Universidade de Aveiro
2023

Pedro Miguel
Ferreira Marques

Localização de Ativos com Sistemas Definidos por Software

Asset Localization with Software Defined Systems

Universidade de Aveiro
2023

Pedro Miguel Ferreira Marques

Localização de Ativos com Sistemas Definidos por Software

Asset Localization with Software Defined Systems

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à conclusão da unidade curricular Dissertação, condição necessária para obtenção do grau de Mestre em Engenharia Informática , realizada sob a orientação científica do Doutor João Paulo Silva Barraca, Professor associado do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Doutor Mário Luís Pinto Antunes, Professor auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

Este trabalho é co-financiado pelo Fundo Europeu de Desenvolvimento Regional (FEDER), através do Programa Operacional Competitividade e Internacionalização (COMPETE 2020) do Portugal 2020 [Projeto SDRT com o nº 070192 (POCI-01-0247-FEDER-070192)]

Este trabalho é co-financiado pela FCT/MCTES através de fundos nacionais e quando aplicável cofinanciado por fundos comunitários no âmbito do projeto UIDB/50008/2020-UIDP/50008/2020

o júri / the jury

presidente / president	Prof. Doutor José Luis Guimarães Oliveira professor catedrático da Universidade de Aveiro
vogais / examiners committee	Prof. Doutor João Nuno Lopes Barata professor auxiliar do Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade do Coimbra
	Prof. Doutor João Paulo Silva Barraca professor associado da Universidade de Aveiro

agradecimentos / acknowledgements

Agradeço toda a ajuda que o meu orientador, Professor Doutor João Paulo Barraca, e corrientador, Mário Luís Antunes, pelo acompanhamento dado durante o desenvolvimento da dissertação. Também quero agradecer à minha família, amigos e colegas pelo apoio e motivação ao longo deste tempo.

Palavras Chave Digital Twins, Internet of Things, Asset Management, Indoor Location.

Resumo Esta dissertação tem como objectivo a criação de um protótipo de uma plataforma desenvolvida à volta do conceito de Digital Twins, capaz de gerir diferentes entidades dentro de uma clínica médica. Numa fase inicial, é apresentado uma introdução aos conceitos essenciais para o desenvolvimento da plataforma, bem como diferentes soluções para problemas semelhantes já existentes. De seguida o foco vira-se para o planeamento e implementação da plataforma, testando as suas principais funcionalidades com o objectivo de determinar a sua eficácia de gestão das diferentes entidades e recolha de dados no espaço físico.

Keywords Digital Twins, Internet of Things, Asset Management, Indoor Location.

Abstract This dissertation aims to create a prototype of a platform developed around the concept of Digital Twins, capable of managing different entities within a medical clinic. Initially, an introduction is given to the essential concepts for developing the platform, as well as different solutions to similar existing problems. The focus then shifts to planning and implementing the platform, testing its main features in order to determine its effectiveness in managing the different entities and collecting data in the physical space.

Contents

List of Figures

List of Tables

Lista de Excertos de Código

Glossário

IoT	Internet of Thing
RFID	Radio Frequency Identification
IPS	Indoor Position System
UUID	Universally Unique Identifier
DT	Digital Twin
DM	Digital Model
DS	Digital Shadow
DSL	Domain Specific Language
NASA	National Aeronautics and Space Administration
SVM	Support Vector Machine
LoRa	Long Range
BIM	Building Information Model
LPWAN	Low Powered Wide Area Network
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
API	Application Programming Interface
MQTT	Message Queuing Telemetry Transport
AMQP	Advanced Message Queuing Protocol
SVM	Support Vector Machine
BIM	Building Information Model
ISMS	Indoor Safety Management System
DAM	Detection of Abnormal Montionless
ML	Machine Learning
CC	Cloud Computing
CPS	Cyber-Physical System

CHAPTER 1
Introduction

The management of organizations within National Health Systems, encompassing hospitals and health centers, encounters efficiency challenges not only because it is a complex area of intervention given its specificity and demands, but also because of external factors and unpredictable circumstances.

One such factor, is the rise of elderly population. In recent years the life expectancy of the population has been increasing, which, being expected due to better living conditions and medical assistance, has aggravated the already existing problems relating to management of resources (human and material) in the National Health Systems [1].

An aging population, given its condition, translates into an increase in demand for the services provided in these organizations, leading to a worsening of the problem, more patients, requires a greater availability of resources, which is not always possible [2].

1.1 Motivation

The problem the project is designed around is related to the healthcare industry. With an aging population, ever more prominent management issues in hospitals and clinics due to lack of workforce, and the possibility of high-scale disease events, a system that can ease the management of the healthcare body is becoming more desirable.

While the presented problem is vast, the context of this project will only go to the scale of a prototype solution for a clinic. The solution will focus on visualizing the position and determining the required entities' state at any moment.

Since this is one of the most sensitive areas of any society, one that at any given moment will have to be able to act or react depending on the circumstances, it is urgent to find solutions that can help optimize management to achieve economic viability of National Health Systems.

Awareness of this reality has led to the search for new alternative solutions that allow for cost reduction, without ever neglecting maximum patient satisfaction, for example, maximizing the efficiency of the service provided.

The increased search for new solutions that can help in the management of the organizations that make up the National Health Systems has triggered the development of new solutions for hospital management, which include the automation of processes [3].

Many hospital organizations have already implemented these process automation solutions, using active localization technologies through Radio Frequency Identification (RFID).

Using radio frequency technology and Digital Twin (DT), it is possible to monitor the movements of patients through the various hospital sectors, how the various equipment is being used, the existing stocks, as well as the supply of medicines to patients.

1.2 Objectives

This dissertation aims to explore system integration solutions, in a perspective of high reconfigurability and alignment with Internet of Thing (IoT) techniques in the scope of DT. Thus, we intend to develop a solution that uses low-cost RFID location devices that are activated in a centralized and highly parameterized way, making them more effective than traditional ones, not forgetting the reduced energy impact and flexibility to specific scenarios.

The goal of the system is to be able to receive data related to human and equipment position from a clinic like space in the outside world (be it real world or simulated), using it to create a digital representation of the entities present in the physical world. These representations, together with all data received, will be stored in a platform that can also utilize the gathered data to determine the state in which the digital representations of each entity finds themselves in at any moment. These new versions should be able to showcase themselves in an easy-to-use platform, evaluate the current state of the clinic and send notifications in specific scenarios.

1.3 Thesis Outline

This document presents 6 chapters, the first being the one already shown, that is, the introduction, where the motivation and goals of the project are explained. The remaining chapters are:

1. Chapter 2 - State of the Art: Presentation of the core concepts of the thesis, such as Digital Twins and indoor tracking methods and technologies. It is also showcased similar solutions as a basis to the developed project and various tools that were used to implement the solution.
2. Chapter 3 - Architecture: Describes the main scenarios expected for the system to handle and how it is structured,in depth look at modeled entities.
3. Chapter 4 - Prototype Implementation: Describes of how the system was built and implemented and how each component is structured and how they interact with each other and the physical world.
4. Chapter 5 - Results: Describes of what will be evaluated in the system, how and the results of said tests.
5. Chapter 6 - Conclusion: A brief overview of the work done over the course of the thesis and future work that can be done.

CHAPTER 2

State of the Art

The goal of this thesis can be summarized as a system that combines DTs and Indoor Position System (IPS) to create a tool capable of handling and managing the state of not only all entities residing inside a closed space, but also being able to classify what they are currently doing.

Both of these systems are complex topics that need to be understood before being able to be applied effectively. The former can be seen as a digital representation of a physical entity, connected to each other, while an IPS, as the name implies, is a system focused around providing and gathering positional data in an indoor scenario, useful for, in the case of this thesis, managing and keeping track of a cluster of different types of entities.

The research made for this work was done with the objective of gathering and grasping all necessary information to build an efficient IoT system capable of monitoring assets and workers in the healthcare sector in the scope of DT.

This chapter is comprised of four main sections, the first being where the concept of DT is presented, then commonly proposed architectures take the focus, the third part elaborates about the main enabling technologies for DTs and ending the chapter with previous work related to the thesis, culminating on a summary of how these works built their frameworks of DTs for their project.

2.1 Digital Twins

The notion of DT is not recent, as it was first introduced by Michael Grieves in 2003 at the University of Michigan [4] during a presentation on product life cycle management. He defines a DT as the virtual representation of a product, which contains three main components: a physical entity, a virtual representation of said entity and the data links connecting both. These elements are intertwined with one another, and the digital representation should try to reflect as much as possible all information about the object it represents by detailed inspection of the real world.

While the concept was introduced back in 2003, the first implementation of it was years later in 2010 by National Aeronautics and Space Administration (NASA) in the Technology Roadmap document, where a DT is used to reproduce conditions in space and perform flight readiness tests [5].

Through the integration with enabling technologies, DT are able to be applied in a multitude of fields where the fusion between the physical and virtual spaces benefits it, such as healthcare, aviation and manufacturing [6]. However due to a multitude of different solutions for each of these fields a diverse and incomplete understanding of its concept exists.

Based on the given definitions of a DT, the most common way to describe it is as a digital version of some physical entity [7], which leads to terms such as Digital Model, Digital Shadow and DT being used as if they were synonymous.

A Digital Model (DM) is a digital representation of an existing or planned physical object that does not use any form of automated data exchange between the physical object and the digital object. The digital representation might include a more or less comprehensive description of the physical object.

These models might include, but are not limited to simulation models of planned factories, mathematical models of new products, or any other models of a physical object, which do not use any form of automatic data integration. Digital data of existing physical systems might still be in use for the development of such models, but all data exchange is done in a manual way. A change in state of the physical object has no direct effect on the digital object and vice versa [8].

Based on the definition of a DM, if there further exists an automated one-way data flow between the state of an existing physical object and a digital object, one might refer to such a combination as Digital Shadow (DS). A change in state of the physical object leads to a change of state in the digital object, but not vice versa [8].

If further, the data flows between an existing physical object and a digital object are fully integrated in both directions, one might refer to it as DT. In such a combination, the digital object might also act as a controlling instance of the physical object. There might also be other objects, physical or digital, which induce changes of state in the digital object. A change in state of the physical object directly leads to a change in state of the digital object and vice versa [8].

Despite the great number of different types of highly specific implementations of DT, according to [9], a DT can be a set of technologies that enable the virtualization and optimization of a real-world object or system. In addition to this, a DT must have the following properties [10]:

1. Real-time and bi-univocal connection with the physical entity it represents. [11]
2. Self-evolution, meaning that any change done to the physical side of the DT will affect the digital side in real-time and vice versa [12].
3. Continuous analysis using machine learning [10]
4. Availability of the data acquired over time [12]

5. Domain dependence, as in providing services specific to the industry it is being built for [10]

2.2 Architectures

Because there are several definitions of DTs, depending on the context and the area applied, there are also several architectures associated with it.

These different implementations, labeled as DT architectures, are built around the concept of components, in which each component is responsible for a specific task and provides information to other components while being from one another.

The architectures studied by reading the scientific documentation surrounding DT for this document were the Three Component and Five Component architectures [13].

2.2.1 Three Component Architecture

This architecture applies the three components described by M. Grieves in its original introduction [4], these being the physical space, the virtual space and and information processing space.

The authors of [14] explain in detail the implementation of each of the three concepts of a DT, demonstrated in Figure 2.1:

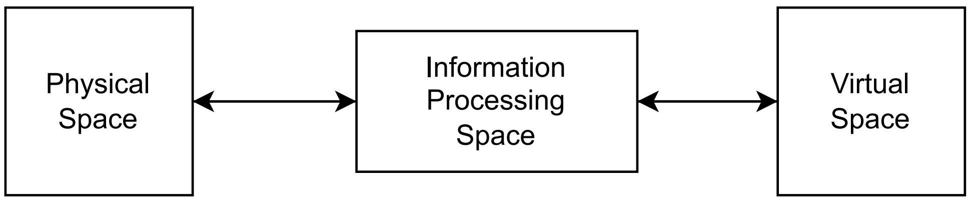


Figure 2.1: Digital Twin Architecture with 3 components

The physical space consists of machines, people, materials, or systems separated and distributed across the physical space itself, interconnected by IoT technologies. These technologies consist of different types of sensors and communication equipment that collect data from an associated physical object. All useful information linked to the physical space and to each object present in it is collected thanks to these sensors, such as an RFID tag for example, and is then processed to be used by the virtual space. In addition, the physical space reacts to all the feedback sent by the virtual counterpart.

The virtual space is the digital representation built using the data collected in the physical space. It consists of virtual models that, with the data obtained by the information processing space originating in the physical space, will be used by functions present in this DT component to be later delivered to the physical space.

The physical and virtual spaces are linked thanks to the Information Processing Space. This component is responsible for storing, processing, and mapping the data it receives.

2.2.2 Five Component Architecture

The next example of an architecture related to DTs is the five component architecture, as is the case of [13] who propose that a complete DT should include five components, showcased in Figure 2.2:

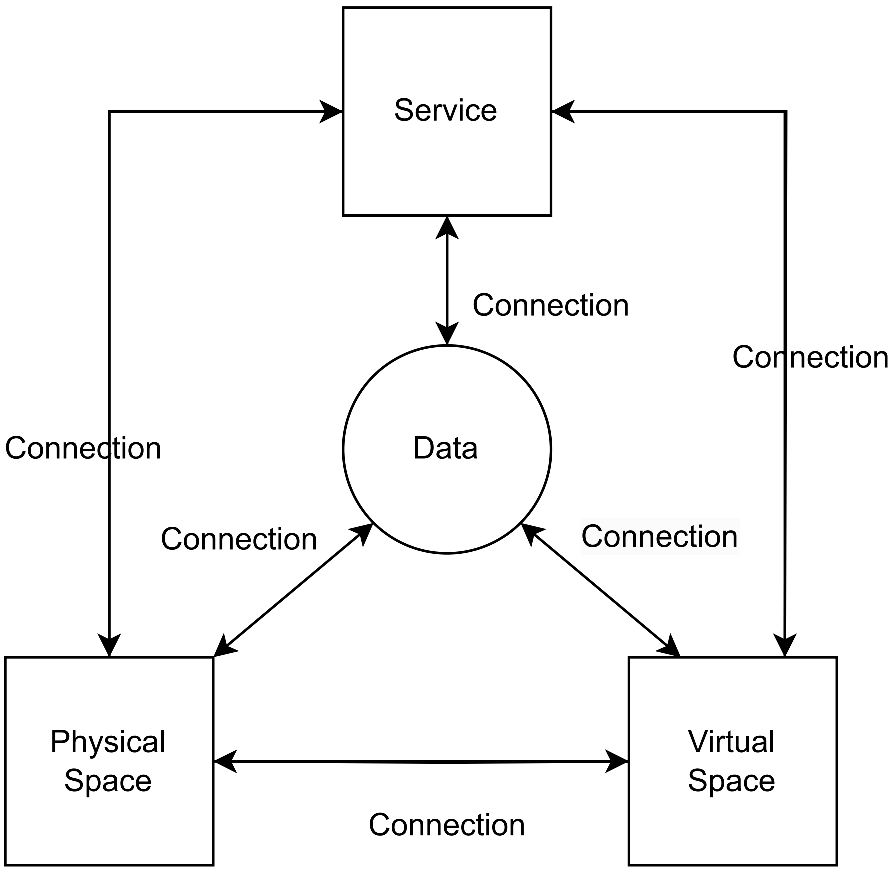


Figure 2.2: Digital Twin Architecture with 5 components

These five dimensions are equally important and necessary for the application of a DT, according to the previously referenced document, because each component has its own functionality that distinguishes it from the others, but it also needs to communicate and share data with other DT components to fulfill its function.

