Article

Microservices and IoT Architectures: Convergence and Application in Smart Agriculture

Imen Ismail *

Posted Date: 6 October 2025

doi: 10.20944/preprints202510.0400.v1

Keywords: IoT; microservices; microservice based architecture; IoT architecture; smart agriculture; container; containerization

Article

Microservices and IoT Architectures: Convergence and Application in Smart Agriculture

Imen Ismail

NOCCS Lab, Eniso, Innovation City, Sousse, Tunisia; imen.ben.ismail@gmail.com

Abstract

The emergence of Internet of Things (IoT) systems has provided another round of distributed, real-time, data-rich applications in different domains by enabling pervasive data collection and automation. However, traditional monolithic system designs struggle to accommodate the heterogeneity, scalability, and dynamic behavior of IoT environments. This paper explores how microservices architecture can enhance IoT systems by promoting modularity, resilience, and independent scalability. We discuss the synergy between IoT and microservices, and present the potential challenges and mitigation strategies. In these circumstances, using microservices and containerization allows IoT functionalities to be decomposed into fine-grained, loosely coupled, and independently deployable services. This modularization enables efficient continuous integration/ continuous delivery (CI/CD) practices, improves fault isolation, and guarantees system-wide robustness without compromising the performance of the IoT ecosystem. This paper explores the technical foundations and benefits of this architectural paradigm, focusing on smart agriculture as a case study.

Keywords: IoT; microservices; microservice based architecture; IoT architecture; smart agriculture; container; containerization

---

1. Introduction

The Internet of Things (IoT) is a network of interconnected systems that enables the direct identification of digital entities and physical objects and the retrieval, storage, transfer, and processing of associated data between the physical and virtual worlds through standardized and unified electronic identification systems and mobile wireless devices [1]. Given the anticipated proliferation of connected devices, what would be the implications if their management were to become a substantial challenge? In this regard, monolithic architectures represent rigid and explicit structures that lack the flexibility to adapt to changes in individual components or to reorganize rapidly. In contrast, the microservices architecture—recognized as a robust design pattern for cloud-native and large-scale distributed systems [2]—enables the decomposition of applications into smaller, relatively independent services. When applied to the Internet of Things, microservices can provide solutions to architectural limitations while maintaining rich, reliable, and responsive systems. Therefore, it is apparent that microservices architectures and the Internet of Things are two powerful, new technological approaches to modern computing systems. In this article, we will start with an introduction to the theoretical concepts that form the basis for understanding the different paradigms. Then, we will critically examine existing cases to study the current situation. Finally, we will present our proposed solution and approach to leveraging the advantages of microservice architectures to design and implement an IoT-based smart farm management system. We will focus on the irrigation part of this system. The other parts will be the subject of future work.

2. Theoretical Background and Related Work

This section provides a theoretical foundation and introduces key concepts, definitions, and the overall logic behind how IoT systems operate and how microservices enable scalable, modular, and efficient system design. These fundamentals form the basis for our proposal and practical implementation.

2.1. IoT-Based Architectures

IoT-based architectures can vary significantly, depending on the implementation. However, there is no single standard reference architecture for the IoT, as it encompasses a wide variety of technologies [1]. This means that there is no single model that can be followed for all possible implementations. A typical IoT architecture consists of several interconnected layers. In fact, IoT architectures are based on a layered organization that structures the flow of information from data collection to analysis. This structure facilitates the design, integration, and management of complex IoT systems. Typically, an IoT architecture consists of four to five main layers (see Figure 8).

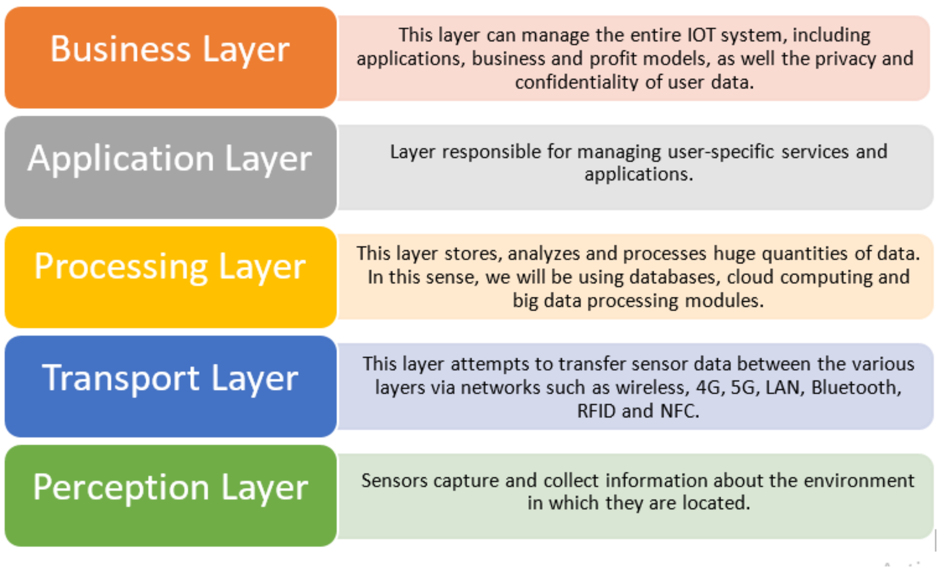
Business Layer
This layer can manage the entire IOT system, including applications, business and profit models, as well the privacy and confidentiality of user data.
Application Layer
Layer responsible for managing user-specific services and applications.
Processing Layer
This layer stores, analyzes and processes huge quantities of data. In this sense, we will be using databases, cloud computing and big data processing modules.
Transport Layer
This layer attempts to transfer sensor data between the various layers via networks such as wireless, 4G, 5G, LAN, Bluetooth, RFID and NFC.
Perception Layer
Sensors capture and collect information about the environment in which they are located.

