Rwanda

[3]Department of Soil Science and Biosystems Engineering, Fiji National University, Fiji

Abstract

Precise water application is crucial to meet the crop-water requirement especially in areas where rain-fed agriculture is considered inadequate. While some AI-driven IoTs have been deployed successfully to coordinate sensors capable of measuring irrigation parameters, there was a problem with their scalability for multiple users. This paper presented the development of a Scalable Micro-Services IoT Irrigation System (SMIIS) for the growth of *Talinum triangulare* based on micro-services IoT. SMIIS was developed to irrigate water leaf planted in a 60 cm by 60 cm loamy soil filled-box for 30 days where irrigation time was determined through threshold-based logic to optimize water use. The ESP32 microcontrollers was connected with soil moisture, temperature, and humidity sensors for data collection, followed by data transmission through a Message Queue Telemetry Transport (MQTT) to a *Mosquitto* broker. A mobile application was developed, Independent Flask micro-services employed to manage user authentication, farm registration, data monitoring, and irrigation control, supported by PostgreSQL for transactional data and InfluxDB for storage. The finding validated the feasibility of combining IoT, micro-services, and cloud infrastructure to deliver reliable, low-maintenance smart farming solutions, with potential for future AI-driven irrigation. SMIIS seamlessly demonstrated the capability for scalability for over 300 multiple concurrent users.

Keywords: IoT; Micro-Services; Automated Irrigation; Cloud Computing

OPEN ACCESS

Correspondence:
Eganoosi Esme Atojunere, Department of Soil Science and Biosystems Engineering, Fiji National University, Fiji,
E-mail: eganoosi.atojunere@fnu.ac.fj

Received Date: *13 May 2026*
Accepted Date: *15 Jun 2026*
Published Date: *17 Jun 2026*

Citation:
Ipinnimo O, Munezero IJ, Atojunere EE, Ogochukwu UC. Development of a Scalable Micro-Services IoT Irrigation System (SMIIS) Application for the Growth of Talinum Triangulare. WebLog J Comp Sci Technol. wjcst.2026.f1706. https://doi.org/10.5281/zenodo.21312642

Introduction

Irrigation farming enabled all-year-round food production, increasing crop yields and farmer incomes [1]. This is essential in regions where rain-fed agriculture is considered inadequate. Therefore, irrigation served as the main source of water to meet the crop water requirement. Because of water scarcity, water application to the growth of crops needs to be precise, as some irrigation methods often result in under- or over-irrigation, resulting in inefficient use of the scarce water resources [2]. However, modern irrigation techniques, characterized as "Agriculture 5.0," utilize smart devices to achieve precise irrigation and efficient freshwater management [3]. These are Internet of Things (IoT) devices that communicate with each other to monitor soil-water variables required for effective irrigation, such as soil moisture, evapotranspiration, effective rainfall, and temperature. As a result of these data-driven systems, irrigation farmers were prepared to make informed decisions on when and how to irrigate. While IoT devices have wide applications, pathogen control [4], wastewater treatment [5], pollutants detection in water supply [6], the use of this technology in precise water application for crop growth is increasing. Several researchers have explored IoT applications in irrigation systems with different report. For instance, in the SWAMP project reported by [2], shown in Figure 1, an IoT-based platform for precision irrigation utilizing FIWARE components and achieving scalability tests with up to 45,000 sensors. However, their system experienced instability at 15,000+ devices due to MongoDB bottlenecks and IoT Agent memory leaks. Moreso, the use of Arduino Controller System to supply water for the growth of Roselle Plant (*Hibiscus sabdariffa Linn*) has been tested [7], as well as Internet of Things (IoT) Enabled Drip Irrigation System (DIS) for the growth of *Allium fistulosum* [8]. While the software architecture of some of these IoTs were found remarkable for their specific function, delivering the accurate amount of water, theses existing solutions lack scalability with setbacks experienced when designed to maintain performance of large users such supplying water to large numbers of irrigation farmers at a go. It was discovered that