The responsibilities of each component are listed as follows:

- Physical Space Component: basis used to build the virtual component, since it is the source of the data needed to perform simulations and monitoring.
- Virtual Space Component: supports the simulation, decisions, and control of the physical component.
- Service: provides support processes for managing and controlling the physical component and other processes for operating and evolving the virtual component.
- Data: needed to do any kind of operation and to create new results. It's the core of the architecture and includes data from the physical component, virtual component, and the service. It also contains data fusion of these three components and methods for modeling, optimization, and prediction.
- Connections: bridge that connect all the other four components of the DT and allows communication between them.

2.3 Enabling technologies

With the advancement of a select group of fields, several DT related concepts have evolved considerably, and because of it, not only did it make it possible to implement DT, but also made it feasible to perform realistic simulations in virtual environments and gave rise to high interest in the research area related to them.

The following sections explores the commonly cited enabling technologies for DT [15].

2.3.1 Machine Learning

Machine Learning (ML) techniques are used in conjunction with DT to interpret information from the data gathered from its physical component to be later used in optimization and monitoring purposes.

Depending on the specific implementation, use cases and services provided by the DT, a vast array of techniques can be implemented to suit its needs, such as parameter optimization and future predictions. Some of the most commonly used algorithms for these purposes are deep learning, regression or supervised/unsupervised learning [15].

DT can also benefit ML, because they can generate data to train ML models with [16].

One negative aspect of ML is that it is perceived as a “Black Box” because they lack transparency which is key for DT for anomaly detection, fault identification and root cause analysis [17]. A way to address this problem is to integrate data-driven models into the DT.

2.3.2 Cloud Computing

Because of its characteristics, DT can be used to mirror systems with varying complexity, from small and simple to large and complex.

While small systems are easy to implement, to deal with large and complex systems, distributed and parallel computing becomes a necessity.

Cloud Computing (CC), acting as a data warehouse and providing heavy-processing capabilities, creates an environment suitable for any complex simulations that the DT [18] may need to do to fulfill its use cases.

2.3.3 Internet of Things

Internet of Things is an enabling technology that connects the virtual and physical counterparts due to its ability to aggregate data from multiple sources via a diverse selection of communication mediums that facilitate data mining and analytics throughout distributed systems [19]. A few devices that make this possible are smart sensors, wearables and RFID tags.

The choice of which IoT devices, communication protocols and platforms can influence the synchronization rate between the physical and virtual components, which in turn depends on the specific use case of the DT.

2.3.4 Cyber-Physical Systems

Cyber-Physical System (CPS) refers to the union of real and virtual systems, where the physical side is represented by the array of equipment that captures data such as sensors and actuators, networking and computation capabilities, while the virtual side is filled with computer-based algorithms that control the physical side and provide self-configuration, self-adaptation and self-preservation.

The majority of CPS integrating DT come from the manufacturing industry [20], where the conventional factory equipment is converted into a CPS by adding IoT equipment such as sensors to connect them to their Digital component to generate intelligent results.

2.4 Indoor Safety Frameworks with Digital Twins

This subsection of the document is aims to analyze two Indoor Positioning systems that implemented DT and understand what work was done and what can be applied or improved during implementation of the project proposed for this dissertation.

2.4.1 Framework 1 - Indoor Safety Management System based on Digital Twin

The project presented in [21] has the objective of creating a framework for a Indoor Safety Management System (ISMS), based on the DT model. As it is explained, the building in which the ISMS is going to be implemented is equipped with various IoT sensors that can detect temperature, humidity, smoke, and any factor that needs to be monitored that could cause harm to the building and to its inhabitants. While the building is being constructed, its Building Information Model (BIM) is simultaneously developed by engineers and all relevant information, including sensors and their location, are saved. This data is then converted into a 3D visualization of the entire building in a webpage, using WebGL technology.

Beyond just representing the building in a 3D manner, all data gathered by the IoT sensors are then sent to the webpage for data visualization and to be analyzed by ML processes, namely Support Vector Machine (SVM).

After the gathered data has been processed, it will be used by the Web platform built-in security models developed for the system, such as the visual safety status monitoring, danger alarm and positioning, danger classification and level assessment, and danger response suggestions to evaluate and display warnings for the safety management staff in case of an emergency. This staff can access the webpage for useful information and suggestions on how to handle dangerous situations, by using computers, smartphones, tablets, or any other device that can access the network. This process is displayed in Figure 2.3:

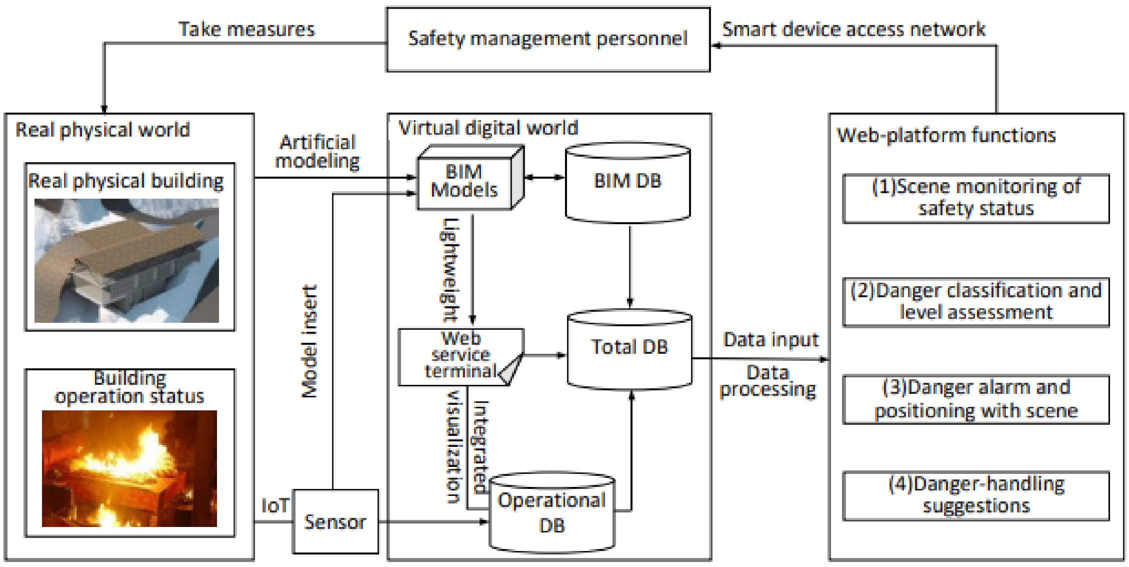
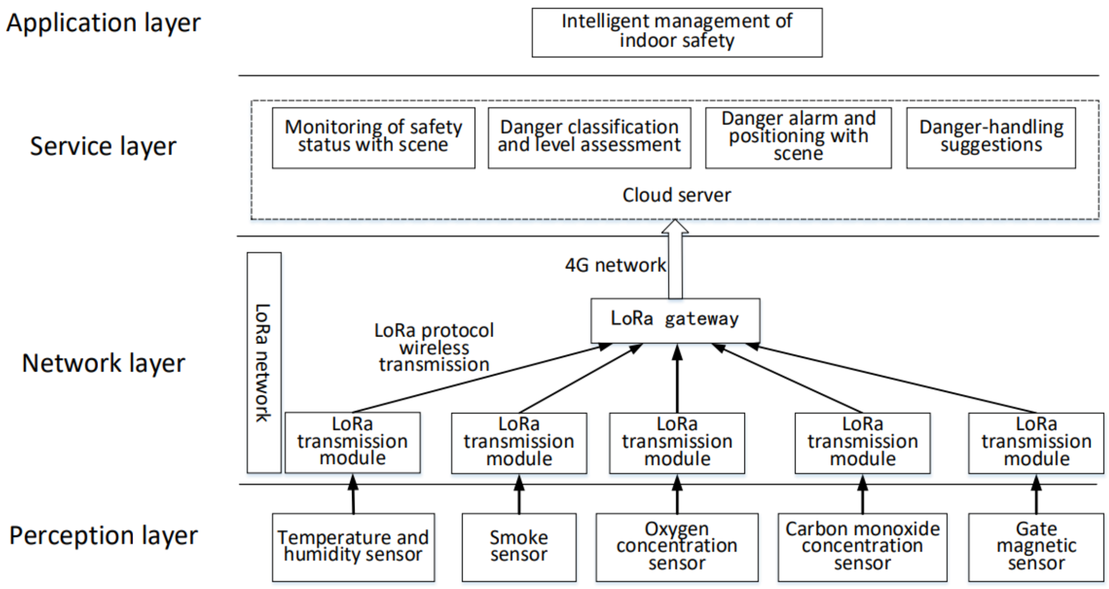


Figure 2.3: Proposed ISMS [21].

Data collection is possible using a proposed IoT system, developed by the team, showcased in Figure 2.4, comprised of four layers, the first being the perception layer, that includes all sensors that are used to acquire data, the second layer is the network layer, built using Long Range (LoRa) technology responsible to send data from the Perception Layer to the layer above it. This third layer that receives data from the Network Layer, is known as the Service Layer, responsible for data storage and analysis using SVM. After all data is analyzed, the results are sent to the final fourth layer, the Application Layer, that will display warnings based on the results.

According to the results of this work, the authors concluded that the feasibility of DT for solving problems related to building safety, the intuitiveness of indoor safety management, the feasibility of using IoT systems to evaluate danger in an indoor scenario was confirmed. They also note that improvements can be made, for example adding more aspects that where not tested, such as monitoring areas to verify if any harmful gases are present in that environment.



Application layer
Intelligent management of indoor safety
Service layer
Monitoring of safety status with scene
Danger classification and level assessment
Danger alarm and positioning with scene
Danger-handling suggestions
Cloud server
4G network
Network layer
LoRa network
LoRa protocol wireless transmission
LoRa gateway
LoRa transmission module
LoRa transmission module
LoRa transmission module
LoRa transmission module
LoRa transmission module
Perception layer
Temperature and humidity sensor
Smoke sensor
Oxygen concentration sensor
Carbon monoxide concentration sensor
Gate magnetic sensor

Figure 2.4: IoT structure of the indoor safety management system based on LoRa technology [21].

2.4.2 Framework 2 - IoT and Digital Twin enabled smart tracking for Safety Management

Because of the extreme conditions of working in warehouses and fatal accidents, the authors of this document [22] develop a DT enabled framework for indoor safety tracking named iSafeTrack that links the assets of the physical world to the cyber world using IoT devices.

This framework, shown in Figure 2.5, is comprised of four layers, these being the Physical World, IoT devices and Services, the Cyber world and, the stakeholders. The physical world is comprised not only of physical assets, such as the workers or any machine or material they need to interact with during work, but also attributes related to physical world itself, such as time and space, which are crucial for tracking workers. The IoT devices and services is made up of all IoT sensors that gather data from the physical world and send it to the cyber world. They use a wireless sensor tag as the main device to gather data, being made up of a 1D/2D barcode for identification, an accelerometer to recognize movement patterns and a Bluetooth low energy communication model that transmits the sensed data but also functions as a signal strength emitter for indoor positioning purposes.

All data gathered by the tags is then collected by the IoT edge gateway. This device also provides three essential services, these being registration services that guarantee the authorized edge gateway to be recorded, the configuration services that ensure data destination is correct, settings for geo location and time-window scanning, and execution services that realize fundamental functions such as executing fainting detection algorithms, scan sensed accelerometer data, filter Bluetooth low energy signals through a Kalman Filter and transmit organized data to the cyber world layer of the framework.

