Faculdade de Engenharia da Universidade do Porto

Digital Twin for Plant Floor Kit

Francisco José Machado Coutinho

Mestrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Prof. Mário Jorge Rodrigues de Sousa

31st July 2023

Francisco Coutinho, 2023

Resumo

No contexto da Indústria 4.0, a integração de tecnologias de ponta transformou os processos de fabrico, aumentando a eficiência e produtividade. Entre essas tecnologias, a tecnologia digital twin tem recebido uma atenção significativa pela sua capacidade de criar réplicas virtuais de sistemas físicos, possibilitando monitorização, análise e otimização em tempo real. Esta tese foca-se em utilizar os avanços dos digital twins para simular o chão de fábrica nos laboratórios de automação da Faculdade de Engenharia da Universidade do Porto (FEUP). O objetivo é criar uma representação precisa do ambiente de produção, oferecendo uma plataforma para estudo, análise e otimização sem necessidade de rearranjos físicos.

O digital twin foi construído, utilizando um motor de jogos capaz de simular as leis da física, possibilitando interações físicas realistas. Esse alto nível de precisão permite a simulação, avaliação e otimização de vários cenários de produção sem necessidade de rearranjos físicos dispendiosos e demorados. Integrando protocolos de comunicação de automação industrial, o digital twin interage e opera consistentemente com as práticas industriais do mundo real, facilitando estratégias de controle inovadoras, avaliação de cenários e validação de algoritmos. A construção de um sistema modular e configurável permite uma personalização e extensão fácil do digital twin para representar diferentes layouts e configurações.

Além disso, ao longo da tese, um estudo abrangente de motores de jogo levou à escolha da Unity como a plataforma mais adequada para o projeto. O simulador existente nos laboratórios de automação da FEUP serviu de base para definir os requisitos e implementar o digital twin utilizando a Unity. Componentes-chave, como tapetes simples, tapetes rotativos e tapetes com máquinas, foram integrados e testados com sucesso. A implementação passou por testes rigorosos, validando a funcionalidade e confiabilidade de cada elemento.

No geral, a dissertação alcançou o seu principal objetivo ao desenvolver um digital twin baseado em física capaz de simular o chão de fábrica nos laboratórios de automação da FEUP. O sistema demonstrou eficiência, robustez e adaptabilidade, fornecendo um layout de fábrica realista e replicando eficazmente as funcionalidades do simulador existente. Este digital twin serve como uma ferramenta para otimizar os processos de fabrico, beneficiando tanto a academia como a indústria em cenários da Indústria 4.0.

Palavras-chave: Indústria 4.0; Digital twin; Motor de Jogo; Unity; GameObject; Modbus.

Abstract

In the context of Industry 4.0, the integration of cutting-edge technologies has transformed manufacturing processes, enhancing efficiency and productivity. Among these technologies, digital twin technology has gained significant attention for its ability to create virtual replicas of physical systems, enabling real-time monitoring, analysis, and optimization. This thesis focuses on utilizing digital twin advancements to simulate the plant floor in the automation laboratories of the Faculty of Engineering at the University of Porto (FEUP). The objective is to create an accurate representation of the production environment, providing a platform for experimentation, analysis, and optimization without the need for physical rearrangements.

The digital twin was built on a robust foundation using a game engine capable of simulating the laws of physics, enabling realistic physical interactions. This high level of accuracy allows for simulation, evaluation, and optimization of various production scenarios without the need for costly and time-consuming physical rearrangements. By integrating industrial automation communication protocols, the digital twin interacts and operates consistently with real-world industrial practices, facilitating innovative control strategies, scenario assessment, and algorithm validation. The construction of a modular and configurable system allows easy customization and extension of the digital twin to represent different layouts and configurations.

Furthermore, throughout the thesis, a comprehensive study of game engines led to the selection of Unity as the most suitable platform for the project. The existing simulator in FEUP's automation labs served as the foundation for defining requirements and implementing the digital twin using Unity. Key components, such as simple conveyors, rotating conveyors, and conveyors with machine tools, were successfully integrated and tested. The implementation underwent rigorous testing, validating each element's functionality and reliability.

Overall, the dissertation achieved its primary objective by developing a physics-based digital twin capable of simulating the plant floor in FEUP's automation labs. The system demonstrated efficiency, robustness, and adaptability, providing a realistic plant layout and effectively replicating the functionalities of the existing simulator. This digital twin serves as a powerful tool for optimizing manufacturing processes, benefiting both academia and industry in Industry 4.0 scenarios.

Keywords: Industry 4.0; Digital Twin; Game Engine; Unity; GameObject; Modbus.

Acknowledgments

First, I would like to express my gratitude to my supervisor, Prof. Mário Jorge Rodrigues de Sousa, for the high-level scientific knowledge transmitted, for offering invaluable advice, and for the opportunity to be part of this project.

To my friends, I am truly grateful for their support throughout this entire journey. Thank you for listening, offering valuable advice, and providing help whenever I needed it. Your presence has made a significant difference in my life.

To my family, thank you for encouraging me in all of my pursuits and inspiring me to follow my dreams. I am especially grateful to my parents, who supported me emotionally and financially. Their countless sacrifices, unconditional love, support and continuous encouragement throughout my life have led me to where I am now. This accomplishment would not have been possible without them.

Also, I would like to deeply thank all of those who contributed directly or indirectly to the completion of my dissertation and have helped me overcome the challenges in the past five years.

Index

List of Figures

List of Tables

Notation and Glossary

AI	Artificial Intelligence
ASCII	American Standard Code for Information Interchange
CPS	Cyber-Physical Systems
DDS	Data Distribution Service
FEUP	Faculdade de Engenharia da Universidade do Porto
HMI	Human Machine Interface
IoT	Internet of Things
ML	Machine Learning
MQTT	Message Queuing Telemetry Transport
OPC UA	Open Platform Communications Unified Architecture
PBR	Physically Based Rendering
PLC	Programmable Logic Controllers
ROS	Robot Operating System
RS	Recommended Standard
RTU	Remote Terminal Unit
SCADA	Supervisory Control And Data Acquisition
SFS	Shop Floor Simulator
TCP/IP	Transmission Control Protocol/Internet Protocol
TSN	Time Sensitive Networking

1. Introduction

1.1. Context

This dissertation aims to use the advances in digital twin technology, to simulate the plant floor in the automation laboratories of the Faculty of Engineering of the University of Porto (FEUP). The envisioned digital twin intends to accurately replicate the physical layout and dynamics of the production environment, providing a powerful platform for experimentation, analysis, and optimization. By developing a physics-based digital twin that accurately simulates the plant floor of FEUP's automation laboratories, this dissertation seeks to offer a valuable tool for academia, industry and researchers interested in optimizing manufacturing processes in Industry 4.0 scenarios.

1.2. Industry 4.0

In the era of Industry 4.0, the integration of cutting-edge technologies has revolutionized manufacturing processes, leading to increased efficiency and productivity [1]. One such technology that has gained significant attention and momentum is the digital twin technology, which offers a virtual replica of physical systems to enable real-time monitoring, analysis, and optimization. Digital twins have also developed as essential tools for simulating and managing the plant floor of production facilities, allowing for control, and exploration of alternative layouts for more efficient production lines [2].

To achieve this objective, the digital twin was built upon a robust foundation using a game engine capable of simulating the laws of physics. With the incorporation of realistic physical interactions, such as collisions, forces, and movements, the digital twin will provide a highly accurate representation of the plant floor, allowing the simulation, evaluation, and optimization of distinct production scenarios without costly and time-consuming physical rearrangements.

1.3. Objective

To accomplish this, the combination of the digital twin with standard industrial automation communication protocols will allow interaction and control of the virtual environment consistently with real-world industrial best practices, aiming at the implementation of innovative control strategies, the assessment of different automation scenarios, and the validation of algorithms. Furthermore, the construction of a modular and configurable system that accurately represents the various components and processes found in a typical production plant will be possible by assembling different blocks, each representing a specific unit within the production environment, such as simple conveyors, rotating conveyors, and conveyors with machine tools. Thus, digital twins can be easily customized and extended to have different layouts and configurations [3].

1.4. Document’s structure

This dissertation is comprised of seven chapters.

The first chapter corresponds to the **Introduction**, which includes a general framing of the issue and a study overview.

The Chapter 2, **State of The Art**, introduces the importance and properties of digital twins and their use in the industry. The studied digital twins and game engines are presented as well as the concepts related to communication protocols.

During Chapter 3, **Requirements and Modelling**, the real shop floor is presented with the current shop floor simulator and the requirements. The game engine to be used and the reason for it to be chosen are also discussed.

Then, in Chapter 4, **Implementation**, information about the materials and equipment used in the project, as well as the followed procedures, are described.

Following, in Chapter 5, **Results and Discussion**, the outcomes of this work are summarized and discussed. The challenges and potential difficulties are also mentioned.

Next, in Chapter 6, **Conclusion**, the main findings, discussions and final remarks of this project are assessed.

Lastly, in the **References**, the used pre-existing materials developed by others are properly cited.

2. State of The Art

2.1. Industry 4.0

2.1.1. Historical Context

Since the beginning of industrialization, significant advancements in technology have triggered fundamental changes that are now referred to as "industrial revolutions". These revolutions comprehend distinct periods of radical transformation: the first marked by mechanization, the second characterized by the extensive utilization of electrical energy, and the third characterized by the pervasive digitization of various industries [4], as represented in Figure 2.1. The first industrial revolution introduced mechanization using water and steam power, while the second brought mass production and assembly lines powered by electricity. Finally, the third revolution, also known as the digital revolution, introduced computerization and automation [5].

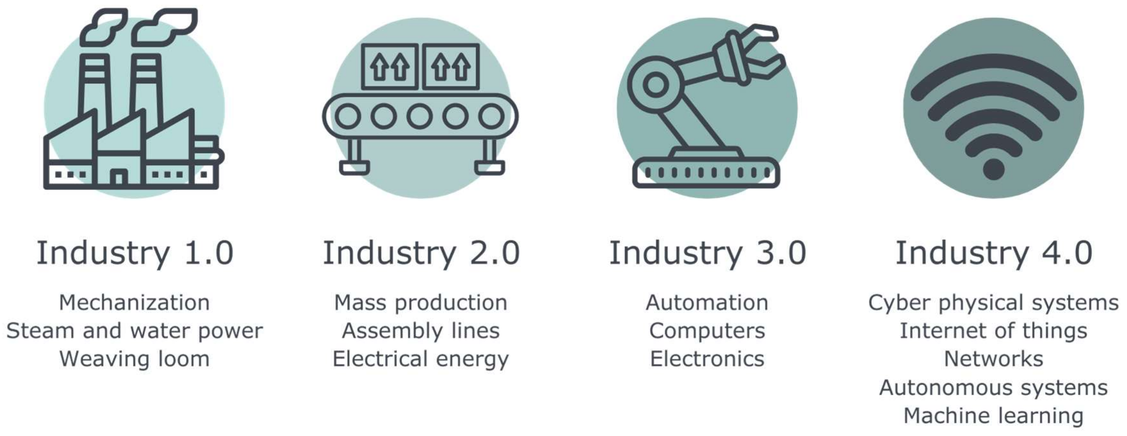


Figure 2.1. Advancements in each industrial revolution [4].

In recent years, the world has witnessed rapid advancements in various technologies, including artificial intelligence, digital twin, the Internet of Things (IoT), and big data analytics. The envisioned future of manufacturing revolves around highly efficient and modular systems, where products are capable of controlling their own production processes [4]. These technologies have had a profound impact on various sectors, including manufacturing, leading to the emergence of a new paradigm known as Industry 4.0. This concept originated in Germany in 2011 represents the fourth industrial revolution, which combines cyber-physical systems with advanced digital technologies to create smart, connected, and automated manufacturing environments. Industry 4.0 represents a leap forward, integrating physical and digital systems to enable intelligent and autonomous manufacturing processes [5].

2.1.2. Implications for the Manufacturing Sector

Industry 4.0 has extensive implications for the manufacturing sector, promising to transform traditional factories into smart factories improving efficiency since Industry 4.0 technologies enable real-time monitoring and optimization of manufacturing processes, leading to increased productivity, reduced downtime, and improved resource utilization. Since the technology of Industry 4.0 is cloud-based, there is the ability to connect machines, systems, and processes in a smart factory that allows for rapid reconfiguration and customization of production lines, enabling manufacturers to respond quickly to changing market demands. Another benefit of the Industry 4.0 is that it enables the integration of digital technologies into the products themselves, leading to the development of products that offer new functionalities and value-added services [6, 7].

On the other hand, the adoption of Industry 4.0 technologies also comes with some challenges, including a skilled workforce capable of managing and leveraging these technologies. Additionally, it also leads to a shift in job roles, with an increased focus on data analysis, programming, system integration, and cybersecurity since, due to its interconnected nature, the manufacturers become more vulnerable to cyber-attacks. So, ensuring the security and integrity of data and systems is a critical challenge in this new era of manufacturing [6].