scalability is a function of the software architecture use, which defined the system components and their interactions. Traditionally, software architects employed Monolithic Architecture, where all services and components existed within a single codebase [9-12]. Although functional, this approach presented scalability challenges, as optimizing one service inevitably affected others, compromising fault tolerance [13]. In view of this, micro services architecture provided vast options, offering independent, loosely coupled, and modular services that can be optimized individually without impacting others [14-15]. Microservices architecture is particularly suited for large-scale agricultural applications, effectively managing the complex data generated by sensors that monitored soil and weather conditions needed for irrigation. The IRRISENS platform developed by [16], represented one of the most relevant research works that implemented a microservices-based IoT platform for commercial agriculture as shown in Figure 2. The system demonstrated robustness through data isolation across farms and handled 45,000 sensors through service replication. However, its limitations were basic linear regression for predictive analytics, inconsistent cloud latency (50-330 seconds), and a lack of automated failure alerts.

In addition, the AgriSens project reported by [17], presented the design of an IoT-based dynamic irrigation scheduling system. Based on their study, they proposed an algorithm for dynamic irrigation in the different phases of a crop's life cycle with manual irrigation based on the experience of the farmers or experts. This information is delivered to the farmers via a web portal, a visual display, and a cell phone respectively. There was a significant results concerning different performance metrics, such as data validation, packet delivery ratio, energy consumption, and failure rate in various climatic conditions and with dynamic irrigation variables. This gap in literature revealed that there was a need for IoT irrigation system that integrated the benefit of scalability obtainable with microservices architecture with robust cloud-native deployment strategies suitable for commercial-scale agriculture.

This paper addressed this gap by presenting a SMIIS irrigation development through hardware prototyping, microservices implementation, and performance evaluation under realistic load conditions.

Materials and Methods

System architecture development

The SMIIS system utilized three-layer architecture: IoT Device Layer, Backend Layer, and Mobile Application Layer. These adequately maintained seamless data flow and communication between components. Several studies reported by [18-28] on IoTs were thoroughly reviewed, and some of their findings were utilized in the development of SMIIS. The step-by-step operation of SMIIS is as shown in Figure 3. SMIIS was developed to irrigate water leaf planted in a 60 cm by 60 cm loamy soil filled-box for 30 days where irrigation time was determined through threshold-based logic to optimize water use. Information on the developed SMIIS's system backend and the aqua grow mobile application procedures were available at https://github.com/Optimus-Goch1/aqua-grow and https://github.com/Optimus-Goch1/aqua-grow-mobile respectively.

The IoT device layer of SMIIS

the hardware layer of SMIIS was purchased in Lagos Nigeria, consisted of an ESP32 microcontroller as the central processing unit equipped with a suite of sensors and control mechanisms as shown

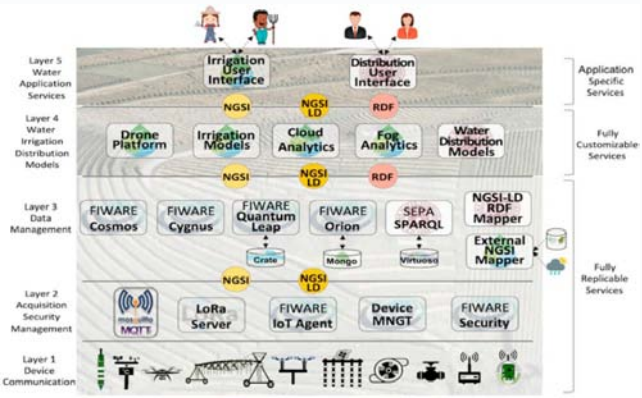

Figure 1: Architecture of the SWAMP [2].

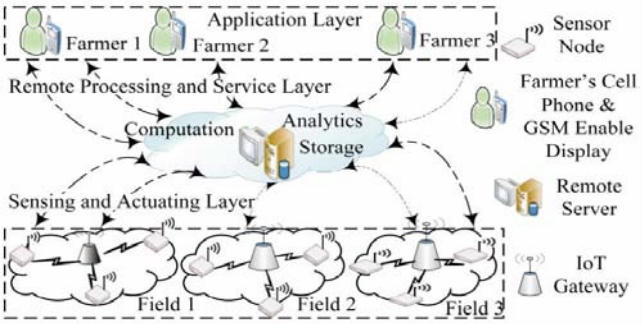

Figure 2: Modules and micro-services with the IRRISENS IoT platform [17].

in Figure 4. The sensors were capacitive soil moisture sensors that provided accurate, corrosion-resistant measurements of soil moisture and a DHT11 temperature and humidity sensors determined temperature and humidity data. If the moisture content were below the threshold for the growth of *Talinum triangulare* water delivery done by a 12V relay modules and a solenoid valve, which facilitated precise water flow control during irrigation cycles. Data were collected by the ESP32 from sensors, transmitted *via* Wi-Fi using the Message Queue Telemetry Transport (MQTT) protocol.