To determine if a worker is abnormally stationary and the stakeholders need to be warned of potential danger, the Detection of Abnormal Montionless (DAM) method was proposed. This method is executed every time tri-axial accelerator data from the smart tag is broadcasted

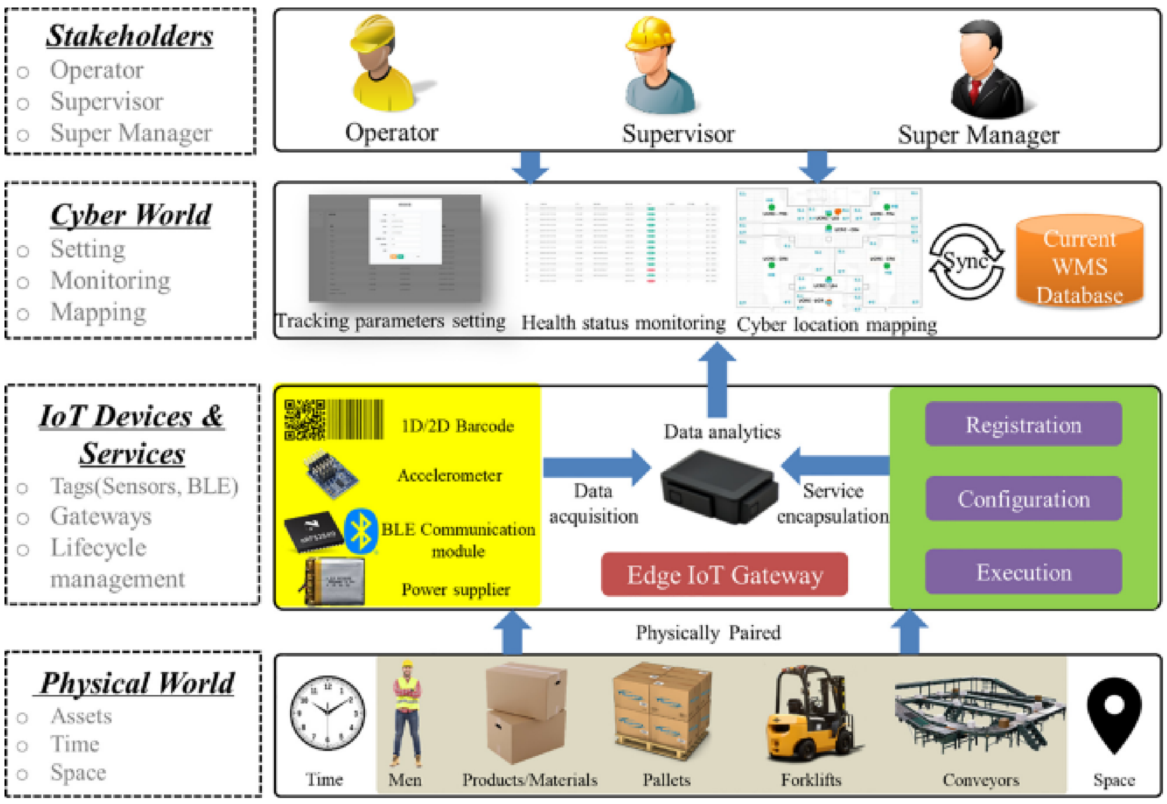


Figure 2.5: The iSafeTrack Framework [22]

and received by the gateway. This data is transformed into coordinates using a self-learning genetic position algorithm proposed by the authors, that will then be used to calculate the total amplitude of the three axes. This value will be compared to a threshold and in case it is bigger, and the operator is not in a labeled area where a supervisor can verify their status, a warning is sent to the cyber world, to notify the supervisors.

The cyber world is the third layer of the framework, and it acts as an interactive virtual representation of the physical world for the stakeholders to manipulate and gather useful information as they see fit. A few characteristics of this layer are the ability to change parameters such as fainting detection threshold values and location environment noise values. Data from the physical world is mapped into the cyber world.

The last layer is made up of the stakeholders, such as operators, superiors, and super managers of the warehouse, that will be interacting with the system and act when it alerts them in case of an emergency. Different types of stakeholders have different types of access to the system. Operators are the main tracking targets, and because of it they only provide basic information about themselves, while the supervisors are responsible for view the status of health and location of the operators by interacting with the cyber world, and in case of an emergency, they can trigger professional help. The super manager is the only element of the stakeholders that can change parameter settings because he is the one responsible of how effective the tracking system needs to be.

The authors end the document stating that the adoption of DT and IoT technologies to make a safety system was deployed and implemented successfully and the proposed self-learning genetic positioning algorithm can recognize abnormal motionless behavior and improve over time. They further state that the system can be improved in three major ways, the first being the addition of new parameters to monitor related to human health, such as blood pressure, by integrating multi-function IoT sensors. The second upgrade would be how data relating to human health is handled, due to privacy issues. And finally, the last improvement is to also virtualize equipment and cargo for better asset coordination.

Both projects successfully implement an indoor system based on DT which focus on the safety of their inhabitants. The proposed frameworks can be used for building different systems, based on indoor positioning, and tracking and had a similar structure, being made up of, in most cases, similar layers. However, the first framework only dealt with the environment, and did not track any physical target through the building and the second, even though it tracked workers, not only it did not do the same for objects, but it was also focused on verifying their behavior, and it did not predict their future positions and status.

2.4.3 Analysis of Relevant Architectures

The architecture for this project, that will be mentioned on the next chapter, was built with the proposed frameworks referenced as a starting foundation, in other words, it's layer based, each layer having a different structure and a unique function.

The way these layers were built depends on the project itself and the context surrounding it, meaning that even though they ended up with various differences and functionalities, analysing their structured showed a way to generalise each layer into one of four categories, that fulfill the definition proposed by M. Grieves of a DT and any type of interaction with it. These categories are as follows: Real World and Data Gathering, Data Transformation and Transportation, Virtual World and Data Processing, Application and Data Displaying.

Real World and Data Gathering

This first type of layer stands at the bottom any framework and represents the Physical Space in the 3 component DT architecture, and includes any type of physical entity that could influence the gathering of data for the system to use, this includes the actual entities being modeled and virtualized in the system and the sensors and data gathering equipment being used. These entities can range from a small to a large amount, and from few to many, depending on the case.

The first framework described in the state of the art [21] uses a "Perception" Layer, that uses indoor environment information acquisition terminals relying on LoRa as a transmission protocol. These consist of various parts, such as a LoRa module and various sensors used to measure the operating conditions of a building, that can sense oxygen, carbon-monoxide and smoke concentration, temperature and door/window opening and closing statuses. Other type of equipment present in this layer are cameras, which capture indoor images to determine the number of people in the various rooms.

Because it's objective is to manage indoor safety, the building itself is also part of this layer, in the form of a BIM model.

Another interpretation of this type of layer is showcased in [22], which proposes a "Physical World" layer, and any asset present in a warehouse as a part of this layer, this includes Space that refers all humans, machines, materials and their precise location information, and Time that represents current and any historical points during the lifecycle of a physical entity. Part of the "IoT Devices and Services" layer present in this last document also falls under this category, because it includes the physical data gathering sensors.

Data Transformation and Transportation

This type of layer is representative part of the "Information Processing Space", focusing entirely on managing and allowing communication to be possible between the real physical entity and its virtual counterpart.

The work [21] implements a "Network Layer", that is built using LoRa wireless communication technology in a Low Powered Wide Area Network (LPWAN) that receives indoor safety information data from its sensing terminals mentioned before and sends it to a cloud server through a 4G network. This cloud server can then be accessed by its local server (part of the next layer) through the Internet.

As mentioned above, the other document referenced has a layer called [22] that mixes physical IoT devices and the transport of data, and because of this, it also falls under this category.

Virtual World and Data Processing

This layer is a fusion of two components of the DT model referenced above, being made up of the "Information Processing Space", focused on storage, processing and mapping data, and also being made up of the "Virtual Space" that represents the digital versions of the entities present in the real world, their current state and their operations.

An application of this type is the "Service Layer" present in [21], that uses a SQL database to store data related to the building and its parameters and performs data analysis and processing using an SVM algorithm for danger classification, categorization and alarms and assisting the management of the building by making suggestions based on the data it currently possesses.

On the other hand, [22] proposes a "Cyber World" that is manly seen as a portal for different stakeholders to view and monitor the Warehouse, however, because it also deals with data storage and some data processing, it also falls in this category.

Application and Data Displaying

The final type of category,unlike the previous three, does not represent any component in the architecture proposed by M. Grieves in [4], instead it portrays the interactions of outside agents, such as humans, with the DT architecture itself.

As mentioned above, the "Cyber World" layer fits perfectly in this category, as it enables interaction of the stakeholders of the project with the different DT present in it, such as altering parameter settings of the indoor tracking procedure such as fainting detection and noise value thresholds. The status of the assets is also showcased to any user and it can illustrate the assets situations and necessary actions. Another layer presented in the same document is the Stakeholders, such as operators, superiors and super managers are considered part of this type of layer because they interact with the system itself, being informed of the status of entities present in the physical world and being able to change data inside the digital world.

An “Application Layer” is showcased in [21], that includes a platform that allows for safety status monitoring, danger alarm and positioning both with a scene viewer, danger handling suggestions and danger classification, which is vital to give the users feedback on how the system that is being managed evolving through time, and in the case that a critical situation happens, they get notified.

CHAPTER 3

Architecture

Developing an effective system requires a structured approach to ensure both the quality of the development and the product itself. Following the previous chapter, enough information has been given to start introducing how the project was developed and why it was built in the first place.

This chapter will serve as a bridge between the theoretical analysis of the underlying technologies and concepts that need to be understood and a more practical application of these concepts in building a foundation for the project's construction. As we transition from theory to practice, various concepts on building the proposed framework will be explored to ensure that all context behind the system's building is understood.

In the following sections, the focus will shift to several critical aspects. It starts with showcasing the problem the project seeks to solve clearly and concisely. Then, its requirements will be exposed to understand the trajectory of how the project was built. Subsequently, the stakeholders will be identified with the intent to understand for whom the project is being built. Lastly, the alignment of the technologies used to build this system with the context of the problem will be explained and made clear to the reader.

After that there is an overview of the system's architecture, providing a bird's-eye view of how the various layers and components interact. This serves as a roadmap for the detailed exploration that follows. Then visiting the essential design considerations is essential, which include the definition of entities. These considerations lay the groundwork for the practical implementation steps that follow.

3.1 Problem Statement

As a stepping stone, the previously analyzed frameworks have features that can be particularly useful in such a context because they use gathered data to conclude whether or not to inform the primary users, operators, or staff of the abnormal behavior of the monitored entities(the building itself or the workers) to take preventive action.

Solving the presented issue will be helpful as a foundation for other software to help automate clinic management and, hopefully, scale it up to the level of large hospitals.

3.2 System Requirements

System requirements are helpful when designing a new system, as they impose on the developer the necessary features and the primary objectives the system must fulfill to be considered ready for use. However, it also guides the project's development, prevents unnecessary features from being implemented, and serves as a reference point during development and discussion.

These requirements can be divided into two sets: functional and non-functional. The former relates to what the system must accomplish, and often are conceptualized as the system's operational blueprint, delineating the array of tasks, processes, and operations that the system must perform. These operations encapsulate the essential capabilities of the system and are integral to achieving the core objectives it is designed to do.

On the other hand, the latter goes beyond the basic functionality and captures a broader spectrum of criteria that determine how the system should accomplish its goals and executes its functionalities. Parameters such as performance efficiency, reliability, scalability and maintainability are all parameters that can form a non-functional requirement. Adherence to this type of requirements augments the system's overall efficacy, resilience, and user satisfaction.

Following this, the rest of the chapter will be focused on exposing the main requirements identified for the project and explaining their importance for its development.

3.2.1 Functional Requirements

Essential requirements for our IoT system to handle multiple DT for our clients and track their position in the clinic were identified. These requirements were chosen to make the system proposed in this thesis to be able to meet the expectations of the thesis and the stakeholders that could benefit from the project.

The list of Functional Requirements goes as follows:

1. Asset Digital Representation: All entities detected inside the clinic must have a corresponding digital representation stored inside the system.
2. Remote Asset Management: The system must be able to manage all assets that it detects remotely, using all available information and methods that it has at its disposal.
3. Entity Location Prediction: A fundamental requirement in this project is the ability to predict where an entity is in real-time within the clinic, which is essential for understanding how many entities are inside the clinic.
4. Entity Identification: The system must be able to differentiate between every entity, or else it is impossible to know where it is and what its state could be.
5. Entity State Prediction: Knowing which entity the system is dealing with and its position, the system can begin predicting the state of the currently observed entities.

6. Concurrent Entity Handling: The system must be capable of dealing with multiple entities at once at any given moment, or it will not be able to deal with the average load of information that a clinic needs to process daily.
7. User Interface: An easy-to-use interface must be available for the target users to get the information they need from the system. For them, it is not relevant how we deal with the data coming from the clinic, and it is more relevant for them to access the data already processed and ready to use as they see fit.
8. Real-Time Data Visualization: The data that reaches the end user needs to come in real time because this data is used to make decisions and help manage the clinic.

3.2.2 Non Functional Requirements

The non-functional requirements were chosen to make the system the project structure more concise, up gradable, and easy to implement.

The list that follows show these requirements:

1. Use Open-Source technologies, protocols, and platforms: Open-source technologies have advantages as being transparent, cost-effective, and having an active community of developers, which makes it ideal for a project on the scale of this thesis.
2. Microservice Architecture: Microservices can be developed independently, have their individual tasks, and be decoupled from one another. Another advantage is that every microservice has its environment; they are modular, easy to maintain, and make faults easier to identify.
3. Layered Architecture: The application being divided into layers has advantages such as having a clear structure, separating tasks by layer responsibility, having less complexity, being scalable, and being easier to set up. Another advantage to this architecture is that it follows a similar approach as the projects found and discussed in the state-of-the-art.
4. Scalability: While the project does not have a scope that requires the system to adapt to substantial workloads, having the option to make it adapt in case the need arises is always useful.
5. Uniform communication protocol for inter-layer communication: To make the project more concise and easy to implement, it was chosen to maintain the same communication protocols between layers. It also makes collaboration between services easier, promotes system maintainability, and reduces needless complexity with multiple communication types.

3.3 Stakeholders and Expectations

Stakeholders are individuals or groups interested in the technology prototype and its outcome, and have influence and input on how it should be built and how it should function.

Because of its small-scale, this section will be brief, but it will go over who the stakeholders are and what they expect the results to be:

- Clinic Medics and Staff: These are the primary users of the system; they are the ones who will get practical improvement using it in their daily activities.

- Patients: These are secondary users of the system; they get a passive improvement in their experience because they do not use the system directly but are benefiting from the fact that the clinic will be more organized and give a better service to them.

Overall, the expectations are simple: improved clinic management, which in turn gives a better experience for the patients.

3.4 Alignment with Digital Twins

To understand why DT and IoT are technologies that benefit a project like this is essential.

The problem involves monitoring an indoor system, where IoT technologies excel at, because they can create intelligent systems of objects that communicate with each other, acquiring data that can be used for management purposes.

This is beneficial to apply DT. They encapsulate every entity that needs to be monitored and can update the virtual version in real time. The virtual version can also give notifications that can affect the physical counterpart.

In the context of a clinic management system, the DT mirrors the physical system, which includes the patients, staff and equipment, and even intangible aspects like workflows, behaviors and stats. This digital mirroring facilitates real-time monitoring, predictive analytics, and strategic decision-making, all underpinned by real-world, real-time data.

All this encapsulates the three pivotal components of a DT: the physical world, the virtual counterpart and the data link that connects and updates the other two in a continuous synchronization and bi-directional way.

3.5 System Architecture Overview

The architecture of any software project forms the backbone of its design and functionality. In this section, the fundamental aspects of the system's architecture will be explained in detail, going over the type of architecture used, the tasks of each element, how they achieve said tasks, and the data flow from start to finish of execution.