Industry 4.0 in industrial practice is driven by two major forces: the application-pull and the technology-push. On one hand, there is a substantial demand for applications or solutions, which creates a significant need for changes in response to shifting operational circumstances. These changes are triggered by social, economic and political changes, such as the need for short development periods, individualization on demand, flexibility and resource efficiency [4]. On the other hand, there is a notable drive for technological advancements within industrial practices. However, when it comes to job-related aspects, particularly in industrial settings, innovative technologies are not widely adopted or implemented on a broad scale. Some approaches of a technology-push are further increasing mechanization and automation, digitalization and networking, and miniaturization [4].

2.1.3. Technologies of Industry 4.0

Industry 4.0 relies on a range of technologies that work together to create a connected and intelligent manufacturing ecosystem. Some examples are shown in Figure 2.2.

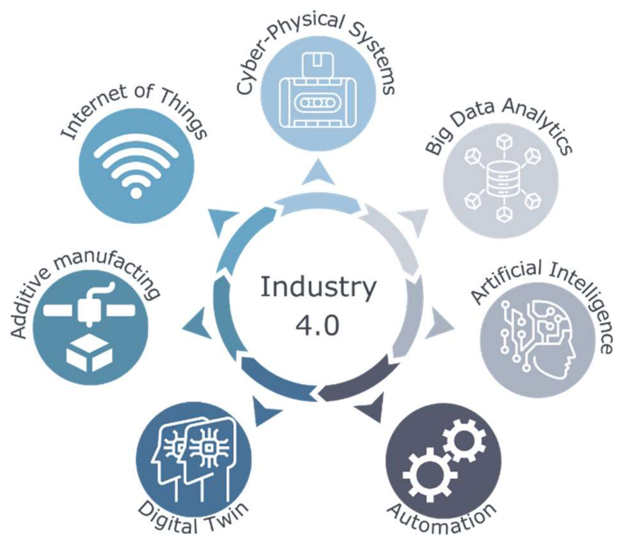


Figure 2.2. Examples of technologies of Industry 4.0 [7].

One of these technologies is the Internet of Things (IoT) which refers to the network of physical devices, sensors, and actuators embedded in machines and products, enabling them to communicate and exchange data with each other and with humans enabling real-time monitoring, control, and optimization of manufacturing processes [8].

Cyber-Physical Systems (CPS) is also one of the Industry 4.0 technologies and it refers to the integration of physical and computational components that interact with each other and the physical world combining sensing, actuation, and control capabilities to enable the seamless integration of the physical and virtual realms in manufacturing systems [9].

Big Data Analytics is another technology that generates vast amounts of data from various sources, including sensors, machines, and production processes. Big data analytics techniques are used to process and analyse this data to extract valuable insights, optimize operations, and support decision-making [10].

Finally, Artificial Intelligence (AI) and Machine Learning (ML) are two technologies that have gained a lot of popularity in recent years. These technologies play a crucial role in Industry 4.0 by enabling machines and systems to learn from data, make decisions, and perform tasks that traditionally required human intelligence. AI and ML algorithms are used for predictive maintenance, quality control, intelligent automation, and optimization in manufacturing [11].

2.1.4. Reference Architectural Model Industry 4.0 (RAMI 4.0)

Various diverse interests intersect in the debate surrounding Industry 4.0. To establish a shared comprehension of the essential standards, use cases, and other requirements for Industry 4.0, it was crucial to create a standardized architectural model. This model acts as a common reference point, facilitating discussions about its connections and intricacies. This resulted in the reference architecture model for Industry 4.0 (RAMI4.0), shown in Figure 2.3 [12].

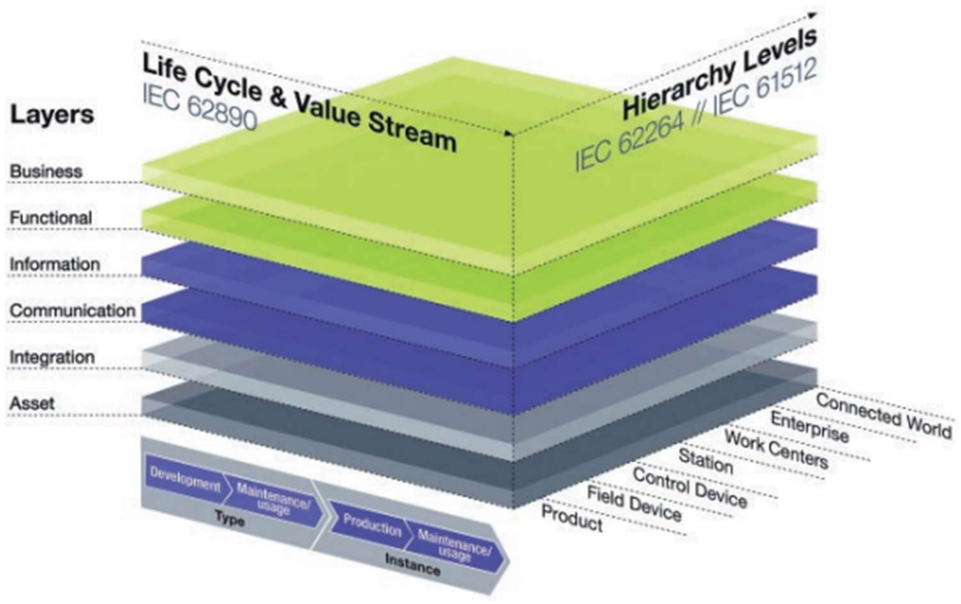


Figure 2.3. Reference architecture model for Industry 4.0 (RAMI4.0) [12].

So, RAMI 4.0 utilizes a three-dimensional coordinate system to capture essential aspects of Industry 4.0. By doing so, it simplifies complex relationships into smaller and more manageable clusters. This model allows to classify objects, like machines, based on the model, enabling the description and implementation of highly flexible Industry 4.0 concepts [13].

These three axes are, the Hierarchy Levels Axis, the Life Cycle & Value Stream Axis, and the Layers Axis.

The Hierarchy Levels Axis, shown on the right horizontal side, represents hierarchy levels from IEC 62264, which is an international standards series for enterprise IT and control systems. These levels define different functionalities within factories or facilities. To represent the Industry 4.0 environment, these functionalities have been expanded to include workpieces labelled as "Product" and the connection to the Internet of Things and Services, labelled as "Connected World" [14].

Additionally, the Life Cycle & Value Stream Axis, the left horizontal axis, represents the life cycle of facilities and products, based on IEC 62890 for life-cycle management. It distinguishes between "types" and "instances." A "type" refers to the design and prototyping stage, while an "instance" is when the actual product is being manufactured [14].

Lastly, on the Layers Axis, the six layers on the vertical axis describe the decomposition of a machine into its structured properties, layer by layer. This virtual mapping of a machine's properties draws inspiration from information and communication technology, where complex systems' properties are commonly broken down into layers [14].

2.1.5. Digital Twin

The concept of the Digital Twin emerged in the early 2000s and has since evolved with no standardized definition, highlighting its versatility across various sectors [13]. It can be seen as a simulation that spans multiple domains, a virtual representation of a physical object or product that has been manufactured, and a virtual substitute for a real-world object, among other interpretations [13, 15, 16].

Nonetheless, there is a general framework for the architecture of a Digital Twin, comprising three main elements: the physical world, the virtual world and the connectivity that links the two together [17]. Each element incorporates a variety of components dependent on the specific requirements of the designer. However, some fundamental components include sensors in the physical world (to collect real-world data), a physical twin, edge processing capabilities, data security measures, the digital twin itself, data processing capabilities (facilitated by artificial intelligence, machine learning, among other) and communication interfaces such as satellite, the internet, Bluetooth, and more [17].

To lay a foundation for understanding digital twin technology, at its core, a digital twin is a virtual representation of a physical object, process, or system that uses real-time data and simulation models to mimic and predict its physical counterpart’s behaviour and performance [17]. It is divided into three elements that consist of the physical entity, which represents the real-world object, the digital counterpart, which comprises a combination of data-driven models, algorithms, and analytical tools, and the connection between them, enabling continuous data exchange and interaction [17].

Digital twin enables real-time monitoring and analysis of the physical entity and by capturing and analysing data from sensors and other sources, they can provide valuable insights into the performance and condition of the physical object. They can also simulate future scenarios and test innovative ideas, preventing potential failures or malfunctions in designs before physical production. This information allows for a proactive decision-making, optimizing efficiency, and reducing downtime, as well as minimizing unexpected breakdowns, accelerating the innovation process, and reducing costs [18].

2.2. Digital Twin Platforms

There are several software platforms and companies that offer Digital Twins solutions and features for various industries, such as Dassault Systèmes (a 3DEXPERIENCE platform), ANSYS, PTC (which offers Industrial IoT capabilities), Microsoft Azure Digital Twin, IBM Watson IoT, SAP Digital Supply Chain, Oracle IoT, Siemens MindSphere, AVEVA, Bentley Systems.

This subchapter explores the features and benefits of two software platforms, Factory I/O and Siemens Digital Twin.

2.2.1. Factory I/O

Factory I/O is a modern virtual environment designed for the simulation and testing of industrial automation systems and provides a realistic and immersive platform for training, programming, and debugging industrial processes [19].

Replication of the physical components and processes of a factory floor, allows for the experimentation and optimization of the automation solutions without the need for a physical setup. Additionally, Factory I/O offers a vast library of industrial components such as sensors, actuators, conveyors, and robots, as well as support for various communication protocols commonly used in industrial automation, such as Modbus and OPC [19]. It also provides real-time feedback and visualization of process data, enabling to monitor, analyse and assess the behaviour of automation systems.

Furthermore, Factory I/O finds applications in different areas of industrial automation. It is commonly used for training purposes and control strategies before implementing them in real-world scenarios and can be utilized for debugging and optimizing existing automation systems, allowing to identify and resolve issues without interrupting the production process.

As a result, Factory I/O brings several benefits and advantages in industrial automation scenarios. Firstly, it significantly reduces costs by eliminating the need for physical prototypes and setups during the development and testing phases. Secondly, Factory I/O enhances safety by experimenting with potentially hazardous scenarios without risking injury or damage to equipment. Finally, it promotes innovation and creativity by providing a flexible and dynamic environment for exploring new automation concepts and ideas.

The Figure 2.4 shows an example of a shop floor simulation created using Factory I/O.

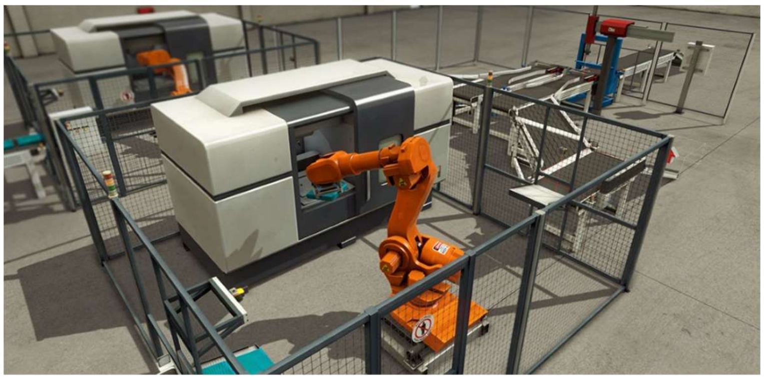


Figure 2.4. Example of a Factory I/O simulation [19].

2.2.2. Siemens Digital Twin

Siemens Digital Twin is a technology that enables the creation of virtual replicas of physical assets and processes and combines the power of real-time data, advanced analytics, and simulation models to provide a comprehensive digital representation of industrial systems. With the integration of the virtual and physical worlds, it offers new possibilities for monitoring, optimizing, and predicting the behaviour of complex industrial processes [20].

Additionally, Siemens Digital Twin comprises several key features and components, such as data acquisition systems, cloud computing infrastructure, simulation models, and analytics algorithms. The data acquisition systems collect real-time sensor data from the physical assets and feed it into the digital twin, enabling continuous monitoring and analysis. The cloud computing infrastructure provides the computational power required to process and analyse the large volumes of data generated by the digital twin. Finally, the simulation models and analytics algorithms leverage this data to simulate and predict the performance of the physical assets, enabling proactive maintenance and optimization [20].

Furthermore, Siemens Digital Twin solution has practical applications across several industries, including manufacturing, where it can be used to monitor and optimize production lines, identify bottlenecks, and simulate the impact of process changes [20].

So, Siemens Digital Twin offers numerous benefits and advantages for industrial processes. Firstly, it enables predictive maintenance, allowing one to identify and address potential issues before they cause significant disruptions or failures. This reduces downtime, improves asset reliability, and extends equipment lifespan. Secondly, Siemens Digital Twin enhances operational efficiency by providing real-time insights into process performance and enabling optimization through simulation and scenario analysis. Finally, it facilitates agile decision-making by providing a virtual platform for testing and evaluating innovative ideas and strategies, before their implementation in the physical environment [20].