Backend micro-services layer of the SMIIS

The backend infrastructure controlled the four independent microservices constructed using the Flask framework, designed to operate independently to communicate through well-defined interfaces (Figure 5). The User Management Service handled user authentication and authorization, with JSON Web Token (JWT) providing a secure access control and farm registration capabilities. The Monitoring Service subscribed to MQTT topics published by ESP32 devices, processed real-time sensor data streams and storing temporal information in InfluxDB for efficient retrieval and analysis. SMIIS implemented irrigation control logic, supporting both automated threshold-based irrigation on predefined soil moistures, temperature and humidity data and manual control mechanisms accessible via the mobile application. The notification service generated and managed alerts for threshold violations, system anomalies and irrigation status updates. This was to keep the irrigation farmers abreast of critical system events. The communication between services was facilitated through RESTful APIs and MQTT messaging via a *Mosquitto* broker. This was an approach that ensured loose coupling and enables independent scalability of individual services.

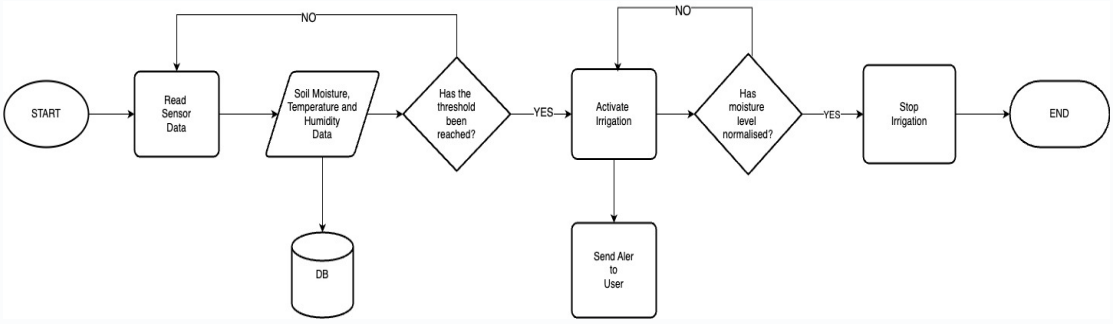

Figure 3: The algorithm of SMIIS System Architecture.

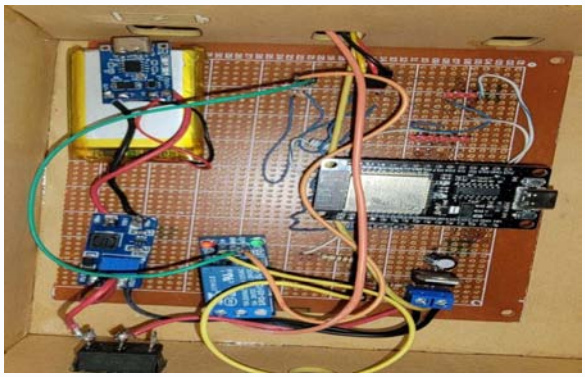

Figure 4: The interior view of the SMIIS hardware prototype.

Data storage architecture and the mobile application layer

This system employed a hybrid database approach tailored to different data characteristics and access patterns. For example, PostgreSQL managed transactional data encompassing user accounts, farm information, and configuration settings, providing robust ACID compliance for consistency-critical operations, whereas InfluxDB handled the high-frequency time-series sensor data, offering specialized optimization for write-heavy workloads and temporal queries that characterized sensor monitoring applications. A cross-platform mobile application was developed using React Native, an open-source framework, that provided potential irrigation farmers comprehensive system interaction capabilities.

The application enabled real-time visualization of sensor data alongside historical trending analysis, which facilitated farm management such as registration and configuration, permits irrigation threshold customization and manual control, delivers push notifications for system alerts, and carries out irrigation events.