As mentioned in previous chapters, the system is built using a Layered Architecture to satisfy the established non-functional requirements and because it is used commonly in systems similar to the one proposed in this document. Each layer is imbued with a specific responsibility. However, this does not mean that each layer is just one service executing tasks. It is the opposite. In most cases, each layer is composed of multiple smaller microservices, each with an even more atomic objective that it must fulfill while working together with other microservices in the same layer or outside of it.

The system's design, visible in Figure 3.1, can be related to what was introduced in Chapter Two regarding DT architectures. A small-scale project such as the one developed will have the three main components of a 3-Component Architecture. To remind the reader these are the Physical, the Digital, and the Information Processing Component. Each piece can be identified in a corresponding architecture layer: The Gathering Layer represents the Physical component, as it comprises all IoT technologies that gather data on the target that needs to

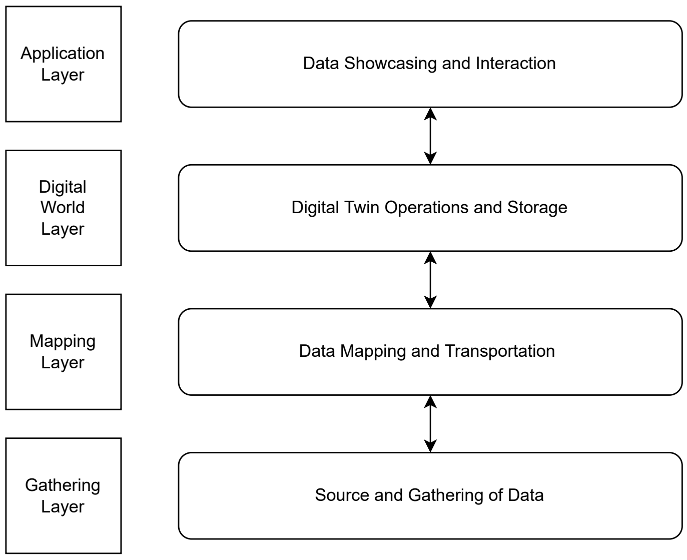

Application Layer
Data Showcasing and Interaction
Digital World Layer
Digital Twin Operations and Storage
Mapping Layer
Data Mapping and Transportation
Gathering Layer
Source and Gathering of Data

Figure 3.1: System Architecture Overview

be tracked and digitized. The Mapping Layer converts the raw data collected into a usable state that can be applied later, just like the described Information Processing component. Finally, the Digital World Layer is a direct implementation of the Digital Component, being responsible for transforming the processed data into helpful predictions and simulations that can influence the Physical component directly or indirectly. There is one last layer that is left out of the 3-Component Architecture, and that is the Application Layer. The lack of a matching component to the mentioned architecture is because a DT does not require one. However, it is helpful to exist as it gives a clear and visible representation of what is being processed in the whole DT ecosystem.

It is essential to state that changing the overall framework of the DT is possible, and having a different architecture, like the 5-Component Architecture, is feasible, especially in more extensive and more complex scenarios where the data load and overall scope of the space the system is dealing with is far more extensive than a clinic. In such a case, having more layers, each with its specialized theme, could benefit the performance and outcome of the system.

Next, a quick summary of each layer will give further insight into how they work.

3.5.1 Gathering Layer

The Gathering Layer fits the role of a "Real World and Data Gathering" layer, comprising any technology focused on indoor positioning data acquisition, such as antennas, RFID tags, Bluetooth sensors, or any other IoT sensor, the physical entities, such as patients or equipment.

The purpose of this layer is to gather positioning data in real-time. The data is in a raw state that must be processed for any state predictions. This layer also sends the collected data to the layer above it for this data processing.

Because it interacts directly with the physical world, it is the lowest layer in the architecture.

3.5.2 Mapping Layer

The Mapping Layer serves two purposes: identifying the type of entity with the data it receives and transforming the data from a raw state to a usable form.

Entity identification is made using the RFID associated with an RFID tag, and depending on the result, a specific payload will be generated to be sent to the layer above.

While most data it receives comes from the Gathering Layer, it will also receive notifications from the layer above for confirmation that the entity it is dealing with already exists. When an entity is already saved, it will simply create an update message to be sent instead of creating a new DT to speed up the overall data mapping process.

3.5.3 Digital World Layer

The Digital World Layer is dense with responsibilities. It stores various forms of data, such as state transitions and DT; it determines the current state and state changes, notifies if entities are already part of the system, and updates the user interface in the layer above.

State transitions are made possible due to the implementation of state machines that use the processed data from the Mapping Layer to verify various parameters with the current payload to change the state if certain conditions are met.

These state machines were built using specific scenarios of the typical behavior of the entity associated with said machine. Both of these concepts will be further explained later in this chapter.

3.5.4 Application Layer

The Application Layer serves the purposes of data display and user interaction.

It comprises a backend that receives data from the Digital World Layer. The data in the notifications it receives is then sent to the frontend.

While it is a simple application, the objective is to showcase the current state of the DT dwelling inside the clinic and to help the staff manage any asset as they see fit.

The system was built with a specific data flow in mind, it starts with detecting a tag by one or multiple antennas, that will then generate raw positional data sent to the Mapping Layer, where the entity associated with the tag will be identified.

The Mapping Layer will try to fetch the recognized entity, and in case it does, a payload to update is generated, otherwise, a payload create a digital version of the entity will be sent to the Digital World Layer that, when received, will be saved and used to predict the entity's current state. Any entity change will be saved onto a database and then it will notify the Application Layer.

The Application Layer will receive the data from below and update the User Interface for the staff to get a High-Level view of the state of all entities being tracked by the system. Depending on their evaluation, they may act as they see best.

3.6 State Machine Implementation

This section will delve into the critical considerations of the software system design relating to the state machine for dealing with patients.

The main factor that guided the development of the system was how data had to be transported and showcased; because all data was related on the entities of the clinic and in their virtualization, the answer lied in them.

By gaining insight into these entities and their connection with DT, we can comprehensively understand how the system operates and achieves its objectives. Real-time monitoring of these entities is essential for creating a responsive and dynamic healthcare environment that allocates resources based on real-time needs.

By seamlessly integrating physical entities into our DT framework, we can bridge the gap between the physical and digital aspects of the clinic ecosystem, which empowers us to make data-driven decisions, automate processes, and enhance patient experience.

With this in mind, state machines were built for the project to achieve its goals of monitoring and predicting behavior, and will be showcased, as well as the thought process for their development.

This section will delve into how these machines were developed, starting with all relevant entities that dwell within the clinic, the scenarios that encapsulate their behavior and ending on the state machine itself and how it works.

3.6.1 Physical Entities

In the context of our IoT system designed for a healthcare clinic, many physical entities play a crucial role in shaping the functionality and efficiency of the project.

These entities, including patients, medical staff, objects, and antennas, form the backbone of our ecosystem.

Patients are at the core of any healthcare facility, and their well-being is paramount. They are the recipients of medical care and attention. In this thesis, tracking patients within the clinic is essential to monitor their movements. This tracking aids in optimizing patient flow by predicting their current behavior, which will help reduce waiting times and enhance overall healthcare delivery.

The medical staff, including doctors, nurses, and support personnel, are the caregivers responsible for diagnosing, treating, and ensuring the well-being of patients. Tracking medical staff within the clinic facilitates the efficient allocation of resources, such as assigning doctors to specific patients or areas. It ensures that the right medical expertise is readily available when needed, leading to improved patient care and optimized workflow.

Objects within the clinic encompass a wide range of medical equipment, devices, and assets essential for diagnosis and treatment. Monitoring the location and condition of objects is critical for inventory management and ensuring that the necessary tools are available when required. This tracking minimizes downtime due to equipment unavailability and enhances the quality of healthcare services.

Antennas are the communication nodes that enable the IoT system to collect data, transmit information, and interact with other entities. They play a pivotal role in data collection, ensuring real-time updates on the location and status of patients and medical equipment. Antennas enable seamless communication between physical entities and their DT, facilitating data-driven decision-making and quick responses to critical situations. While they are considered entities in the project context, they are not modeled as a DT. They are only used to gather information from the environment in a periodic fashion. More information about how antennas are handled in the project is available later in this chapter.

3.6.2 Scenarios

Scenarios are representations of different behaviors considered typical for entities to display in the context of the system.

They are used to guide the development of state machines, to compare simulated results with what is expected, and to aggregate testing data.

Initially, all entities modeled as DT were planned to have scenarios to describe their behavior; unfortunately, due to problems during development, the scenarios for the medical staff and objects were cut out of the project.

For client entities, the scenarios used were:

- Appointment Behavior - The patient enters the clinic, awaits their turn, goes to the first corridor, enters the office, has an appointment, leaves the office, and then leaves the clinic. This behavior is expected when a patient has a doctor's appointment in the clinic.
- Treatment Behavior - The patient enters the clinic, awaits their turn, goes to the first corridor, goes to the second corridor in the direction of the treatment cabins, enters the treatment room, enters a cabin, is treated, returns to the reception room, and leaves the clinic. This behavior is to deal with patients that need to be treated by a doctor in the clinic.
- Giving Up Behavior - The patient enters the clinic, stays in the reception room, and leaves after waiting too long. This represents a patient who waited for too long in the reception area.

3.6.3 Client State Machine

Using the behaviors showcased in the previous section, a state machine was developed to simulate the expected states a patient can have when dealing with the clinic environment.

The absence of state machines associated with medical staff and objects results in their need for predefined scenarios, and, as stated previously, they originally were built with some standard behavior in mind but were cut in the final stretch of the thesis development.

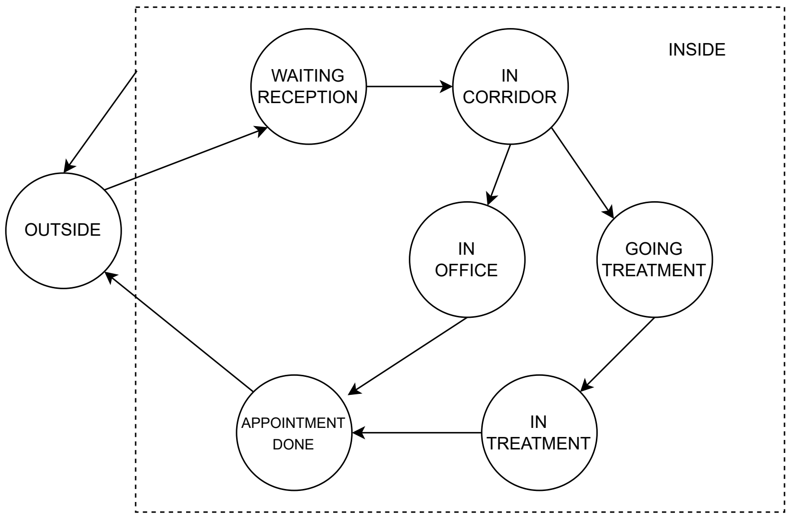


Figure 3.2: Client State Machine

Figure 3.2 showcases a patient state machine, where any possible patient life cycle can be observed.

The state machine has 7 possible states, but it is possible to identify two types of states: those of when a user is active inside the clinic, and the outside state that represents that the user either exited the clinic or timed out because no antenna could find its tag in a certain time frame.

A patient, when created, always starts in the Outside state and later moves on to other ones as data is received from the Gathering Layer.

All possible states that the state machine has are the following:

- OUTSIDE - Every client starts and ends its life cycle in this state. It represents the client staying outside the clinic and only has a state it can go to "WAITING_IN_RECEPTION".
- WAITING_IN_RECEPTION - The client enters the clinic and waits in reception. It only has a possible state to go "IN_CORRIDOR". It can timeout and return to being in the "OUTSIDE" state, indicating that the client waited too long, gave up, and left the clinic.
- IN_CORRIDOR - This state indicates that the patient was called for an appointment or treatment.
- IN_AN_OFFICE - This state means the client is in an office with a doctor, begins when the client enters the office and ends when the client exits it. This state means that the patient is in an Appointment Scenario.
- GOING_TO_TREATMENT - This state of the client's life cycle is triggered when he is in a Treatment Scenario, which represents the client walking through the corridor to the treatment area.

- IN_TREATMENT - This state can only happen in a Treatment Scenario after the "GOING_TO_TREATMENT" state. It represents that the patient entered the treatment area successfully and is currently receiving treatment with a doctor.
- APPOINTMENT_DONE - This is the last state for the patient in both the Treatment and Appointment Scenarios. It means the client successfully attended the clinic, either just a quick checkup or complete treatment. The resulting state will be "OUTSIDE", ending the life cycle.

The state transitions don't change by chance, and instead have associated parameters and data from inside the clinic. This data is gathered by the antennas.

The state machine will change the patient's current state upon receiving that necessary data. This process involves considering its current state, one specific antenna that will trigger the change,the distance to this specified antenna, and whether the patient is moving closer to or farther from the antenna. The module identifies the next state in the patient's behavioral sequence by evaluating these factors.

The state machine also has an associated timeout period to avoid indefinite waiting. If no data arrives within the specified timeout, it will reset the itself by returning the patient's state as if they were outside of the clinic.

The transitions of patient states within the state machine are driven by a series of predefined parameters and the data acquired from within the clinic, acquired through the antennas scattered throughout the clinic.

The state machine is programmed to initiate a transition in the patient's current state upon receiving data gathered by the antennas. This transition process incorporates multiple factors: the patient's immediate preceding state, the identification of a specific antenna acting as the trigger for the state change, the measured distance between the patient and this antenna, and the trajectory of the patient's movement. Based on these criteria, the state machine predicts the subsequent state in the patient's behavioral sequence, thus facilitating a dynamic and responsive patient tracking system.

Incorporated into the state machine's is a fail-safe mechanism in the form of a predefined timeout period. This feature is essential for avoiding a scenario of indefinite waiting in the absence of incoming data. Should the system encounter a situation where requisite data is not received within the established timeout threshold, it resets the patient's state, as if the patient were positioned outside the clinical premises, thereby ensuring the system's robustness in data-limited contexts.

CHAPTER 4

Prototype Implementation

In the preceding chapters, we've explored the conceptual foundations, and critical considerations of the DT based clinic asset management system. With a solid understanding of the DT framework and the layered microservices architecture, the implementation of the project will be explained in this chapter.

The chapter goes through each layer of the system, and, layer by layer, the functionality, data flow, and communication mechanisms that underpin the system are explored.

4.1 Gathering Layer

The Gathering Layer stands as the primary interface between the system and the physical world. Entrusted with the crucial task of collecting and processing raw data, this layer captures information generated by various physical entities within the clinical environment. Its operations are integral to transforming this raw data into a format that is usable for subsequent layers of analysis and application.

In terms of composition, the Gathering Layer encompasses a diverse array of components: modeled entities, their physical environment, and an assortment of IoT devices, such as RFID tags and antennas. Furthermore, the layer employs a suite of microservices and tools, designed to facilitate the conversion of gathered data into a format that is readily usable by the Mapping Layer.

As illustrated in Figure 4.1, the architectural construct of the Gathering Layer is composed of several key elements: the Data Source, entities inside the clinic, antennas that serve as the primary data collectors, and the predictor that is responsible for data analysis and interpretation. Each of these components plays a unique role in the data gathering process, ensuring the efficient and accurate acquisition of information that is pivotal for the system.

This section is dedicated to an in-depth exploration of the Gathering Layer, providing a comprehensive overview of its components and their functionalities and the significance of this layer in the system.

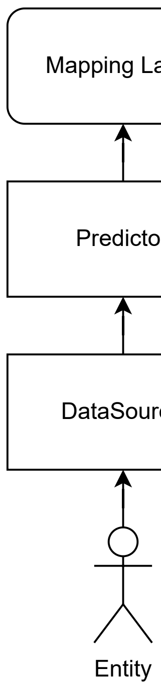


Figure 4.1: Gathering Layer

4.1.1 Data Sources

One notable feature of the Gathering Layer is its adaptability. The architecture of this layer can vary significantly based on the data source it interacts with. The system utilizes two primary data sources: the simulator and the clinic. The architecture and components of the Gathering Layer are tailored to efficiently handle data from each of these sources.

Both of these sources were used throughout the development lifecycle of the system, with different objectives in mind.

Simulator

The simulator is a web tool developed to generate indoor positioning data. It comprises two components, the frontend and the backend, and all communication between these components is done by sending a JavaScript Object Notation (JSON) message through Message Queuing Telemetry Transport (MQTT) Protocol using Mosquitto MQTT.

The previously mentioned protocol is a lightweight and efficient messaging protocol for reliable communication between devices in low-bandwidth, high-latency, or unreliable networks. It is used in systems where efficient real-time communication is essential, such as IoT systems like the one proposed in this project.

MQTT follows a publish-subscribe communication model, where clients can publish messages on specific topics, and other clients can subscribe to other topics of interest. This model enables efficient data distribution and decouples message producers and consumers.

This protocol is used throughout most microservices in the system to maintain consistency of communication between layers.

Another key feature of the simulator is that it uses Mapbox Gl, a suite of open-source libraries that allows the creation of dynamic web applications with the ability to use maps and is built using JavaScript and Python.

During the initial state of the system, a reliable data source was necessary during the developmental phases. The simulator emerged as the ideal solution, providing simulated data in GeoJSON files.

GeoJSON data was generated manually and served as a crucial asset during the early stages of the project. It was used to prototype and refine entity behavior, entity tracking, and state determination, and comprehensively test and validate the system's initial design and logic using simulated data.

Throughout the development process, the simulator provided a controlled environment for testing under various scenarios.

It is important to note that the Simulator is a separate project developed alongside the one described in this document, which means that while they are integral parts of the system, their development and effectiveness are out of the grasp of this project.

When dealing with data from the simulator, the Gathering Layer is designed to interact with GeoJSON data files. It uses this data to mimic real-world scenarios for testing and development purposes. The architecture incorporates modules and data processing pipelines optimized for simulator data ingestion.

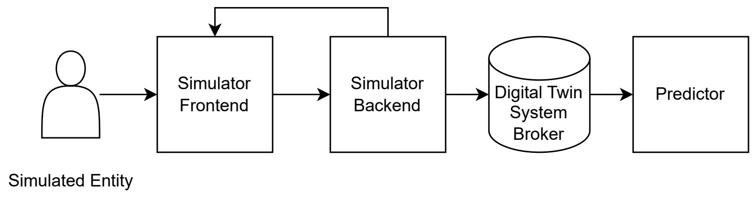


Figure 4.2: Gathering Layer using the Simulator

As seen in Figure 4.2, the Gathering Layer includes the simulator and the predictor. All data generated comes from the predictor and all communication in the layer and between layers is done using the MQTT Protocol.

Clinic

While the simulator was instrumental in system development, the final goal for the Digital Twin system lay in its ability to operate in the real-world clinical environment.

The introduction of clinic data marked a significant transition from development to real-world validation because it embodies the complexity and dynamics of an actual healthcare clinic, including information on patients, medic staff, objects, and their movements within the clinic.

This real-world data source challenged the system to adapt and perform seamlessly in a dynamic and unpredictable environment while also having to deal with the management of antenna behavior, which was done automatically by the simulator.

In this environment, testing for actual values related to entity identification and type counting was possible, as they were unpredictable, unlike with the simulator.

In contrast, when interacting with clinic data, the Gathering Layer employs a different architecture. This architecture is customized to handle real-time data streams from physical entities within the clinic, such as patients, medical staff, objects, and antennas. The Gathering Layer ensures that this data is efficiently collected, processed, and integrated into the system.

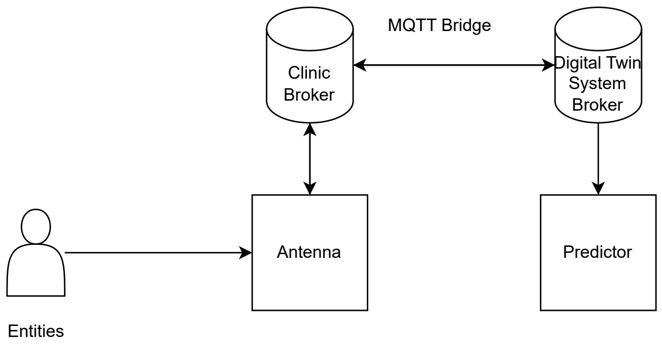


Figure 4.3: Gathering Layer using the Clinic

As seen in Figure 4.3, this architecture includes the entities that will be modeled and the predictor. Communication, however, is slightly different.

The clinic has its own MQTT Broker that communicates with the one responsible for the communication of the rest of the system using a Bridge between them.

All traffic that handles antenna communication and data gathering goes to the Clinic Broker, while the rest of the communication goes to the Digital Twin System.

The Gathering Layer handles real-time data streams efficiently. It establishes and manages connections to various sensors, RFID antennas, and data collection points throughout the clinic. This capability enables the system to receive and process data in real-time, supporting dynamic decision-making and management within the healthcare environment.

4.1.2 Predictor

The predictor is a component built in Python and is responsible for calculating distance using ML and different open-source libraries to do these operations.

This service, much like the Simulator, was developed alongside the one described in this thesis, and it links the “Gathering Layer” and the “Mapping Layer,” sending indoor information data in a JSON structure through the MQTT protocol.

Regardless of the data source, the Gathering Layer performs essential tasks such as data transformation and normalization. This process of data handling is done by the Predictor, which seen by the architectures mentioned previously, will always receive data regardless of the source.

It converts raw data into a standardized format that can be easily processed by subsequent layers. This process ensures consistency and reliability in the data flowing through the system.

4.2 Mapping Layer

The Mapping Layer plays a pivotal role in the overall architecture by serving as a vital bridge between the Gathering Layer and the Digital World Layer while also facilitating data flow within the system.

Its primary objective is to process incoming data from the Gathering Layer, determine the entity type associated with RFID tags, and oversee the creation and update of Digital Twins in the Digital World Layer. By fulfilling these tasks, the Mapping Layer significantly enhances the efficiency and effectiveness of the system's twin management and synchronization processes.

Within this layer, the Twin Creator, working with the Mapping Agent, seamlessly communicates with the Digital World Layer. This combination of services establishes a robust layer that enables effective twin management and synchronization throughout the system.

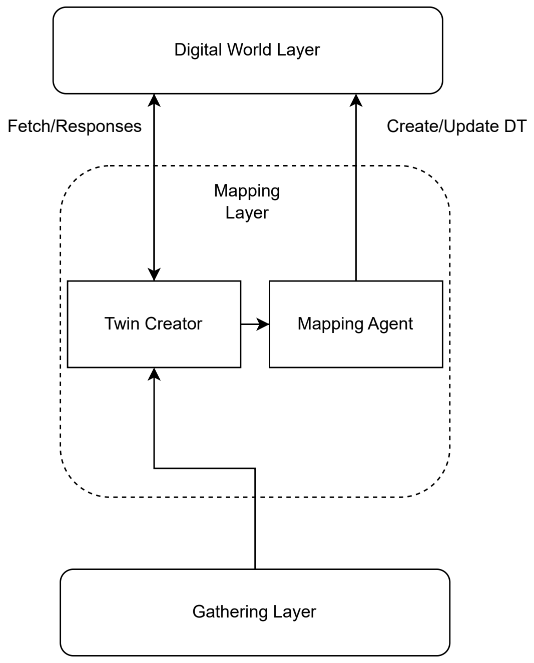


Figure 4.4: Mapping Layer Architecture

The structure of this layer can be seen in Figure 4.4

This section is dedicated to the Mapping Layer, providing an overview of its components, their functionalities and the tools used in conjunction with the layer that enables it to fulfill it's objectives.

4.2.1 Eclipse Vorto

Eclipse Vorto [1] is a fundamental tool used in this layer.

Eclipse Vorto was adopted not just because it is an open-source tool that is used for entity modeling into DT but because it also provides a way to map the results of this modeling into a protocol used to communicate with Eclipse Ditto, the Ditto Protocol, which facilitates communication with this central component of the project.

This open-source tool is also developed by the Eclipse Foundation. It gives modeling language a repository for the modeled entities and a mapping engine that can convert JSON data into different Protocols, such as AWS Cloud and Ditto Protocol.

These three tools given by Vorto are as follows: the Vorto Language, the Metamodel, and the Repository.

The Vorto Language is a Domain Specific Language (DSL) that describes DT using a simple grammar that lets developers focus only on describing expressively all of the attributes that characterize their DT modules. It uses two high-level classes, the Information Model for DT descriptions and Function Blocks for their capabilities.

The Metamodel defines how different entities, such as Information Models and Function Blocks, relate. The Information Model describes an entity such as a physical device and is made up of Function Blocks. It is common to see specific products, such as sensors, described using Information Models, while Function Blocks describe their capabilities.

The capabilities that a Function Block can describe are seen below:

- Properties - Properties can be a Status or a Configuration. The former is a read-only property that describes static information, while the latter is a read-write property that describes data that can be updated.
- Events - Events define the data emitted from a device or entity.
- Operations - Operations represent a function that can be executed,it can have arguments and return types.

The Repository provides two main functions: storage and mapping. It can store, manage, and distribute all Information Models created and re-utilize Function Blocks. When describing a DT, an interactive text editor is also provided to write code in Vorto Language. The Repository also provides the Mapping Engine capable of mapping incoming data in JSON to another Protocol, such as the previously mentioned Ditto Protocol.

4.2.2 Twin Creator

The Twin Creator is an integral part of the Mapping Layer. Its main objectives are discerning the type of entity associated with a tag received from the Gathering Layer and creating data payloads for DT creation or updating them if they already exist in the Digital World Layer.

[1] https://eclipse.dev/vorto/

To achieve the previously described objectives, it comprises three queues for data transportation and four threads, two for communication and two for data processing.

Starting with threads, each has its functionality, whether communicating with other microservices or layers, or data processing and management.

The two threads tasked with handling communication are the Incoming and Outgoing threads. The first receives data from the MQTT broker, either from the Mapping Layer or the Digital World Layer, and sends it to the following thread, while the latter sends messages it receives to the Mapping Agent.

The two threads tasked with data processing are the Message Processing Thread, tasked with handling MQTT messages, verifying the integrity of their payload, and determining what the message should be used for, while the Twin Processing Thread creates JSON messages with instructions for the Mapping Agent to use.

The Twin Creator microservice utilizes three queues to manage message flow and communication within the Mapping Layer and within itself.

The first queue is the Incoming Queue, which receives incoming messages from external sources, such as the Predictor or the Digital World Layer, and it links the Incoming Thread to the Message Processing Thread.

The second queue is the Twin Queue, which stores the messages processed by the Message Processing Thread. It connects this thread to the Twin Processing Thread.

The last queue is the Outgoing Queue, which connects the Twin Processing Thread to the Outgoing Thread.

4.2.3 Mapping Agent

The Mapping Agent plays a vital role in communication and synchronization between physical entities and their digital twins, acting as a mediator between the Twin Agent and the Digital World Layer. Its primary purpose is to receive data from the Twin Agent, convert it into the Ditto protocol, and facilitate the creation or updating of digital twins, ensuring data consistency and synchronization across the system.

The Mapping Agent integrates the Eclipse Vorto Mapping Engine to transform incoming data. This integration allows the Mapping Agent to map source data to Infomodel values representing DT. It supports separate mapping engines for different entity types, ensuring accurate representation of various entities.

After successful data mapping, the Mapping Agent creates or updates DT in the Digital World Layer. It utilizes the TwinPayloadFactory from the Eclipse Vorto library to transform the mapped results into JSON payloads. The payloads are modified to include the necessary topic and path information for DT creation or update in the Ditto Protocol. Finally, to ensure proper order and prevent data races, the Mapping Agent employs a queue for pending create or update requests, which are then published with the modified payloads to the MQTT broker to be sent to the Digital World Layer.

Vorto Model

To understand how the Mapping Agent converts JSON payloads into Ditto Protocol payloads, it is required to understand the underlying concept of Vorto Models and the ones developed for the entities that reside in the clinic.

In the project context, using Vorto models plays a pivotal role in streamlining the already-mentioned tracking and monitoring of entities.

Vorto models are a standardized framework for describing and modeling devices and their functionalities in the IoT ecosystem. As explained previously, these models provide a common language for IoT devices, making integrating, managing, and interacting with them within our system easier.

With Vorto models, the system can communicate seamlessly with various devices, regardless of manufacturer or type. This interoperability is crucial in a healthcare setting where different devices and equipment may come from different vendors.

