INSTITUTO POLITÉCNICO DE BRAGANÇA Escola Superior de Tecnologia e Gestão

ՀԱՅԱՍՏԱՆԻ ԱԶԳԱՅԻՆ ՊՈԼԻՏԵԽՆԻԿԱԿԱՆ ՀԱՄԱԼՍԱՐԱՆ

FIWARE-based Context Management and Visualization for an IoT-enabled Olive Grove

Lusine Avetisyan - a62875

Thesis presented to the School of Technology and Management in the scope of the Master in Electrical and Computer Engineering.

Supervisors:

Prof. João Paulo Coelho

Prof. Surik Khudaverdyan

Bragança.

2024-2025

INSTITUTO POLITÉCNICO DE BRAGANÇA Escola Superior de Tecnologia e Gestão

ՀԱՅԱՍՏԱՆԻ ԱԶԳԱՅԻՆ ՊՈԼԻՏԵԽՆԻԿԱԿԱՆ ՀԱՄԱԼՍԱՐԱՆ

FIWARE-based Context Management and Visualization for an IoT-enabled Olive Grove

Lusine Avetisyan - a62875

Thesis presented to the School of Technology and Management in the scope of the Master in Electrical and Computer Engineering.

Supervisors:

Prof. João Paulo Coelho

Prof. Surik Khudaverdyan

Bragança.

2024-2025

Dedication

I dedicate this work to my family, whose support, unwavering love, patience, and belief made it possible to lay a foundation for every success and gave me the strength to overcome every challenge throughout this journey.

This achievement is as much yours as it is mine.

Aknowledgments

I would like to express my deepest gratitude to my supervisor, Professor João Paulo Coelho, for his guidance, patience, and valuable recommendations throughout this work. His support and expertise were of great value to the success of this project.

I am also deeply thankful to my colleagues and friends who shared their ideas, feedback, and encouragement during this research process. They motivated me in moments of doubt, helped me move forward when the goal seemed out of reach, and reminded me to believe in myself again.

My sincere thanks go to my home institution, the National Polytechnic University of Armenia, where I built the foundation for my career, gained essential knowledge, and had the opportunity to join the Erasmus+ program — an experience that ultimately led me to the Polytechnic Institute of Bragança (IPB).

I am genuinely grateful to IPB for providing the resources, guidance, and environment that made this study possible. The knowledge and experiences I gained during this year have been truly invaluable.

Finally, I owe my deepest gratitude to my family for their unconditional love, encouragement, and belief in me. Their support has been my greatest strength and the reason I was able to reach this stage and complete this thesis.

Resumo

Para monitorizar as condições ambientais e apoiar decisões baseadas em dados, a agricultura de precisão depende cada vez mais das tecnologias da Internet das Coisas (IoT). Nos olivais, a gestão eficiente de recursos como a água e a fertilidade do solo é essencial para uma produção sustentável. Esta dissertação apresenta a integração do ecossistema FIWARE numa rede de sensores IoT baseada em LoRa, instalada num olival na região de Mirandela.

Com base no projeto anterior *Man4Health*, este trabalho contornos algumas das suas limitaçõe. Nomeadamente, a falta de contexto nos dados, de interoperabilidade e de capacidade analítica. O sistema proposto adota o Orion Context Broker do FIWARE como componente central para a gestão de dados de contexto em tempo real através do padrão NGSI-LD. Foi desenvolvido um painel Web para visualizar indicadores ambientais, como temperatura, humidade do solo e radiação solar, e emitir alertas quando os limiares são excedidos.

Embora o sistema tenha sido testado em campo, problemas técnicos e ambientais impediram a recolha contínua de dados. Para garantir uma avaliação consistente, foi criado um gerador de dados que simula medições realistas com a mesma estrutura dos dispositivos reais. O sistema demonstrou fiabilidade com dados reais e simulados, oferecendo visualização eficiente e gestão escalável de informação.

Palavras-chave: FIWARE, IoT, NGSI-LD, agricultura de precisão, olival, monitorização ambiental.

Abstract

To monitor environmental conditions and support data-driven decision-making, precision agriculture is increasingly relying on IoT technologies. In olive groves, the efficient management of resources such as water and soil fertility is vital for sustainable production. This thesis presents the integration of the FIWARE platform into an existing LoRa-based IoT sensor network deployed in an olive grove in the Mirandela region.

Building on the previous *Man4Health* project, this work addresses its main limitations, namely the lack of data context, interoperability, and analytical capability. The proposed system adopts the FIWARE Orion Context Broker as the central component for managing real-time contextual data through the NGSI-LD standard. A web-based dashboard was developed to visualize environmental indicators such as soil temperature, humidity, and solar radiation, and to generate alerts when thresholds are exceeded.

Although the system was tested in the field, technical and environmental issues prevented continuous data collection. To ensure consistent evaluation, a data generator was created to simulate realistic sensor readings using the same structure as the real devices. The system proved reliable with both real and simulated data, offering responsive visualization and scalable data management. Overall, this work demonstrates how FIWARE can strengthen smart farming platforms by promoting efficiency, interoperability, and environmental sustainability.

Keywords: FIWARE, IoT, NGSI-LD, Precision agriculture, olive grove, Environmental monitoring

Contents

List of Tables

List of Figures

Acronyms

API Application Programming Interface.

CIM Context Information Management.

CMF Context Management Framework.

CoAP Constrained Application Protocol.

DB Database.

DT Digital Twin.

ETSI European Telecommunications Standards Institute.

FIWARE Future Internet WARE.

GE Generic Enabler.

HTTP HyperText Transfer Protocol.

IoT Internet of Things.

LoRaWAN Long Range Wide Area Network.

MQTT Message Queuing Telemetry Transport.

NB-IoT Narrowband IoT.

NGSI-LD Next Generation Service Interface with Linked Data.

QoC Quality of Context.

RDF Resource Description Framework.

URN Uniform Resource Name.

Chapter 1

Intoduction

In recent years, agriculture has undergone a profound transformation driven by technological innovation. The emergence of the Internet of Things (IoT) and the increasing availability of low-cost sensors, wireless communication technologies, and cloud-based platforms have enabled a new generation of intelligent, data-driven agricultural systems, commonly known as precision agriculture.

Precision agriculture focuses on optimizing the use of natural and technical resources by monitoring and analyzing environmental variables such as soil moisture, temperature, humidity, and solar radiation. Through the integration of sensors, automation, and data analytics, farmers can make informed decisions that improve crop productivity, reduce waste, and promote sustainable practices.

However, one of the major challenges in implementing such systems lies in the integration and management of heterogeneous data sources. Many agricultural IoT solutions are built using proprietary data models and closed systems, resulting in fragmented and non-interoperable architectures. This limits data sharing, complicates system scalability, and hinders the adoption of advanced analytics and decision-support tools.

To overcome these challenges, open-source platforms such as the Future Internet WARE (FIWARE) ecosystem have emerged as key enablers for the development of standardized, interoperable, and scalable IoT systems. FIWARE provides a set of Generic Enabler (GE)s that enable real-time Context Information Management (CIM), semantic

interoperability through the Next Generation Service Interface with Linked Data (NGSI-LD) data model, and integration with other technologies such as Database (DB)s and visualization tools.

This thesis builds upon the Man4Health project, which focused on improving soil quality and olive production through experimental field studies in the Mirandela region. The project employed a network of IoT sensors to collect environmental and soil data; however, the initial architecture lacked semantic context, interoperability, and long-term data management capabilities. This limitation made it difficult to correlate measurements, analyze long-term patterns, and integrate external tools for visualization and analysis.

In this context, the present work proposes the integration of the FIWARE ecosystem into the existing IoT infrastructure of the Man4Health project, transforming it into a context-aware, interoperable, and scalable smart agriculture platform. The new system standardizes data flows, improves real-time monitoring, enables structured data persistence, and facilitates visualization through modern web-based dashboards.

1.1 Objectives

The main objective of this thesis is to design and implement a FIWARE-based data management system for environmental monitoring in precision agriculture, specifically applied to olive grove management.

To achieve this goal, three specific objectives were defined:

1. **Implement Context Data Management:** Integrate the FIWARE Orion-LD Context Broker and IoT Agent components to provide standardized, NGSI-LD-based context information for sensor data collected in the olive grove.

2. **Ensure Data Persistence and Accessibility:** Establish a scalable data pipeline that separates real-time context management from historical data storage, ensuring efficient long-term analysis through InfluxDB.

3. **Develop a Web-based Visualization Platform:** Design and deploy a Grafana-based dashboard to visualize environmental parameters, trends, and alerts, providing actionable insights for decision-making.

These objectives address the primary limitations of the previous system: lack of semantic data representation, poor interoperability, and absence of automated visualization tools.

The implementation of a FIWARE-based platform contributes to both research and technological innovation in the field of smart agriculture. From a scientific perspective, this work demonstrates how context-aware computing principles can be applied to real-world agricultural systems to enhance data usability and interoperability. From a technological standpoint, it provides a modular and replicable architecture that can be easily extended to other domains or integrated with complementary technologies such as artificial intelligence, predictive analytics, or blockchain-based traceability systems.

The resulting system represents a practical and scalable solution that supports sustainable agricultural practices by providing farmers and researchers with a comprehensive, real-time understanding of environmental dynamics.

1.2 Structure of the Document

This thesis is organized into six chapters, each addressing a distinct stage of the research and implementation process:

- **Chapter 2** – *State of the Art:* Provides a literature review and conceptual background on precision agriculture, IoT architectures, CIM, and the FIWARE ecosystem, highlighting the technologies and standards that support this work.
- **Chapter 3** – *Methodology:* Describes the experimental environment, including the original Man4Health infrastructure, the IoT sensor network, and the data collection procedures. It also introduces the analysis of the existing system and the methodological approach used for integrating FIWARE components.

- **Chapter 4** – *Implementation:* Details the development and deployment of the FIWARE-based architecture, including Dockerized services, data flow design, entity modeling, and integration with InfluxDB and Grafana for visualization.
- **Chapter 5** – *Conclusions and Future Work:* Summarizes the main outcomes and contributions of the project, discusses its limitations, and proposes directions for future research and technological development.

In summary, this thesis aims to demonstrate how FIWARE can be effectively used as an enabling technology for precision agriculture, turning isolated sensor networks into intelligent and interoperable data ecosystems that support sustainable agricultural management.