The SMIIS development and cloud deployment

The Initial development of SMIIS utilized Docker Compose for local service orchestration, which enabled rapid prototyping and testing. This was followed by the containerization of each micro service using Docker to ensure consistent deployment environments and simplified dependency management. Production deployment was implemented on Amazon Web Services (AWS) infrastructure to ensure scalability and reliability at operational scale. Elastic Container Service (ECS) with Fargate technology provided serverless container orchestration while eliminating the need for manual server management. Request routing and service discovery across microservices were handled by Application Load Balancer (ALB) while Amazon RDS delivered managed PostgreSQL hosting with automated backup and recovery capabilities. InfluxDB was deployed on EC2 instances to provide customized time-series data management optimized for sensor workloads. The network configuration incorporated a Virtual Private Cloud (VPC) with appropriately configured security groups, establishing secure internal communication between services while maintaining public accessibility for mobile applications and IoT devices. System performance was evaluated using Locust, which was an open-source load testing framework capable of simulating thousands of concurrent users and IoT devices suitable for coordinating irrigation operations. The test environment consisted of all microservices (User, Monitoring, Irrigation), *Mosquitto* broker, InfluxDB, PostgreSQL, and the Locust load generator deployed together via Docker Compose. This setup enabled realistic stress tests to be run locally while maintaining service isolation. The SMIIS Hardware Prototype tested on waterleaf planter-box as shown in Figure 6.

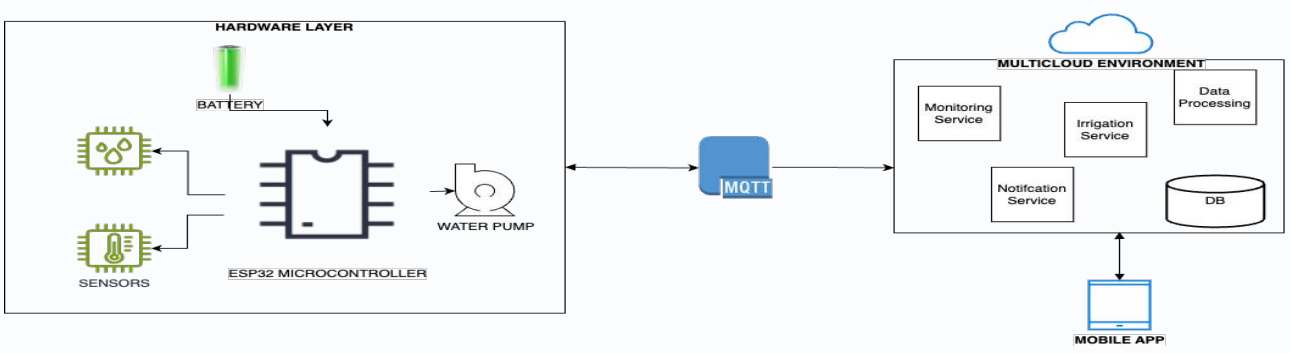

Figure 5: Software Architecture of the Application.

Figure 6: The testing of SMIIS for water application on the growth of water leaf.

The stress testing framework for scalability and robustness of SMIIS

1. The load testing tool was used to create a stress testing framework to evaluate the scalability and robustness of the SMIIS. It was selected based on the capability of Locust to allow simulation of thousands of concurrent users and devices and seamless integration into a containerized environment. The test scenario was designed to replicate real-world usage as fo>300 concurrent users (farmers) were simulated signing up, logging in, and interacting with the system.

2. Each user created approximately 10 farms, leading to a total of >300 simulated farms.

Each farm had a virtual ESP32 client that continuously published sensor data via MQTT at regular intervals. Users periodically retrieved the latest sensor values *via* REST API endpoints and occasionally triggered irrigation actions (manual ON/OFF).