By employing standardized Vorto models, we reduce the complexity of device configuration and data interpretation, streamlining the development process and ensuring that data from RFID tagged entities is efficiently processed and utilized.

Patients, medical staff, and objects within the clinic environment are all equipped with RFID tags. These tags serve as the primary means of tracking and gathering positional information. Due to this commonality in RFID tag usage and the fact that the information we gather from the physical world is the same, it is practical to utilize similar Vorto models for these entities.

The Information Model, seen in Listing 4.1 used for every entity, uses two Function blocks, Identification, Listing 4.2, and Position, Listing 4.3. The former is responsible for the positional data of the entity and represents the RFID tag itself. At the same time, Identification is used for entity type differentiation and represents the physical entity.

Listing 4.1: Entity Infomodel

```
infomodel Entity {
    functionblocks {
        mandatory identification as Identification "user data"
        mandatory position as Position "user position using tag"
    }
}
```

Listing 4.2: Identification Function Block

```
functionblock Identification {
    status {
            optional userId as string
            optional userType as string
```

```
        }
    }
```

Listing 4.3: Position Function Block

```
    functionblock Position {
        configuration {
            optional lat as float "current user longitude"
            optional lng as float "current user latitude"
            mandatory antenna_dist as string "min distance to
               an antenna"
        }
        status {
            mandatory tagId as string
        }
    }
```

4.3 Digital World Layer

The next Layer, and the center of the entire system because its primary responsibility is to manage the Virtual Component of Digital Twins, is the Digital World Layer.

As described in this chapter, its main functionalities are storing all of the Digital Twin data, processing it, and notifying certain services whenever a Digital Twin is updated with data from the physical space.

It comprises two distinct services, the first one being Eclipse Ditto, which has already been described in this chapter, and the other being the Digital Twin Agent.

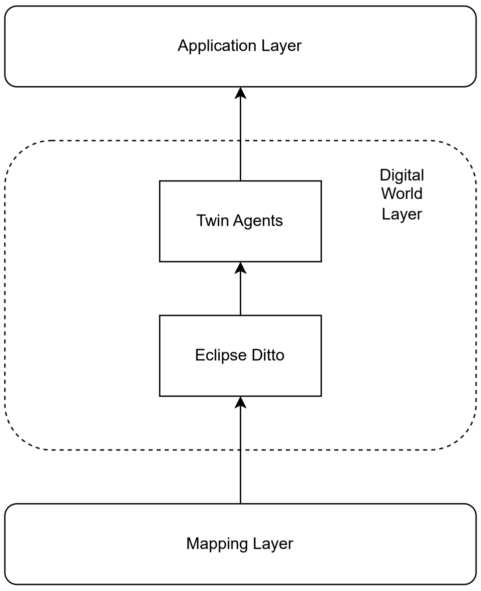


Figure 4.5: System Architecture Overview

The architecture of the Layer can be seen in Figure 4.5

4.3.1 Eclipse Ditto

Eclipse Ditto [2] is an open-source framework developed by Eclipse Foundation that acts as a centralized service for creating, managing, and interacting with Digital Twins in IoT.

It is the leading platform to store Digital Twins and manage communication in the Digital World Layer. It facilitates the integration between physical objects and the Digital World Layer, enabling real-time data collection and remote control of objects through an Application Programming Interface (API). It offers advanced features such as Digital Twin lifecycle management, event notifications, and access control, ensuring efficient and secure management of digital objects.

Furthermore, Ditto is highly extensible and interoperable, as it supports various communication protocols such as Hypertext Transfer Protocol (HTTP), MQTT, and Advanced Message Queuing Protocol (AMQP), allowing integration with different IoT devices and platforms. The platform also provides features for data mapping, message transformation, and integration with other services and ecosystems.

Ditto Connections

Connections, in the context of Eclipse Ditto, act as bridges between the DTs managed by Ditto and external entities in the rest of the system. These connections come in various types, each tailored to specific communication protocols and use cases.

Through custom connections, Eclipse Ditto offers the flexibility to integrate with external messaging services, including Eclipse Hono, RabbitMQT brokers, or Apache Kafka brokers.

[2]https://eclipse.dev/ditto/

This capability extends Ditto's reach to a broader ecosystem of messaging systems, facilitating data exchange and interoperability.

Connections in Ditto rely on transport protocols to transmit Ditto Protocol messages. This framework supports one-way and two-way communication, catering to various use cases, from simple data ingestion to complex command-response scenarios.

Ditto's connectivity model is extensible, allowing for protocol-specific customizations and adaptability. These protocols are responsible for sending data from outside to inside Ditto and vice-versa. Currently, Ditto supports several connection types, including AMQP, MQTT, HTTP, and Kaftka. These connection types accommodate diverse communication requirements and scenarios.

To maintain data integrity and security, Ditto employs enforcement mechanisms that validate the identity of devices and enforce precise message mappings to DT. These measurements guarantee that messages from external systems correspond accurately to their intended DT, enhancing overall system security.

Within connections, two fundamental components play pivotal roles: sources and targets.

Sources connect to external systems or message brokers to consume messages, encompassing various message types such as commands, messages, live commands, live responses, live events, and acknowledgments. They define multiple addresses, consumer counts, authorization contexts, and enforcement information, ensuring efficient data consumption and processing.

A critical aspect of sources is the ability to define reply targets. These reply targets are responsible for publishing responses to incoming commands. They inherit payload mapping from the parent source, encompassing address definitions, header mapping, and expected response types. Reply targets enable effective communication of responses, acknowledgments, and error messages.

Targets connect to message brokers or external systems to publish messages. They encompass different message types, including Thing messages, Thing events, Thing live commands, Thing live responses, Thing live events, policy announcements, and connection announcements. Targets specify one address, topics for message publication, authorization contexts, and header mappings for external headers.

The types of messages published to target addresses can be defined through configurations, allowing precise control over which messages are sent to external systems. Parameters for message filtering are specified, akin to HTTP query parameters, enabling fine-grained management of message publication.

Ditto empowers users to define which types of messages are published to target addresses through configurable parameters. This fine-grained control over message filtering ensures that only relevant data is sent to external systems. These filtering parameters are specified, like HTTP query parameters, enabling effective management of message publication.

4.3.2 Digital Twin Agent

This Agent receives data from Eclipse Ditto, processes the Digital Twin data by predicting the current state of an entity using state machines, storing the current active entities within the clinic, and sending data to the Application Layer to be showcased to the clinic staff.

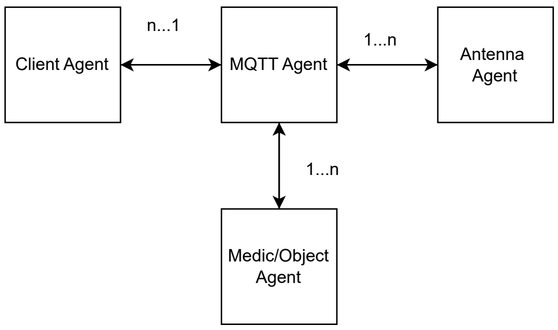


Figure 4.6: Digital Twin Agent Architecture

It is one Python microservice that is composed of multiple types of threads that deal with different virtual instances of physical entities. They will be called modules for the sake of the project.

The composition of this microservice, as seen in Figure 4.6, is:

- MQTT Agent - Main module of the microservice, it deals with communication between modules and other microservices, work distribution for each module, and module instantiation.
- Antenna Agent - Module that deals with physical antenna life cycle, sends and receives data from the physical antennas in the clinic.
- Client Agent - Module that represents and predicts the state of a patient.
- Medic/Object Agent - Modules that represent the medical staff/objects.

4.3.3 MQTT Agent

The MQTT Agent module serves as the central component of the Digital Twin Agent, as it acts as the bridge between external MQTT sources and facilitates seamless communication and coordination within the service and the layer, ensuring the proper flow of information and enabling effective management of the virtual entities and associated data.

Another vital function this module is responsible for is instantiating all Digital Twin logic whenever a new entity enters the clinic.

To deal with incoming messages from other modules and to sending jobs to the right modules, a command system was implemented. A command is a string that represents a job for the MQTT Agent or any other module to do. This command is added to a list with a payload for the task to be completed, and a module will only do its work if it recognizes the command.

Regarding data storage, this module maintains a virtual representation of the clinic's floor plan using a dictionary, where each room is associated with its geometry. It also uses dictionaries for storing access to the antenna, patient, medic, and object instances. However, data gathered on these entities is stored using MongoDB.

Architecture

This module follows a similar architecture to the one in the Twin Creator but is altered to fulfill its requirements.

It is comprised of the Incoming Thread, a thread that deals with incoming messages from the MQTT broker, then the Incoming Message Handler, which is a thread that deals with messages that come from MQTT or other modules within the Digital Twin Agent microservice to be distributed to the correct module, and finally the Outgoing Message Handler, which is responsible for sending messages through MQTT to services outside the Digital Twin Agent.

This module uses dictionaries to store both antenna macs and tag Universally Unique Identifier (UUID), and it has access to different queues to communicate with other modules of the MQTT Agents.

The final components of the architecture for miscellaneous tasks such as state change storage and future updates, such as the MongoDB database, which stores data related to state transition, loads the clinic map to predict in which area the entity might currently be, and connect to the MQTT broker to communicate with outside microservices.

4.3.4 Antenna Module

The Antenna Module represents the digital counterpart of a physical antenna. This module is responsible for managing the life cycle and variables associated with an instance of an antenna, and it fulfills this primary function by communicating with its physical component through the use of MQTT protocol through the transmission of data to the MQTT Agent, this in turn is made possible by using an input and an output queue to deal with messages that come from the MQTT Agent and has only one thread to deal with its task.

This module is only used when the Gathering Layer is communicating the clinic and not with the Simulator.

4.3.5 Entity Modules

Entity Modules represent the three entities that dwell within the clinic, the patient, the medic and the objects, and much like the Antenna Module, the Entity Modules have an input and output threads that communicate with the MQTT Agent.

The Client Module module plays the role of serving as the primary receptor of processed positional data collected throughout the system. This module tracks and predicts the patient's actions by utilizing state machines fed with data from other services. Additionally, it captures and stores the room where the patient is likely located using the provided data.

The Medic Agent, and the Object Agent were suppose to also have state machines associated with them, but due to problems with communication and time constraints, they never were implemented, instead they are just object instances that feed information to the Application Layer, as they have a similar data flow to the Client Agent, but they do not state associated with them.

4.4 Application Layer

The Application Layer stands at the top of the architecture, just after the Digital World Layer.

The primary purpose of the Application Layer is to provide a user-friendly and intuitive interface for healthcare professionals, staff, and administrators. This layer facilitates real-time monitoring, management, and decision-making by offering insights into the clinic's operations, patient status, and resource allocation.

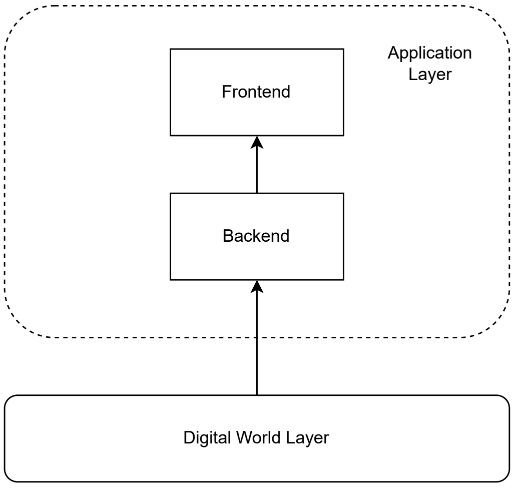


Figure 4.7: Application Layer Architecture

This layer architecture can be seen in Figure 4.7, and is composed of two parts: the backend and the frontend.

Smooth communication is a necessity for the Application Layer's functionality. It seamlessly connects with the Digital World Layer beneath it using the MQTT Protocol. However, communication between the frontend and the backend is done using WebSockets, because an open channel is required for real-time visual updates of the entities inside the clinic for the requirements setup for the project to be met.

WebSocket is a communication protocol that provides a full-duplex, bidirectional, and real-time communication channel over a single, long-lived connection between a client and a server.

WebSocket's full duplex communication enables two-way communication between client and server. Both parties can send and receive data independently, which is essential for the application layer because the processed data, notably positional data in a map, must be rendered as it is available. Unlike an HTTP request, which is stateless and follows a request-response pattern, WebSocket makes the desired outcome possible by allowing continuous, low-latency communication between the front and backend of the application layer.

The frontend of the Application Layer is the visual aspect that users interact with through web-based dashboards and interfaces. It's designed for ease of use and accessibility. The frontend leverages web technologies, such as Bootstrap, HTML, and JavaScript to ensure responsive design and compatibility across various devices. It also uses MapBox GL to render the map where the data is showcased.

The frontend gives a map of the clinic and a table for the users so they have access to patient, resources and clinic staff monitoring. Data presentation is a critical aspect, and we describe how real-time data from Digital Twins is visualized and made actionable.

The backend of the Application Layer handles the heavy lifting. It processes data from the underlying layers, and manages the communication with the frontend. The backend is built using Node.Js to ensure that it is small and effective at fulfilling its job.

CHAPTER 5

Evaluation and Results

The objective of this thesis was to develop a system that would accurately track specific entities, these being either patients, objects or staff, inside a clinic using DT and RFID technology and techniques.

The final stretch of work to be done is to evaluate the performance and results of the system, to understand if the proposed requirements were met and if it is usable for a real-life scenario, or at least, has the potential to be.

This chapter will go over the main functionalities that needed to be evaluated and tested, these being entity identification, entity creation, patient state transition, antenna control.

5.1 Entity Identification Evaluation

The entity identification system serves as a keystone in our Digital Twin model and a fundamental requirement for its work.

This functionality was built using the fact that the employed RFID tags have a UUID that, by definition, is unique to each entity that is in the clinic. All clinical staff and objects are known at the moment the system is running because it has a local file that maps the UUID to a name of an object or staff member.

When the system, in this case the Twin Creator, starts, it reads the file and converts it into an internal dictionary that serves as reference every time it receives data from the predictor. If the UUID in the message is in the dictionary it can either be of a staff member or an object, this will be identified immediately and an appropriate request to either create or update a Digital Twin will be made. However if the message contains an UUID that is not present in the dictionary, that means that the message contains data related to a client.

This procedure can be observed in Figure 5.1.

The message path within the Mapping Layer is based on the origin of the message it receives. However, they follow a similar data flow to ensure efficient processing and communication.

The Twin Creator is prepared to receive Predictor and Eclipse Ditto messages. These will trigger a data flow scenario to achieve a concrete purpose, seen in Figure 5.2.

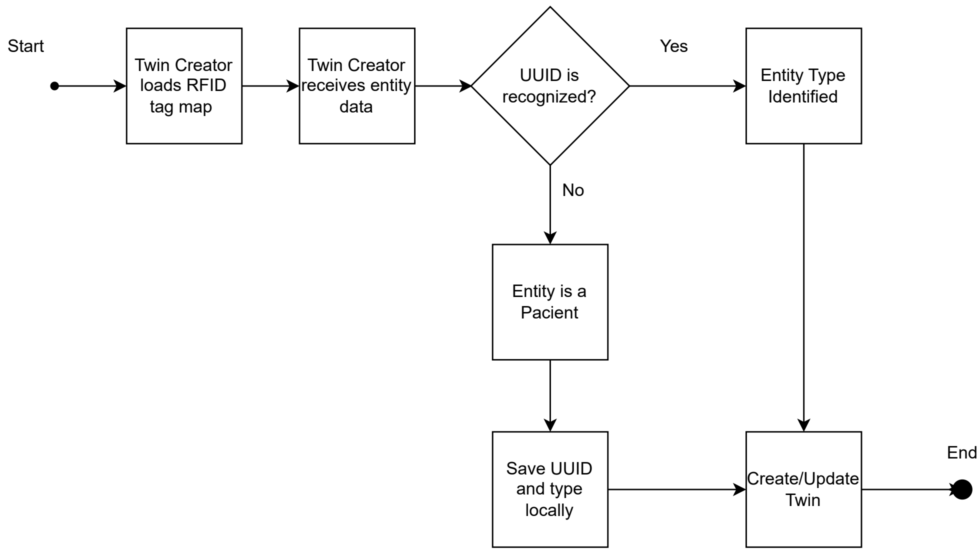


Figure 5.1: Entity Identification Flow Diagram

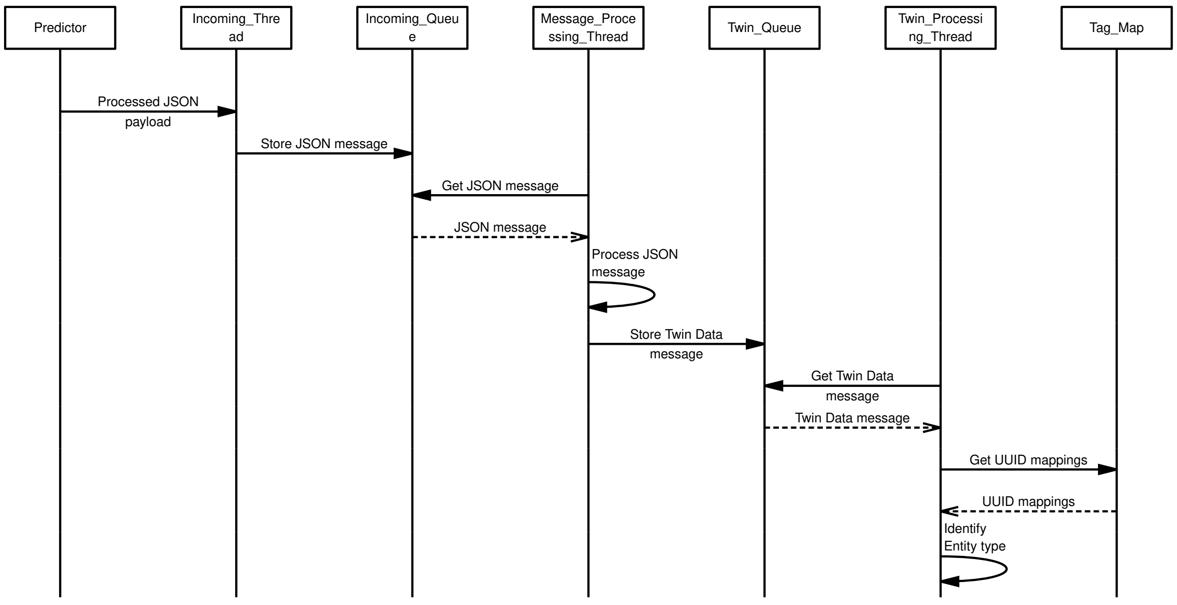
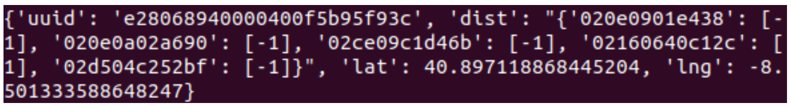


Figure 5.2: Entity Identification Message Sequence

The flow starts with the Predictor sending a message from the Gathering Layer. These messages always has the same type of content, the UUID of the tag, the predicted latitude and longitude, and the distance to the closest antenna. The message arrives in the Incoming Thread through the MQTT broker and is stored in the Incoming Queue.

The Message Processing Thread will fetch this message and try to convert it into a JSON object to make dealing with it easier. The MQTT topic of origin of the message will be determined, and if it is from the Predictor, it will proceed to the next step.

A JSON object is created, seen in Figure 5.3, with the UUID and distance to the closest



```
{'uuid': 'e28068940000400f5b95f93c', 'dist': "{'020e0901e438': [-
1], '020e0a02a690': [-1], '02ce09c1d46b': [-1], '02160640c12c': [
1], '02d504c252bf': [-1]}", 'lat': 40.897118868445204, 'lng': -8.
501333588648247}
```

Figure 5.3: Entity Data Message

antenna, extracted from the original message and the processed latitude and longitude. This object is sent to the Twin Processing Thread through the Twin Queue.

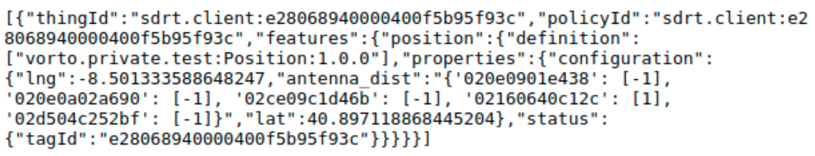
The next phase of this scenario is in the Twin Processing Thread, where the entity type is determined based on the UUID of the entity. This thread will consult the dictionary that maps a tag UUID to a type of entity, if the UUID is present then it will send the message with the DT data to the next service with the corresponding entity type, however if there is no match, the system will assume that the entity is a patient and send the same message to the one previously mentioned.

5.2 Entity Creation

The process of creating a new entity takes place after its type identification identification which, as seen previously, is performed by the Twin Creator microservice. When this happens, the Mapping Agent receives the data associated with the entity and its his responsibility to assimilate the data linked to the identified entity and construct a Ditto Protocol message to create an entry for this new entity in the Digital World Layer. This process has two steps: the integration into Eclipse Ditto, and creating an instance of an Entity Module in the Twin Agents microservice for the new entity.

The Mapping Agent initiates this stage by disseminating a Ditto Protocol message, encapsulating all the necessary data pertaining to the entity. This message effectively signals the integration of the new entity into Eclipse Ditto, utilizing the data structured in accordance with the Vorto Model. As illustrated in Figure 5.4, the first step is accomplished when Eclipse Ditto adds to its storage the data contained by the message published by the Mapping Agent.

Concurrently, the integration into Eclipse Ditto activates a Ditto connection. This connection plays a pivotal role in transmitting the entity data to the Twin Agent microservice. Upon receiving this information, the Twin Agent microservice undertakes the creation of a new instance of an Entity Module. This step in the workflow is depicted in Figure 5.5, showcasing the establishment of a running instance corresponding to the newly integrated entity.



```
[{"thingId":"sdrt.client:e28068940000400f5b95f93c","policyId":"sdrt.client:e2
8068940000400f5b95f93c","features":{"position":{"definition":
["vorto.private.test:Position:1.0.0"],"properties":{"configuration":
{"lng":-8.501333588648247,"antenna_dist":"{'020e0901e438': [-1],
'020e0a02a690': [-1], '02ce09c1d46b': [-1], '02160640c12c': [1],
'02d504c252bf': [-1]}","lat":40.897118868445204},"status":
{"tagId":"e28068940000400f5b95f93c"}}}}}]
```

Figure 5.4: Entity data stored in Eclipse Ditto

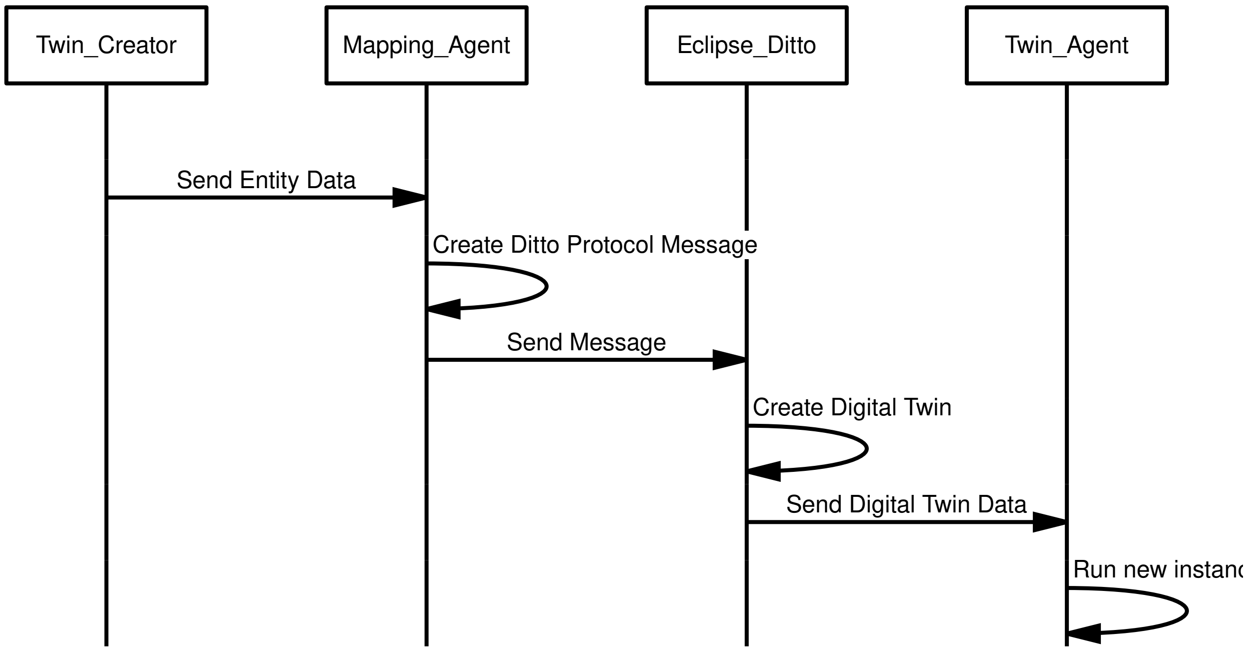


Figure 5.5: Entity Creation Sequence

The message flow between all services is seen in Figure 5.5.

5.3 Patient State Transition

When an entity already exists inside the Digital World Layer, all data that is gathered serves to update it. For the patients, however, the update will also affect the state machine.

The Client Agent starts its data flow by receiving incoming messages through a queue in contact with the MQTT Agent. These messages consist of two types of data: data for state determination and data specifying the patient's room.

The lifecycle of the Client Agent module follows a simple pattern. Upon start-up, the module sets the state to the default "OUTSIDE" and the current location to "Outside" and awaits new data in the queue. The queue also has an associated timeout period to avoid indefinite waiting. If no data arrives within the specified timeout, the module resets the state machine, returning the patient's location and state as if they were outside of the clinic.

The module determines the patient's current state upon receiving the necessary data. This process involves considering its current state, the distance to the specified antenna, and whether the patient is moving closer to or farther from the antenna. The module identifies the next state in the patient's behavioral sequence by evaluating these factors.

When a state change occurs, the module records the previous state and the transition to the new state in the database.

To end the life cycle, seen in Figure 5.6,this module will then send the positional data and state to the MQTT Agent, so it can be sent to the Application Layer to be displayed in a table, as seen in Figure 5.7.

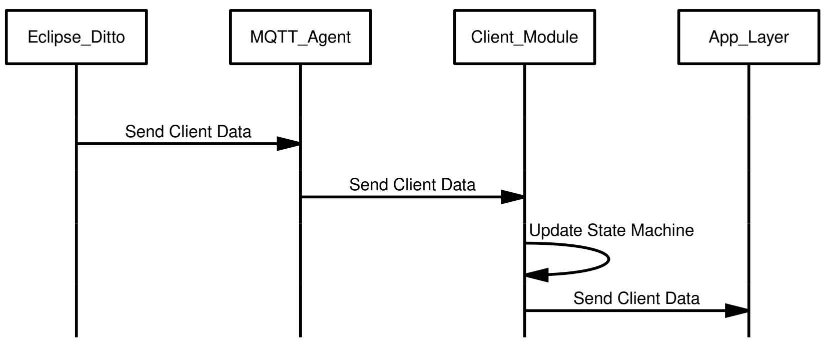
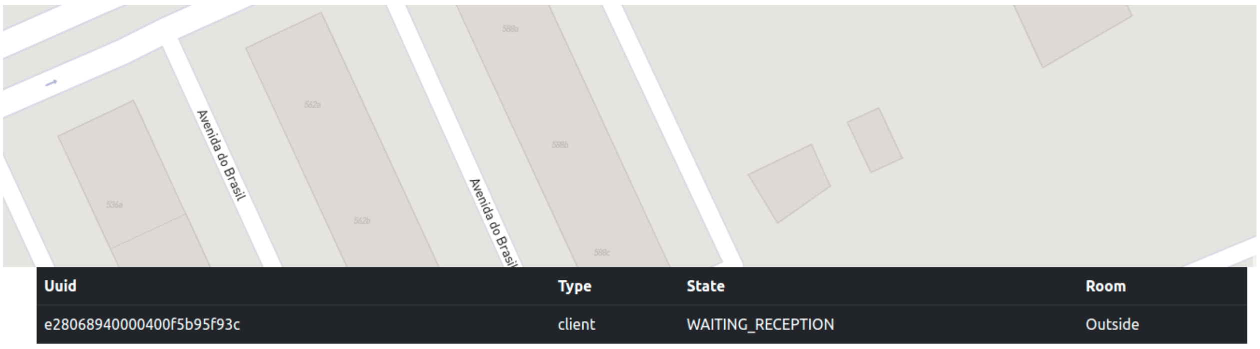


Figure 5.6: Entity Update Message Sequence



Uuid	Type	State	Room
e28068940000400f5b95f93c	client	WAITING_RECEPTION	Outside

Figure 5.7: Entity data displayed in the Application Layer

5.4 Antenna Control

The antenna module is responsible for managing the activity of the physical antennas inside the clinic. This module accomplishes this by following the life cycle seen in Figure 5.8 , which encapsulates the core logic required for a virtual antenna to interact with its physical counterpart and participate in the data-gathering process of the system.