Figure 1. IoT based architecture layers.

2.1.1. Perception Layer (or Physical Layer)

This layer is the foundation of the IoT architecture. It includes all the physical devices required to interact with the environment. These include: sensors (temperature, humidity, pressure, biometrics, etc.), actuators (switches, motors, water valve, etc.), and sometimes RFID or NFC devices. The role of this layer is to collect physical data and convert it into usable digital signals.

2.1.2. Network (or Transmission) Layer

After the data is collected, it must be transmitted to the processing units. The "network layer" is responsible for this, ensuring reliable and secure data transmission using technologies adapted to the system's requirements. Wireless communication technologies are generally used, and they are selected based on the required range, data rate, and energy consumption (ZigBee, Wi-Fi, Bluetooth Low Energy, NB-IoT, LoRaWAN and 5G). Communication protocols such as MQTT, HTTP and CoAP are also used to manage data exchange between devices and servers.

2.1.3. Processing Layer (or Middleware)

This is the intelligence layer of the architecture. It receives raw data from the network layer and is responsible for filtering, processing, storing, or redistributing it. Depending on the chosen architecture, processing can be done locally near the sensors (Edge computing), on intermediate nodes such as gateways (Fog computing), or centrally on remote servers (Cloud computing). The key functions encompass data analysis leveraging artificial intelligence and machine learning—stream management, automated decision-making processes, and integration with databases or business applications.

2.1.4. Application and Business Layers

The insights gained from data analysis are used to develop applications that offer value to end-users. These applications could be web or mobile apps that provide visualizations, alerts, control, and interaction with IoT devices. These applications aimed also, at delivering real-time information based on user needs and adjusting to their preferences. This layer is generally user-oriented. It provides interfaces, services, and dashboards that leverage processed data across various domains such as smart homes, healthcare and remote monitoring, smart agriculture, intelligent transportation systems and Industry 4.0. Key challenges include personalization, interface usability, and service reliability. The Business and user Layer is where end users and businesses interact with the IoT system. In this setting, they can receive insights, control devices, and make decisions based on the system's provided data and applications.

2.2. An Overview of Microservices Architecture

This section provides a brief description of the microservice paradigm and microservice-based architectures.

2.2.1. What are Microservices?

Microservices refer to a software development approach where an application is divided into a set of small, modular services that are loosely coupled and independently deployable. Each service is designed to perform a specific task such as data collection or analysis and interacts with others through standardized APIs (application programming interfaces), typically using lightweight protocols like HTTP/REST or MQTT (Figure 2).

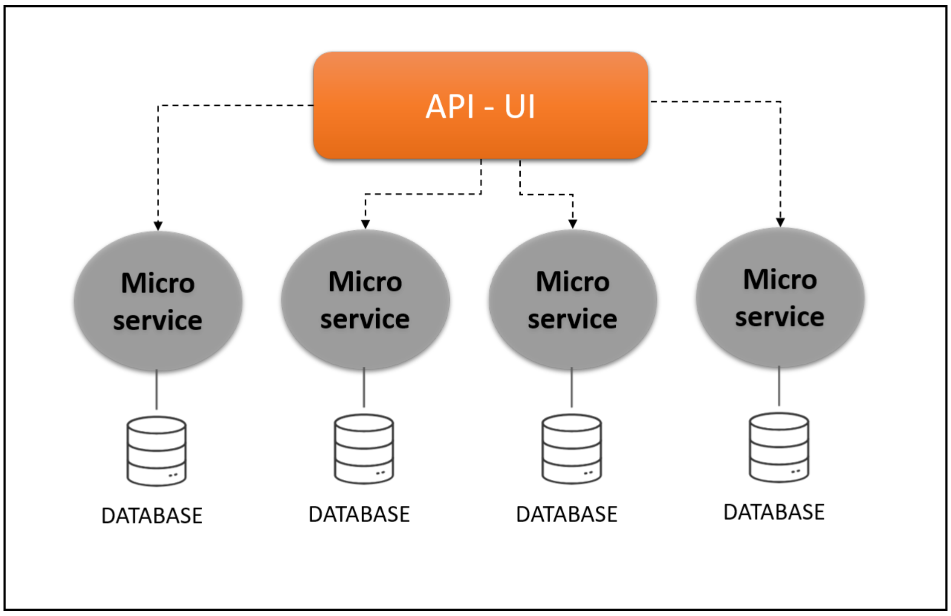

Figure 2. Typical example of microservices architecture.

2.2.2. Microservices-Based Architectures

Unlike monolithic architectures, which integrate all components into one system, microservices promote decentralized management and enable more precise scalability (see Figure 3). Microservices architecture [3] is characterized by several key features that enhance the flexibility and efficiency of modern software systems. Independence is a central aspect that allows each service to be developed, deployed, and updated independently without disrupting other services. This autonomy facilitates continuous integration and delivery. Modularity also improves system maintainability because each service is designed to perform a specific function, which makes debugging and updating easier. Additionally, lightweight communication mechanisms, such as application programming interfaces and messaging systems, enable efficient, standardized interactions between services. Finally, containerization, commonly through tools like Docker [7], plays a crucial role in microservices deployment by providing portability, scalability, and resource optimization across various environments.