Results and Discussions

The functional validation of the SMIIS

The initial testing of SMIIS confirmed successful end-to-end system integration when the ESP32 prototype was deployed to irrigate *Talinum triangulare*. Irrigation parameters were seamlessly collected and transmitted to backend services and triggered automated irrigation when the threshold soil moisture in the box was exceeded. The growth of the *Talinum triangulare* with the SMIIS-fed irrigation was steady, with more green leaf production throughout the experiment, indicating that the water requirement of the crop was met by the SMIIS-fed irrigation system. Sensor readings were also displayed on the mobile application for manual irrigation control. Data flow validation demonstrated seamless integration: sensor readings published to MQTT topics were immediately processed by the monitoring service and stored in InfluxDB, with mobile application queries returning current and historical data within acceptable latency bounds. SMIIS application precisely supplied the required amount of water for the growth of *Tilanum Traingulare*. This finding agreed with previous works on the efficiency of IoTs to precisely delivered water,for, household usage [6, 7], the growth of roselle (*Hibiscus sabdariffa Linn*) and *Allium Fistulosum* [8, 9], respectively. However, the advantage of SMIIS was its scalability, to accommodate more user up to 300 irrigation farmers at a time. This buttressed SMIIS suitability for commercial agriculture based on the micro service employed.

The scalability performance of SMIIS

Load testing results revealed strong system performance under increasing user loads. The system successfully handled 350+ concurrent users with corresponding farm and device simulations. Response time analysis showed initial latency spikes during load ramp-up due to connection pool initialization and resource allocation overhead. However, the performance stabilized as load distribution balanced, with median response times approaching near-zero values and 95th percentile latencies plateauing between 250-300 milliseconds. The throughput measurements demonstrated consistent request processing rates, with the system maintaining stable performance even under sustained high-load conditions. Failure rates remained minimal during moderate load scenarios, indicating robust system stability as shown in Figure 7. The response time metrics exhibited a transient spike followed by stabilization. Specifically, both the median (50th percentile) and high-tail (95th percentile) response times increased sharply as the number of concurrent users grew, reaching values above 300 milliseconds. However, as the load continued to increase, the response times subsequently decreased and plateaued. The 50th percentile stabilized near zero, while the 95th percentile plateaued between 250 and 300 milliseconds. This pattern suggested that the system initially experienced contention due to resource allocation overheads or database connection pooling, but once the load distribution stabilized, the system was able to process most requests efficiently. The plateau in the 95th percentile indicates that a minority of requests consistently incurred higher latency, reflecting bottlenecks that may require optimization at the database or service orchestration layers. Failure rates remained very low during moderate load conditions, indicating robust performance (Figure 8). However, as the number of simulated farms exceeded 250 and message frequency increased, MQTT message queues began to

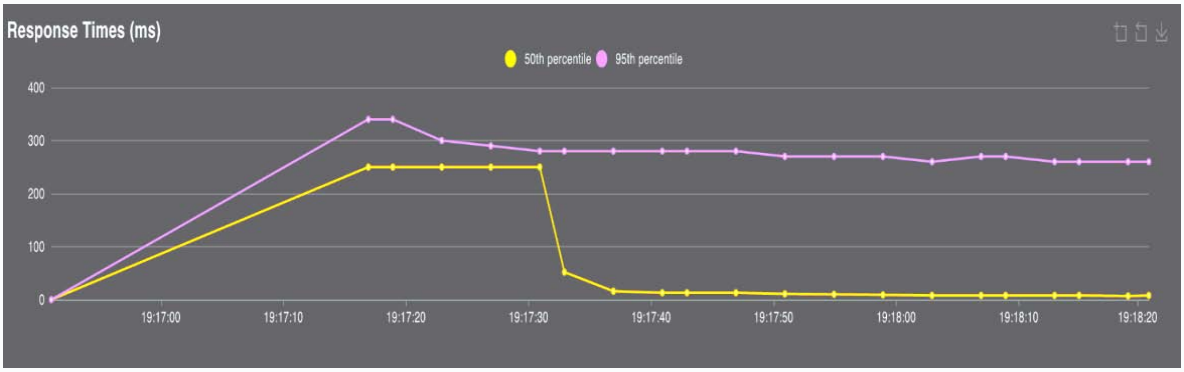

Figure 7: Total Response and Failures for SMIIS.