This process only begins if the physical antenna is turned on and both the physical and digital antennas are connected to the MQTT Broker.

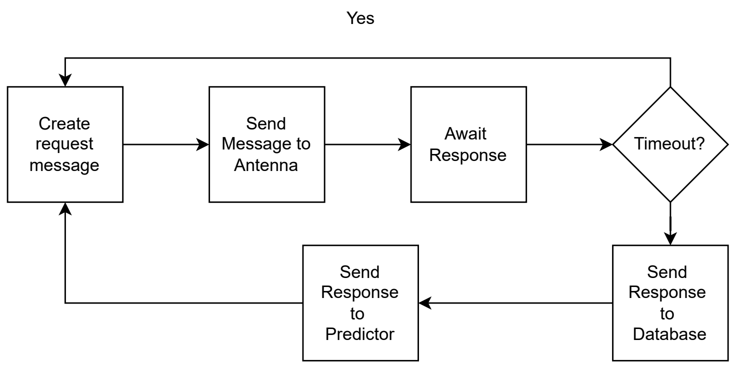
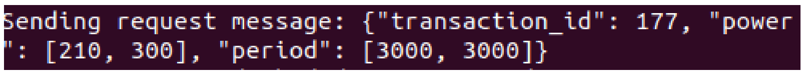


Figure 5.8: Antenna Agent Data Flow

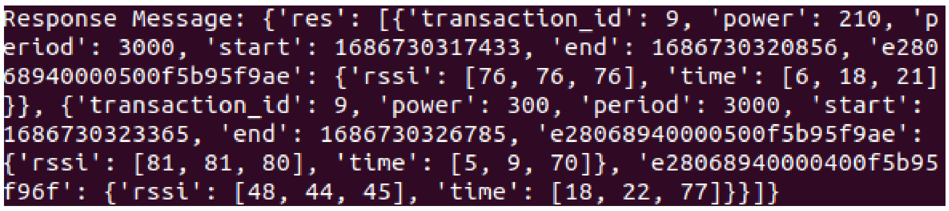
The life cycle starts with the antenna module preparing a message that will be used to activate the physical antenna, so that it can gather data on the position of entities inside the clinic. This message contains three essential values. The first value the transactional_id that tracks the number of times a virtual antenna tries to activate its physical counter-part, and it's shared across all virtual antennas. The second value is a list of energy powers for the antenna to use when gathering, and the final value is the time it needs to scan for each power. This message can be seen in Figure 5.9.

When the message is created, a it packaged into a payload for the MQTT Agent to send to the physical antenna. This package is also comprised of three variables.



```
Sending request message: {"transaction_id": 177, "power
": [210, 300], "period": [3000, 3000]}
```

Figure 5.9: Request Message for the Antenna Module



```
Response Message: {'res': [{'transaction_id': 9, 'power': 210, 'p
eriod': 3000, 'start': 1686730317433, 'end': 1686730320856, 'e280
68940000500f5b95f9ae': {'rssi': [76, 76, 76], 'time': [6, 18, 21]
}}, {'transaction_id': 9, 'power': 300, 'period': 3000, 'start':
1686730323365, 'end': 1686730326785, 'e28068940000500f5b95f9ae':
{'rssi': [81, 81, 80], 'time': [5, 9, 70]}, 'e28068940000400f5b95
f96f': {'rssi': [48, 44, 45], 'time': [18, 22, 77]}}]}
```

Figure 5.10: Response Message for the Antenna Module

First, the ANTENNA_GET command signals to the MQTT Agent that a message will be sent to the physical antenna. The second parameter is an MQTT topic specific to the antenna. Lastly, a JSON message specifies the antenna's data retrieval parameters. Subsequently, this array is enqueued into the output queue, awaiting transmission to the physical antennas via MQTT.

The virtual antenna then enters a waiting state, allowing a configurable time period in seconds to receive a message from the physical antenna. This message should contain the anticipated data. If the waiting time elapses without receiving the expected message, the virtual antenna considers the delivery unsuccessful and repeats the entire process.

Upon receiving the awaited message, the virtual antenna extracts the MQTT topic and payload. This message will not be processed, instead it packaged into a payload for the MQTT Agent to send to the Predictor. The message received contains a parameter named res that houses a list of gathered data for each level of power sent by the antenna module. Each member of the list contains the following attributes: transaction_id, the power level, the scanning time, the start time of scanning, the end time of scanning, and a dictionary for each tag it found during the search. This can be seen in Figure 5.10.

With this information, the antenna performs two crucial actions. First, it creates a list with two attributes: the ANTENNA_GATHERED_DATA command. This command instructs the MQTT Agent to save the received data to the antenna collection in MongoDB. The list also includes a dictionary that encapsulates the MQTT topic, payload, and the timestamp of the message's arrival.

Additionally, the virtual antenna creates another list with three attributes: the "ML" command, which prompts the MQTT Agent to transmit the gathered data to the Predictor for further processing. The list includes the data itself, which the Predictor will utilize to generate data for later use, and the MAC address of the antenna, allowing proper identification and association with the originating device.

The Antenna Agent Module ensures seamless operation and data flow within the system by effectively managing the communication and synchronization between the virtual and physical antennas. It enables data collection, processing, and transmission, facilitating subsequent analysis and decision-making by the Machine Learning Agents.

The sequence described can be observed in Figure 5.11.

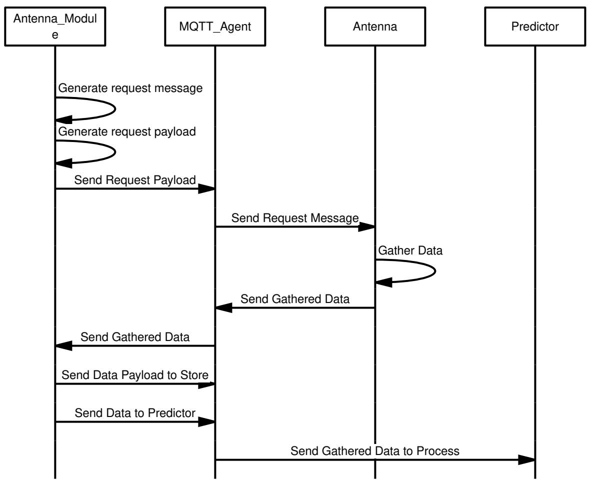


Figure 5.11: Antenna Control Message Sequence

CHAPTER 6

Conclusion

Digital Twin based systems have the potential to build powerful indoor management systems, as seen with previous projects that used indoor data to rebuild buildings and their inhabitants to simulate and manage the flow of data inside them.

The general Layered Architecture was analysed from previous done work, to build a new one for the use of this thesis. The new system, while having a similar general objective of managing an indoor space, it differs from the other analysed work in its system requirements and ability to change the direct data of a twin and use it for others analysis.

To make the system possible to run, various different Digital Twin open-source tools and platforms were employed, such as Eclipse Ditto and Eclipse Vorto, and a Simulator that represented a physical space and fed the system with indoor position and data relating to different asset points, that represented the various entities.

The original vision for the project had RFID technology and techniques in mind for the gathering of positional data, however it was not implemented as such. Because of this, features such being able to determine the rooms were left impossible to implement, only being able to guess based on the current state of the entity.

Even with this setback, the results were satisfying, and they are able to show the current state of the entities dwelling within the analysed space, but they can still be improved.

6.1 Future Work

In terms of possible work that could be done to improve the overall performance and usage of the system in the future boils down to two factors, the first being its application in a real physical space. Real data is fundamentally different than that generated in a simulated environment and it is impossible to create organic scenarios and paths that can enter in conflict with the system, and because of this it is essential to have real entities roaming around an actual real space to detect the faults of the system and improve it. Another positive of dealing with a real space is the limitation regarding the amount of entities that can be placed in the “Gathering Layer” will become less prevalent and the creation of a map that converts raw entity messages to a specific type stops being randomly generated by the system and instead is simply data that can be fed to the system and it handles the rest.

The second possible improvement is an expansion of the showcasing layer and its interactions with the physical world. This could be achieved through an expansion of the back-end to support operations that would notify entities when certain events would occur, this could be achieved by having a device that would produce sound and vibrate to warn a medic that an object that he needs is available. This could affect the way the state machines would operate. The current version of the “Showcasing Layer” can be seen as base that can be developed into a larger clinical platform.

One last addition that could be made is a better algorithm to determine the distance parameters, as well as a better dataset. The latter can be improved with more use of the system, specially if the data comes from a real world scenario.

References

[1] *Retrato da saude 2018.*

[2] *Retrato mundial de envelhecimento e saude.*

[3] B. Farahani, F. Firouzi, and K. Chakrabarty, “Healthcare iot,” in *Intelligent internet of things*, Springer, 2020, pp. 515–545.

[4] M. Grieves, “Digital twin: Manufacturing excellence through virtual factory replication,” *White paper*, vol. 1, no. 2014, pp. 1–7, 2014.

[5] M. Shafto, M. Conroy, R. Doyle, *et al.*, “Modeling, simulation, information technology & processing roadmap,” *National Aeronautics and Space Administration*, vol. 32, no. 2012, pp. 1–38, 2012.

[6] B. R. Barricelli, E. Casiraghi, and D. Fogli, “A survey on digital twin: Definitions, characteristics, applications, and design implications,” *IEEE Access*, vol. 7, pp. 167 653–167 671, 2019. DOI: `10.1109/ACCESS.2019.2953499`.

[7] A. Sharma, E. Kosasih, J. Zhang, A. Brintrup, and A. Calinescu, “Digital twins: State of the art theory and practice, challenges, and open research questions,” *Journal of Industrial Information Integration*, vol. 30, p. 100 383, 2022, ISSN: 2452-414X. DOI: `https://doi.org/10.1016/j.jii.2022.100383`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S2452414X22000516`.

[8] 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`.

[9] T. Sanislav, G. D. Mois, and S. Folea, “Digital twins in the internet of things context,” in *2021 29th Telecommunications Forum (TELFOR)*, IEEE, 2021, pp. 1–4.

[10] A. Sharma, E. Kosasih, J. Zhang, A. Brintrup, and A. Calinescu, “(2020). digital twins: State of the art theory and practice, challenges, and open research questions.,”

[11] J. Ríos, J. C. Hernandez, M. Oliva, and F. Mas, “Product avatar as digital counterpart of a physical individual product: Literature review and implications in an aircraft,” *Transdisciplinary Lifecycle Analysis of Systems*, pp. 657–666, 2015.

[12] F. Tao, J. Cheng, Q. Qi, M. Zhang, H. Zhang, and F. Sui, “Digital twin-driven product design, manufacturing and service with big data,” *The International Journal of Advanced Manufacturing Technology*, vol. 94, no. 9, pp. 3563–3576, 2018.

[13] F. Tao and M. Zhang, “Digital twin shop-floor: A new shop-floor paradigm towards smart manufacturing,” *Ieee Access*, vol. 5, pp. 20 418–20 427, 2017.

[14] Y. Zheng, S. Yang, and H. Cheng, “An application framework of digital twin and its case study,” *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 3, pp. 1141–1153, 2019.

[15] S. Mihai, M. Yaqoob, D. V. Hung, *et al.*, “Digital twins: A survey on enabling technologies, challenges, trends and future prospects,” *IEEE Communications Surveys & Tutorials*, vol. 24, no. 4, pp. 2255–2291, 2022. DOI: `10.1109/COMST.2022.3208773`.

[16] X. He, Q. Ai, R. C. Qiu, and D. Zhang, “Preliminary exploration on digital twin for power systems: Challenges, framework, and applications,” *arXiv preprint arXiv:1909.06977*, 2019.

[17] S. E. Ahmed, O. San, K. Kara, R. Younis, and A. Rasheed, “Multifidelity computing for coupling full and reduced order models,” *Plos one*, vol. 16, no. 2, e0246092, 2021.

[18] L. Lopez-Estrada, M. Fajardo-Pruna, S. Gualoto-Condor, J. Rios, and A. Vizan, "Creation of a micro cutting machine tool digital-twin using a cloud-based model-based plm platform: First results," *Procedia Manufacturing*, vol. 41, pp. 137–144, 2019, 8th Manufacturing Engineering Society International Conference, MESIC 2019, 19-21 June 2019, Madrid, Spain, ISSN: 2351-9789. DOI: https://doi.org/10.1016/j.promfg.2019.07.039. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2351978919310698.

[19] Z. Jiang, Y. Guo, and Z. Wang, "Digital twin to improve the virtual-real integration of industrial iot," *Journal of Industrial Information Integration*, vol. 22, p. 100 196, 2021, ISSN: 2452-414X. DOI: https://doi.org/10.1016/j.jii.2020.100196. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2452414X20300716.

[20] S. M. Bazaz, M. Lohtander, and J. Varis, "5-dimensional definition for a manufacturing digital twin," *Procedia Manufacturing*, vol. 38, pp. 1705–1712, 2019, 29th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM 2019), June 24-28, 2019, Limerick, Ireland, Beyond Industry 4.0: Industrial Advances, Engineering Education and Intelligent Manufacturing, ISSN: 2351-9789. DOI: https://doi.org/10.1016/j.promfg.2020.01.107. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2351978920301086.

[21] Z. Liu, A. Zhang, and W. Wang, "A framework for an indoor safety management system based on digital twin," *Sensors*, vol. 20, no. 20, p. 5771, 2020.

[22] Z. Zhao, L. Shen, C. Yang, W. Wu, M. Zhang, and G. Q. Huang, "Iot and digital twin enabled smart tracking for safety management," *Computers & Operations Research*, vol. 128, p. 105 183, 2021.