The Figure 2.5 shows an example of a shop floor simulation created using the Siemens Digital Twin technology.

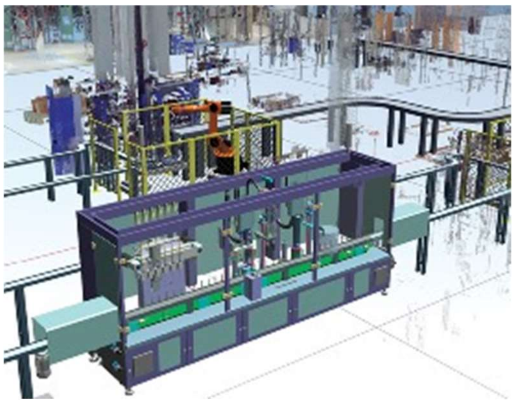


Figure 2.5. Example of a shop floor using Siemens Digital Twin [20].

2.3. Game Engines

Game engines offer game developers a framework to create video games without starting from scratch on all fundamental systems like physics, graphics and artificial intelligence [21, 22]. By providing pre-built tools, game engines save developers time and resources, eliminating the need to create and integrate these supporting systems manually [21, 22]. This allows developers to concentrate their efforts on the core aspects of the game, enhancing the overall game development process [21, 22]. Some of the most popular game engines are Unreal Engine, Unity, Godot, Amazon Game Engines, CryEngine and GameMaker.

In this subchapter, three of the powerful game development engines available, Unity, Unreal Engine and CryEngine, will be explored, examining their key features and advantages.

2.3.1. Unity

Unity, developed by Unity Technologies, is a cross-platform game development engine that provides developers with a comprehensive set of tools, resources, and workflows to create interactive experiences across various platforms. This game engine offers a wide range of features, including a robust editor, scripting capabilities, asset management, physics simulation, and integrated support for programming languages [23]. Its appeal list in its ability to create both 2D and 3D games for different platforms such as desktop, mobile, web and virtual reality [24].

Moreover, Unity's popularity among developers can be attributed to its accessibility, flexibility, efficiency and low power consumption. It is compatible with Mac, Linux and Windows, offering an artist-friendly range of tools for immersive designing and game world creation, alongside a strong developer toolkit for high-performance gameplay and game logic implementation. The editor offers a visual environment that allows one to design scenes, create game objects, and implement logic through a drag-and-drop system. Additionally, the asset store offers an extensive library of pre-built assets, scripts, and plugins, enabling them to accelerate their workflows and focus on creativity [25].

Unity's game development features include processing, asset tracking, scripting, and physics, all of which contribute to reducing game development costs and time. Furthermore, one of Unity's strengths is its stable code and well- designed architecture, leading to better game performance. It also supports high-quality video and audio effects, ensuring smooth and effective game development. The engine's runtime debugging feature allows developers to quickly identify and fix issues during gameplay [24].

Moreover, the Unity engine supports high-quality video and audio effects, providing significant benefits to game development in terms of intelligence and effectiveness. Remarkably, videos can be adjusted and displayed in various devices and screens without any compromise or distortion on picture quality [23]. The ability to deploy games across multiple platforms is a significant advantage, because not only saves development time but also expands the potential audience for a game, reaching players on different devices and operating systems [24].

Unity's extensibility allows developers to customize the engine to their needs, utilizing C# scripting to create custom tools, complex game mechanics, and integrate external libraries, pushing the boundaries of game development for unique experiences. The Unity integrated development environment (IDE) supports C# and JavaScript for scripting, offering essential functions suitable for game development [25, 26]. Some advantages of Unity engine in comparison with other game engines are presented in Table 2.1.

Table 2.1. Advantages of Unity in comparison with other game engines [24].

Scenario of Game Design	Unity game engine solution	Comparative analysis with other game engines
Develop a 3D environment that reconciles game performance with visual quality	Maximum real-world visual quality: shadows, light, texture map, alpha channel, independent animation time	Less graphic quality structures
Develop a navigation system for maximum user freedom	Enable maximum freedom to navigate and explore the virtual environment	High control but limited freedom in movement and predetermined animation sequences
Develop a navigation system that allows end-users to scrutinize a particular object for multiple perspectives	Method to augment spatial understanding	High degree navigation system, view direction
Develop a method to integrate several information types	Rich content and other data leaping technique	Require server-based interaction and script language
Develop a method for flexible 3D data exchange	Method for 3D data identical to the external tool	Less supple to accomplish data synchronization
Develop a translate conventional analysis study method	The basic visual-based analysis method is present	Less flexible
Flexibility for the expansion of game design	Possible due to object-oriented programming	Not possible because of structural programming

2.3.2. Unreal Engine

Unreal Engine, developed by Epic Games, is a comprehensive game development engine containing a robust suite of tools, libraries, and frameworks and is renowned for its advanced real-time rendering capabilities, making it the preferred choice for realistic visualization games. Its feature set includes a powerful editor, Blueprint visual scripting, asset management, physics simulation, and extensive support for various platforms [27-30]. Moreover, Unreal Engine's real-time rendering capabilities allow it to achieve high levels of visual fidelity and realism and, its advanced rendering pipeline, coupled with the physically based rendering (PBR) system, allows for the creation of visually stunning environments, realistic lighting, and lifelike character models. Additionally, accurately simulating light interactions and shadows in real-time are obtained with the engine's dynamic lighting and global illumination systems [27-29].

The physics simulation capabilities of this game engine contribute to the immersion and realism of games, by having the NVIDIA PhysX engine, which enables accurate and dynamic simulations of objects, collisions, and environmental interactions enhancing gameplay mechanics, allowing for more engaging and dynamic experiences [31]. With Unreal Engine's Blueprint visual scripting system, developers can employ a visual programming interface to create gameplay mechanics, AI behaviour, and interactive systems without writing traditional code, enhancing accessibility and empowering the prototypes and the implementation of ideas more efficiently, reducing reliance on dedicated programmers for certain tasks [31].

Furthermore, this game engine provides a comprehensive networking framework that supports seamless multiplayer experiences and offers a high-level networking architecture, a built-in replication system, and support for dedicated servers, making it easier to implement multiplayer features and to ensure smooth gameplay interactions [32].

2.3.3. CryEngine

Crytek designed the gaming engine known as CryEngine. It originated with the development of Far Cry, a first-person shooter game, and its subsequent instalments, utilizing the C++ and Lua programming languages for scripting. The engine allows game development for various platforms like Xbox One, PlayStation 4, Windows, Linux, PSVR, and Oculus Rift. It manages real-time asset conversion and optimization, enabling developers to make cross-platform modifications efficiently throughout the game development process. This enhances both the speed and quality of development while reducing the risks associated with multiplatform games. CryEngine grants access to source codes and the gaming engine itself, providing more flexibility and customization options. CryEngine is renowned for its exceptional graphics and game performance [33].

CryEngine games primarily revolve around the first-person shooter genre. It boasts numerous features that contribute to immersive and realistic game and virtual environment creation, such as a real-time editor, bump mapping, dynamic lights, a network system, an integrated physics system, shaders, shadow support, and a dynamic music system. The engine is equipped with all necessary development tools, including the CryEngine Sandbox world editing system. It supports all available hardware and receives updates for further hardware compatibility. Licensed developers gain access to the full source code and documentation for the engine and tools [34].

Using the CryEngine offers several advantages, notably its ability to produce high-quality graphics and visuals. The engine comes with necessary development tools that can be accessed from games employing it. The Sandbox editor is particularly user-friendly as it allows real-time level editing, providing immediate feedback on design changes. Additionally, a freely downloadable SDK includes partial source code and documentation [34].

2.4. Communication Protocols

Communication protocols are essential components of modern networking systems that enable devices and systems to exchange information in a structured and efficient way. A communication protocol can be seen as a set of rules and conventions that control how data is transmitted, received, and interpreted between different entities within a network. These protocols are specifically designed to meet the unique requirements and challenges faced by industrial applications, such as real-time control, deterministic behaviour, scalability, and interoperability [35].

Several widely used protocols in the fields of industrial automation and the Internet of Things (IoT) include OPC UA, Modbus, MQTT, DDS and ROS [36].

In this subchapter, Modbus and OPC UA, will be explored, examining their key features and advantages.

2.4.1. Modbus

The Modbus transmission protocol was specifically designed for controlling processes in various systems and has become the prevailing standard for serial communication in the industry. It facilitates communication between devices connected to the same network, typically operating in a client/server configuration. Modbus messages are categorized into two types, query/response, and broadcast/no response. In both cases, the client is responsible for initiating the communication, and report-by-exception (RBE) is not supported except in the case of Modbus TCP [37].

Modbus is versatile and can be employed by different types of devices, such as PLCs, HMIs, control panels, drivers, and I/O devices, enabling them to execute remote operations [37, 38].

The protocol defines a set of standard function codes that facilitate diverse types of operations, such as reading and writing data registers, coils, input registers, and discrete inputs and supports both analogue and digital data types, making it versatile for a wide range of applications [39, 40].

To implement the Modbus protocol, various transmission protocols are available, such as Asynchronous Serial Transmission and TCP/IP. Asynchronous Serial Transmission is used for serial connections, such as those over wire RS-232, RS-422, RS-485, fiber optics, or radio links. Within this mode, there are two distinct transmission modes, the Modbus RTU and the Modbus ASCII. Modbus RTU utilizes a compact, binary representation of data, leading to faster communication and its primary use during regular operations. It is often represented in hexadecimal format. Modbus ASCII, in contrast, uses a human-readable format, making it more verbose. It is commonly used for testing and debugging purposes. TCP/IP enables the use of Modbus over Ethernet networks, extending its capabilities and facilitating communication between devices connected through TCP/IP [37].

The figure 2.6 shows the Modbus mapping onto the OSI model.

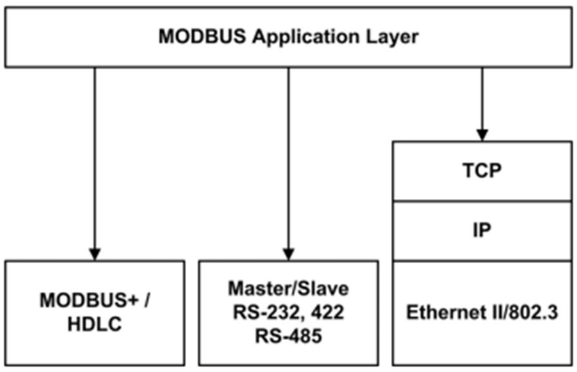


Figure 2.6. Modbus Model [37].

Modbus provides support for four primary data types. Firstly, there are 1-bit inputs, which are read-only and can be accessed from an I/O system. Secondly, we have 2-byte input registers, also read-only, and accessible through an I/O system. On the other hand, we have 1-bit discrete outputs, commonly known as coils, which are used for output operations. These coils are read/write enabled, meaning they can be both accessed and modified by an application program. Finally, there are 2-byte holding registers, also read/write enabled, allowing an application program to both read from and write to them [37].

Modbus protocol defines a simple PDU independent of the underlying communication layers. A typical Modbus Application Data Unit (ADU) is shown in Table 2.2.

Table 2.2. Typical Modbus Application Data Unit [37].

	Protocol Data Unit (PDU)		
Application Data Unit (ADU)			
1 Byte	1 Byte	Variable	2 Bytes
Address Field	Function Field	Data Field	Error Checking Fields

2.4.2. Modbus TCP

Modbus TCP is a communication protocol, an extension of the Modbus protocol, widely used in industrial automation and control systems to establish communication between devices such as programmable logic controllers (PLCs), supervisory control and data acquisition (SCADA) systems, and various other devices [38].

Modbus TCP operates over TCP/IP networks, allowing faster and more reliable communication over long distances. It is based on a client/server architecture, where a client initiates a request to a server device, and the server responds with the requested data or performs the desired action and uses the Transmission Control Protocol (TCP) as the transport layer, providing features such as error checking, data packets acknowledgment, and reliable data transmission. It also utilizes the Internet Protocol (IP) for addressing and routing packets within an Ethernet network [39, 40].

2.4.3. OPC UA

OPC UA, which stands for Open Platform Communications Unified Architecture, is a communication protocol that facilitates machine-to-machine interaction, primarily employed in industrial automation. It is meticulously defined in the IEC 62541 specification, and its key objective is to enable seamless communication across different platforms while utilizing an information model to describe the data being exchanged [36].

This protocol is widely adopted in the European manufacturing industry, driving its significance in the realm of industrial automation and gaining global recognition as one of the leading communication protocols [36].

One of the notable developments in OPC UA is the introduction of the Publish/Subscribe specification, where servers can publish data, and clients can subscribe to this data without being concerned about the data's source. An essential aspect to note is that OPC UA Publish/Subscribe does not inherently incorporate quality of service (QoS) mechanisms. However, when combined with technologies like Time Sensitive Networking (TSN) on layer 2, or other protocols such as MQTT, it becomes feasible to incorporate additional QoS principles [36].

The OPC Foundation, responsible for shaping and maintaining the OPC UA standard, has released multiple specification parts that are accessible to the public. This approach fosters transparency and allows the community to understand the various features and components of OPC UA comprehensively. As a result, OPC UA continues to evolve and adapt, providing a robust and versatile framework for machine-to-machine communication in the industrial landscape. Its semantic-rich architecture and compatibility with different domains make it an ideal choice for a wide range of applications in the industrial sector, propelling the progress of automation and smart manufacturing processes [36].

3. Requirements and Modelling

An in-depth understanding of the shop floor will be undertaken, exploring the functioning of all its components. To achieve this, an examination of the existing shop floor as well as the shop floor simulator will be conducted, given its alignment with the requirements of the automation laboratories at FEUP, represented in Figure 3.1.

Subsequently, the reason for Unity to be chosen to develop the digital twin is presented. Also, an analysis of the components will be performed to ensure the new elements meet the same requirements as those in the current shop floor setup. This meticulous process will lead to the creation of an efficient and effective shop floor simulator for research and experimentation purposes.

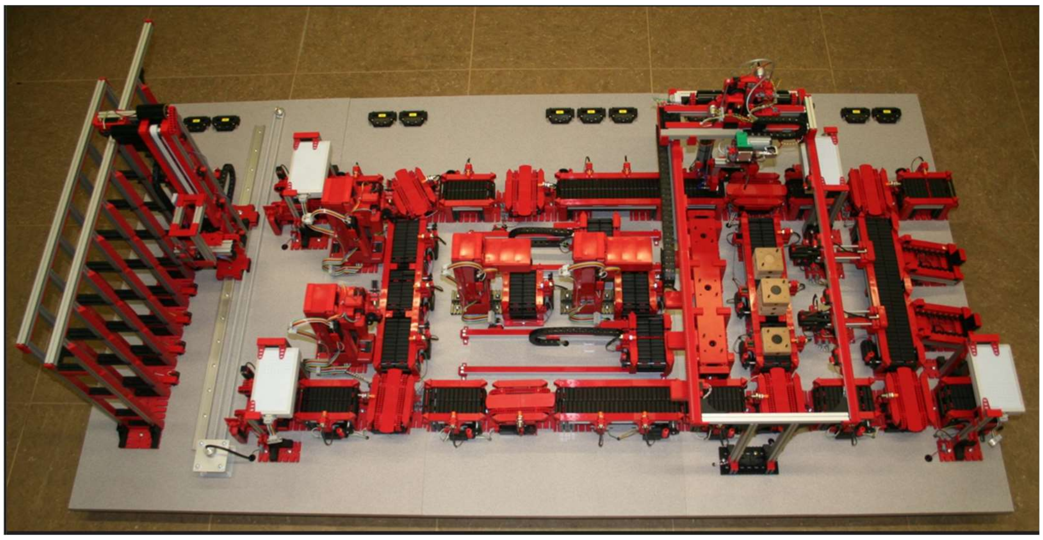


Figure 3.1. Plant floor in the automation laboratories of FEUP.

3.1. Shop Floor

The shop floor has a well-defined fixed layout, comprising a diverse range of equipment. In the subsequent sections, a comprehensive and detailed description of the operation of each piece of equipment will be provided. Furthermore, explicit mention will be made of the actuators and sensors associated with each equipment, offering a comprehensive understanding of their functionalities and interactions.

3.1.1 Simple conveyor

The simple conveyor possesses the capability to transport a piece in two opposing directions, depending upon the conveyor's positioning. Regardless of the direction, the movement consistently occurs at a uniform speed, facilitating the control mechanism for the conveyor's motor, which can be achieved using separate binary signals for each direction.

The conveyors are equipped with one or more sensors that facilitate the detection of a piece on the conveyor, with the number of sensors varying depending on the conveyor's size. Each of these sensors corresponds to a binary output, providing a clear and concise indication of whether a piece is present or absent on the conveyor's surface.

The Table 3.1 has the simple conveyor’s input and output signal.

Table 3.1. Simple conveyor’s input and output signal.

Type	Acronym	Name
Binary Actuator	mp	Positive direction movement
Binary Actuator	mm	Negative direction movement
Binary Sensor 1	p1	Presence of piece
Binary Sensor 2	p2	Presence of piece
Binary Sensor 3	p3	Presence of piece

3.1.2 Rotative conveyor

The rotative conveyor belt functions much like a simple conveyor belt but comes with the added advantage of controllable rotation. This means that besides its linear movement, it can also be precisely rotated to a desired angle.

To facilitate this extra rotational control, the conveyor belt incorporates two additional binary signals, one for each direction of rotation. These signals allow for smooth and controlled rotations within a range of 90 degrees. The conveyor belt is equipped with two limit switches that indicate the endpoints of this 90-degree rotation.

The Table 3.2 has the rotative conveyor’s input and output signals.

Table 3.2. Rotative conveyor's input and output signals.

Type	Acronym	Name
Binary Actuator	mp	Positive direction movement
Binary Actuator	mm	Negative direction movement
Binary Actuator	rd	Direct rotation
Binary Actuator	ri	Reverse rotation
Binary Sensor 1	p	Presence of piece
Binary Sensor 2	d	End of direct rotation
Binary Sensor 3	i	End of reverse rotation

3.1.3 Sliding conveyor

The sliding conveyor belt functions in a manner quite similar to the rotative conveyor belt, with the only distinction being the mode of movement. Instead of rotational motion, the sliding conveyor belt employs linear translation for its operations. As a result, the actuators and sensors used in the sliding conveyor belt are identical to those found in the rotative conveyor belt, although with distinct names, as indicated in the Table 3.3.

Table 3.3. Sliding conveyor's input and output signals.

Type	Acronym	Name
Binary Actuator	mp	Positive direction movement
Binary Actuator	mm	Negative direction movement
Binary Actuator	tp	Positive direction translation
Binary Actuator	tm	Negative direction translation
Binary Sensor 1	p	Presence of piece
Binary Sensor 2	fp	End of positive direction translation
Binary Sensor 3	fm	End of negative direction translation

3.1.4. Workbench

The worktable has a specific purpose of providing a temporary location for positioning pieces within the gantry-type robot's reach. It permits the robot to deposit pieces onto the table, facilitating the stacking of these pieces on top of each other.

Regarding its control mechanism, the worktable operates as an immobile conveyor belt, essentially static and devoid of any movement. It relies on a single presence sensor to identify the presence of a piece on its surface. Unlike traditional conveyor belts, the worktable does not possess any actuators for mobility since it remains fixed in one position throughout its use.

The Table 3.4 has the workbench's input and output signals.

Table 3.4. Workbench's input and output signals.

Type	Acronym	Name
Binary Sensor	p	Presence of piece

3.1.5. Warehouse

In order to simplify warehouse operations, the interaction with the warehouse is streamlined to include two primary actions, requesting the storage of a piece or the retrieval of a specific type of piece.

The warehouse cell's physical interface with other cells is facilitated through two distinct conveyor belts, each serving a unique purpose. One type of conveyor belt enables the retrieval of pieces from the warehouse, while the other type facilitates the storage of pieces into the warehouse.

Control of each conveyor belt aligns with the familiar setup of a simple conveyor belt, utilizing two digital outputs for the movement actuators and one digital input to detect the presence of a piece on the conveyor belt.

To store a piece within the warehouse, it must first be positioned in the middle position on the appropriate conveyor belt. The request for storage is initiated by a designated digital signal transitioning from 0 to 1.

On the other hand, when it comes to removing a piece from the warehouse, the conveyor belt should be initially free. Subsequently, the desired piece type is indicated using a numerical value. Notably, for the removal request to be acknowledged, the corresponding register should have an initial value of 0.

The Tables 3.5 and 3.6 have the input and output signals for the warehouse's entrance and exit respectively.

Table 3.5. Warehouse's entrance input and output signals.

Type	Acronym	Name
Binary Actuator	mp	Positive direction movement
Binary Actuator	mm	Negative direction movement
Binary Actuator	in	Insert piece into the warehouse
Binary Sensor	p	Presence of a piece

Table 3.6. Warehouse's exit input and output signals.

Type	Acronym	Name
Binary Actuator	mp	Positive direction movement
Binary Actuator	mm	Negative direction movement
Word Actuator	tp	Type of piece to remove
Binary Sensor	p	Presence of a piece

3.1.6. Machine tool

The machine tool efficiently performs operations on workpieces positioned on the attached conveyor belt, which forms an integral part of the machine tool and is controlled by actuators similar to those used in simple conveyor belts.

There exist two variants of the machine tool, one equipped with a single tool and another with three tools. In the latter case, these three tools are housed on a turret. When switching between tools on the turret, a simple and consistent process is followed, ordering the rotation of the turret in a single direction until the desired tool is correctly aligned for machining. The activation of a sensor confirms the presence of any tool in the machining position.

Upon startup, the machine assumes Tool T1 to be in the machining position. The tools are strategically mounted in the following order: T1, T2, T3. The turret, housing the tools, possesses independent movement capabilities along the ZZ and YY axis. Each axis is controlled by two binary actuators and is accompanied by two sensors that signal the arrival at the extreme positions.

For machines equipped with a single tool, the control is limited to maintaining a fixed speed, and the turret's movement is confined solely to the ZZ axis. The indication of the fixed-speed tool operating in the machining position is communicated through a straightforward binary actuator.

The Table 3.7 has the machine tool’s input and output signals.

Table 3.7. Machine tool’s input and output signals.

Type	Acronym	Name
Binary Actuator	mp	Positive direction movement
Binary Actuator	mm	Negative direction movement
Binary Actuator	tc	Tool change
Binary Actuator	--	Tool change
Binary Actuator	tr	Tool rotate
Binary Actuator	vp	Positive tower movement along Y-axis
Binary Actuator	vm	Negative tower movement along Y-axis
Binary Actuator	zp	Positive tower movement along Z-axis
Binary Actuator	zm	Negative tower movement along Z-axis
Binary Sensor	p	Presence of piece
Binary Sensor	pt	Presence of tool
Binary Sensor	vp	End of positive Y-axis movement
Binary Sensor	vm	End of negative Y-axis movement
Binary Sensor	zp	End of positive Z-axis movement
Binary Sensor	zm	End of negative Z-axis movement

3.1.7. Pusher

A pusher serves as a valuable device utilized for effectively displacing pieces on a conveyor belt, primarily used in sorting and selecting operations. The pusher and the conveyor belt are inherently interconnected, with the latter being controlled by actuators similar to those used for simple conveyor belts.

The pusher operates through two actuators, enabling movements in both right and left directions. The boundary of each movement is precisely determined by two limit switch sensors. To ensure the preservation of equipment integrity, strict measures are in place to prevent the pusher from surpassing these limits during its movements.

The Table 3.8 has the pusher's input and output signals.

Table 3.8. Pusher's input and output signals.

Type	Acronym	Name
Binary Actuator	mp	Positive direction movement
Binary Actuator	mm	Negative direction movement
Binary Actuator	pr	Retract the pusher
Binary Actuator	pe	Extend the pusher
Binary Sensor	p	Presence of piece
Binary Sensor	fr	End of pusher retraction
Binary Sensor	fe	End of pusher extension

3.1.8. Robot 3D

The 3D robot is equipped to perform movements along three axes, with two movement actuators for each axis, enabling motion in both directions. The ZZ axis is monitored by two sensors indicating its extreme positions, namely up and down. Similarly, the XX axis employs two sensors to identify its extreme positions, ensuring correct alignment with the conveyors or tables beneath the robot. On the other hand, the YY axis utilizes five sensors to establish alignment with any of the five conveyors or tables along this axis.

To control the claw, a single binary actuator is utilized. However, due to mechanical constraints, after the claw's closing actuator is engaged, it should pause for approximately one second to ensure a secure grip on the object. A sensor inside the claw detects the presence or absence of a piece.

To prevent collisions, all movements of the claw along the XX and YY axes are restricted to occur only when the claw is positioned in the uppermost position along the ZZ axis. Additionally, movements along the XX, YY, and ZZ axes must never exceed the prescribed limits of the robot's operating area.

At the system's startup, the robot's initial position is unknown. Hence, an initialization procedure is required to place the robot in a known position, establishing a reliable starting point for its operations.

The Table 3.9 has the Robot 3D‘s input and output signals.

Table 3.9. Robot 3D‘s input and output signals.