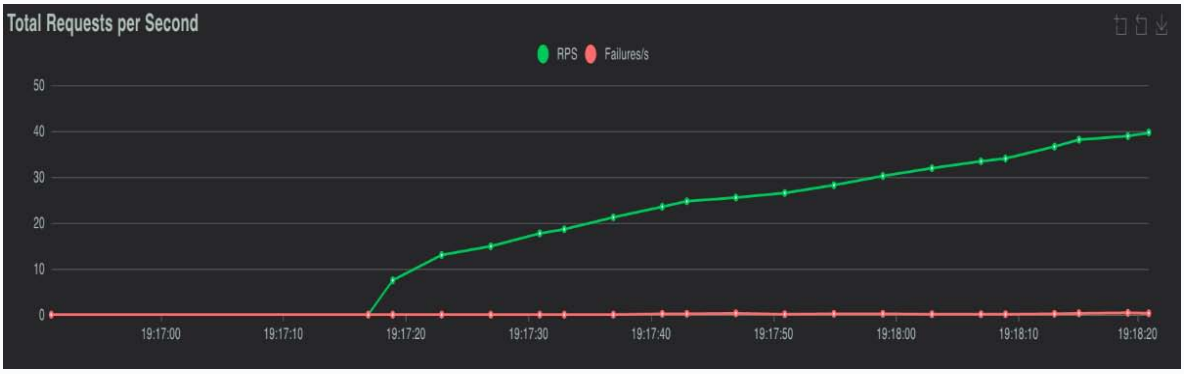

Figure 8: Graph of Total RPS and Failures for SMIIS.

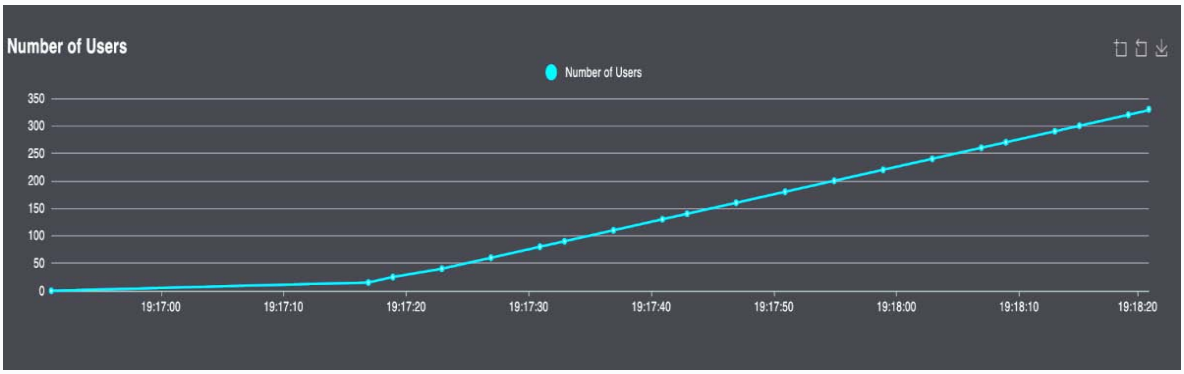

Figure 9: Number of Users on SMIIS.

exhibit slight delays. This suggested that while *Mosquitto* was stable at smaller scales, broker clustering or load balancing would be required to handle larger deployments.

The fault tolerance evaluation of the micro-services architecture of SMIIS

The micro-services architecture demonstrated an effective fault isolation capability. Simulated service failures in individual components, for example, Irrigation Service downtime did not affect the operation of other services, such as the User Management and Monitoring. This validated the loose coupling design principle. Authentication overhead analysis revealed that JWT verification introduced slight latency increases under high-frequency API calls, but remained within acceptable bounds for agricultural application requirements.

The stress testing framework for scalability and robustness of SMIIS

This structured testing framework provided a controlled environment to measure how the system behaves as load increases, highlighting performance limits and bottlenecks. In addition, it has provided valuable insights into the performance of the system. Locust generated graphs detailing the happenings of the test. The first graph detailed is the user graph, which shows the increase in users over time, reaching up to 350 users (Figure 9). Each simulated user performed actions on the system. The metrics captured during the stress test were:

- Response Time (Latency): Time taken for requests to complete.
- Requests per Second (RPS): Throughput achieved by the system under load.
- Failure Rate: Number of failed requests or dropped messages.

The bottleneck analysis of SMIIS

Performance testing identified the Monitoring Service and InfluxDB as primary bottlenecks under heavy sensor data ingestion loads. Query response times increased proportionally with time-series data volume, suggesting the need for database optimization or horizontal scaling strategies. MQTT broker performance remained stable for moderate device counts but showed signs of message queuing delays when simulated farms exceeded 250 concurrent devices publishing at high frequencies. This might suggest the need for broker clustering or load balancing for larger deployments.