Chapter 2

State of the Art

Precision agriculture is a data-driven approach to farming that relies on advanced technologies and automation to manage fields and livestock more efficiently and sustainably. By collecting and analyzing environmental and operational data, farmers and researchers can identify the specific needs of different areas within a field and apply resources where and when they are required.

In recent years, research in intelligent and context-aware computing has increasingly focused on integrating open-source data management platforms such as FIWARE into real-world agricultural environments. These studies demonstrate how interoperable architectures can enhance automation, scalability, and data sharing across diverse domains including the IoT, smart cities, and precision agriculture. [1].

By adopting the FIWARE platform—particularly the Orion-LD Context Broker and the NGSI-LD data model—the project ensures semantic interoperability, standardized data representation, and the ability to integrate additional sensors or analytical tools in the future. This work therefore contributes to the advancement of precision agriculture by enabling automated environmental monitoring, improved decision-making, and sustainable management of agricultural resources.

This chapter presents the theoretical and technological background that supports the development of the proposed system. It begins by contextualizing the problem and the relevance of precision agriculture, followed by an overview of the IoT and its application

in agricultural monitoring. Next, the FIWARE platform and the NGSI-LD standard are described in detail, along with related research and the tools and technologies employed.

2.1 Precision Agriculture and Context Management

The evolution of precision agriculture has been driven by the need to optimize agricultural resources through the integration of data-driven technologies. However, despite its proven potential to increase efficiency and sustainability, the adoption of precision agriculture practices remains limited in several regions, particularly in developing agricultural contexts. Katke and Goud [2] conducted a study, highlighting the main barriers that farmers face when adopting precision agriculture technologies. According to their findings, the most common challenges include the high initial cost of equipment, insufficient technical knowledge among farmers, lack of data infrastructure, and limited interoperability between devices from different manufacturers. These barriers result in fragmented data collection and underutilization of available information, ultimately reducing the effectiveness of technology-driven decision-making processes.

These limitations reinforce the need for standardized and accessible solutions that can integrate heterogeneous data sources, reduce operational complexity, and support decision-making in real time. Open-source ecosystems such as FIWARE have emerged as a response to this challenge, offering scalable and interoperable frameworks that enable the efficient management of contextual data across multiple domains, including agriculture. By providing a common information model and open interfaces, FIWARE facilitates the creation of applications capable of connecting diverse IoT devices and data platforms into a unified system.

In parallel, research on CIM has focused on structuring and processing dynamic data generated by distributed systems. Liao, He, and Tang [3] proposed a conceptual framework for CIM that outlines the processes of data collection, representation, reasoning, and dissemination. Their model emphasizes the need to transform raw sensor data into higher-level contextual knowledge that can be interpreted by intelligent systems. This

approach is particularly relevant for IoT applications, where vast amounts of data are continuously produced by sensors operating in heterogeneous environments.

The principles outlined by Liao et al. [3] align closely with the context management mechanisms employed in FIWARE. The FIWARE Orion-LD Context Broker, for instance, applies similar concepts by maintaining real-time representations of entities, their properties, and relationships in accordance with the NGSI-LD standard. Through this architecture, data from the field—such as soil temperature, humidity, or CO_2 concentration—can be aggregated, structured, and shared with analytical tools or dashboards, transforming raw measurements into actionable insights. In the context of precision agriculture, this ability to manage and contextualize environmental data is essential for developing efficient decision-support systems and promoting sustainable farming practices.

The combination of standardized context management and open-source interoperability offered by FIWARE represents a significant step toward overcoming the technological and organizational challenges identified in earlier studies such as Katke and Goud [2]. By bridging the gap between data collection and actionable intelligence, such architectures enable the practical implementation of smart farming systems that are both scalable and accessible to a wide range of users. This thesis builds upon these concepts by applying the FIWARE framework to the management of environmental data collected from an olive grove, demonstrating its potential to enhance data integration, visualization, and real-time decision-making in precision agriculture.

2.2 Internet of Things in Agriculture

The Internet of Things (IoT) refers to a network of interconnected physical devices—such as sensors, actuators, and embedded systems—that can collect, exchange, and process data autonomously through the Internet [4]. IoT has become a key enabler of digital transformation across multiple domains, allowing real-time monitoring, intelligent control, and data-driven decision making. In agriculture, IoT technologies play a vital role in addressing the challenges of resource optimization, environmental sustainability, and

productivity improvement through continuous monitoring and automation [5].

IoT-based agricultural systems typically consist of several interconnected layers [4]. At the perception layer, various sensors and actuators are deployed in the field to collect physical and environmental parameters such as soil moisture, temperature, humidity, and solar radiation. The network layer is responsible for transmitting this data to gateways or cloud servers through communication protocols such as Long Range Wide Area Network (LoRaWAN), Narrowband IoT (NB-IoT), Zigbee, or Wi-Fi, depending on the range and bandwidth requirements of the application [6]. The middleware layer handles data aggregation, storage, and management, often through platforms such as FIWARE or Message Queuing Telemetry Transport (MQTT) brokers [1]. Finally, the application layer provides user interfaces, dashboards, and analytics tools that transform the raw sensor data into actionable insights for farmers, researchers, or decision makers.

Among the various communication technologies used in agriculture, LoRaWAN is one of the most widely adopted due to its long-range communication capability and low power consumption, which make it suitable for large-scale rural deployments. It operates in unlicensed frequency bands and allows battery-powered sensor nodes to transmit data over several kilometers with minimal energy usage. Alternatively, NB-IoT provides a cellular-based solution that supports higher data reliability and integration with existing mobile network infrastructure, albeit with greater power demands and operational costs. Protocols such as MQTT are commonly used to ensure lightweight and efficient message exchange between IoT devices and data servers, especially in bandwidth-constrained environments.

Despite the advantages of IoT adoption in agriculture, several challenges persist. These include the high initial cost of equipment, limited network coverage in rural areas, energy constraints of field devices, and the lack of standardized data formats for interoperability across different platforms. [7] Additionally, managing the large volume of heterogeneous data produced by sensors requires robust data management frameworks and context-awareness mechanisms to extract meaningful information. The integration of IoT with

open-source platforms such as FIWARE addresses many of these issues by enabling standardized data representation, real-time context management, and seamless interaction between devices and analytical services.

In the context of this project, IoT technologies form the backbone of the data acquisition process. Sensor nodes deployed in the olive grove continuously measure parameters such as soil temperature, humidity, and CO_2 concentration, transmitting the collected data via LoRaWAN to a central gateway. From there, the data is processed and stored within the FIWARE platform, where it is structured according to the NGSI-LD information model. This integration allows for real-time monitoring, historical data analysis, and the development of intelligent tools for supporting decision making in precision agriculture.

2.3 The FIWARE Ecosystem

The FIWARE ecosystem is an open-source platform designed to accelerate the development of smart, data-driven solutions across multiple domains, including smart cities, industry, energy, and agriculture. Rather than being a single product, FIWARE represents a framework of interoperable software components, known as GEs that provide standardized interfaces and Application Programming Interface (APIs) for CIM, data processing, and service orchestration. Its open and modular architecture allows developers and organizations to integrate heterogeneous data sources, IoT devices, and applications within a unified, interoperable environment.

The NGSI family of standards, originally defined under the European Open Platform for Smart Cities initiative and later extended to NGSI-LD by European Telecommunications Standards Institute (ETSI), provides the communication and data-modeling layer envisioned in early frameworks. NGSI-LD introduces a linked-data representation that supports semantic interoperability by expressing entities, their properties, and relationships in a machine-readable format based on JSON-LD. This allows different systems, sensors, and services to exchange contextual information in a unified and meaningful way.

Within the FIWARE architecture, the Orion-LD Context Broker serves as the operational realization of the "context provider" concept described by van Kranenburg et al. It maintains a dynamic digital representation of the real world—referred to as the context—by continuously updating entity states as new observations are received from IoT devices, applications, or external data sources. Orion-LD enables both real-time queries and event-driven subscriptions, ensuring that any change in context can immediately trigger corresponding actions or analyses. This event-oriented design directly reflects the reasoning and distribution layers of the early context-management frameworks but extends them with RESTful APIs, linked-data semantics, and scalable containerized deployments (e.g., via Docker and Kubernetes).

In this way, FIWARE operationalizes the theoretical foundations of context management into a practical, open-source platform capable of supporting modern IoT and smart-environment applications. By leveraging components such as Orion-LD, IoT Agents, and Cygnus, the framework ensures a seamless flow of information from heterogeneous data sources to decision-support systems—transforming the conceptual notion of context into a functional, standardized infrastructure for digital innovation.

2.3.1 Context Information Management

Context information refers to any data that helps describe the circumstances surrounding an entity, such as a person, location, or object, which is crucial for the interaction between a user and an application, including both the user and the application themselves. This key concept allows computing systems to adjust their actions and responses according to the surrounding environmental or operational conditions, thereby facilitating adaptive, intelligent, and context-aware applications.

An essential component of any context-aware system is the framework responsible for acquiring, processing, and distributing contextual information. One of the earliest and most influential models was proposed by van Kranenburg et al. [8], who introduced a

generic Context Management Framework (CMF) designed to support context-aware distributed applications. Their work defined the functional layers required for handling heterogeneous information sources and providing a unified interface to applications, thereby simplifying the development of adaptive and intelligent services.

According to van Kranenburg et al. [8], context management involves several key activities: *(i)* capturing or gathering context from distributed sensors or data providers; *(ii)* processing, interpreting, and reasoning about this data to derive meaningful information; and *(iii)* acting upon or distributing this information to consumers through standardized interfaces. The proposed CMF introduced three core components—Context Wrappers, which abstract low-level sensor data; Context Reasoners, which interpret and combine heterogeneous inputs; and Context Providers, which expose processed information to applications. This modular approach enables context-aware systems to operate across multiple domains while maintaining scalability and semantic consistency.

Figure 2.1 illustrates this continuous cycle of CIM. Data is first captured from sensors and devices, becoming contextual information when associated with entities, time, and location. This information is then processed to extract knowledge or detect relevant situations. Finally, the system actuates—either by triggering an application response, adjusting a parameter, or distributing updated context to other components. The cycle reflects the adaptive nature of modern context-aware architectures and forms the conceptual foundation for frameworks such as FIWARE's Orion-LD Context Broker, where context is dynamically captured, processed, and disseminated across IoT systems.