Type	Acronym	Name
Binary Actuator	xp	Positive direction movement along XX axis
Binary Actuator	xm	Negative direction movement along XX axis
Binary Actuator	yp	Positive direction movement along YY axis
Binary Actuator	ym	Negative direction movement along YY axis
Binary Actuator	zp	Positive direction movement along ZZ axis
Binary Actuator	zm	Negative direction movement along ZZ axis
Binary Actuator	g	Claw actuator
Binary Sensor	xp	End of movement in positive direction along XX axis
Binary Sensor	xm	End of movement in negative direction along XX axis
Binary Sensor	zp	End of movement in positive direction along ZZ axis
Binary Sensor	zm	End of movement in negative direction along ZZ axis
Binary Sensor	y1	Position 1 of YY axis

Table 3.9. *Cont.*

Type	Acronym	Name
Binary Sensor	y2	Position 2 of YY axis
Binary Sensor	y3	Position 3 of YY axis
Binary Sensor	y4	Position 4 of YY axis
Binary Sensor	y5	Position 5 of YY axis
Binary Sensor	p	Presence of piece

3.1.9. Interlock

The shop floor is equipped with an interlock system capable of filtering out student actuators that could potentially cause harm or damage to the physical equipment. This safety feature prevents the execution of certain types of actuators known to be hazardous. For instance, it blocks any attempts to perform simultaneous movements in both directions on any conveyor or trying to rotate the rotative conveyor in both directions simultaneously. Additionally, if a piece becomes jammed between two conveyors, being on of them a rotative conveyor, the system prevents any attempts to rotate it. It also intervenes when there are attempts to force pieces against each other, as such actions could lead to damage. Furthermore, actuators that could result in displacements exceeding their designated limits, such as those in the warehouse or robot3d, are also restricted by the interlock mechanism.

3.2. SFS - Flexible Production Line

FEUP's current simulator is a 2D representation of the shop floor that contains the various objects like conveyors, warehouses, and machines. The simulator, unlike its real counterpart, can be customized, meaning it can have different layouts. The Figure 3.2 shows an example of those layouts.

For this layout, the warehouses are depicted as large vertical rectangles with brown squares. All conveyors are bi-directional, and the yellow dots represent sensors capable of detecting pieces' presence. Additionally, some conveyors, indicated by a blue tinge, can rotate about their vertical axis, allowing pieces to move in both horizontal and vertical directions. While conveyors with an orange arrow can insert or remove pieces from the respective warehouses.

Furthermore, the simulator has two types of machines, namely M1 and M2, with different tool sets. M1 is equipped with tools T1, T3, and T4, while M2 uses tools T2, T3, and T4.

To simulate this production line, the Shop Floor Simulator (SFS) software is employed. The SFS is developed in Java and is compatible with Windows, Linux, and OSX. The logic signals, including sensors and actuators, are accessed using the Modbus/TCP protocol. Specifically, the SFS simulator acts as a Modbus/TCP server, mapping sensors to Input Discrete and actuators to Coils. The addresses for each signal are available in a csv file.

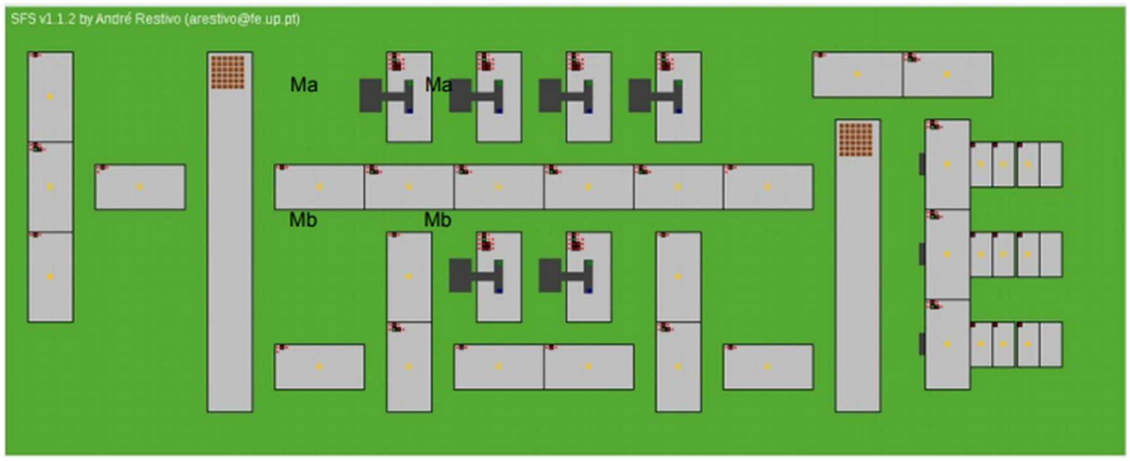


Figure 3.2. Shop Floor Simulator.

3.3. Requirements analysis

Considering both the physical shop floor and the shop floor simulation, several essential requirements arise for this project. Each of these objects serves distinct purposes and possesses specific requirements.

The conveyor's primary function is to efficiently transport pieces from one location to another. Its key requirement is to ensure seamless dislocation of the pieces along the intended path.

The rotator, on the other hand, plays a pivotal role in altering the direction of the pieces. Hence, it is imperative that the rotator can smoothly and precisely rotate, in order for the items to change to the direction needed.

For the machines, their primary objective is to work on the pieces by modifying their colour. As a result, the system must provide the necessary tools and functionalities to enable the machines to execute this colour-changing process flawlessly.

Finally, the warehouses function as storage units for the pieces. Ensuring that the system can effectively manage and handle the storage of pieces within the warehouses.

Additionally, the simulation should be configurable using file formats like txt or JSON. These files will serve as a means to customize the properties and behaviour of objects in the simulation. Consequently, the project should also include functionalities to handle file communication effectively, enabling seamless interaction with these configuration files. By doing so, different and flexible layouts of the shop floor can be created, meeting specific needs and scenarios.

Lastly, it is crucial for the project to offer support for the Modbus protocol to facilitate smooth interactions with external industrial devices.

3.4. Unity as a digital twin simulation foundation

The reason behind selecting Unity as the preferred platform for creating a digital twin capable of simulating the plant floor when comparing it with other popular options such as Unreal Engine and CryEngine are represented on Table 3.10.

Table 3.10. Comparation of Game Engines[33].

Parameters	Unity	Unreal	CryEngine
Cross Platform	Consoles (Xbox, PlayStation, Wii U, Nintendo), OS or Desktop (macOS, Windows and Linux), Mobile devices (Android, Windows, iOS, Blackberry), WebGL	Consoles (Xbox, PlayStation, Switch), OS (Windows, macOS and Linux), Mobile devices (Android, iOS), HTML5	Consoles (Xbox, PlayStation, Oculus Rift), OS (Windows Linux), Mobile devices (not supported)
OS Support	Windows, Linux, Mac	Windows, Mac	Windows, Linux

Table 3.10. *Cont.*

Parameters	Unity	Unreal	CryEngine
Programming Languages	C#, JavaScript, Boo	C++	C++,Lua
Multifunctionality	2D,3D	2D,3D	3D
Documentation	Best	Good	Poor
Difficulty Level (for Beginner)	Low	High	High
Artificial Intelligence	RAIN	Kynapse	Lua Driven AI
Physics Engine	PhvsX	PhvsX	Soft-Bodv
Network/ Multiplayer	Supported	Supported	Supported
Network/ Multiplayer	Supported	Supported	Supported
Development Tools	Visual Studio, MonoDevelop	BluePrint Editor, Visual Studio	FlowGraph, Visual Studio
Terrain Design using Engine Tools	Medium	High	Medium (Good in-built assets)
Graphic Effects	Shadow effects, Particle system, Different types of lighting, Lens Flare	Shadow effects, Particle system, Different types of lighting, Lens Flare	Shadow effects, Particle system, Different types of lighting, Lens Flare
Libraries & Plugins	Maximum	Less than Unity	Least
Technical and Community Support	Vast, active and supportive community	Between Unity and CryEngine	Relatively small community

Table 3.10. *Cont.*

Parameters	Unity	Unreal	CryEngine
Animations	Basic animation	SkeletalControl and supports dynamic animation (better than CryEngine)	Supports SkeletalControl and Facial Editor
Modeling	Supports external modeling assets created in (Blender, 3ds Max, Maya, etc.) and it has built in asset store	Static Mesh Editor(not better when compared to CryEngine)	Provides Designer Tool, CryEngine's 3D object modelling tool
Pricing	Personal – Free Plus - $399/yr per seat Pro - $1800/yr per seat Enterprise - $2000/mo per 10 seats	Personal use – Free Personal Version – if revenue from the game is more than $3,000 /quarter than 5% of the game's gross income is charged or else free	CryEngine is available for free including the full engine code. There is a 5% royalty fee levied after the first $5000 of revenue earnt
VR/AR	Supports Oculus Rift, HTC Vive, Google DayDream, Cardboard, Gear VR, Steam VR	Supports Oculus Rift, HTC Vive, Steam VR, OSVR Google VR/DayDream, Samsung Gear VR	Supports HTC Vive, Oculus Rift

Considering Table 3.10, Unity stands out as the optimal choice for several reasons. Firstly, its cross-platform compatibility with macOS, Windows, and Linux makes it incredibly versatile.

Moreover, Unity's exceptional documentation, extensive libraries, and robust community support make it the leader among game engines in these aspects. The wealth of resources available allows for swift and efficient troubleshooting, enabling to overcome any challenges or setbacks that may arise during development.

One of Unity's most significant advantages is its beginner-friendly nature, with the lowest difficulty level compared to other engines. This ease of use facilitates a smooth learning curve for newcomers, granting them a better understanding of the engine and game development process.

So, considering all these factors and ensuring compatibility with the requirements outlined in the previous sub-chapter, it becomes evident that Unity stands out as the optimal choice for simulating a shop floor.

3.4.1. Unity GameObjects

To develop the components for the Shop Floor Simulator, understanding Unity's GameObjects is crucial.

GameObjects are the basic building blocks of any project in Unity and represent the entities, objects, or elements that make up a game world. To define the specific functionality and behaviour of each GameObject, they can have more than one component attached to it. Unity provides a wide range of built-in components, being the main ones the Transform which allows to define and change the position, rotation, and scale of the GameObject, the Renderer which gives the GameObject a visual appearance, the Rigidbody that allows the GameObject to suffer for physics simulation, and the Collider that allows for physical collisions between GameObjects. These components can be added, modified, or removed to customize the behaviour and appearance of GameObjects. Scripts are also components that can be added to a GameObject.

Scripts can be attached to GameObjects as components, providing them with specific functionality like modifying already existing components, creating new ones, interacting with other scripts, and responding to events or input from the player. In Unity, scripts are typically written in an external code editor using the C# language and then attached to GameObjects using the Inspector window. Moreover, when scripts are attached to any GameObject in Unity, they remain active throughout the GameObject's existence. These scripts typically have two primary functions by default, the Start() and the Update() functions. The Start() function is executed only once each time the class is called or the GameObject is initialized. On the other hand, the Update() function is executed once every frame, providing continuous updates and allowing for real-time interactions and changes in the game or application.

4. Implementation

To address the shortcomings of the old simulator, a new 3D simulator was developed that offers a more realistic representation of the shop floor. The new simulator was built using the old simulator as a reference, allowing us to incorporate the knowledge gained from it into the development of the new one.

4.1. Architecture

The architecture of the new simulator was carefully designed to enable efficient communication between different classes and seamless interactions between various elements of the shop floor simulation.

At the heart of the architecture lies the “Shopfloor” class. This class serves as a central connection point that facilitates communication between different components of the simulator. Its main responsibilities include reading a JSON file that contains crucial information about the shop floor layout, such as specifications for different objects, types of pieces, and actions associated with various machines. By parsing this JSON data, the Shopfloor class can set up the initial state of the simulation.

Furthermore, the Shopfloor class acts as a server for the Modbus communication protocol, where Modbus is implemented to facilitate real-time data exchange between the simulation and the user's program. The Shopfloor class continuously updates the sensors’ values on the shop floor and sends this information to the user's program using the Modbus protocol. Simultaneously, it receives actuator values from the user's program, enabling real-time control of the simulation.

The user's program, acting as a Modbus client, connects to the Shopfloor class and interacts with the simulation. By sending actuator values, the user can manipulate the behaviour of various elements in the shop floor environment and observe the effects in real time through the simulation's graphical interface.

Additionally, the Shopfloor class establishes connections with various classes responsible for handling different objects within the simulation, which include the Simple Conveyor, Rotator Conveyor, Machine Conveyor. Each of these classes manages the behaviour and functionality of its corresponding object.

For instance, the Simple Conveyor class manages the movement of pieces along the conveyor, updates sensor data for detecting piece positions, and communicates with the user's program through the Shopfloor class to reflect changes in the conveyor's behaviour.

Similarly, the Rotator Conveyor class and Machine Conveyor class handle specific functionalities unique to their respective objects. They also interact with the user's program via the Shopfloor class, providing the user with comprehensive control over the simulation.

The architecture promotes modularity and flexibility, allowing for easy expansion and incorporation of new elements into the simulation. By establishing well-defined communication channels and encapsulating functionalities within specific classes, the new simulator achieves a robust and scalable design.

The Figure 4.1 shows the architecture implemented for this project.

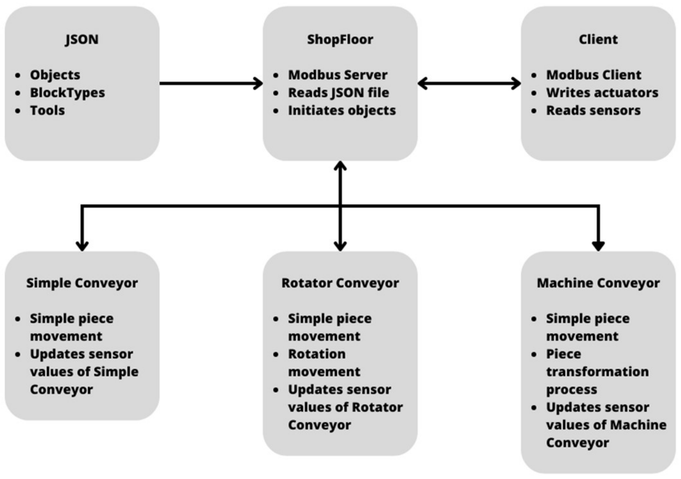


Figure 4.1. Architecture of the implementation.

4.2. Materials

Each material used in the simulator served a specific purpose in distinguishing various elements within the shop floor environment. So, the Black material was applied to the conveyors, giving them the appearance of a typical conveyor belt, the red material was utilized for the side barriers, indicating restricted areas for the movement of pieces and the yellow material was employed for the sensors located in the middle of the conveyors, facilitating the detection of pieces during their transit.

The Grey material provided a neutral colour for the floor, contributing to the overall aesthetics. In contrast, the GreyMet material, sharing the same grey colour, added a metallic sheen to the machines, making them appear more robust and industrial in nature.

Additionally, the implementation of a physical material with drag components added a layer of realism to the simulation. This allowed for more accurate modelling of object interactions, such as pieces being affected by friction as they moved across the conveyor systems.

4.3. GameObjects

To enhance the visual fidelity and realism of the simulation, careful consideration was given to the design of GameObjects.

4.3.1. Floor

The GameObject Floor is used for the Shopfloor class and serves as the base layer upon which all other elements are positioned and interact, providing the necessary backdrop for the entire shop floor simulation.

The Figure 4.2 shows the scene view of the GameObject Floor.

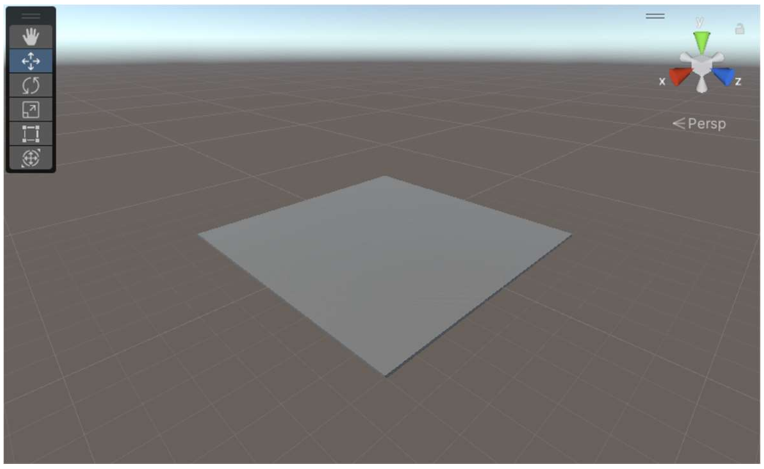


Figure 4.2. GameObject Floor.

Since the Shopfloor class is responsible for the communication with the JSON file, the parameters of configuration that come from it for each GameObject are presented on this script. The Table 4.1 shows those parameters.

Table 4.1. JSON parameters.

Parameter	Simple Conveyor	Rotate Conveyor	Machine Conveyor
Id	X	X	X
Type	“conveyor”	“rotator”	“machine”
DirectionX	X	X	X
DirectionZ	X	X	X
PositionX	X	X	X
PositionZ	X	X	X
OffsetSen	X	X	X
OffsetAct	X	X	X
Speed	X	X	X

The table 4.1 clearly illustrates that all GameObjects share the same parameter configuration, except for the "Type" parameter, which determines the specific object to be created. The "Id" parameter is used for unique identification of each object. Moreover, the "directionX" and "directionZ" parameters control the orientation of the GameObject, while the "positionX" and "positionZ" parameters define its coordinates.

Additionally, the "offsetSen" and "offsetAct" parameters dictate the positions in the Modbus memory where the GameObject's sensor and actuator values are stored, respectively. Lastly, the "speed" parameter influences the movement velocity of the pieces on the conveyors.

Utilizing the “Instantiate()” function with these specific parameters, the objects can be seamlessly spawned onto the floor, enabling the automatic creation of the shop floor the moment the game is initiated.

4.3.2. Simple Conveyor

Representing one of the core elements of the shop floor, the Simple Conveyor was designed as a 3D object with a cube shape, subsequently transformed to simulate a conveyor belt. The modification of component Transform values resulted in the conveyor's characteristic shape, facilitating the movement of pieces. To create a more realistic representation, the conveyor was complemented with side barriers and a central sensor to detect the presence of pieces as they travelled along the conveyor belt.

To the Simple Conveyor it was attached the “SimpleConveyorMove” script. This script is responsible for controlling the movement of objects on the conveyor belt and uses variables such as “direction”, a Vector3 variable representing the movement direction of objects, and “onBelt” , a List of GameObjects to store objects on the belt.

In the Update() function, the script evaluates the sensor state to detect objects on the conveyor belt and, based on the actuator values, the appropriate movement direction is determined. The script iterates through objects on the belt, updating their velocity for smooth movement. Instead of moving the conveyor itself, it manipulates the velocity of each individual piece, resulting in their seamless movement.

The script also manages collisions with other objects. When a collision occurs, the “OnCollisionEnter(Collision collision)” function adds the collided object to the “onBelt” list. On the other hand, the “OnCollisionExit(Collision collision)” function removes the object from the list when the collision ends.

The Figure 4.3 shows the scene view of the GameObject Simple Conveyor while the Figure 4.4 shows the laboratory of automation counterpart.

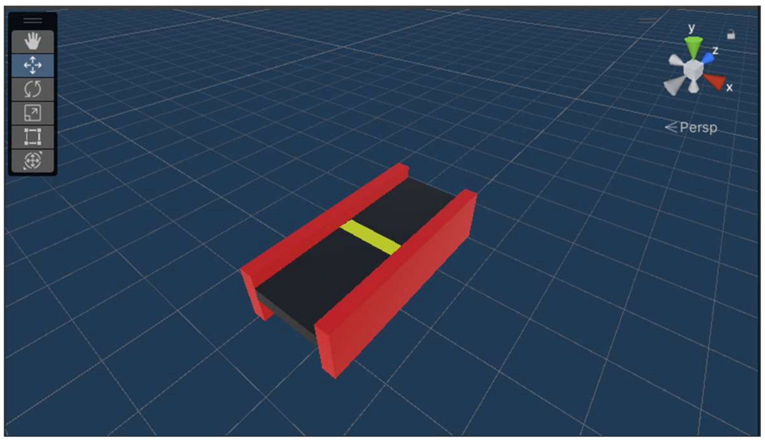


Figure 4.3. GameObject Simple Conveyor.

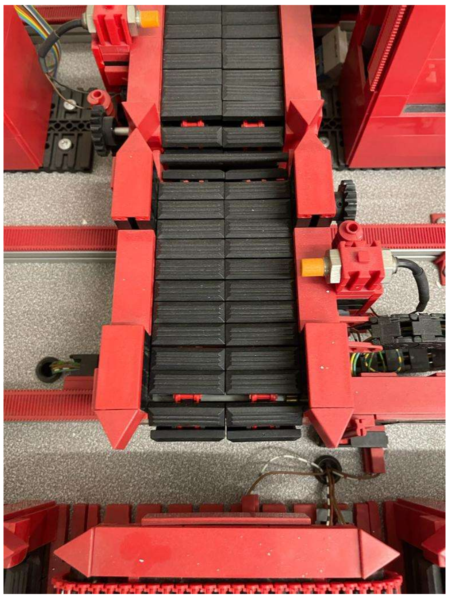


Figure 4.4. Simple Conveyor of Automation Laboratory

4.3.3. Rotator Conveyor

Building upon the foundation of the Simple Conveyor, the Rotator Conveyor shares similar design principles but includes additional sensors. These sensors detect changes in the conveyor's orientation, allowing the simulator to accurately model pieces being rotated during their movement.

To the Rotator Conveyor, the “RotatorConveyor” script was attached. This script controls the rotation of the conveyor belt and object movement using the same logic as the “SimpleConveyorMove” script for moving objects.

In the Update() function, the script updates the time variable based on the elapsed time since the previous frame and checks the status of two rotation sensors, updating their values, and determining the rotation direction based on the active rotation sensor. The script manages the conveyor belt's rotation based on the actuator states, and if the appropriate actuator is active and the time delay has passed, it rotates the conveyor belt in the desired direction using “eulerAngles” of the transform. Moreover, the script forces the piece to rotate in sync with the conveyor, creating a visually realistic behaviour.

The Figure 4.5 shows the scene view of the GameObject Rotator Conveyor while the Figure 4.6 shows the laboratory of automation counterpart.

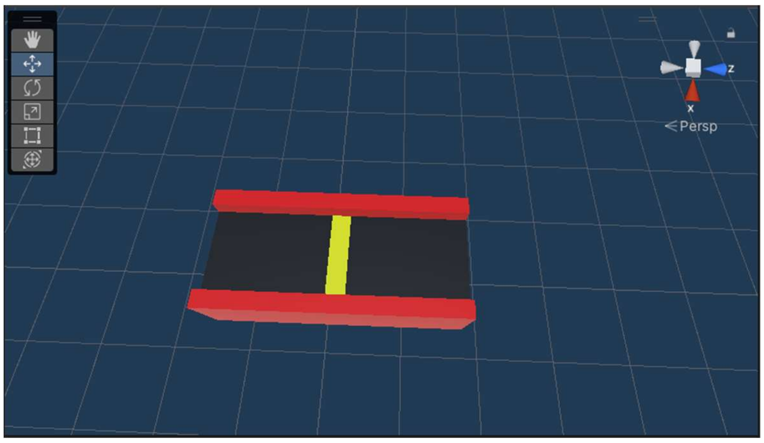


Figure 4.5. GameObject Rotator Conveyor.

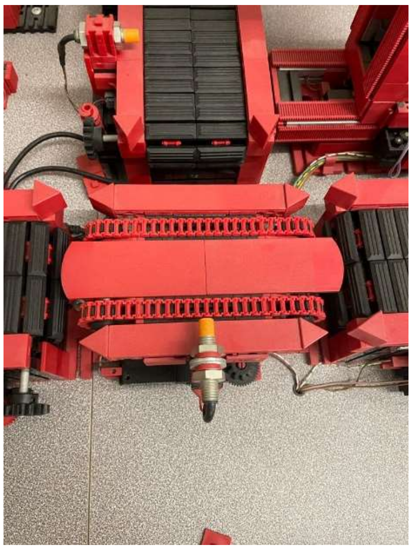


Figure 4.6. Rotator Conveyor of Automation Laboratory

4.3.4. Machine Conveyor

Extending the conveyor system, the Machine Conveyor featured elements indicative of a machine. Three supplementary 3D objects were included to visually represent the presence of a machine connected to the conveyor adding depth and detail to the simulation, allowing the simulator to accurately model pieces being transformed by the machines.

The “MachineMove” script was attached to the Machine Conveyor. This script controls the interactions of a machine and the movement of objects on the conveyor belt following the same logic as the previous scripts for the movement of objects.
The script begins by declaring variables related to colour, tool status, and timing, initializing the timing variable in the Start() function.

In the Update() function, the script handles tool activation and deactivation based on specific actuators and sensors. Additionally, triggers a tool selection process based on timing and tool type conditions. The tool selection process determines which values should be set to true, representing the usage of different tools.

If the sensor detects an object, the script performs loops to identify the initial and final states of the object on the belt. Based on the identified states and the current tool status, the script updates the object's colour, sets the tool work status, and resets the timing variable.

The Figure 4.7 shows the scene view of the GameObject Machine Conveyor while the Figure 4.8 shows the laboratory of automation counterpart.

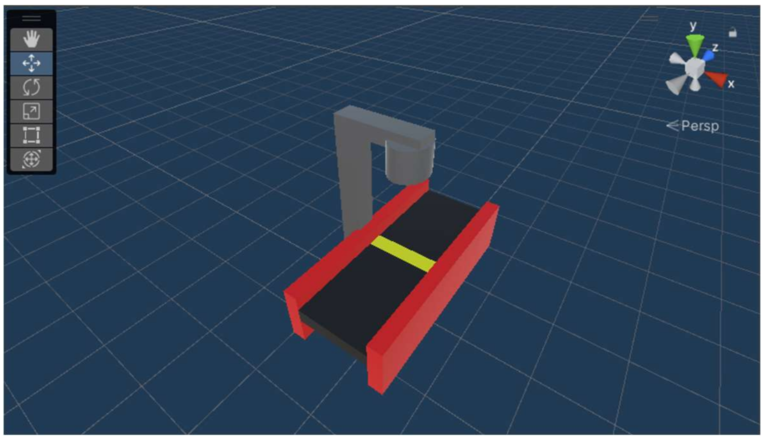


Figure 4.7. GameObject Machine Conveyor.

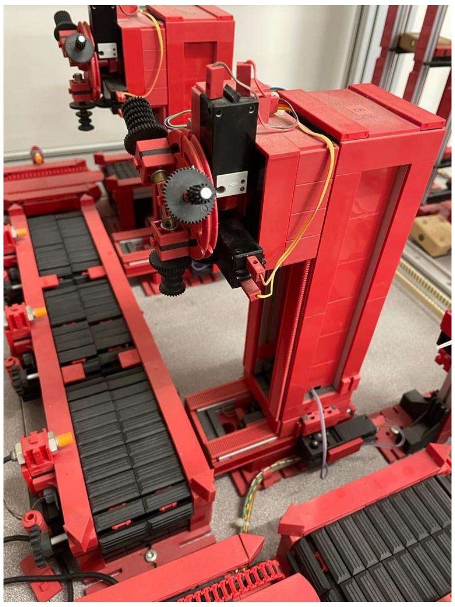


Figure 4.8. Machine Conveyor of Automation Laboratory

4.3.5. Piece

A fundamental element of the shop floor, the Piece was represented as a simple 3D cube object. To enable interactions with other GameObjects, a Rigidbody component was added, allowing the Piece to react to external forces and, for instance, remain atop the conveyors during transit.

The Figure 4.9 shows the scene view of the GameObject Piece.

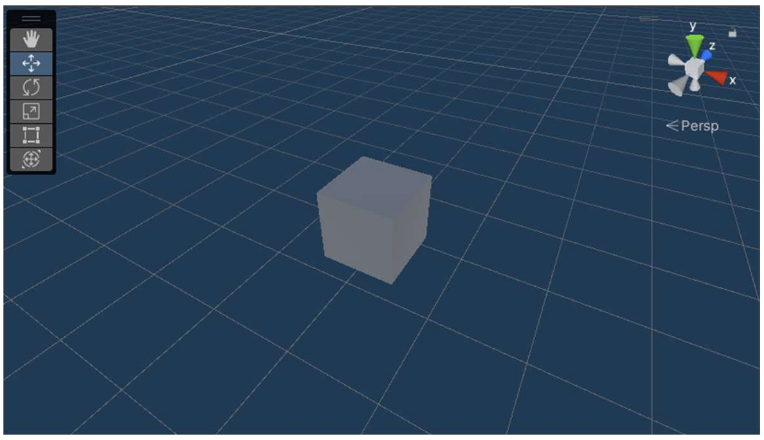


Figure 4.9. GameObject Piece.

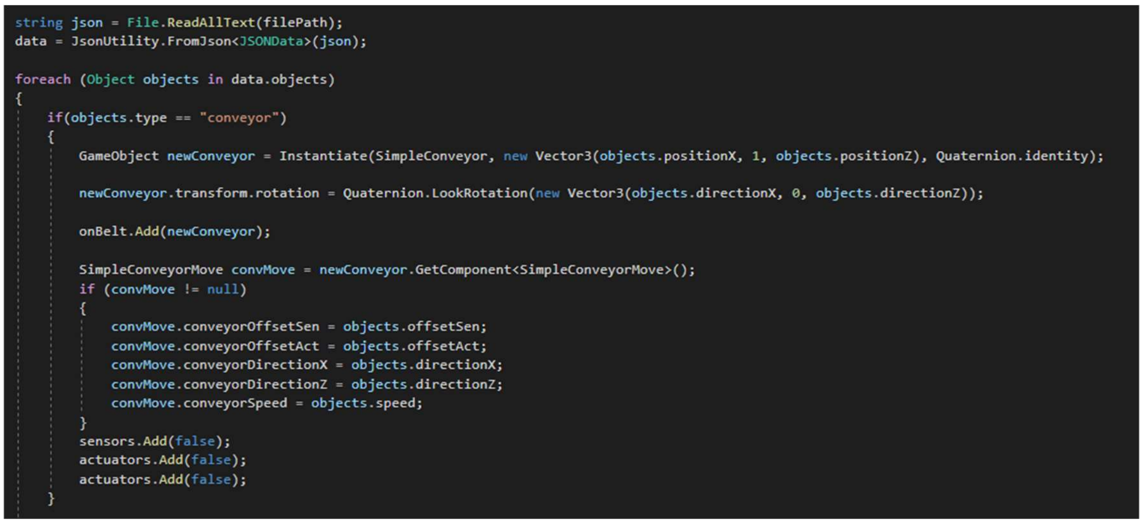
4.4. Deployment Code Sample

To implement the simulation with objects in the Unity project, the code shown in Figure 4.10 is utilized when the project starts running. Although the provided code example only demonstrates the creation of a simple conveyor, it serves as a blueprint for creating all other objects in the simulation.

The process begins by reading the parameters from a JSON file. These parameters contain essential information about the objects to be instantiated. Depending on the type of object specified in the JSON file (in this case, a conveyor for the simple conveyor), the corresponding object will be created.

The Instantiate() function is then employed to generate the objects in the scene. By using the parameters obtained from the JSON file, each object is positioned accurately on the floor with the correct orientation.

Following the instantiation process, the relevant parameters are sent to the “SimpleConveyorMove” script. This script handles the movement and behaviour of the created conveyor object, allowing it to function as intended within the simulation.



```
string json = File.ReadAllText(filePath);
data = JsonUtility.FromJson<JSONData>(json);

foreach (Object objects in data.objects)
{
    if(objects.type == "conveyor")
    {
        GameObject newConveyor = Instantiate(SimpleConveyor, new Vector3(objects.positionX, 1, objects.positionZ), Quaternion.identity);

        newConveyor.transform.rotation = Quaternion.LookRotation(new Vector3(objects.directionX, 0, objects.directionZ));

        onBelt.Add(newConveyor);

        SimpleConveyorMove convMove = newConveyor.GetComponent<SimpleConveyorMove>();
        if (convMove != null)
        {
            convMove.conveyorOffsetSen = objects.offsetSen;
            convMove.conveyorOffsetAct = objects.offsetAct;
            convMove.conveyorDirectionX = objects.directionX;
            convMove.conveyorDirectionZ = objects.directionZ;
            convMove.conveyorSpeed = objects.speed;
        }
        sensors.Add(false);
        actuators.Add(false);
        actuators.Add(false);
    }
```

Figure 4.10. Implementation of the Simple Conveyor on the simulation.

5. Results and Discussion

A series of test were conducted on the created objects. The primary objective of these tests was to validate the functionality and performance of the objects, including their movements, sensors detection, and Modbus communication with the user's program.

5.1. Tests

The first test involved the implementation of a single conveyor to assess the efficiency of Modbus communication. A piece was placed on the conveyor's sensor, and the corresponding sensor value was updated and transmitted to the client. This test aimed to ensure that the Modbus communication protocol was correctly established between the Shopfloor class and the user's program. A visual representation of this test is shown on the Figure 5.1.

Figure 5.1. Modbus communication test.

Building upon the previous test, a tester was designed to verify the bi-directional movement of the simple conveyor. Two simple conveyors were used, and a piece was made to move back and forth between them, triggered by the sensors. This test enabled the validation of both actuators of the conveyor, as the movement occurred in both directions. Additionally, the change in motion also served as a validation of the sensors' accuracy. A visual representation of this test is shown on the Figure 5.2.

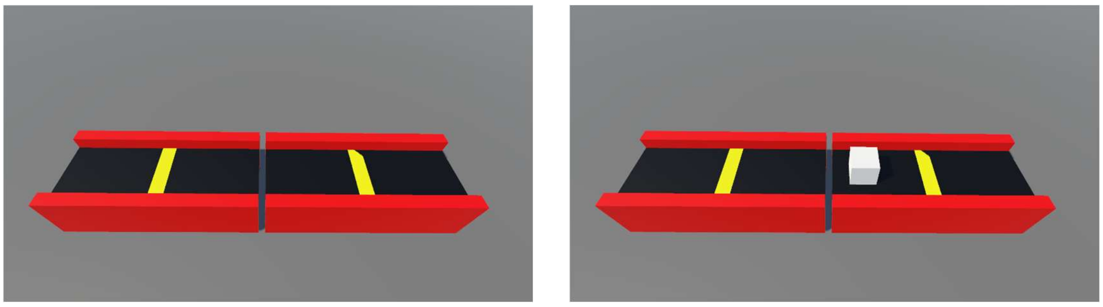


Figure 5.2. Simple Conveyor test.

Next, the focus shifted to testing the rotator conveyor. The objective was to implement a tester that could validate the rotation functionality of the rotator. For this purpose, two simple conveyors were arranged in an inverse L shape, with a rotator at the joint. The rotator was programmed to perform a 90º rotation in both orientations. This test was essential to confirm the proper functioning of the rotator's actuators and sensors responsible for the rotational movement. A visual representation of this test is shown on the Figure 5.3.

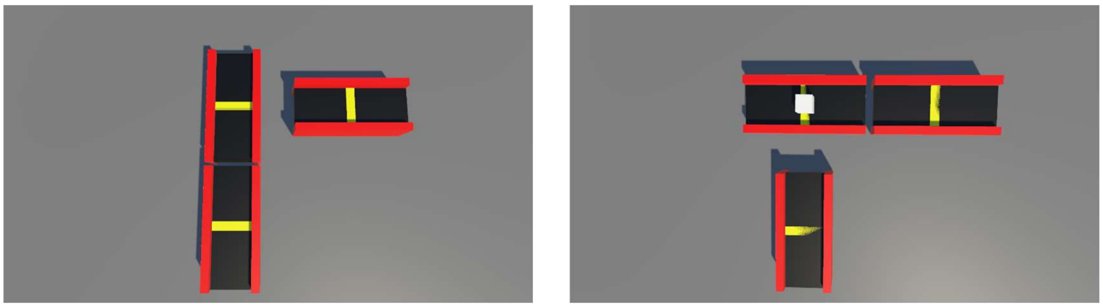


Figure 5.3. Rotator Conveyor test.

Dedicated tests were performed on the machine to assess its ability to transform pieces. Multiple iterations of this test were conducted, using different tools of the machine with various types of pieces and transformation times. The objective was to evaluate the machine's reliability and versatility in executing different transformation processes. The distinct tests are presented on the Table 5.1 and an example of this test is shown on the Figure 5.4.

Table 5.1. Work-piece transformations.

Starting Piece	Produced Piece	Tool	Processing Time
P1	P6	T1	20s
P2	P4	T2	10s
P2	P5	T3	15s
P3	P6	T1	20s
P4	P7	T3	10s
P6	P8	T2	30s

P7	P9	T2	10s

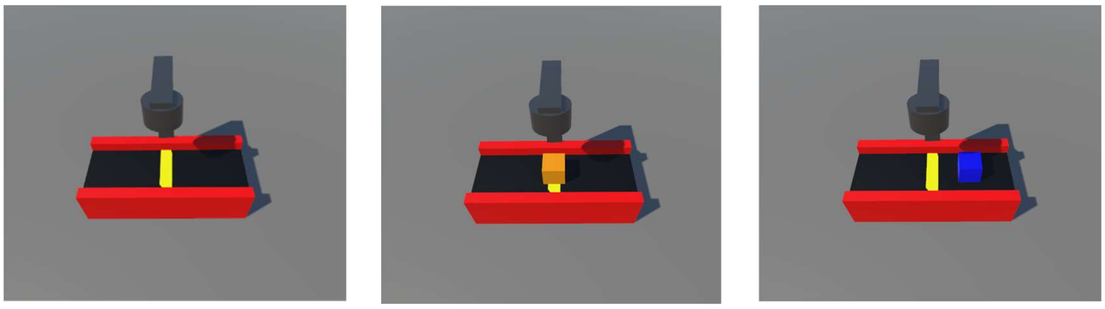


Figure 5.4. Machine test.

In this final test, a more realistic shop floor simulation was created to validate all the objects' functionalities in a practical setting. The aim was to replicate cells A and C of the current FEUP simulator with a simplified layout. This comprehensive test allowed for the evaluation of the objects' performance under real-world scenarios, providing valuable insights into their practical applicability. A visual representation of this test is shown on the Figure 5.5.

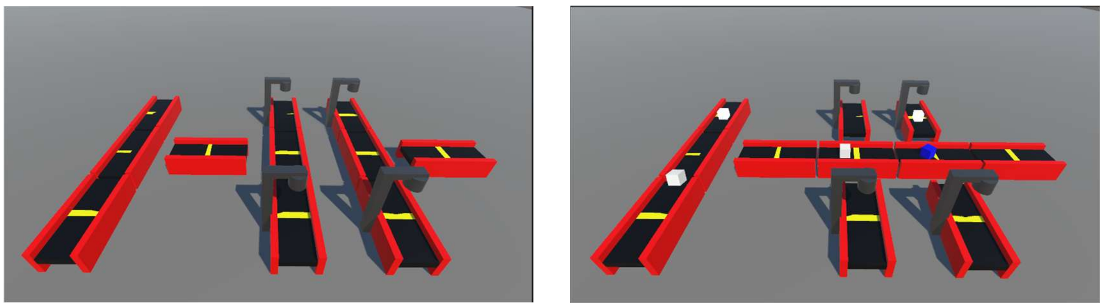


Figure 5.5. Shop Floor test.

6. Conclusion

This dissertation successfully achieved its primary objective of developing a digital twin capable of simulating the plant floor simulator in FEUP's automation labs. Through an in-depth exploration of various game engines able to create a digital twin capable of simulating the plant floor simulator, it was determined that Unity was the most suitable choice for this project due to its extensive documentation and user-friendly nature, making it accessible even to beginners.

Additionally, a comprehensive study of the existing simulator was conducted to gain a thorough understanding of its components, which served as the foundation for defining the requirements for the implementation of the new digital twin using Unity. The identified requirements included the integration of the Modbus communication protocol, along with the implementation of various elements such as a simple conveyor, a rotating conveyor, and a conveyor with a machine tool.

Following the successful implementation of the requirements, a series of comprehensive and robust tests were conducted to carefully validate each element of the system. The initial test involved the movement of a piece between two simple conveyors, validating their consistent logic and functionality. Subsequently, the second test focused on assessing the rotator conveyor's in executing rotation movements efficiently and accurately. In the final test, the machine's capability to change tools and alter the colour of the piece was rigorously examined. The flawless execution of these tasks within the designated time frames unequivocally confirmed the effectiveness and reliability of the implementation.

Through these meticulous tests, the system has demonstrated its efficiency, robustness, and adaptability, leaving no doubts about the successful validation of all elements. The results underscore the system's capacity to meet the required functionalities.

Lastly, the creation of the digital twin was a successful endeavour, resulting in the development of a realistic plant layout that effectively replicated the functionalities of the existing simulator. Throughout the implementation process, all the identified requirements were seamlessly integrated and operated as intended, leading to a fully functional digital twin.

6.1. Future work

It is important to acknowledge that like any project, this dissertation also has its limitations. Due to the complexity of industrial systems, some requirements have not been fully captured in the simulation. Future work could focus on refining the digital twin to include more advanced features and precise modelling, like adding the warehouses and allowing for different communications protocol, making it even more representative of real-world scenarios.

Additionally, to enhance result validation, tests conducted by running the simulation and the real shop floor simultaneously, would allow for a better comparation of the object interactions and execution times, ensuring a stronger evaluation of the outcomes.

6.2. Final assessment

The development of this project enabled personal and educational growth through a combination of self-organization and pursuit of excellence, ensuring the achievement of the project objectives and deadlines. Despite the difficulty of self-evaluation in lengthy projects, the deep interest in the subject, intense concentration, and unwavering dedication helped maintain a high level of focus throughout the semester, resulting in an outstanding final outcome.

References

[1] H. Fatorachian and H. Kazemi, "A critical investigation of Industry 4.0 in manufacturing: theoretical operationalisation framework," *Production Planning & Control,* vol. 29, pp. 1-12, 01/11 2018, doi: www.doi.org/10.1080/09537287.2018.1424960.

[2] N. Mason, P. Hailey, D. Mifsud, and J. Urquhart, "Systems Astrochemistry: A New Doctrine for Experimental Studies," *Frontiers in Astronomy and Space Sciences,* vol. 8, 12/08 2021, doi: www.doi.org/10.3389/fspas.2021.739046.

[3] A. M. Madni, C. C. Madni, and S. D. Lucero, "Leveraging Digital Twin Technology in Model-Based Systems Engineering," *Systems,* vol. 7, doi: www.doi.org/10.3390/systems7010007.

[4] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, and M. Hoffmann, "Industry 4.0," *Business & Information Systems Engineering,* vol. 6, pp. 239-242, 2014/08/01 2014, doi: www.doi.org/10.1007/s12599-014-0334-4.

[5] M. Piccarozzi, B. Aquilani, and C. Gatti, "Industry 4.0 in Management Studies: A Systematic Literature Review," *Sustainability,* vol. 10, doi: www.doi.org/10.3390/su10103821.

[6] M. Correia, "Industrie 4.0 Framework, Challenges and Perspectives," 2014.

[7] H. Cañas, J. Mula, M. Díaz-Madroñero, and F. Campuzano-Bolarín, "Implementing Industry 4.0 principles," *Computers & Industrial Engineering,* vol. 158, p. 107379, 2021/08/01/ 2021, doi: www.doi.org/10.1016/j.cie.2021.107379.

[8] S. Rajput and S. P. Singh, "Identifying Industry 4.0 IoT enablers by integrated PCA-ISM-DEMATEL approach," *Management Decision,* vol. 57, 07/31 2018, doi: www.doi.org/10.1108/MD-04-2018-0378.

[9] E. Mueller, X.-L. Chen, and R. Riedel, "Challenges and Requirements for the Application of Industry 4.0: A Special Insight with the Usage of Cyber-Physical System," *Chinese Journal of Mechanical Engineering,* vol. 30, pp. 1050-1057, 2017/09/01 2017, doi: www.doi.org/10.1007/s10033-017-0164-7.

[10] M. Khan, X. Wu, X. Xu, and W. Dou, "Big data challenges and opportunities in the hype of Industry 4.0," presented at the 2017 IEEE International Conference on Communications (ICC), 2017.

[11] R. Cioffi, M. Travaglioni, G. Piscitelli, A. Petrillo, and F. De Felice, "Artificial Intelligence and Machine Learning Applications in Smart Production: Progress, Trends, and Directions," *Sustainability*, vol. 12, doi: www.doi.org/10.3390/su12020492.

[12] P. Adolphs and U. Epple, "Reference Architecture Model Industrie 4.0 (RAMI4.0)," ZVEI: Die Electroindustrie, 2015.

[13] W. Kritzinger, M. Karner, G. Traar, J. Henjes, and W. Sihn, "Digital Twin in manufacturing: A categorical literature review and classification," *IFAC-PapersOnLine,* vol. 51, pp. 1016-1022, 2018/01/01/ 2018, doi: www.doi.org/10.1016/j.ifacol.2018.08.474.

[14] B. Lydon, "RAMI 4.0 Reference Architectural Model for Industrie 4.0," *International Society of Automation,* 2019.

[15] M. Grieves, "Digital Twin: Manufacturing Excellence through Virtual Factory Replication," 03/01 2015.

[16] F. Jaensch, A. Csiszar, C. Scheifele, and A. Verl, "Digital Twins of Manufacturing Systems as a Base for Machine Learning," in *2018 25th International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*, 20-22 Nov. 2018 2018, pp. 1-6, doi: www.doi.org/10.1109/M2VIP.2018.8600844.

[17] D. M. Botín-Sanabria, A.-S. Mihaita, R. E. Peimbert-García, M. A. Ramírez-Moreno, R. A. Ramírez-Mendoza, and J. d. J. Lozoya-Santos, "Digital Twin Technology Challenges and Applications: A Comprehensive Review," *Remote Sensing*, vol. 14, doi: www.doi.org/10.3390/rs14061335.

[18] Y. Lu, C. Liu, K. I. K. Wang, H. Huang, and X. Xu, "Digital Twin-driven smart manufacturing: Connotation, reference model, applications and research issues," *Robotics and Computer-Integrated Manufacturing,* vol. 61, 2020, doi: www.doi.org/10.1016/j.rcim.2019.101837.

[19] "Factory I/O." factoryio.com (accessed 2023).

[20] "Siemens." siemens.com (accessed 2023).

[21] P. Mishra and U. Shrawankar, "Comparison between Famous Game Engines and Eminent Games," *International Journal of Interactive Multimedia and Artificial Intelligence,* vol. 4, pp. 69-77, 2016, doi: www.doi.org/10.9781/ijimai.2016.4113.

[22] S. Pavkov, I. Franković, and N. Hoić-Božić, "Comparison of game engines for serious games," in *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 22-26 May 2017 2017, pp. 728-733, doi: www.doi.org/10.23919/MIPRO.2017.7973518.

[23] J. K. Haas, "A History of the Unity Game Engine," 2014.

[24] O. C. Agbonifo, O. A. Sarumi, and Y. M. Akinola, "A chemistry laboratory platform enhanced with virtual reality for students' adaptive learning," *Research in Learning Technology,* vol. 28, 2020, doi: www.doi.org/10.25304/rlt.v28.2419.

[25] T. Nieminen, "Unity game engine in visualization, simulation and modelling," 2021.

[26] M. Foxman, "United We Stand: Platforms, Tools and Innovation With the Unity Game Engine," *Social Media + Society,* vol. 5, 2019, doi: www.doi.org/10.1177/2056305119880177.

[27] A. Šmíd, "Comparison of unity and unreal engine.," Czech Technical University in Prague, 2017.

[28] J. Lee, *Learning Unreal Engine Game Development*. 2016, p. 274.

[29] A. Sanders, *An Introduction to Unreal Engine 4*. 2016, p. 256.

[30] W. Qiu and A. Yuille, "UnrealCV: Connecting Computer Vision to Unreal Engine," in *Computer Vision – ECCV 2016 Workshops*, Cham, G. Hua and H. Jégou, Eds., 2016// 2016: Springer International Publishing, pp. 909-916.

[31] D. Valente de Macedo, M. A. F. Rodrigues, and Y. R. Serpa, "Desenvolvimento de Aplicações Gráficas Interativas com a Unreal Engine 4," *Revista de*

Informática Teórica e Aplicada, vol. 22, pp. 181-202, 11/25 2015, doi: www.doi.org/10.22456/2175-2745.56371.

[32] R. Pennanen, "Virtual Reality Multiplayer in Unreal Engine 5 with C++," 2022.

[33] C. Vohera, H. Chheda, D. Chouhan, A. Desai, and V. Jain, "Game Engine Architecture and Comparative Study of Different Game Engines," in *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, 6-8 July 2021 2021, pp. 1-6, doi: www.doi.org/10.1109/ICCCNT51525.2021.9579618.

[34] S. Smith and D. Trenholme, "Computer game engines for developing first-person virtual environments," *Virtual Reality*, vol. 12, 09/01 2008, doi: www.doi.org/10.1007/s10055-008-0092-z.

[35] A. Ioana and A. Korodi, "DDS and OPC UA Protocol Coexistence Solution in Real-Time and Industry 4.0 Context Using Non-Ideal Infrastructure," *Sensors*, vol. 21, doi: www.doi.org/10.3390/s21227760.

[36] S. Profanter, A. Tekat, K. Dorofeev, M. Rickert, and A. Knoll, "OPC UA versus ROS, DDS, and MQTT: Performance Evaluation of Industry 4.0 Protocols," in *2019 IEEE International Conference on Industrial Technology (ICIT)*, 13-15 Feb. 2019 2019, pp. 955-962, doi: www.doi.org/10.1109/ICIT.2019.8755050.

[37] S. Mohagheghi, J. Stoupis, and Z. Wang, "Communication protocols and networks for power systems-current status and future trends," in *2009 IEEE/PES Power Systems Conference and Exposition*, 15-18 March 2009 2009, pp. 1-9, doi: www.doi.org/10.1109/PSCE.2009.4840174.

[38] S. Figueroa-Lorenzo, J. Añorga Benito, and S. Arrizabalaga, "Modbus Access Control System Based on SSI over Hyperledger Fabric Blockchain," *Sensors*, vol. 21, doi: www.doi.org/10.3390/s21165438.

[39] G. Meza, C. d. Carpio, N. Vinces, and M. Klusmann, "Control of a three-axis CNC machine using PLC S7 1200 with the Mach3 software adapted to a Modbus TCP/IP network," in *2018 IEEE XXV International Conference on Electronics, Electrical Engineering and Computing (INTERCON)*, 2018, pp. 1-4, doi: www.doi.org/10.1109/INTERCON.2018.8526429.

[40] A. Swales, "Open modbus/tcp specification," *Schneider Electric*, vol. 29, p. 19, 1999.