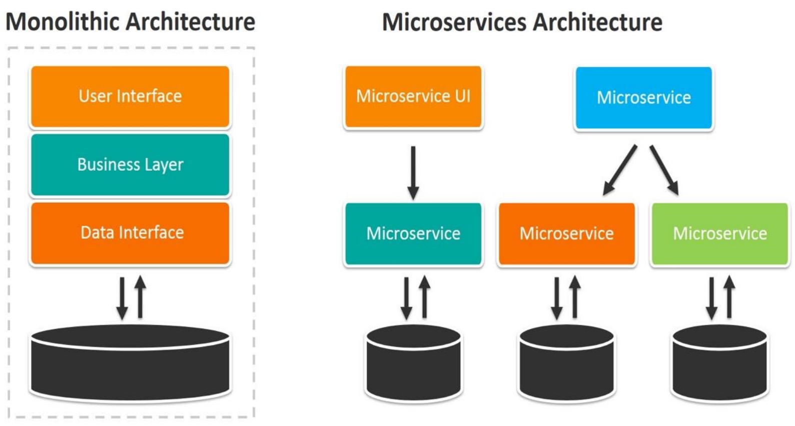

Figure 3. Monolithic vs Microservices architectures.

2.3. Related Work

Up to now, research converging towards the use of microservices architectures to support the Internet of Things (IoT) remains scarce. Nonetheless, there is a substantial ongoing effort to explore and implement such architectures in the IoT domain. From this perspective, three recent research efforts have been considered [4] [5] [6] while examining the potential limitations for implementing a large-scale and high-performance IoT ecosystem. These three works collectively offer a recent and valuable overview of microservice-oriented IoT systems, focusing on architecture design, service orchestration, and intelligent data analytics. For example, [4] brings practical insight by presenting a real-world deployment of a cloud–edge architecture. This work emphasizes orchestration and interoperability in heterogeneous environments. In addition, [6] advances the field by integrating Federated Learning and Transfer Learning into microservice architectures, addressing privacy-preserving analytics in distributed IoT scenarios. Finally, [5] just provides a broad state-of-the-art analysis while highlighting current trends in microservice adoption for IoT.

2.4. Critical Analysis of Existing Work

The reviewed literature reveals a consistent set of limitations across recent microservice-based IoT architectures. First, as highlighted by [5], problems with scalability and reliability persist, especially due to the lack of effective fail-over and recovery mechanisms. These issues are particularly pronounced at the edge, where services are inherently fragile. In [4] further underscore orchestration challenges are presented, noting that cloud–edge coordination often relies on manual configuration, which hinders automation and scalability. All studies acknowledge the fundamental constraint imposed by limited computational resources on IoT devices, which significantly restricts the deployment and execution of microservices at scale. Furthermore, the [6] study fails to consider the significant communication overhead and latency associated with Federated Learning, particularly in bandwidth-constrained environments. In terms of security, federated architectures are vulnerable to data manipulation, partial model updates, and centralized aggregation attacks. Despite efforts toward standardization, there is still no comprehensive middle-ware solution that addresses the heterogeneity of IoT systems, resulting in persistent interoperability issues.

3. Importance of Microservices Architecture in IoT Systems

After reviewing previous literature and conducting a general critical study, we will identify the main advantages of using microservice architectures to manage an IoT system.

3.1. *General Critical Study*

As previously mentioned, traditional IoT architectures typically follow a layered model. This model comprises a perception layer, which is responsible for data acquisition through sensors and actuators; a network layer, which enables communication; and an application layer, where data is processed and analyzed. While this architecture has proven effective for small-scale systems, it is limited for large-scale IoT deployments. Notably, scalability is constrained as centralized systems struggle to manage millions of interconnected devices. Furthermore, these types of architectures tend to be rigid. Updates often require full re-deployments, which can cause service disruptions. Latency issues also arise due to cloud-based processing, which can be critical in time-sensitive applications. Additionally, the complexity of managing heterogeneous devices and data streams grows with system scale. These limitations have driven the transition towards microservices-based architectures, which offer enhanced agility, scalability, and resilience in IoT systems.

3.2. *Objectives Study*

The integration of microservices into IoT architectures presents a natural and effective evolution toward addressing the growing complexity of interconnected systems. In fact, microservices architecture significantly simplifies the development and maintenance of large-scale, complex applications by enabling modular design and independent service deployment. In the context of IoT, users expect a seamless experience encompassing reliable connectivity, intuitive user interfaces, precise functionality, and high performance. Meeting these expectations necessitates frequent updates across various layers of the ecosystem, including the platform, devices, and associated applications. The adoption of DevOps practices, such as continuous integration, deployment, testing, and monitoring supports rapid iteration and delivery, enhances this agility even further. In addition, by decomposing functionalities into specialized, autonomous services, microservices enable a modular design in which individual components can handle distinct tasks such as sensor data collection, data analysis, or actuator control. These services can be deployed flexibly across edge devices, on-premise servers, or cloud infrastructures depending on operational requirements. This architectural approach directly addresses several challenges inherent to large-scale IoT systems. In terms of scalability, microservices allow independent scaling of each service. This makes it easier to accommodate an increasing number of devices and data flows without overloading the system, which is a key priority. Flexibility is enhanced because services can be updated or replaced independently, supporting seamless integration of new technologies. Additionally, the system's resilience improves because failures in individual services do not compromise the entire application. Eventually, microservices also promote agility by enabling parallel development and deployment. Additionally, real-time processing becomes more viable through edge deployment, which minimizes latency by processing data locally. This architectural style also aids in managing complexity by simplifying coordination across diverse components. Finally, well-designed microservices support re-usability, allowing components to be repurposed across multiple IoT projects, thereby reducing development costs.

3.3. *The Role of Containerization in Microservices-Based IoT Architectures*

The adoption of microservices and containerization emerges as a fundamental necessity for building scalable, maintainable, and future-ready IoT solutions. Microservices are typically organized around distinct business functionalities, deployed through automated pipelines, and interact using lightweight communication protocols such as RESTful APIs. However, the effective deployment and management of microservices in heterogeneous environments require a level of abstraction and portability that traditional deployment methods often fail to provide. This is where containerization becomes a critical enabler. Containers encapsulate microservices along with their runtime environment, dependencies, and libraries. This ensures consistent behavior across development, testing, and production environments. In containerized deployments, services can be launched or scaled independently without disrupting other containers running on the same host. This isolation simplifies resource management and significantly reduces deployment time.

3.4. How Microservices and Containerization Address Core Challenges in IoT

Containers offer the runtime isolation and dependency management (See Figure 9) necessary for seamlessly deploying these heterogeneous services [8]. This flexibility improves agility and reduces time-to-deployment when rolling out updates to devices or applications within the IoT environment. In contrast, when deployed through containerization, microservices provide a lightweight, scalable, and decentralized alternative. Each service can be independently developed, deployed, and scaled without impacting others, while containers ensure resource isolation and efficiency. This architectural synergy reduces build times, improves scalability, and minimizes infrastructure overhead, making it especially well-suited for dynamic and large-scale IoT environments.

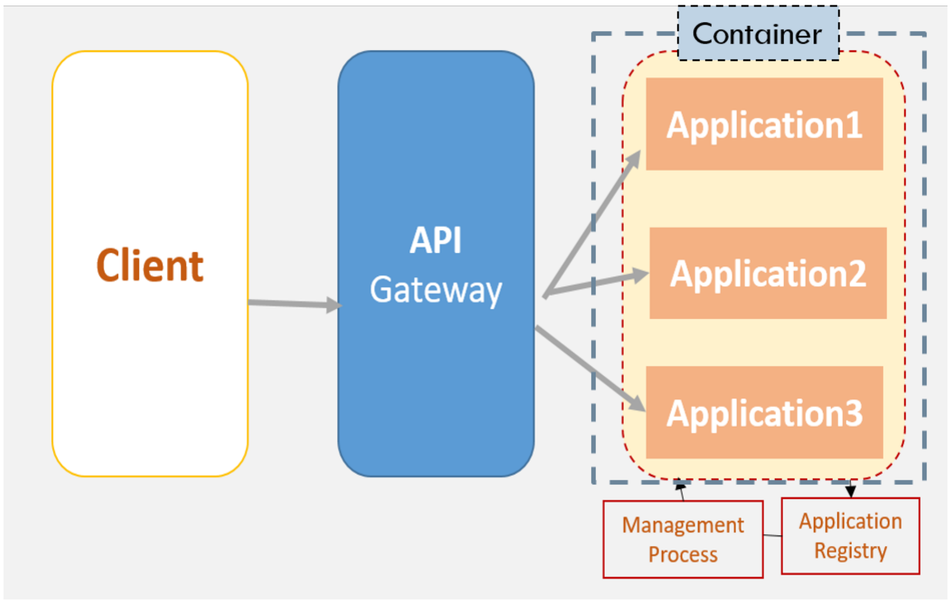

Figure 4. Containerization Process.

4. A General Microservices-Based Architecture Proposal

In response to the identified challenges, this paper proposes a pioneering solution in the form of an IoT-enabled smart irrigation system. By leveraging real-time data, advanced sensor technologies, and sophisticated data analytics, this system addresses the inefficiencies of traditional irrigation methods. The integration of soil moisture sensors and weather stations, coupled with actuation mechanisms and a central control unit, forms the backbone of the proposed solution. The smart irrigation system uses adaptive decision-making algorithms that respond dynamically to changing soil conditions, weather forecasts, and crop-specific requirements. This adaptability ensures an optimized irrigation schedule, minimizing water waste and maximizing resource efficiency. Actuation mechanisms driven by real-time data deliver precise, targeted irrigation, contributing to sustainable agricultural practices. Furthermore, integration the system with cloud platforms and mobile applications enhances accessibility and control. Farmers can remotely monitor and manage irrigation processes, making real-time adjustments based on evolving field conditions. This solution addresses the limitations of existing irrigation systems and aligns with the broader goals of sustainable and environmentally conscious agriculture. The proposed smart irrigation system is an innovative and comprehensive solution to the identified challenges. It promises to redefine precision agriculture and contribute to a more resilient and efficient future for global food production.

4.1. Irrigation System: Proposed Iot-based Approach

The following section presents a detailed illustration and description of the smart irrigation system's architecture (See Figure 5). Each component is explained, from the field sensors to the central control unit and the cloud-based platform. We will also discuss the role of IoT in facilitating seamless communication and data exchange.

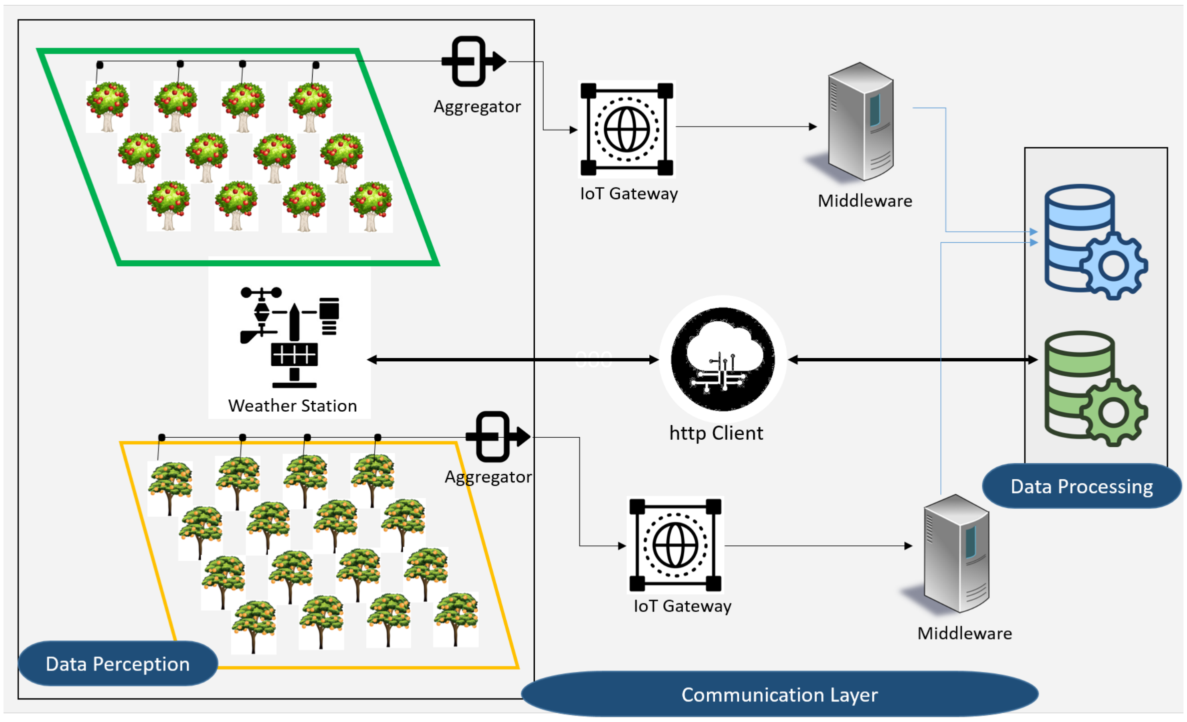

Figure 5. Global IoT based Architecture.

4.1.1. Field Sensors

At the core of the system are soil moisture sensors and weather stations deployed in the field. These sensors are strategically positioned to provide a comprehensive representation of environmental conditions. Soil moisture sensors measure hydration levels, while weather stations capture data such as temperature, humidity, and weather forecasts. This information is collected in real-time.

4.1.2. Central Control Unit

Sensor data are transmitted to the central control unit. This unit processes data using advanced analytics algorithms. These algorithms take into account soil conditions, weather forecasts, and specific crop requirements to determine an optimal irrigation schedule. The central control unit makes real-time adaptive decisions, ensuring a dynamic response to changing conditions.

4.1.3. Actuation Mechanisms

Decisions made by the central control unit are transmitted to the actuation mechanisms. These mechanisms, such as drip irrigation or sprinklers, are activated to implement the optimized irrigation schedule. This ensures precise and targeted water distribution to the crops, minimizing waste.

4.1.4. Cloud Platform

The system transmits the data it collects (may include the decisions it makes) to a cloud platform. This platform serves as a central hub for storing and processing data. It provides remote access and a scalable solution for handling large volume of data. Next, the cloud platform allows farmers to interact with the system from any location.

4.1.5. IoT Communication

IoT plays a crucial role in communication between the system components. Field sensors use IoT protocols to transmit data to the central control unit. Likewise, the central control unit communicates decisions to the actuation mechanisms via IoT protocols. This inter-connectivity enables the system to adapt to changing field conditions in real time, ensuring rapid communication.

4.2. Design and Implementation of a Microservice-Oriented Smart Irrigation System

The design and implementation methodology for the smart irrigation system involves a systematic approach that includes requirements analysis and sensor selection. The ultimate goal is to create a coherent system architecture by strategically placing sensors and formulating an accurate irrigation scheduling decision algorithm. Prototype implementation (Figure 6) involves the integration of selected components such as microcontrollers, soil moisture sensors, temperature sensors, light intensity sensors, ultrasonic sensors, and the sprinkler system. Real-world testing and validation is critical, with simulated testing ensuring functionality under various conditions and prototype testing validating the system's decision making and water distribution accuracy.

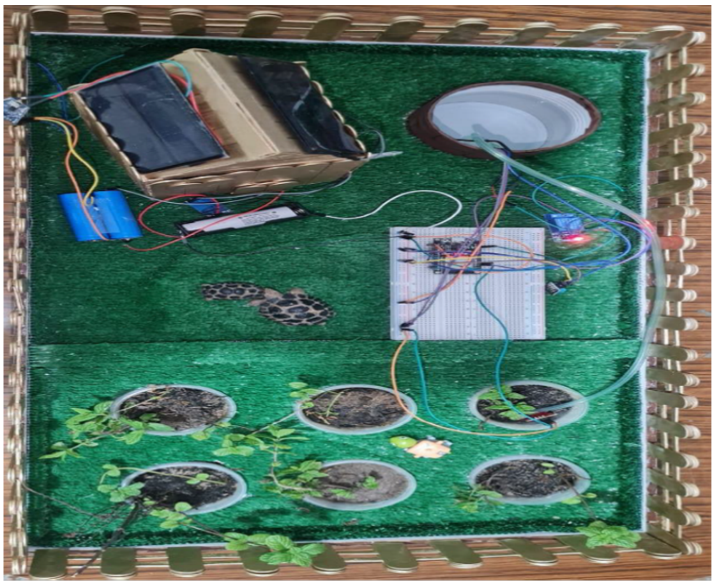

Figure 6. Test Prototype.

Analysis of the collected data is performed and an offline data storage mechanism via a micro-SD card module is implemented for comprehensive data tracking. User interfaces for monitoring and controlling the system are developed. An iterative refinement process, incorporating user feedback and continuous improvement, will ensure continuous improvement of system performance and responsiveness to environmental conditions, contributing to sustainable agricultural practices.

The ESP8266 module (see Figure 7) serves as data processor and Wi-Fi network server. The sensor reading is transmitted through the Wi-Fi network and sent to the web server. The data reading is displayed in the web browser, which can be accessed on an internet-connected computer.

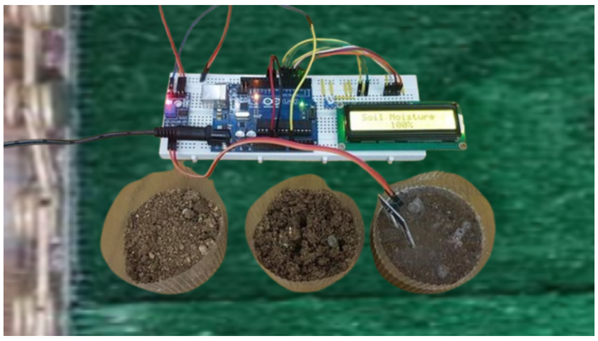
Figure 7. Unit Test Prototype.

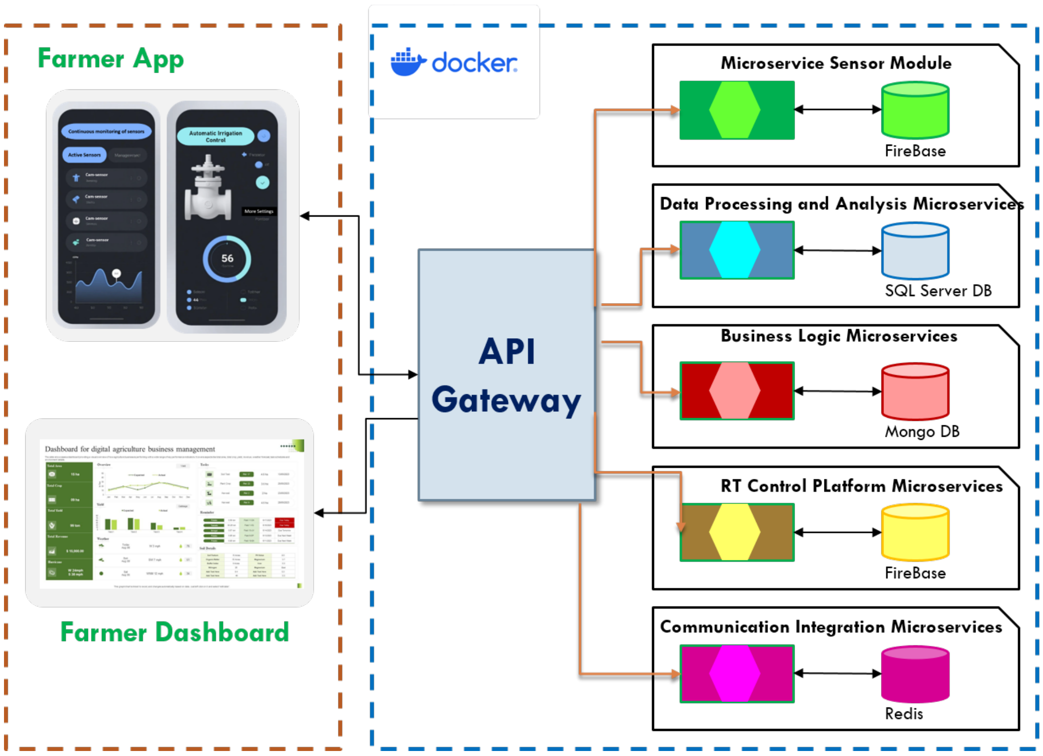
Farmer App
docker
Microservice Sensor Module
FireBase
Data Processing and Analysis Microservices
SQL Server DB
API Gateway
Business Logic Microservices
Mongo DB
RT Control PLatform Microservices
FireBase
Communication Integration Microservices
Redis
Farmer Dashboard

Figure 8. Microservices based Architecture.

4.3. Synergy Between IoT and Microservices in Our Proposed Architecture

The integration of the Internet of Things with microservices architectures provides substantial advantages in terms of performance and flexibility. Scalability is enhanced, as microservices allow specific functions—such as data processing— to scale independently without requiring an overhaul of the entire system. Additionally, modularity enables each IoT component, like data collectors or analytics engines, to be developed and maintained as separate services, simplifying updates and system evolution. Microservices are also compatible with edge computing, allowing local data processing on edge nodes. Furthermore, the use of APIs promotes interoperability among heterogeneous IoT devices and services, ensuring seamless system integration. A practical example of this approach can be seen in smart agriculture, where microservices for sensor data collection, weather forecasting, and irrigation control operate independently while communicating effectively.

4.4. Microservices for Sensors and IoT

In a microservice-oriented architecture for smart agriculture, sensor management and data acquisition represent fundamental components of the system. The Data Collection Service is responsible for continuously receiving information from field-deployed sensors, including measurements such as temperature, humidity, soil pH, and ambient light. This service interfaces with IoT devices through lightweight and efficient communication protocols ensuring reliable and realtime data transmission to the central infrastructure. In parallel, the Sensor Management microservice handles the registration, configuration, and monitoring of sensors, taking into account parameters such as sensor type, geographical location, and operational status. This functional separation promotes greater flexibility, improved maintainability of the system, and scalability suited to the dynamic demands of agricultural environments

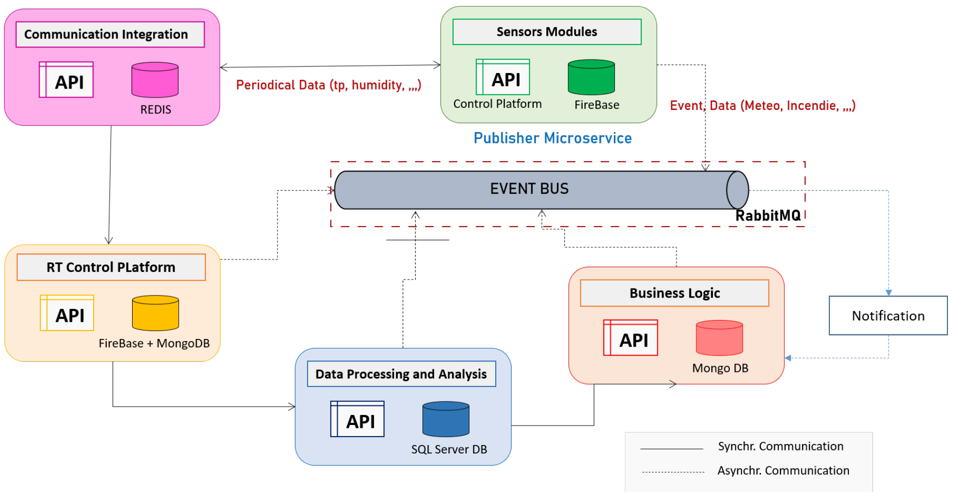

Figure 9. Logic Architecture.

4.5. Data Processing and Analysis Microservices

The Data Processing Microservice is designed to manage large-scale data flows in a distributed architecture. It leverages message queues and event-driven communication to receive raw data, applies validation and normalization rules, and enriches the datasets before publishing them to storage or analytics pipelines. Built for scalability, it runs in containerized environments such as Kubernetes and integrates with monitoring tools to ensure reliability, fault tolerance, and high availability.

4.6. Business Logic Microservices

The Business Logic Layer Microservice in a precision agriculture architecture is responsible for orchestrating decision-making processes based on processed data. It applies domain-specific rules and algorithms to transform agronomic insights—such as soil conditions, crop health, or weather forecasts—into actionable recommendations for farmers and automated systems. By centralizing business rules, it ensures consistency across applications, supports dynamic adjustments to agricultural strategies, and enables seamless integration with field devices, analytics platforms, and user-facing dashboards.

4.7. User Interface Microservices

The User Interface Microservice serves as the interaction point between end-users and the precision agriculture system. It delivers intuitive dashboards, maps, and visual analytics that allow farmers and stakeholders to monitor field conditions, review recommendations, and control automated processes in real time. By providing a responsive and user-friendly experience across web and mobile platforms, it ensures that complex agricultural insights are presented in a clear, actionable format, enabling better decision-making and improved farm management (See Figures 10 and 11).

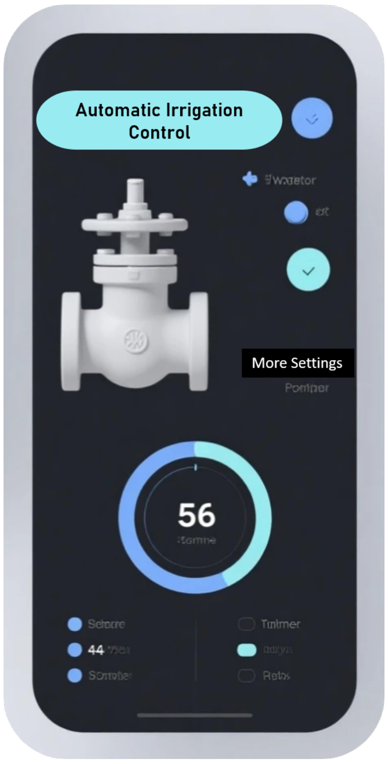

Figure 10. User Interface 1.

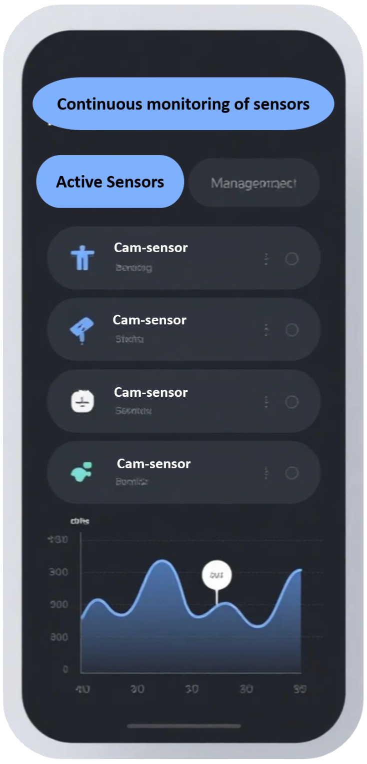

Figure 11. User Interface 2.

4.8. *Communication and Integration Microservices*

The Communication and Integration Microservice ensures seamless data exchange and interoperability between different components of the precision agriculture system. It manages the secure transfer of information across sensors, field devices, external APIs, and cloud services, using standardized protocols and middleware solutions. By enabling real-time synchronization and integration with third-party platforms—such as weather services, satellite imagery, or equipment management systems—it guarantees that all layers of the architecture operate cohesively, supporting efficient and scalable farm management.

5. Evaluation and Discussion

The architectural complexity of IoT ecosystems results from the coexistence of diverse components, including distributed servers, heterogeneous applications, various communication protocols and a wide array of endpoints such as embedded firmware, mobile clients, web interfaces and quality assurance tools. This multiplicity requires deep integration across physical devices, data pipelines, and application layers, which often leads to increased development effort, prolonged deployment cycles, and elevated maintenance costs, even within agile methodologies. In contrast, a microservices-based approach decomposes system functionality into atomic, loosely coupled, independently deployable services. This level of granularity enhances modularity and re-usability, and mitigates the integration challenges inherent in monolithic systems. Consequently, it facilitates the development of more agile and scalable IoT solutions. Combined with microservices, they offer modularity, resilience, and deployment flexibility, making them ideal for complex IoT applications. The precision agriculture use case demonstrates how these technologies enable real-time, sustainable farming practices. Future research should explore security enhancements and energy-efficient edge processing to further advance IoT-microservices integration.

6. Conclusions

Microservices architectures are essential for modern IoT systems, offering unmatched scalability, flexibility, resilience, and agility. They enable the management of complex IoT ecosystems, support real-time data processing, and adapt easily to rapid technological changes. Use cases in smart agriculture, smart ports, and connected vehicles highlight their practical impact. Looking ahead, research should focus on enhancing security, optimizing resource usage, and integrating emerging technologies such as artificial intelligence and edge computing. The integration of microservices architecture into IoT systems offers a transformative shift toward flexible, maintainable, and scalable solutions. This synergy is especially valuable as IoT systems become more complex and larger in scale. Although challenges remain, architectural best practices, tool-chains, and cloud-native platforms are evolving to support this integration. These advancements will enable the development of intelligent and resilient IoT applications in the future. Microservices is an efficient implementation that reduces the infrastructure usage and cost. It also reduces the maintenance cost of the code base. Furthermore, containerization effectively helps deploy microservices with best resource utilization and monitors this entire infrastructure through various frameworks, reducing operational overhead. Hence, an overall architecture for IoT based on containerized microservices for device management, communication protocols, cloud services, web and mobile services with REST APIs would solve many of the above challenges.

References

1. Imen Ismail, *Challenges and Comparative Analysis of IoT-Based Architectures: Towards More Resilient and Scalable Systems*, (Accepted in) Internation Journal of Smart Agriculture Vol3, 2025.
2. S. Newman, *Building Microservices: Designing Fine-Grained Systems.*, Sebastopol, CA, USA: O'Reilly Media, 2015.
3. Di Francesco, Lago and Malavolta, *Architecting with microservices: A systematic mapping study*, Journal of Systems and Software 150, pp.77–97 2019.
4. Roda-Sanchez, L. et al. *Cloud–edge microservices architecture and service orchestration: An integral solution for a real-world deployment experience*, Internet of Things, vol.23, 100777, 2023.
5. Siddiqui, H., Khendek, F., and Toeroe, M. *Microservices based architectures for IoT systems – State-of-the-art review* Internet of Things, Vol23, 100854. 2023
6. Ben Atitallah, S., Driss, M., and Ben Ghezala, H. *A microservices-based framework for IoT data analytics with federated and transfer learning* Internet of Things, Vol23, 100845. 2023

7. Liz Rice. *Container Security: Fundamental Technology Concepts that Protect Containerized Applications* Published by O'Reilly (2020).
8. Saha, Sourav and Rahman, Md Sazzadur and Islam, Md Mostafa and Karim, Md Rezaul. *Evaluation of Docker Containers for Scientific Workloads in the Cloud* Published by arXiv 1905.08415 (2019).