The authors highlighted the significance of semantic reasoning, ontology-based modeling, and quality-of-context Quality of Context (QoC) metrics to guarantee precise and dependable decision-making. These principles are closely aligned with modern context-management solutions such as the FIWARE Orion-LD Context Broker, which also utilizes standardized interfaces and linked-data semantics for the representation and exchange of contextual entities. Consequently, the work by van Kranenburg et al. [8] can be viewed as a conceptual forerunner to the current NGSI-LD-based frameworks used in IoT and smart-environment applications.

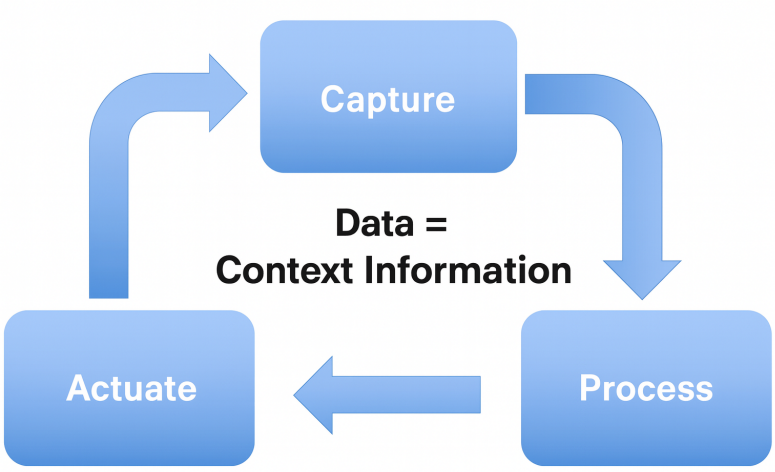

Figure 2.1: Cycle of CIM: captured, processed, and used to actuate decisions or services.

This conceptual principles laid the foundation for today's context-aware architectures. Their model—built around the acquisition, reasoning, and distribution of context information—anticipated many of the design choices that are now formalized within the FIWARE ecosystem. In modern implementations, these ideas have been materialized through standardized interfaces and reusable components that enable large-scale, cross-domain interoperability.

2.3.2 Generic Enablers in the FIWARE Ecosystem

The FIWARE ecosystem is composed of modular and reusable software components known as GEs. Each Generic Enabler provides a set of standardized and open interfaces that address specific functionalities required to build smart solutions in diverse domains, such as Smart Cities, Industry 4.0, Energy, and Smart Agriculture. Rather than being a monolithic platform, FIWARE offers a building block approach, where developers can combine different GEs to create tailor-made applications that share a common foundation of interoperability and open standards.

Definition and Purpose

A Generic Enabler is a software component that provides a general-purpose function, accessible through an open API (usually RESTful). The key objectives of GEs are to:

- promote reusability across different projects and domains,
- ensure interoperability between heterogeneous devices and systems,
- support open innovation by allowing third-party developers to integrate their own modules seamlessly,
- maintain standard compliance, primarily with NGSI and NGSI-LD specifications.

Each GE can operate independently but achieves its full potential when combined with others through FIWARE's standardized interfaces. The flexibility of this modular approach allows integrators to build distributed, scalable, and easily maintainable systems.

Types of Generic Enablers

FIWARE GEs are commonly grouped into four functional categories, each responsible for a specific layer of the system architecture:

1. Context Management: Handles real-time acquisition, storage, and distribution of contextual information. The most notable component in this group is the Orion-LD Context Broker, which acts as the central hub for managing and updating context entities based on the NGSI-LD standard.
2. Interface with IoT, Robots, and Third-Party Systems: Enables communication between heterogeneous IoT devices and the context management layer. The IoT Agents fall into this category, functioning as protocol translators that convert native device data into the standardized NGSI-LD format understood by the Context Broker.

3. Data Processing, Analysis, and Storage: Responsible for long-term data persistence, stream processing, and historical analysis. Examples include Cygnus, which exports context data to external databases or systems, and QuantumLeap, which stores time-series data in databases such as InfluxDB or TimescaleDB. In this project, a lightweight custom persistence adapter was developed to fulfill a similar function.

4. Visualization and User Interaction: Provides interfaces for data exploration and decision-making. While not part of the official FIWARE catalogue, visualization tools such as Grafana or Kibana are commonly integrated with FIWARE deployments to display data retrieved from the context or persistence layers.

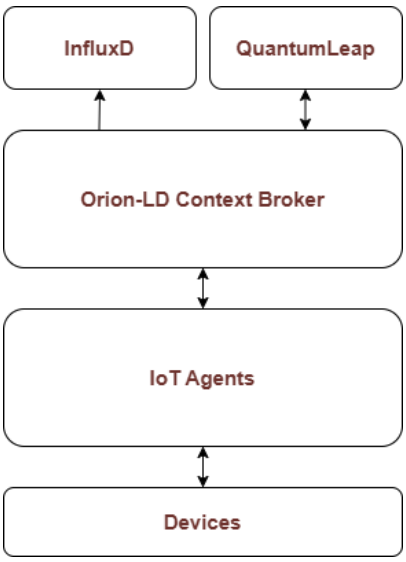

Figure 2.2: Diagram of FIWARE architecture

This modular design ensures that each component can be replaced, scaled, or upgraded independently, preserving the interoperability and flexibility that define the FIWARE philosophy. In the context of precision agriculture, this allows easy integration of new sensors, data sources, or analytical services as the system evolves.

2.3.3 NGSI-LD Information Model

The NGSI-LD information model defines a standard approach for representing, exchanging, and linking contextual information in distributed systems. Developed by the ETSI, NGSI-LD extends the original NGSI specification by incorporating principles of Linked Data and the Resource Description Framework (RDF) to achieve semantic interoperability across heterogeneous platforms [9]. It provides a unified data model and a RESTful APIs that allow applications to describe entities, their properties, and relationships in a consistent and machine-readable format.

In NGSI-LD, every entity represents a real-world object—such as a sensor, device, or geographic area—and is uniquely identified using a Uniform Resource Name (URN). Each entity has one or more properties that describe its characteristics (e.g., temperature, humidity, or status), and may include relationships that connect it to other entities (e.g., a soil sensor belonging to a specific olive plot). This semantic representation allows data from different systems to be linked, queried, and reasoned about in a context-aware manner, thus facilitating interoperability between IoT devices, services, and analytical tools.

The following example illustrates a simplified NGSI-LD entity representing a soil sensor used in this project:

```
{
  "id": "urn:ngsi-ld:SoilSensor:001",
  "type": "SoilSensor",
  "temperature": {
    "type": "Property",
    "value": 19.4,
    "unitCode": "CEL",
    "observedAt": "2025-06-01T09:00:00Z"
  },
  "humidity": {
    "type": "Property",
    "value": 0.31,
```

```
    "unitCode": "P1",
    "observedAt": "2025-06-01T09:00:00Z"
  },
  "co2Level": {
    "type": "Property",
    "value": 420,
    "unitCode": "PPM"
  },
  "belongsTo": {
    "type": "Relationship",
    "object": "urn:ngsi-ld:Plot:OliveField01"
  },
  "@context": [
    "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld"
  ]
}
```

Listing 2.1: Example of an NGSI-LD entity representing a soil sensor.

In this example, the `SoilSensor` entity has multiple properties (`temperature`, `humidity`, and `co2Level`) that include both values and metadata such as units of measurement and timestamps (`observedAt`). The `belongsTo` relationship links the sensor to another entity, `OliveField01`, which may represent a specific plot in the olive grove. The `@context` field provides references to ontologies and vocabularies, enabling semantic enrichment and compatibility with other NGSI-LD-compliant systems.

By adopting the NGSI-LD information model, the FIWARE platform enables a structured, semantic, and interoperable representation of context data. This approach allows information collected from IoT devices—such as soil temperature, humidity, and CO_2 concentration—to be shared seamlessly across applications and domains, forming the foundation for intelligent data analysis and decision support in precision agriculture.

2.3.4 Orion-LD Context Broker

The Orion-LD Context Broker is the central component of the FIWARE ecosystem. It implements the NGSI-LD standard defined by ETSI [9], providing a RESTful API for the creation, update, and querying of contextual information. The Context Broker stores real-time data in the form of entities, each representing a physical or logical object (e.g., a sensor, a farm plot, or a weather station). Applications and services can subscribe to context changes and receive notifications when relevant updates occur. This mechanism enables reactive and event-driven architectures, essential for IoT and smart-agriculture systems.

Role and Functionality

The Orion-LD Context Broker acts as a middleware that ensures seamless communication between data producers like sensors and data consumers like visualization tools or analytic applications. It receives, stores, and distributes information about entities and their states using a publish/subscribe model. When the state of an entity changes—such as an updated soil temperature measurement—the Context Broker immediately propagates this change to all subscribed systems, ensuring that every component in the architecture works with the most current and consistent information.

In the context of this project, Orion-LD serves as the single source of truth for all real-time environmental data originating from the olive grove. Every temperature, humidity, or CO_2 reading processed by the IoT Agent is transformed into a structured NGSI-LD update and sent to the Context Broker, which updates the relevant entity in its internal data store.

NGSI-LD Compatibility

Orion-LD fully implements the NGSI-LD API and data model, which extends the previous NGSIv2 standard with support for linked data and semantic relationships. Each entity managed by the broker is described using a unique identifier, a type, and a set of properties

and relationships.

- represent measurable characteristics: `soilTemperature`, `airHumidity`,
- Relationships describe associations between entities and states that a sensor belongs to a specific plot or weather station.

This semantic structure allows different systems to interpret and link data in a meaningful way. For example, two different sensors reporting soil humidity can be identified as instances of the same entity type while being logically connected to different geographical plots within the same dataset.

API Interactions and Entity Management

The Orion-LD Context Broker exposes a RESTful API through which external services can interact using standard HyperText Transfer Protocol (HTTP) methods. The main operations include:

- POST /entities — creates new entities: new sensors or environmental nodes,
- PATCH /entities/{id}/attrs — updates one or more attributes of an existing entity,
- GET /entities — queries existing entities or filters them based on type, location, or specific attribute values,
- DELETE /entities/{id} — removes an entity and its associated context data,
- POST /subscriptions — registers a subscription to receive notifications whenever selected attributes change.

An example of an NGSI-LD entity update, representing a soil sensor node, is shown below:

```
PATCH http://orion-ld:1026/ngsi-ld/v1/entities/urn:ngsi-ld:SoilSensor:001/attrs
Content-Type: application/ld+json

```

```
{
  "soilTemperature": {
    "type": "Property",
    "value": 23.7,
    "observedAt": "2025-10-28T10:45:00Z"
  },
  "soilHumidity": {
    "type": "Property",
    "value": 48.5,
    "observedAt": "2025-10-28T10:45:00Z"
  }
}
```

Listing 2.2: Example of an NGSI-LD entity update message sent to Orion-LD.

When this request is received, Orion-LD validates the payload according to the NGSI-LD schema and updates the corresponding entity in MongoDB. Each attribute is stored along with metadata such as the observation timestamp (`observedAt`), unit of measure, and provenance.

Event-Driven and Scalable Architecture

Orion-LD operates as an event-driven system, ensuring that only relevant components are notified when a specific context changes. This model reduces network traffic and allows the system to scale efficiently across distributed environments. Each event (such as a sensor reading update) is processed in near real-time, enabling responsive visualization and timely alert generation in the Grafana dashboard.

Furthermore, Orion-LD's RESTful API and NGSI-LD data model ensure compatibility with other FIWARE components and external services. This makes it possible to extend the system with advanced modules such as machine learning analytics, predictive irrigation models, or cross-domain integrations.

In summary, the Orion-LD Context Broker forms the core intelligence layer of the FIWARE-based architecture, transforming raw sensor data into semantically enriched,

standardized, and accessible context information that serves as the foundation for smart agricultural decision-making.

2.3.5 IoT Agents

IoT Agents act as translation layers between IoT devices and the Context Broker. They convert data transmitted using protocols such as LoRaWAN, MQTT, or HTTP into the standardized NGSI format, ensuring that sensor data can be ingested and managed consistently. IoT Agents also handle device provisioning, data normalization, and the mapping of raw measurements to NGSI-LD entities and attributes.

2.4 Digital Twins and the IoT

The integration of Digital Twin (DT)s and the IoT has become one of the most transformative paradigms in modern data-driven systems. A Digital Twin is a virtual representation of a physical entity that continuously receives data from sensors and IoT devices to replicate the real-time status, behavior, and performance of its physical counterpart [10]. This synchronization enables simulation, monitoring, and predictive maintenance in diverse domains such as smart manufacturing, agriculture, and urban infrastructure [11].

In an IoT ecosystem, interconnected devices collect and transmit data from the physical environment through communication protocols such as MQTT or Constrained Application Protocol (CoAP), enabling seamless data exchange between sensors, gateways, and cloud services [12]. When coupled with Digital Twins, these data streams empower advanced analytics and decision-making by linking the cyber and physical worlds in near real time [13].

In agriculture, for example, Digital Twins can model crop growth conditions and soil parameters, allowing farmers to simulate irrigation or fertilization strategies before applying them in the field. The IoT infrastructure provides the real-time environmental data—such as temperature, humidity, and soil moisture—needed to keep the virtual model accurate and adaptive [14]. Similarly, in industrial contexts, IoT-enabled Digital Twins

allow predictive maintenance of machinery, reducing downtime and operational costs by forecasting potential failures [15].

Overall, the convergence of IoT and Digital Twin technologies represents a crucial step toward achieving intelligent, autonomous systems that continuously learn, adapt, and optimize their performance based on real-world feedback.

2.4.1 The us of FIWARE in the context of Digital Twins

At the core of FIWARE, the Orion-LD Context Broker, is the actual realization of the Digital Twin. It shifts the system's focus from merely logging data to managing the current, living state (context) of the real world, enabling the creation of dynamic digital representations of physical assets. This functionality directly supports the synchronization between the physical and virtual worlds that defines a Digital Twin.

In a FIWARE-based Digital Twin, IoT Agents act as data acquisition gateways, converting measurements from heterogeneous sensors into NGSI-LD entities and transmitting them to the Context Broker. These entities represent the virtual counterpart of real-world assets, such as sensors, machinery, or agricultural plots, and are continuously updated as new observations are received. Through this mechanism, the Digital Twin can mirror the physical system in near real time, maintaining a consistent and semantically enriched data model that can be accessed or modified by analytical applications and dashboards [12].

FIWARE's interoperability also enables historical data persistence in time-series databases like InfluxDB. This historical layer is essential for Digital Twin applications that rely on predictive modeling, trend analysis, or anomaly detection. Similarly, advanced analytics or machine learning services can subscribe to context updates through the Orion-LD broker, enabling automated reasoning and decision support based on evolving environmental conditions.

Furthermore, the NGSI-LD information model provides the semantic framework required for linking entities and establishing relationships between them, such as connecting

a soil sensor to a specific field plot or correlating environmental parameters with crop health indicators [13]. This semantic linkage enhances the expressiveness and interoperability of Digital Twins, allowing them to operate as part of larger smart systems that span multiple domains—such as smart agriculture, energy management, or logistics.

In the agricultural domain, a FIWARE-based Digital Twin can therefore integrate real-time sensor data, historical records, and analytical models within a unified architecture. Such a system supports farmers and researchers in monitoring field conditions, simulating operational scenarios, and optimizing interventions in a data-driven and sustainable manner. By combining IoT connectivity, semantic data representation, and scalable open-source tools, FIWARE provides a practical and extensible foundation for building Digital Twin applications in precision agriculture and beyond.

The functional architecture relies on this strict segregation of roles provided by the main FIWARE components. Their specific contributions to establishing the authoritative Digital Twin state are detailed in Table 2.1.

FIWARE Component	**Digital Twin Function**	**Contribution to the DT**
Orion-LD Context Broker	Real-Time State Repository	Stores the Digital Twin. All applications query this broker to know the current temperature, humidity, etc.
NGSI-LD Standard	Semantic Modeling Language	Defines the structured, machine-readable language for the DT. It ensures the virtual entity is a standardized model.
IoT Agents	DT Data Ingress/-Translation	Acts as the standard gateway that receives raw data from sensors and translates it into NGSI-LD before feeding it to the Context Broker.

Table 2.1: FIWARE Component Roles in Establishing the DT Context Layer

2.5 Summary

This chapter presented the conceptual and technological foundations of the work. It described the principles of precision agriculture, the role of IoT and LoRa communication,

and the FIWARE platform with its NGSI-LD information model. The next chapter details the design and architecture of the proposed system, describing the interaction between IoT devices, FIWARE components, and data visualization modules.

Chapter 3

Methodology

This chapter describes the setup inherited from the Man4Health project and details the original architecture of the IoT node network. It identifies the inherent data management problems associated with the prior setup, particularly those related to semantic context and data storage. The solution implemented—achieved by designing a FIWARE-based data management system developed in this work—is then thoroughly documented. It also details the logical data flow, the crucial entity modeling according to the NGSI-LD standard, and the visualization approach used to extend the existing infrastructure into a fully automated and context-aware monitoring solution for the olive grove.

3.1 Background: The Man4Health Project

The main purpose of the previous project on which this thesis is based was to improve soil quality through the use of selected plant species. The introduction of specific vegetation aimed to protect the soil, promote olive productivity, and simultaneously reduce pests and diseases.

To evaluate the effectiveness of the proposed plant mixtures, a series of field experiments were carried out both manually and with the support of a sensor network installed in an olive grove located in the Mirandela region (geographical coordinates: 41°29'19.3"N,

7°14’53.6"W). The sensor network enabled the automatic collection of several physic-chemical parameters from both soil and air, providing continuous and reliable environmental data for analysis.

To accurately characterize soil properties, multiple sensor nodes were installed to measure and transmit information related to temperature, humidity, and CO_2 at various heights. In particular, to generate a three-dimensional spatial profile of biological activity by estimating the rate of soil respiration, it was decided to measure humidity, electrical conductivity, temperature, and CO_2 concentration at multiple depths (specifically at 5 cm, 10 cm, and 20 cm below the surface).

A weather station was also installed on a galvanized steel pole, which additionally housed the LoRa gateway, the electrical energy production and storage system, and a set of sensors designed to characterize the vertical profile of the lower atmosphere. This configuration made it possible to measure several atmospheric parameters, including air humidity, temperature, pressure, solar radiation intensity, and precipitation levels.

Since the field where the experiments were carried out did not have access to the electricity distribution network, the entire system had to rely on batteries to ensure continuous data collection over extended periods of time. To avoid the need for periodic battery replacement, a solar power solution was implemented. In particular, photovoltaic solar panels were installed to generate the necessary electrical energy to sustain the operation of all components, compensating for the power consumption of the sensor nodes, weather station, and especially the LoRa gateway.

3.1.1 General architecture of the sensor network

The Figure 3.1 illustrates the general layout of the sensor network. Each quadrangular plot, composed of four olive trees with average heights between two and three meters and represented by black circular markers, measures approximately 7 meters by 7 meters, corresponding to a total area of 49 m^2, dividing the land into 24 plots each of them being identified by an alphanumeric sequence consisting of a letter (A,B,C,D) and one

digit (1,2,3,4,5,6). In each of these plots, a sensor node was added, which was capable of acquiring environmental data and transmitting them to a LoRa hub.

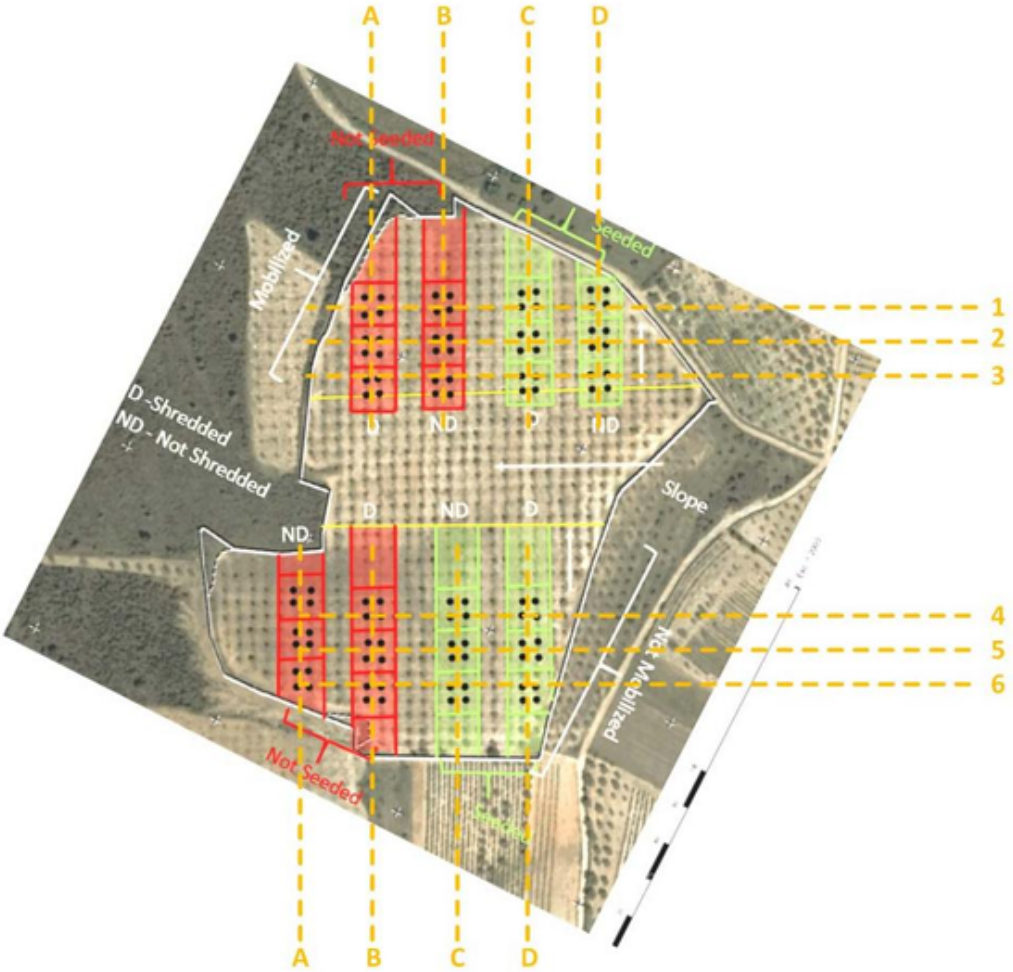

Figure 3.1: Map of the plots and their identification, positioning of measurement nodes.

3.1.2 IoT Node Network

As described in the previous sections, each experimental quadrant within the olive grove is equipped with an IoT node responsible for acquiring environmental data such as soil moisture and temperature at different depths. Each node operates as an energy-autonomous unit, powered by a local energy source (e.g., solar power), and includes a LoRa transceiver

that enables the wireless transmission of telemetry data [16]. This architecture allows for continuous and distributed environmental monitoring without dependence on external power or wired communication infrastructure.

The network employs the LoRaWAN protocol as its primary wireless communication technology. LoRaWAN was selected for its low power consumption, long transmission range, and robustness against interference—key characteristics for deployments in rural and agricultural environments [17]. The IoT nodes are organized in a star topology, with all nodes transmitting their data to a central LoRa gateway. This gateway acts as the hub of the local network and forwards the collected data to the project's central server through a 4G/LTE backbone connection.

Meteorological data collected from the weather station, along with telemetry related to the power supply system and other auxiliary devices, are also transmitted to the central server through the same 4G communication channel. Figure 3.2 illustrates the overall network architecture, showing the relationship between the sensor nodes, the LoRa gateway, and the project's data server.

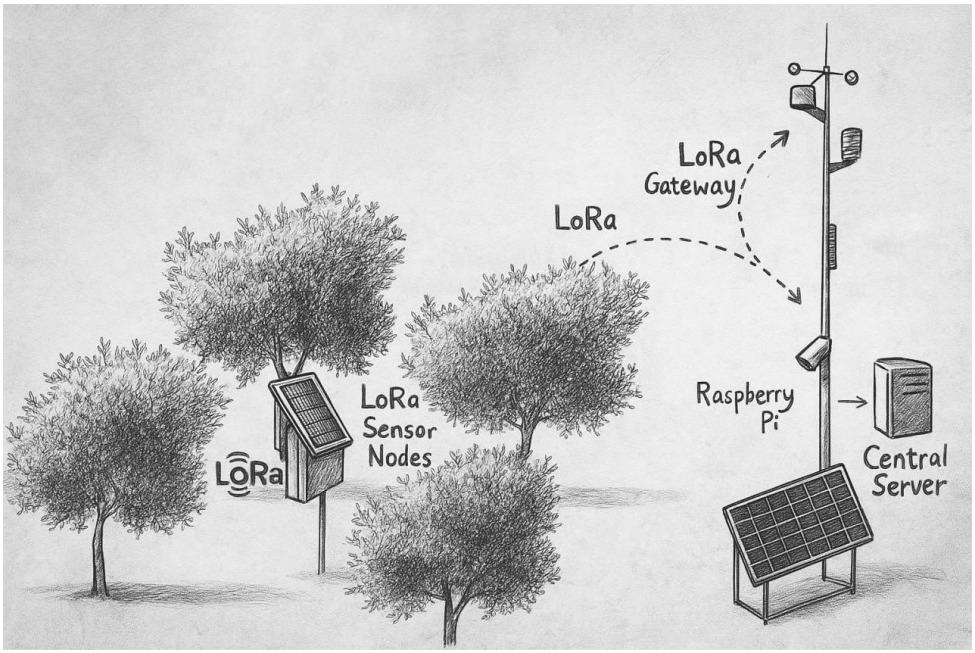

Figure 3.2: IoT network topology used in the olive grove.

Since the LoRa gateway must be installed in an area without access to the electrical

grid, an independent power supply system was required. This system was designed to ensure continuous 24/7 operation of the gateway and the associated electronics, including the energy management, communication, and environmental monitoring components. The following section details the design of this infrastructure, including the energy production, storage, and regulation mechanisms that sustain the IoT network in the field.

3.2 Analysis of the Inherited System and Data Flow

The operational foundation of this project is the existing IoT network deployed in the olive grove, characterized by its distributed sensor nodes and centralized communication hub.

3.2.1 Network Components and Data Aggregation

The field system relies on the following components to push data to the central server:

- **LoRa Sensor Nodes:** These are the primary data producers, collecting environmental parameters (soil temperature, humidity, CO_2) and transmitting data wirelessly via the LoRaWAN protocol.
- **LoRa Gateway:** This central hub receives all LoRa packets and is responsible for forwarding the data stream to the remote server using a robust 4G/LTE connection.
- **Raspberry Pi:** This device acts as a local data aggregator for auxiliary devices. It is responsible for collecting data from the weather station (atmospheric conditions, solar radiation) and the power supply regulator (system telemetry). The Raspberry Pi converts these specialized data streams into a network-ready format and publishes them to the central server via the same 4G/LTE channel, utilizing the Mosquitto MQTT Broker.

In the existing architecture, the central server receives all sensor data combined under

various topics in the Mosquitto MQTT Broker, which is the starting point for all server-side processing.

Actor	Role / Data Generated	Communication Path
LoRa Sensor Nodes	Primary data producers (T, H, CO_2, etc.).	LoRaWAN → LoRa Gateway
LoRa Gateway	Receives LoRa packets; provides 4G backhaul.	4G/LTE → Central Server
Raspberry Pi	Local Aggregator: Collects data from auxiliary devices	MQTT (via 4G/LTE) → Central Server
Central Server (VM)	Hosts Mosquitto MQTT Broker and legacy processing tools.	Receives all field data streams.

Table 3.1: System Components and Data Flow in the Inherited Architecture

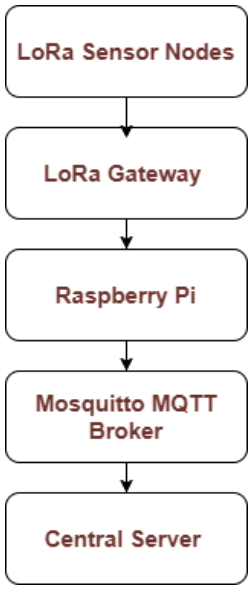

Figure 3.3: Diagram of the data flow in the Inherited Architecture

3.3 Problem Definition: Architectural and Context Deficiencies

While the infrastructure successfully collects data, its data management approach suffers from two critical deficiencies that the thesis aims to solve:

3.3.1 Lack of Context Information and Interoperability

The most significant problem is the unstructured nature of the data entering the server.

- **Raw Payloads:** Data arrives via MQTT as simple payloads (e.g., ‘{"t": 25.5, "h": 60}‘). Crucially, the system lacks an intrinsic context dictionary to differentiate between distinct sensor readings (e.g., whether ’t’ means air temperature from the weather station or soil temperature from a LoRa node).

- **Application Dependency:** Interpreting these raw keys requires external, hard-coded logic within every consuming application (e.g., Node-RED flows). This tightly couples data interpretation with application code, preventing the use of standardized tools and severely limiting interoperability.

3.3.2 Failure to Decouple Data Layers (Objective 2: Data Persistence)

The system lacks the necessary architectural separation between data concerns, compromising efficiency and scalability:

- **Inefficient Status Management:** There is no dedicated service to maintain the single, real-time current state of an entity (a Digital Twin). Consequently, any application needing the current soil temperature must query the large, heavy historical database, which leads to slow performance and unnecessary resource consumption.

- **Tight Coupling:** Data logging (history) and status monitoring (real-time) are managed by the same process, violating the principle of separation of concerns and making system maintenance complex.

3.4 Proposed Solution Architecture: The FIWARE Context Layer

The proposed solution implements the FIWARE CIM layer to impose standardization, context integrity, and efficient data access, thereby directly addressing the three project objectives.

3.4.1 Achievement of Objective 1: Implement Context Data Management

The Context Broker becomes the new core of the system:

- **Context Gateway (IoT Agent):** The IoT Agent for UltraLight 2.0/MQTT is deployed to intercept all raw MQTT data streams. Through a one-time provisioning step, the Agent is configured to map raw payload keys to standardized NGSI-LD Attributes (e.g., mapping 't' to 'soilTemperature' with unit metadata).
- **Digital Twin Engine (Orion-LD):** The Orion-LD Context Broker receives these context-rich updates, creating and managing the Digital Twin entities (e.g., `AgriSoil`) that represent the current, authoritative state of the olive grove. This is the implementation of efficient, standardized data access.

3.4.2 Achievement of Objective 2: Data Storage and Processing Pipelines

The solution decouples historical and real-time data for optimized performance:

- **Real-Time Access:** All applications now query the low-latency Orion-LD Broker for the current status.
- **Decoupled Archiving:** A standardized NGSI-LD Subscription is configured on Orion-LD to notify the dedicated Persistence Adapter service of any context change.

- **Historical Pipeline:** This adapter is responsible for receiving the standardized notification and translating it into the InfluxDB format. The InfluxDB database then efficiently archives the time-series data. This automated, decoupled pipeline ensures both real-time status and long-term analytical history are properly stored and accessible.

3.4.3 Achievement of Objective 3: Web-based Dashboard

- The project uses Grafana as the visualization layer. Grafana is connected directly to the InfluxDB database to display detailed historical trends and analysis.
- The dashboard provides crucial operational insights and features threshold-based alerts that notify users when Context Information attributes (e.g., soil moisture) cross critical operational boundaries, leveraging the integrity of the data provided by the FIWARE context layer.

Actor / Component	**Type**	**Primary Task(s)**
LoRa Sensors / RPi	Producer (Existing)	Publish raw payloads to Mosquitto.
IoT Agent (UL/MQTT)	FIWARE GE (New)	Translate raw MQTT to NGSI-LD format.
Orion-LD Context Broker	FIWARE GE (New)	Manage and centralize the Digital Twin (current state).
Persistence Adapter	Microservice (New)	Receive NGSI-LD notification and translate data for InfluxDB.
InfluxDB	Storage (New)	Store clean, time-series data for analytics.
Grafana	Consumer (New)	Visualize data and provide threshold-based alerts.

Table 3.2: System Actors and Tasks in the FIWARE Solution Architecture

3.5 Summary

The integration of FIWARE into the existing Man4Health infrastructure transformed a conventional sensor network into a context-aware, interoperable IoT platform. The system

now provides:

- Standardized, semantically meaningful data through NGSI-LD entities;
- Decoupled, scalable data persistence using Orion-LD and InfluxDB;
- Real-time visualization and analytics through Grafana.

This methodology ensured that the system could operate autonomously in the field, collect and process environmental data, and deliver both real-time and historical insights in a consistent, open, and scalable manner.

Chapter 4

Implementation

This chapter presents the implementation of the FIWARE-based architecture developed for the olive grove monitoring system. It details the deployment of the system components, the configuration of the data management and persistence layers, the integration of the IoT network, and the visualization of results through Grafana [18].

4.1 Dockerized Deployment and Infrastructure Foundation

The entire system architecture was implemented using Docker and Docker Compose, establishing a microservice-based environment where every component runs in an isolated container. This approach ensures maximum modularity, consistency, and reproducibility across different environments (from development to the final server deployment) [19].

4.1.1 Core Service Deployment

The foundational stack consists of four primary services, each decoupled and orchestrated via a docker-compose.yml file. This file dictates the service configurations, network dependencies, persistent storage volumes, and environment variables necessary for interoperability [20].

The following snippet illustrates the essential services required to establish the Context Management and Persistence layers, including the IoT Agent, Orion-LD Context Broker, and the InfluxDB/Grafana components:

```
version: "3.8"

services:
mongodb:
image: mongo:4.4
container_name: db-mongo
volumes:
  - mongodb_data:/data/db
restart: unless-stopped

orion-ld:
image: fiware/orion-ld:latest
container_name: fiware-orion-ld
depends_on:
  - mongodb
ports:
  - "1026:1026"
command: ["-logLevel","DEBUG","--dbURI","mongodb://mongodb:27017"]
restart: unless-stopped
healthcheck:
  test: ["CMD", "curl", "-f", "http://localhost:1026/version"]
  interval: 10s
  timeout: 3s
  retries: 10

iot-agent:
image: fiware/iotagent-ul:latest
container_name: fiware-iot-agent
depends_on:
  orion-ld:
    condition: service_healthy
```

```
    mongodb:
      condition: service_started
  ports:
    - "4041:4041" # northbound config API
    - "7896:7896" # southbound UL2 HTTP; MQTT comes via broker
  environment:
    - IOTA_LOG_LEVEL=DEBUG
    - IOTA_CB_HOST=orion-ld
    - IOTA_CB_PORT=1026
    - IOTA_CB_NGSI_VERSION=ld
    - IOTA_PROVIDER_URL=http://iot-agent:4041
    - IOTA_MQTT_HOST=mosquitto
    - IOTA_MQTT_PORT=1883
    - IOTA_REGISTRY_TYPE=mongodb
    - IOTA_MONGO_HOST=mongodb
    - IOTA_MONGO_PORT=27017
    - IOTA_DEFAULT_RESOURCE=/iot/ul
  restart: unless-stopped

  influxdb:
  image: influxdb:2.7
  container_name: db-influxdb
  ports:
    - "8086:8086"
  environment:
    - DOCKER_INFLUXDB_INIT_MODE=setup
    - DOCKER_INFLUXDB_INIT_USERNAME=${INFLUXDB_USER}
    - DOCKER_INFLUXDB_INIT_PASSWORD=${INFLUXDB_PASS}
    - DOCKER_INFLUXDB_INIT_ORG=${INFLUXDB_ORG}
    - DOCKER_INFLUXDB_INIT_BUCKET=${INFLUXDB_BUCKET}
    - DOCKER_INFLUXDB_INIT_ADMIN_TOKEN=${INFLUXDB_TOKEN}
  volumes:
    - influxdb_data:/var/lib/influxdb2
  restart: unless-stopped

```

```
67    grafana:
68    image: grafana/grafana:latest
69    container_name: visualization-grafana
70    depends_on:
71      - influxdb
72    ports:
73      - "3000:3000"
74    environment:
75      - GF_SECURITY_ADMIN_USER=${GF_USER}
76      - GF_SECURITY_ADMIN_PASSWORD=${GF_PASS}
77    volumes:
78      - grafana_data:/var/lib/grafana
79    restart: unless-stopped
80
81    mosquitto:
82     image: eclipse-mosquitto:2
83     container_name: mosquitto
84     ports: ["1883:1883"]
85     volumes:
86       - ./mosquitto.conf:/mosquitto/config/mosquitto.conf
87     restart: unless-stopped
88
89    volumes:
90        mongodb_data:
91        influxdb_data:
92        grafana_data:
```

Listing 4.1: Snippet of the essential `docker-compose.yml` configuration.

4.1.2 Service Startup Order and Dependencies

The entire multi-component system was initiated using `docker-compose up -d`, respecting a critical dependency chain to ensure each service received its required connection details before becoming active:

- **MongoDB:** Must start first as it serves as the persistent storage backbone for the Context Broker's live data. Unlike InfluxDB, which stores the historical record, MongoDB is responsible for maintaining the live, current state of your Digital Twin entities.
- **Orion-LD Context Broker:** Starts after MongoDB. It requires the database connection details (`ORION_MONGO_URI`) to be passed via environment variables to initialize its data store. The Context Broker does not store context data in its own memory. It uses an external database, which is almost always MongoDB, to keep a persistent record of every entity, its attributes, and its current value. Every time the IoT Agent sends a new reading (e.g., soil temperature changes from 25°C to 25.5°C), Orion-LD immediately updates the single, specific document in MongoDB that represents that sensor's digital twin.
- **IoT Agent:** Depends on Orion-LD (`IOTA_CB_HOST=orion-ld`) to receive the Context Broker's North Port connection details. The IoT Agent must be running to receive data from the Mosquitto broker (which is assumed to be either external or running separately).
- **InfluxDB:** Starts independently to prepare its data volume and required organization/bucket structure.
- **Grafana:** Depends on InfluxDB. Although configuration of the data source is done later via the Grafana UI, the container ensures the visualization layer is ready once the database is initialized.

4.1.3 Benefits of the Dockerized Approach

The decision to containerize the entire platform delivered several key operational benefits that address the project's sustainability goals:

- **Reproducibility and Scalability:** The `docker-compose.yml` file serves as a single source of truth for the architecture, allowing the entire system to be recreated

identically on any server. This is vital for expanding the project to larger test sites or migrating infrastructure.

- **Modularity and Isolation:** Each component (IoT Agent, Orion-LD, InfluxDB) runs in isolation. This prevents software version conflicts and simplifies maintenance; an update to the Context Broker does not risk breaking the InfluxDB persistence.

- **Easy Maintenance:** Volumes (`mongodb_data`, `influxdb_data`) are used to ensure that data persists across container restarts, providing reliable long-term data management. Network ports (e.g., `1026` for Orion-LD) are explicitly mapped, simplifying network configuration and troubleshooting.

4.2 IoT Data Acquisition and Payload Standardization

This section details the nature of the sensor data collected from the field, which constitutes the input stream for the FIWARE Context Management layer, and describes the supplementary data generation method developed to ensure continuous testing of the new architecture.

4.2.1 Real Sensor Data Transmission and Raw Payloads

Data generated by the physical components—the LoRa sensor nodes and the auxiliary devices managed by the Raspberry Pi—are all aggregated via the Mosquitto MQTT Broker. The data are transmitted in a minimal, raw format optimized for bandwidth-limited environments, lacking inherent context metadata.

- **Field Data Flow:** The LoRa Gateway and the Raspberry Pi's custom scripts publish the data to the broker. The payloads are kept minimal to conserve power and bandwidth.

- **Raw State Problem:** In this state, the generic keys used (e.g., `t`, `h`) are ambiguous, representing a fundamental lack of Context Information that the FIWARE IoT Agent is tasked with resolving.

An example of a typical raw payload received by the broker from a single sensor node would look similar to the structure below:

```
{
  "dev_id": "SN-PLOT-A01",
  "payload": {
    "t": 21.5,
    "h": 55.2,
    "r": 12.0
  },
  "metadata": {
    "time": "2025-10-27T10:30:00Z",
    "rssi": -95
  }
}
```

Listing 4.2: Example of a raw MQTT payload received from a sensor node, illustrating ambiguous keys.

4.2.2 Data Generation for Continuous Testing

Problem Encountered: Due to the experimental nature of the field site, communication interruptions and intermittent power supply issues often resulted in gaps in the live data stream. This posed a significant difficulty for continuously developing, testing, and validating the new FIWARE pipeline.

Solution: A **Python-based data generator** was developed to produce synthetic data streams that perfectly mimic the structure of the real payloads. This generator ensures the Context Broker and Persistence Adapter can be stress-tested with realistic, continuous data, irrespective of the field network status.