Conclusion

This paper successfully demonstrates the design and implementation of a scalable IoT irrigation system utilizing micro-services architecture for commercial agriculture applications. The system addresses critical limitations of existing monolithic approaches by providing independent service scaling, fault isolation, and robust performance under concurrent user loads. The results validated the feasibility of micro-services-based IoT systems for smart agriculture, with the architecture supporting horizontal scaling, fault tolerance, and efficient resource utilization. Performance testing confirmed the system's readiness for commercial deployment while highlighting optimization opportunities in database query performance and MQTT broker clustering.

References

1. Bashir A and Kyung-Sook C. A review of the evaluation of irrigation practice in Nigeria: Past, present, and future prospects. African Journal of Agricultural Research, 2018; 13(40), 2087–2097. https://doi.org/10.5897/AJAR2018.13403
2. Kamienski C, Soininen J. P, Taumberger M, Dantas R, Toscano A, Salmon Cinotti T, Filev Maia R and Torre Neto A. Smart Water Management Platform: IoT-Based Precision Irrigation for Agriculture. Sensors, 2019; 19(2), Article 2. https://doi.org/10.3390/s19020276
3. Pandrea V.-A, Ciocoiu A.-O and Machedon-Pisu M. IoT-Based Irrigation System for Agriculture 5.0. 2023 17th International Conference on Engineering of Modern Electric Systems (EMES), 2023; 1–4. https://doi.org/10.1109/EMES58375.2023.10171631
4. Atojunere E. E and T. A Onaneye. Detecting Malaria Cells with Plasmo-D Expert System Based on Android and Computer, Vision,Vokasi Unesa Bulletin Engineering, Technology and Applied Science, Indonesia. 2025; 2(3): 503-515. https://doi.org/10.26740/vubeta.v2i3.39626
5. Fashanu T. A, Eche J. P. Akanmu, J. A. Adeyeye O. J. and Atojunere E.E. A Virtual Auto-desk-Simulink Reference Plant for wastewater Treatment, Nigerian Journal of Technology (NIJOTECH), Faculty of Engineering, University of Nigeria, 2019; 38(1): 267-277.
6. Atojunere E. E and Amiegbe G. E. Automated Leak and Water Quality Detection System for Piped water Supply, Malaysian Journal of Science, 2024; 43(3): 98-108. https://doi.org/10.22452/mjs.vol43no3.11
7. Atojunere E. E and Ogunleye G. A. Evaluation of a Developed Water Quality Monitoring System (WQMS) alongside weighted Arithmetic Water Quality Index Model, Journal of Environmental Engineering and Land Scape Management, Vilnius Gediminas Technical University, Lithuania. 2026.
8. Atojunere E. E and Ogundipe O. S. Arduino Controller System to supply water for the growth of Roselle Plant (Hibiscus sabdariffa Linn) AMA, Agricultural Mechanization in Asia, Africa and Latin America, 2022; 53(01): 6883-6895.
9. Atojunere E. E and B. Omotoro. Internet of Thıngs (IoT) Enabled Drip Irrigation System (DIS) for the growth of Allium Fistulosum, Selcuk University Journal of Engineering Sciences, Turkey, 2024. https://sujes.selcuk.edu.tr/sujes/article/view/654
10. Blinowski G, Ojdowska A and Przybyłek A. Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation. IEEE Access, 10, 20357–20374. IEEE Access. 2022. https://doi.org/10.1109/ACCESS.2022.3152803
11. Cambra C, Sendra S, Lloret J and Garcia L. An IoT service-oriented system for agriculture monitoring. 2017 IEEE International Conference on Communications (ICC), 2017; 1–6. https://doi.org/10.1109/ICC.2017.7996640
12. Cambra C, Sendra S, Lloret J and Lacuesta R. Smart System for Bicarbonate Control in Irrigation for Hydroponic Precision Farming. Sensors, 2018; 18(5), 1333. https://doi.org/10.3390/s18051333