The generator uses CSV files containing historical or realistic values as input, ensuring the payload structure matches the format generated by the LoRa Gateway.

```
id,time,temp,humid
SN-PLOT-A01,2025-10-27T10:30:00Z,21.5,55.2
SN-PLOT-A02,2025-10-27T10:35:00Z,20.8,56.1
SN-PLOT-A01,2025-10-27T10:40:00Z,21.6,55.1
SN-PLOT-A02,2025-10-27T10:45:00Z,20.9,56.0
```

Listing 4.3: Sample input data used by the Python data generator.

Screenshots of panels in Figures 4.1, and 4.2 show system operation with simulated data.

Figure 4.1: Dashboard section displaying Temperature values

Figure 4.2: Dashboard section displaying Humididy values

The generator iterates through this data, wrapping the relevant fields (e.g., `temp`, `humid`) into the required MQTT JSON structure and publishing them to the Mosquitto Broker at simulated intervals. This method allowed the development team to isolate

software issues from hardware reliability issues and validate the semantic mapping logic in the IoT Agent.

4.3 Entity Modelling (NGSI-LD)

The previous project was aiming for a data description format, based on an older NGSI version, had several architectural and semantic flaws that required a complete refactoring for the current solution.

The entity attempted to model all concerns—device health, environmental context, and generic measurement IDs—in a single structure. This violates the principle of CIM, which dictates that concerns must be separated into specialized entities.

The model incorrectly defines core NGSI fields. In a true NGSI-LD entity, id and type must be top-level JSON keys, not fields nested inside a properties object. This structure would not be accepted by the Orion-LD Context Broker. While refDevice correctly identifies the relationship to hardware, the main entity type is a generic "Sensors." The NGSI-LD goal is to model the subject of the observation (the plot of soil or the air), not the instrument itself.

```
Temperature:
   title: Temperature
   type: object
   description: Defines a given temperature sensor to be used in a -Device-
   properties:
     id:
       description: Sensor identifier
       type: string
       format: uri
       example: urn:device:temperature:sensorNBR
       x-ngsi:
         type: Property
     type:
       description: NGSI Entity type. It has to be temperature
```

```
        title: Type
        enum:
          - Temperature
        type: string
        x-ngsi:
          model: https://schema.org/Text
          type: Property
      measuringMedium:
        description: Define tthe medium from where the measurement is taken
        enum:
          - air
          - soil
          - water
        type: string
```

Listing 4.4: Example of a previous project's monolithic YAML data model for a `Temperature` sensor.

- **Ambiguous Labels:** The column headers use native language shorthand (`Hora`, `Humidade`) or generic terms (`Temperatura`). When this data is transferred across the network, its original meaning and unit are often lost.
- **Missing Context:** The data lacks metadata specifying *where* the measurement was taken (which plot, which depth) and *who* measured it, necessitating standardization by the FIWARE layer.

4.4 Subscription Configuration

The subscription mechanism was implemented using the `/ngsi-ld/v1/subscriptions` endpoint of `Orion-LD`. A typical configuration is illustrated below, where the Context Broker is instructed to notify the persistence adapter at each change of a soil sensor entity.

```
POST http://localhost:1026/ngsi-ld/v1/subscriptions
```

```
Content-Type: application/ld+json

{
    "type": "Subscription",
    "description": "Forward SoilSensor updates to persistence adapter",
    "entities": [{ "type": "SoilSensor" }],
    "watchedAttributes": ["soilTemperature", "soilHumidity"],
    "notification": {
        "endpoint": {
          "uri": "http://persistence-adapter:8080/notify",
          "accept": "application/json"
        }
    }
}
```

Listing 4.5: Example of Orion-LD subscription configuration for context notifications.

Whenever one of the specified attributes changes, `Orion-LD` sends a notification similar to the structure shown below.

```
{
    "id": "urn:ngsi-ld:SoilSensor:001",
    "type": "SoilSensor",
    "soilTemperature": {
        "type": "Property",
        "value": 22.6,
        "observedAt": "2025-10-27T14:32:00Z"
    },
    "soilHumidity": {
        "type": "Property",
        "value": 43.8,
        "observedAt": "2025-10-27T14:32:00Z"
    }
}
```

Listing 4.6: Example of Orion-LD notification message.

4.5 InfluxDB Persistence Adapter

The FIWARE Orion-LD Context Broker does not store data directly in databases. Instead, it follows a publish/subscribe mechanism based on the `NGSI-LD` standard, through which it sends notifications of context changes to subscribed endpoints. In the implemented architecture, this mechanism was used to integrate Orion-LD with the InfluxDB time-series database through a lightweight intermediary service referred to as the *persistence adapter* [21].

The communication workflow between `Orion-LD` and `InfluxDB` is summarized as follows:

1. **Context Update:** Each time a sensor value is updated through the `IoT Agent`, `Orion-LD` registers the change in the corresponding NGSI-LD entity.

2. **Notification Trigger:** A subscription, previously configured on `Orion-LD`, monitors specific attributes (e.g., `soilTemperature`, `airHumidity`). When one of these attributes changes, `Orion-LD` sends an HTTP `POST` request containing the updated data to the persistence adapter endpoint.

3. **Data Transformation and Storage:** The persistence adapter processes the incoming JSON payload and converts it into the **Line Protocol format** accepted by `InfluxDB`. The formatted measurement is then inserted into the database through the `InfluxDB` REST API.

4. **Visualization:** `Grafana` dashboards access the `InfluxDB` database directly to visualize the stored time-series data and compute analytics such as trends, averages, or threshold-based alerts.

The persistence adapter was developed as a lightweight microservice using Python and the Flask framework. The service exposes a single endpoint (`/notify`) that receives the `Orion-LD` notifications and converts them into `InfluxDB` line protocol entries, which are then written to the database.

```python
from flask import Flask, request
    import requests

    app = Flask(__name__)

    @app.route("/notify", methods=["POST"])
    def notify():
        data = request.get_json()
        for entity in data["data"]:
            if "soilTemperature" in entity:
                temp = entity["soilTemperature"]["value"]
                measurement = f"soil,entity={entity['id']} temperature={temp}"
                requests.post(
                    "http://influxdb:8086/api/v2/write?org=myorg&bucket=mybucket&precision=s",
                    headers={"Authorization": "Token <TOKEN>"},
                    data=measurement
                )
        return "", 204
```

Listing 4.7: Simplified implementation of the InfluxDB persistence adapter (Python/Flask).

This simple service acts as the necessary bridge between FIWARE's context management layer and the time-series database. The use of `InfluxDB` provides high performance for large volumes of sequential measurements while maintaining compatibility with `Grafana` for visualization.

4.6 The End-to-End Pipeline Logic

The final solution operates on a strict publish/subscribe model, ensuring that the Context Broker layer (`Orion-LD`) is always decoupled from both the ingestion layer (`IoT Agent`) and the archival layer (`InfluxDB`). The complete data flow is as follows:

1. **Sensor Publication (Field to Broker):** The LoRa Sensor Nodes and the Raspberry Pi publish minimal, raw JSON payloads to the Mosquitto MQTT Broker.

2. **Ingestion and Standardization (`IoT Agent`):** The `IoT Agent for UltraLight 2.0/MQTT` subscribes to the relevant Mosquitto topics. It performs the crucial task of semantic translation: mapping the ambiguous raw keys (e.g., `t`, `h`) to standardized **NGSI-LD Attributes**, thereby injecting Context Information into the data stream.

3. **Context Management (`Orion-LD`):** The `IoT Agent` sends the standardized data to the `Orion-LD Context Broker` via an NGSI-LD update request. `Orion-LD` updates the relevant **Digital Twin Entity** (e.g., `AgriSensorNode`) and saves the new current state to `MongoDB`.

4. **Decoupling Trigger (Subscription):** `Orion-LD` automatically triggers a notification based on a pre-configured **NGSI-LD Subscription** that monitors the attributes that have just changed.

5. **Archival and Storage (Persistence Adapter):** The notification is sent as an HTTP `POST` request to the custom-developed **Persistence Adapter**. The adapter processes the JSON-LD payload, converts the context data into the InfluxDB Line Protocol format, and writes the time-series record to the `InfluxDB` database.

6. **Visualization (`Grafana`):** `Grafana` reads the stored time-series data directly from `InfluxDB` for historical trend visualization and alert processing.

Once deployed, the hybrid system operated continuously, collecting and transmitting data between the olive grove and the FIWARE platform. The Raspberry Pi performed local MQTT data aggregation, while the server handled real-time context management, data persistence, and visualization. All Docker containers were monitored to ensure stable operation and low latency in data updates.

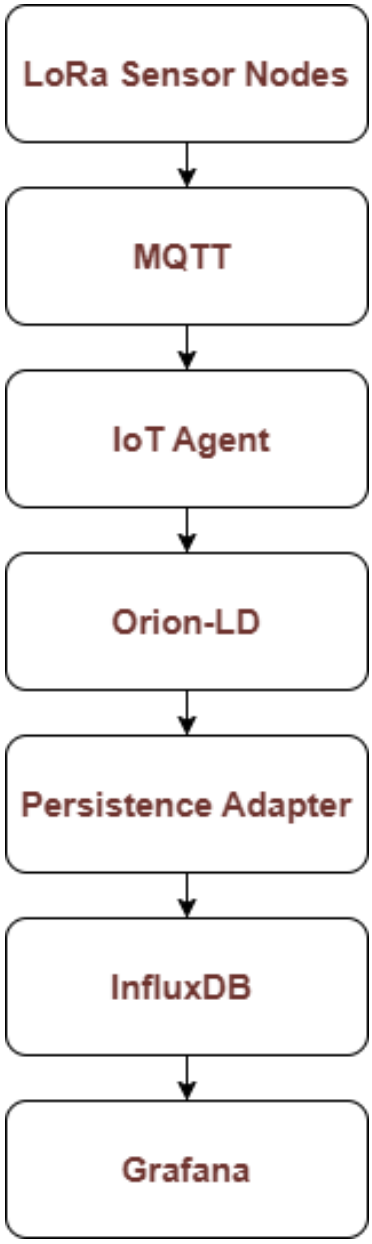

Figure 4.3: Diagram of the end-to-end pipline

4.7 Results and Validation

To test the system's functionality, datasets corresponding to CO_2, weather, pH, pluviometer, and IoT soil sensors were uploaded into the FIWARE environment. Simulated readings were successfully processed and visualized through Grafana dashboards. Trends in environmental parameters, such as temperature fluctuations and humidity variations, were clearly represented, demonstrating the system's ability to process and display both real and simulated data. Figure 4.4 displays a part of the Grafana dashoard in the testing period.

Figure 4.4: Section from the Grafana dashboard

4.8 Summary

This chapter presented the implementation plan for the FIWARE-enabled IoT system. Through the integration of edge computing (Raspberry Pi) and cloud-based context management (FIWARE), the system achieved reliable, scalable, and semantically rich data management. The results confirmed that FIWARE can effectively enhance environmental monitoring applications in precision agriculture

Chapter 5

Conclusions

This work presented the design and implementation of an IoT architecture based on the FIWARE ecosystem to enhance the data management and monitoring capabilities of the *Man4Health* project. The original infrastructure, although effective in collecting environmental data through distributed sensor nodes, suffered from several limitations, namely the absence of semantic context, lack of interoperability, and inefficiencies in long-term data management.

By integrating the FIWARE framework—particularly the Orion-LD Context Broker, the IoT Agent for UltraLight2.0/MQTT, and a custom InfluxDB persistence adapter—the system evolved into a fully context-aware and scalable monitoring platform. This architecture enabled the translation of raw MQTT payloads into standardized NGSI-LD entities, ensuring semantic consistency, interoperability, and efficient information retrieval across all layers of the system.

The adoption of a Dockerized deployment strategy allowed modular, reproducible, and maintainable integration of multiple FIWARE GEs within a single microservice environment. This design choice simplified deployment and maintenance while providing a foundation for future scalability and remote accessibility.

The integration of FIWARE's context management principles introduced a clear separation between the real-time and historical data layers. The Orion-LD Context Broker now acts as the authoritative source for live contextual data (the "Digital Twin" of the

olive grove), while InfluxDB maintains optimized, time-series storage for long-term environmental and operational analysis. Visualization through Grafana further enhanced usability, providing intuitive dashboards that display both instantaneous and historical measurements, such as soil temperature, humidity, CO_2 concentration, and atmospheric parameters.

In summary, the project successfully achieved all the objectives initially defined:

- Implemented a context-based data management system using NGSI-LD and FIWARE components, ensuring semantic interoperability and structured data exchange;
- Developed a decoupled data persistence pipeline capable of separating live and historical data using InfluxDB and a custom persistence adapter;
- Designed and deployed an interactive web dashboard using Grafana for visualization, monitoring, and analysis of environmental parameters in real time.

The resulting system demonstrates the applicability of FIWARE technologies in precision agriculture and their potential to transform isolated IoT networks into intelligent, standardized, and interoperable ecosystems. Beyond the technical outcomes, this implementation highlights the value of open-source technologies in promoting transparency, replicability, and long-term sustainability in agricultural innovation.

5.1 Limitations

Despite the successful implementation, some limitations were observed during development and testing:

- The physical infrastructure in the olive grove occasionally suffered from power interruptions, resulting in data gaps that affected system testing;
- The lack of continuous field connectivity prevented long-term validation of the end-to-end architecture with live data;

- The persistence adapter was developed as a lightweight custom solution and does not yet include advanced data validation, error handling, or redundancy mechanisms.

Nevertheless, these challenges provided valuable insights into the operational difficulties of deploying IoT systems in rural environments and guided design decisions that improved the robustness of the final solution.

5.2 Future Work

The work developed in this thesis lays a strong foundation for future research and technical extensions. The following directions are recommended for further development:

- **Full integration of QuantumLeap:** Replace or complement the current custom persistence adapter with the official FIWARE QuantumLeap Generic Enabler to achieve native long-term storage and simplify system maintenance.
- **Predictive analytics and AI integration:** Implement machine learning models to analyze historical data for predictive irrigation scheduling, pest risk detection, or soil health forecasting.
- **Edge computing enhancements:** Expand the role of the Raspberry Pi to include on-site data pre-processing, local storage, and fault-tolerant caching to mitigate network instability.
- **Scalability tests:** Extend the system to multiple agricultural sites to evaluate the scalability of the FIWARE-based solution and measure performance under higher data loads.
- **User interaction and decision support:** Enrich the Grafana dashboard with map-based interfaces, interactive controls, and role-based access for farmers, researchers, and system administrators.

- **Data sharing and interoperability:** Integrate with external agricultural data platforms or open data initiatives using NGSI-LD federation principles to enable collaborative research and multi-domain interoperability.

5.3 Final Remarks

The integration of FIWARE into the *Man4Health* project demonstrates a viable pathway for transforming experimental IoT infrastructures into intelligent, interoperable, and scalable monitoring systems. The resulting platform not only improves the efficiency of data collection and analysis but also aligns with broader goals of digital transformation and sustainability in agriculture. Through its modular and open design, the system serves as a replicable model for similar environmental monitoring projects and contributes to advancing the state of the art in precision agriculture technologies.

Bibliography

[1] M. Rahman, R. Wójcik, and D. Karagiannis, "A review of fiware-based architectures for smart agriculture," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, pp. 9411–9430, 2021. DOI: 10.1007/s12652-021-03160-9.

[2] K. Katke and V. G. G, "Challenges of precision agriculture technology adoption: A case study of tumkur district, india," *The International Journal of Analytical and Experimental Modal Analysis*, vol. XI, no. XI, pp. 2653–2672, 2020. [Online]. Available: https://www.researchgate.net/publication/338990802.

[3] S. S. Liao, J. W. He, and T. H. Tang, "A framework for context information management," *Journal of Information Science*, vol. 30, no. 6, pp. 528–539, 2004. DOI: 10.1177/0165551504047829.

[4] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013. DOI: 10.1016/j.future.2013.01.010.

[5] A. Kamilaris and F. X. Prenafeta-Boldú, "The rise of iot in agriculture: Smart farming systems and applications," *Computers and Electronics in Agriculture*, vol. 136, pp. 145–159, 2017. DOI: 10.1016/j.compag.2017.02.002.

[6] M. Centenaro, L. Vangelista, A. Zanella, and M. Zorzi, "Long-range communications in unlicensed bands: The rising stars in the iot and smart city scenarios," *IEEE Wireless Communications*, vol. 23, no. 5, pp. 60–67, 2016. DOI: 10.1109/MWC.2016.7721743.

[7] S. Verma, N. Sood, and A. Sharma, "Iot in smart agriculture: Challenges, issues, and future directions," *Sustainable Computing: Informatics and Systems*, vol. 28, p. 100 470, 2020. DOI: 10.1016/j.suscom.2020.100470.

[8] H. van Kranenburg, M. S. Bargh, S. M. Iacob, and A. Peddemors, "A context management framework for supporting context-aware distributed applications," *IEEE Communications Magazine*, vol. 44, no. 8, pp. 67–74, 2006. DOI: 10.1109/MCOM.2006.1678112.

[9] ETSI ISG CIM, *Ngsi-ld api specification, etsi gs cim 009 v1.7.1*, https://www.etsi.org/deliver/etsi_gs/CIM/001_099/009/01.07.01_60/gs_CIM009v010701p.pdf, Standard defining the NGSI-LD information model, Jul. 2021.

[10] F. Tao, M. Zhang, A. Liu, and A. Nee, "Digital twin in industry: State-of-the-art," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 2405–2415, 2018. DOI: 10.1109/TII.2018.2873186.

[11] A. Fuller, Z. Fan, C. Day, and C. Barlow, "Digital twin: Enabling technologies, challenges and open research," *IEEE Access*, vol. 8, pp. 108 952–108 971, 2020. DOI: 10.1109/ACCESS.2020.2998358.

[12] J. Lee, K. Park, and M. Kim, "Iot data management and analytics for digital twin systems," in *Proceedings of the 2020 IEEE International Conference on Big Data*, 2020, pp. 3201–3206. DOI: 10.1109/BigData50022.2020.9378001.

[13] S. Boschert and R. Rosen, "Digital twin—the simulation aspect," in *Mechatronic Futures*, Cham, Switzerland: Springer, 2016, pp. 59–74. DOI: 10.1007/978-3-319-32156-1_5.

[14] K. Zhang, Y. Zhang, Z. Xu, and F. Tao, "Digital twin-driven smart agricultural systems: Concepts, architecture, and case study," *Computers and Electronics in Agriculture*, vol. 189, p. 106 410, 2021. DOI: 10.1016/j.compag.2021.106410.

[15] M. Grieves, *Digital Twin: Manufacturing Excellence through Virtual Factory Replication*. Melbourne, FL, USA: Florida Institute of Technology, 2017.

[16] M. Adam, A. Soeparno, E. Suharjono, and A. Rahmat, "Energy-autonomous lora sensor node for smart agriculture monitoring," *IEEE Access*, vol. 9, pp. 148 504–148 516, 2021. DOI: 10.1109/ACCESS.2021.3125483.

[17] M. Rizzi, P. Ferrari, A. Flammini, and E. Sisinni, "Evaluation of the iot lorawan solution for distributed measurement applications," *IEEE Transactions on Instrumentation and Measurement*, vol. 66, no. 12, pp. 3340–3349, 2017. DOI: 10.1109/TIM.2017.2754438.

[18] M. Shafi, A. Alghamdi, and M. O. Alassafi, "Microservice-based iot platform using docker containers and kubernetes for smart environments," *IEEE Access*, vol. 8, pp. 222 883–222 897, 2020. DOI: 10.1109/ACCESS.2020.3043271.

[19] D. Merkel, "Docker: Lightweight linux containers for consistent development and deployment," *Linux Journal*, vol. 2014, no. 239, p. 2, 2014. [Online]. Available: https://dl.acm.org/doi/10.5555/2600239.2600241.

[20] J. Pereira, J. P. Coelho, and J. J. Rodrigues, "A dockerized fiware architecture for smart and sustainable agricultural environments," *Computers and Electronics in Agriculture*, vol. 190, p. 106 432, 2021. DOI: 10.1016/j.compag.2021.106432.

[21] FIWARE Foundation, *Fiware orion-ld: Next generation context broker for ngsi-ld applications*, https://fiware-orion-ld.readthedocs.io/en/latest/, Accessed October 2025, 2022.