13. Angelakıs A. N, Zaccaria D, Krasilnikoff J, Salgot M, Bazza M, Roccaro P, Jimenez B, Kumar A, Yinghua W, Baba A, Harrison J. A, Garduno-Jimenez A, and Fereres E. Irrigation of World Agricultural Lands: Evolution through the Millennia. Water, 2020; 12(5), 1285. https://doi.org/10.3390/w12051285
14. Ani A and Gopalakirishnan P. Automated Hydroponic Drip Irrigation Using Big Data. 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA), 2020; 370–375. https://doi.org/10.1109/ICIRCA48905.2020.9182908
15. Lewis J and Fowler M. Microservices: A definition of this new architectural term. 2014.
16. Filev Maia R, Ballester Lurbe C, Agrahari Baniya A and Hornbuckle J. IRRISENS: An IoT Platform Based on Microservices Applied in Commercial-Scale Crops Working in a Multi-Cloud Environment. Sensors, 2020; 20(24), 7163. https://doi.org/10.3390/s20247163
17. Roy S. K, Misra S, Raghuwanshi N. S and Das S. K. AgriSens: IoT-Based Dynamic Irrigation Scheduling System for Water Management of Irrigated Crops. IEEE Internet of Things Journal, 2021; 8(6), 5023–5030. IEEE Internet of Things Journal. https://doi.org/10.1109/JIOT.2020.3036126
18. Zaki F. M, Tajjudin M and Ishak N. IoT-Based System for Monitoring Smart Agriculture's Automated Irrigation. 2023 IEEE International Conference on Agrosystem Engineering, Technology and Applications (AGRETA), 2023; 135–141. https://doi.org/10.1109/AGRETA57740.2023.10262688
19. Kumar A, Tiwari R. G and Trivedi N. K. Smart Farming: Design and Implementation of an IoT-Based Automated Irrigation System for Precision Agriculture. 2023 3rd International Conference on Innovative Sustainable Computational Technologies (CISCT), 2023; 1–6. https://doi.org/10.1109/CISCT57197.2023.10351483
20. Palomar-Cosín E and García-Valls M. Flexible IoT Agriculture Systems for Irrigation Control Based on Software Services. Sensors, 2022; 22(24), 9999. https://doi.org/10.3390/s22249999
21. Gill R, Tripathi A and Chawla P. Designing a IoT based Prototype for Crop Monitoring and Smart Irrigation. 2022 2nd International Conference on Technological Advancements in Computational Sciences (ICTACS), 2022; 268–272. https://doi.org/10.1109/ICTACS56270.2022.9987789
22. Li S, Zhang H, Jia Z, Zhong C, Zhang C, Shan Z, Shen J and Babar M. A. Understanding and addressing quality attributes of microservices architecture: A Systematic literature review. Information and Software Technology, 2021; 131, 106449. https://doi.org/10.1016/j.infsof.2020.106449
23. Goap A, Sharma D, Shukla A. K and Rama Krishna C. An IoT based smart irrigation management system using Machine learning and open-source technologies. Computers and Electronics in Agriculture, 2018; 155, 41–49. https://doi.org/10.1016/j.compag.2018.09.040
24. Gupta B. B and Quamara M. An overview of Internet of Things (IoT): Architectural aspects, challenges, and protocols. Concurrency and Computation: Practice and Experience, 2020; 32(21), e4946. https://doi.org/10.1002/cpe.4946
25. Bruinsma J. World Agriculture: Towards 2015/2030 (0 ed.). Routledge. 2017. https://doi.org/10.4324/9781315083858
26. Garlan D. Software Architecture. In J. J. Marciniak (Ed.), Encyclopedia of Software Engineering (1st ed.). Wiley. 2002. https://doi.org/10.1002/0471028959.sof012
27. Kouroshfar E, Mirakhorli M, Bagheri H, Xiao L, Malek S and Cai Y. A Study on the Role of Software Architecture in the Evolution and Quality of Software. 2015 IEEE/ACM 12th Working Conference on Mining Software Repositories, 2015; 246–257. https://doi.org/10.1109/MSR.2015.30
28. Turral H, Svendsen M and Faures J. M. Investing in irrigation: Reviewing the past and looking to the future. Agricultural Water Management, 2010; 97(4), 551–560. https://doi.org/10.1016/j.agwat.2009.07.012llows: