escola superior de tecnologia e gestão
instituto politécnico de leiria

Polytechnic of Leiria
School of Technology and Management
Department of Electrical Engineering
Master in Electrical and Electronic Engineering

# INDUSTRY 4.0: LEGACY DEVICES INTEGRATION WITH OPC UA AND THE DIGITAL TWIN

MARCELLA BARROS DE BRITO CAVALCANTI

Leiria, March, 2023

Polytechnic of Leiria
School of Technology and Management
Department of Electrical Engineering
Master in Electrical and Electronic Engineering

# INDUSTRY 4.0: LEGACY DEVICES INTEGRATION WITH OPC UA AND THE DIGITAL TWIN

MARCELLA BARROS DE BRITO CAVALCANTI
2200085

Project developed under supervision of Professor Hugo Filipe Costelha de Castro (hugo.costelha@ipleiria.pt) and co-supervision of Professor Carlos Fernando Couceiro de Sousa Neves (carlos.neves@ipleiria.pt).

Leiria, March, 2023

# ACKNOWLEDGMENTS

First, I would like to thank my supervisor, Professor Hugo Filipe Costelha de Castro, for his support, attention, and guidance throughout these years. Also a special thanks to my co-supervisor, Professor Carlos Fernando Couceiro de Sousa, whose support and encouragement were equally important during this period.

A special thanks is also to all Professors from the Electrical and Electronic Engineering Master's Programme at IPLeiria, specially to Professor Luís Miguel Ramos Perdigoto, who joined the project I have been working on at the Advanced Robotics and Smart Factories laboratory. His inputs during this time have also contributed in the development described in this report.

I must also express profound gratitude to my parents, Mércia and Ivo, for their continuous love and support throughout these years, even from a distance. It is hard being away, but they understand this opportunity and have always encouraged me to pursue my goals. The same goes to Marco Antonio, who I deeply miss. Thank you very much.

At last, thank you to all the friends and colleagues I have met living in Leiria for the past two and a half years. You all have made this time away from a home more pleasant and fun.

This work was partially supported by the S4PLAST - Sustainable Plastics Advanced Solutions (POCI-01-0247-FEDER-046089), under the Programa Operacional Competitividade e Internacionalização (POCI), and Programa Operacional Regional de Lisboa, PORTUGAL 2020, through Fundo Europeu de Desenvolvimento Regional (FEDER).

# RESUMO

Ao longo dos anos, a constante evolução da indústria tem levado a muitos avanços nas fábricas e nos sistemas de fabricação. Os termos "Fábricas Inteligentes" e "Sistemas de Fabricação Inteligentes" têm sido usados para descrever a última onda de inovações tecnológicas que transformaram a maneira como as fábricas operam. Uma dessas inovações é o conceito de Gémeos Digitais, que é uma cópia virtual realista de um objeto físico. Essa tecnologia permite que todo o chão de fábrica seja digitalizado e que processos físicos estejam intimamente ligados aos seus contrapartes cibernéticos.

O desenvolvimento dos Gémeos Digitais abrange vários desafios, incluindo a precisão do modelo, segurança e integração de diferentes dispositivos e sistemas, incluindo interoperabilidade e padronização entre eles. O objetivo deste trabalho é desenvolver aplicações-chave para apoiar a implementação de Gémeos Digitais em um ambiente de Fábrica Inteligente, descrever um exemplo de desenvolvimento de uma aplicação habilitadora da Indústria 4.0 para um dispositivo legado, bem como o projeto de um Gémeo Digital para um sistema industrial real desde o início.

Um resultado chave deste trabalho é um caso de uso bem-sucedido da criação de um Gémeo Digital para uma célula produtiva real na indústria, usando o Robot-Studio como ambiente de simulação, e OPC UA como protocolo de comunicação entre os dispositivos na célula. O Gémeo Digital desenvolvido é capaz de simular o comportamento dos dispositivos na célula e realizar a lógica de controlo da célula. Também é capaz de armazenar dados históricos do processo, que podem ser analisados e usados para realizar a otimização do processo.

Outro resultado relevante está relacionado com o uso do Gémeo Digital de um dispositivo para apoiar o desenvolvimento de uma aplicação, realizando testes e validação, eliminando assim a necessidade de aceder ao dispositivo real nessas fases. Essa abordagem mostra que esta tecnologia pode ser usada para acelerar o desenvolvimento e reduzir o tempo de inatividade de dispositivos industriais, reduzindo custos e melhorando o processo de produção.

***Keywords***— Indústria 4.0, Fábrica Inteligente, Gémeo Digital, OPC UA

# ABSTRACT

Over the years, the constant evolution of the industry has led to many advancements in factories and manufacturing systems. The terms "Smart Factories" and "Smart Manufacturing Systems" have been used to describe the latest wave of technological innovations that have transformed how factories operate. One of these innovations is the concept of the Digital Twin, which is a realistic virtual copy of a physical object. This technology allows entire manufacturing shop-floors to be digitalized, and physical processes to be tightly intertwined with their cyber counterparts.

The development of Digital Twins encompasses several challenges, including model accuracy, security, and the integration of different devices and systems, including interoperability and standardization across them. The goal of this work is to develop key applications to support the implementation of Digital Twins in a Smart Factory environment, by describing an example of the development of an Industry 4.0 enabling application for a legacy device, as well as the design of a Digital Twin for a real industrial system from the ground up.

A key result of this work is a successful use case of creating a Digital Twin for a quality control cell in the industry, using RobotStudio as the simulation environment and OPC UA as the communication protocol between the devices in the cell. The developed Digital Twin is capable of simulating the behaviour of the devices in the cell, as well as performing the cell's control logic. It is also capable of storing historical process data, which could be analyzed and used to perform process optimization.

Another relevant result is related to the use of a device's Digital Twin to support the development of an application, performing tests and validation, while eliminating the need of accessing the real device. It shows that this technology can be used to speed up development and reduce downtime of industrial devices, thus reducing costs and improving the production process.

***Keywords***— Industry 4.0, Smart Factory, Digital Twin, OPC UA

# CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

l

# LIST OF ABBREVIATIONS AND ACRONYMS

| | |
|---|---|
| AMQP | Advanced Message Queuing Protocol. |
| CoAP | Constrained Application Protocol. |
| CPS | Cyber-Physical Systems. |
| CS | Companion Specifications. |
| DDS | Data Distribution Service. |
| DT | Digital Twin. |
| ERP | Enterprise Resource Planning. |
| FSM | Finite State Machine. |
| HDA | Historical Data Access. |
| HMI | Human-Machine Interface. |
| IIoT | Industrial Internet of Things. |
| IoT | Internet of Things. |
| LwM2M | LightweightM2M. |
| MES | Manufacturing Execution Systems. |
| MQTT | Message Queue Telemetry Protocol. |
| OPC | Open Platform Communication. |

| | |
|---|---|
| PLC | Programmable Logic Controller. |
| QC | Quality Control. |
| QoS | Quality of Service. |
| RDBMS | Relational Database Management System. |
| RS | RobotStudio. |
| RTOS | Real Time Operating System. |
| SC | SmartComponent. |
| SCADA | Supervisory Control and Data Acquisition. |
| SDK | Software Development Kit. |
| SQL | Structured Query Language. |
| UA | Unified Architecture. |
| UART | Universal Asynchronous Receiver/Transmitter. |

1

# INTRODUCTION

The introduction of the Industry 4.0 paradigm has led to many advancements in Smart Manufacturing Systems over the years. By digitalizing traditional industrial processes and bridging the physical and virtual worlds together, this paradigm causes a disruptive transformation and upgrade in intelligent industrialization (C. Zhang et al., 2021). One of these advancements is the possibility to have an entire manufacturing shop-floor digitalized, where physical processes are tightly intertwined with their cyber counterparts. This is the concept of a Digital Twin (DT), which is a realistic virtual copy of a physical object.

DT is one of the key enabling technologies in Smart Factories and it can provide real-time state monitoring, energy consumption analysis, product failure analysis and prediction, product maintenance strategy, as well as intelligent optimization and update. Based on the concept of DT, Tao and M. Zhang (2017) explore the concept of Digital Twin Shop-Floor (DTS), defining four key components: the physical shop-floor, the virtual shop-floor, the shop-floor service system, and shop-floor digital twin data. As these parts keep consistent and optimized with each other, data from both the physical and virtual systems, as well as fused data, can be used to drive production.

According to Lu et al. (2020), DT-driven applications are a core element of future manufacturing, and they will change the fundamentals of manufacturing systems and operations, as the convergence of the digital and physical worlds enables smarter decision making. The work presented in this report aims for the development of key applications to support the implementation of DTs in a Smart Factory environment, describing an example of the development of an Industry 4.0 enabling application for a legacy device, as well as the design of a DT for a real industrial system from the ground up.

To achieve this goal, key technologies were studied. First, an extensive review of industrial communications systems was done, to provide a classification and comparison of the different solutions, based on their most relevant features in the context of Industry 4.0. The result of this study presented Open Platform Communication (OPC) Unified Architecture (UA) as the most complete and applicable middleware solution for industrial communication systems. For that reason, OPC UA is also a key enabling technology for the development of the work, and described in this report. Furthermore, the concepts of DT, Smart Manufacturing and Cyber-Physical Systems (CPS) were also explored, and a significant body of knowledge on these areas was reviewed.

The main contributions of this work are the development of key generic, scalable applications to support DT development using OPC UA, as well as the demonstration of how DT can be used to support hardware development for an industrial device. Moreover, this work describes the use of OPC UA and embedded systems to elevate a legacy device to an Industry 4.0 level, and provide an example of the design of a DT featuring process control and simulation, historical data storage and retrieval, and Industry 4.0 compliant communication. The integration of several software tools and Software Development Kit (SDK) is also discussed, as well as the implementation of OPC UA applications for several different platforms.

Throughout the development of the masters' project, two papers were published, as follows:

- Cavalcanti, Marcella, Hugo Costelha, and Carlos Neves (June 2023). "Industry 4.0 Machine-to-Machine Communication Protocols and Architectures on the Shop Floor". In: pp. 222–234. ISBN: 978-3-031-33889-2. DOI: 10.1007/978-3-031-33890-8_19.
- Cavalcanti, Marcella, Hugo Costelha, Carlos Neves, et al. (July 2023). "Digital Twin Development for a Quality Control Cell". In: 2023 9th International Conference on Control, Decision and Information Technologies (CoDIT).

This report is organized as follows: This first chapter defines the field and the main goals of the research work, as well as the main contributions that are provided. Chapter 2 provides the reader with a description of the fundamental concepts that are the basis of this work and a literature review on those subjects. Chapter 3 describes the generic applications, based on OPC UA, that were developed throughout this work. Chapter 4 describes the developed work in creating an Industry 4.0 enabling application for a legacy device. Chapter 5 details the development of a DT for a Quality Control (QC) cell in a manufacturing process. Chapter 6 presents the tests and results of the applications that were developed. Finally, in Chapter 7, the main conclusions of the work are presented and some paths for future work are suggested.

# 2 FUNDAMENTAL CONCEPTS AND RELATED WORK

This chapter reviews relevant concepts and related work on the topics relevant for the developed work, namely Industry 4.0, industrial communication systems, OPC UA, and Digital Twin, which are significant for the embodiment of subsequent chapters.

## 2.1 INDUSTRY 4.0

Industrial manufacturing systems have been evolving through what is called "Industrial Revolutions" since the late 1700s, as shown in Fig 1. The First Industrial Revolution brought the transition from manual work to the first manufacturing processes through mechanization and mechanical power generation. The Second Industrial Revolution came with the electrification of the industry, and the introduction of work division, assembly lines and mass production. The Third Industrial Revolution is characterized by the automation of processes, with the introduction of the first Programmable Logic Controller (PLC) (Rojko, 2017).

Most recently, the Fourth Industrial Revolution, or Industry 4.0, is a concept introduced by the German government that aims for the transformation of manu-

1 https://mitranslations.com/industrial-revolution

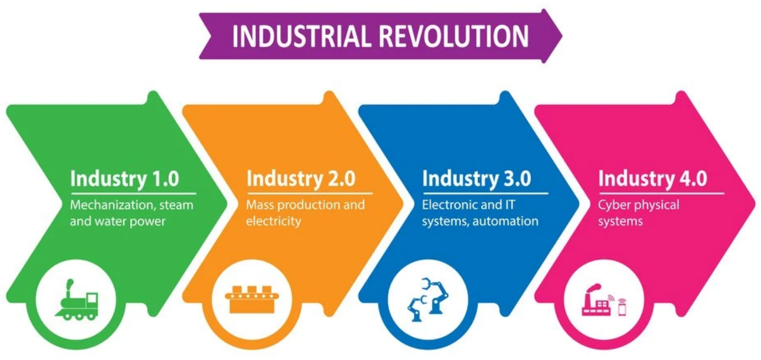

INDUSTRIAL REVOLUTION
Industry 1.0
Mechanization, steam and water power
Industry 2.0
Mass production and electricity
Industry 3.0
Electronic and IT systems, automation
Industry 4.0
Cyber physical systems

Figure 1: Industrial Revolutions[1].

facturing, enabled by advanced technologies such as the Internet of Things (IoT) and CPS (Hermann et al., 2016), to create Smart Factories and achieve industrial processes optimization, increase productivity and lower production costs. In an Industry 4.0 environment, interoperability and connectivity are important elements. This involves a continuous flow of information between devices, components, manufacturing systems, and actors, which is facilitated by Machine-To-Machine (M2M) interaction, as well as Human-To-Machine (H2M) collaboration, particularly when production tasks are too unstructured for full automation (Rojko, 2017). This allows the transformation of the production process from a centralized decision making instance into a decentralized one (C. Zhang et al., 2021).

## 2.2 INDUSTRIAL COMMUNICATION SYSTEMS

Also referred to as “Industrial Networks”, industrial communication systems are networks typically adopted in factory automation, manufacturing and process control, to implement data exchange between controllers, sensors, actuators, input/output devices and industrial equipment in general (Vitturi et al., 2019). These networks are also responsible for the data flow and information sharing to higher levels of automation in factories, such as Supervisory Control and Data Acquisition (SCADA) systems, Manufacturing Execution Systems (MES) and Enterprise Resource Planning (ERP).

In the 1980s, dedicated automation networks, called fieldbus systems, were initially developed and implemented as a replacement for point-to-point links between industrial devices, using serial digital communication. This change allowed more flexibility in the shop floor, enabling remote configuration and diagnostics to be carried out more easily. Moreover, noticeable savings were made in both cabling and deployment costs, because of the shared communication support (Wilamowski and Irwin, 2016).

The downside of fieldbuses was, though, the lack of standardization. Over the years, many proprietary solutions were developed, such as PROFIBUS, INTERBUS and MODBUS, as well as CAN-based implementations, such as Devicenet and CANopen. Most of these protocols are still often used in industry, typically in shop floors, between PLC, I/O-modules, sensors and actuators. In the end of the 1990s, Ethernet-based networks were introduced in the industrial environment, due to the growing use of Internet technologies and Information Technology (IT). At first, the lack of genuine real-time capabilities in standard Ethernet prevented the development of one single Ethernet solution for automation purposes (Wollschlaeger et al., 2017), and more dedicated proprietary ones were created.

Over the years, modifications were made to meet these time constraints and several solutions emerged, such as EtherCAT and PROFINET IRT, which are two of the most used Real-Time Ethernet (RTE) protocols. Ethernet protocols are most commonly being used in industry for communication between process control entities and support systems, such as SCADA (Wilamowski and Irwin, 2016).

Most recently, aiming for the same practicality of the increasing use of wireless technology in daily life, industrial wireless networks (IWNs) have emerged. Starting from the wireless sensor networks (WSNs), such as WirelessHART, these solutions were initially implemented along with wired networks to increase flexibility, mobility and easiness of deployment. Some examples used in industry are Zigbee, Bluetooth and WLAN (Wilamowski and Irwin, 2016).

The main disadvantages of IWNs can be security issues and power consumption, particularly for battery powered equipment, but its advantages match most of the requirements for Industry 4.0, with more of these solutions being adopted within factories (Li et al., 2017). For non-battery operated devices, a physical wired connection must exist for power, thus the advantage of having a complete wireless device is lower when compared to a battery-operated device.

It results that, currently, the field of industrial communication networks is very heterogeneous, with many different, non-interoperable, solutions being used. Table 1 shows some network examples and applications, while Fig. 2 shows the industrial networks market share of new nodes installed in 2022, according to HMS Networks(Carlsson, 2022). The graph shows Industrial Ethernet had the highest growth and continues to take market share from fieldbuses, which went from 28% of new nodes installed in 2021, down to 27% in 2022. Meanwhile, the share for wireless solutions stayed stable compared to past years. The most used Industrial Ethernet protocol was Ethernet/IP, whereas PROFIBUS was still number one in fieldbus usage.

As a consequence of this heterogeneous landscape, automation practitioners resort to the use of middleware software to interconnect systems. In this context, a middleware can be seen as a digital interface used to establish communication between physical (OT-Operations Technology) and digital environments (IT). By connecting hardware and applications with the necessary level of abstraction from heterogeneous systems, it provides a common infrastructure to support communication (Ch et al., 2018). The use of this kind of strategy is also compatible with the idea of

Table 1: Classification of Industrial Networks

| | **Features** | |
|---|---|---|
| **Networks** | ***Solutions*** | ***Typical use*** |
| Fieldbuses | PROFIBUS, INTERBUS, Modbus, CANopen, DeviceNET | Device communication in shop floors |
| Industrial Ethernet | PROFINET IRT, EtherCAT, Ethernet/IP, Modbus TCP | Communication between control units and SCADA systems |
| Wireless Networks | Zigbee, WirelessHART, Bluetooth, WLAN | Sensor networks, parameter monitoring and automation control |

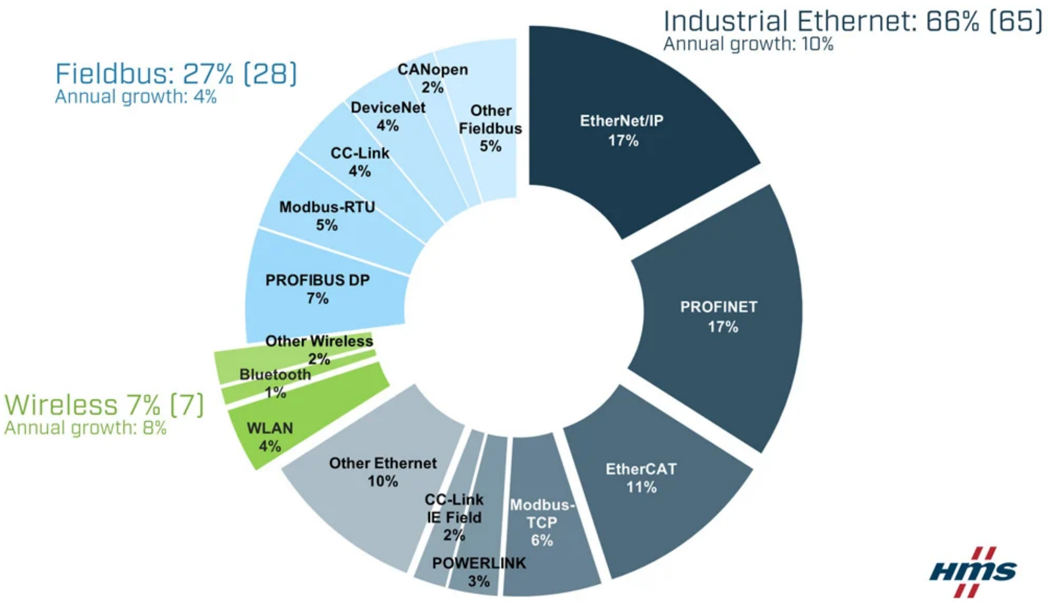


Figure 2: Industrial Networks market share in 2022(Carlsson, 2022).

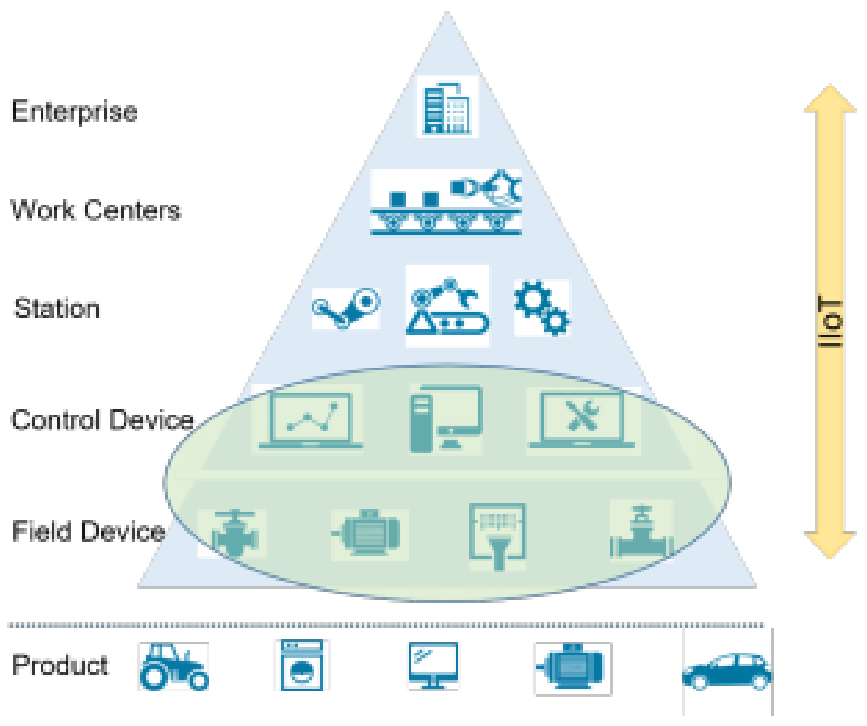


Figure 3: The company pyramid (Tran et al., 2020).

Industrial Internet of Things (IIoT), where every node is connected and data flows not only horizontally, at the device control level, but also vertically, between control, production and planning systems (Tran et al., 2020), which becomes even more relevant considering the typical company pyramid, as shown in Fig. 3. Examples of communication middleware for industrial applications are Message Queue Telemetry Protocol (MQTT), Data Distribution Service (DDS), Advanced Message Queuing Protocol (AMQP), Constrained Application Protocol (CoAP) and OPC UA.

MQTT is a standard developed by the OASIS consortium[2]. It is described as "an extremely lightweight publish/subscribe messaging transport that is ideal for connecting remote devices with a small code footprint and minimal network bandwidth"[3]. It uses the concept of an MQTT-server, also known as a broker, which holds all the data from the connected nodes. Therefore, devices must simply report data to the broker, not storing any data themselves. Such devices can also be controlled by the broker (Profanter et al., 2019). MQTT also provides 3 levels of Quality of Service (QoS), namely, at most once (level 0), at least once (level 1), and exactly once (level 2).

2 https://www.oasisconsortium.com/
3 https://mqtt.org/

As MQTT is an open communication protocol, some companies in the automation sector saw it as an opportunity to develop IIoT solutions using this standard. That is the case of Sparkplug, which is a specification for MQTT, created with the purpose of better defining its infrastructure and semantics, in order to improve interoperability and make data easily available and standardized for SCADA, MES and Human-Machine Interface (HMI) solutions in industry (Céspedes Cubides and Gualdrón, 2020).

DDS is an open middleware standard developed by the Object Management Group (OMG). It works by introducing a virtual Global Data Space where applications are able to share information by reading and writing data-objects addressed by means of an application-defined name and a key[4]. It supports QoS parameters, such as reliability, bandwidth, delivery deadlines and resource limits. It is also real-time capable, with its nodes being able to do peer-to-peer communication using UDP multicast, thus removing the need of a centralized network management system (Balador et al., 2017).

AMQP is referred to as a reliable protocol for business messaging. It allows different systems to interact, as long as they can create and interpret this data format. Using AMQP, the network will be organized in nodes, which can deliver messages or provide them storage. Data is directly transferred between nodes, so any network model can be implemented, the most used being the centralized model, where data is transmitted via the server (Andrei et al., 2020). It also supports QoS.

CoAP is a specialized web transfer protocol for use with constrained nodes and constrained networks in IoT[5]. It was developed as an internet standards document(Leiba, 2008) and it uses the Representational State Transfer (REST) mechanism (keophilavong et al., 2019). On top of CoAP, the Open Mobile Alliance (OMA) has defined the LightweightM2M (LwM2M), which is a client-server protocol with several management functionalities for resource-constrained devices and remote applications (Karaagac et al., 2019). The LwM2M specification also implements a data model, which is organized as a three-level tree, and the levels are Object, Object Instance, and Resource[6]. The OMA also holds the LwM2M Object and Resource Registry, where new objects and resources can be submitted for registration.

OPC is “the interoperability standard for the secure and reliable exchange of data in the industrial automation space and in other industries”[7] developed and

4 https://www.omg.org/omg-dds-portal/
5 https://coap.technology
6 https://avsystem.github.io/Anjay-doc/LwM2M.html#data-model
7 https://opcfoundation.org/about/what-is-opc/

maintained by the OPC Foundation. The standard, named OPC Classic, is based on Microsoft Windows technology using the COM/DCOM (Distributed Component Object Model) for the exchange of data between software components. It brings a series of specifications to define the interface between servers and clients, real-time and historical data access, and alarm and events monitoring. OPC UA is a platform independent service-oriented architecture that integrated all OPC Classic functionalities in one extensible framework[8]. This architecture has two main components: transport, which defines the protocols to serialize or deserialize data sent over the network, and a data model, which defines rules on how to expose an information model (Balador et al., 2017). The semantic description of this model (the address space) is one of the major strengths of OPC UA (Profanter et al., 2019).

Additionally, since 2012, there is the oneM2M initiative, which comprises multiple Standards Developer Organizations (SDOs) worldwide, with the goal of providing a standard interface where incompatible devices can exchange data, manage information, and interact (Willner et al., 2017). There are currently 200 active members in oneM2M, developing the specifications for a common service layer, which can exist between applications and networks, exposing functions needed by IoT across different industry segments. This standard might work as an interoperability hub across industries and industry-specific protocols[9].

## 2.3 OPC UA

OPC UA is a services-oriented architecture for industrial automation systems. It is a platform-independent open standard that enables secure and reliable exchange of data between devices, systems, and applications. It allows for the integration of different devices, regardless of their manufacturer or communication protocol, providing a common data model, which allows for easy communication and data exchange between different systems. This enables the creation of a seamless and secure communication network across various systems and devices, which can be used for process control, monitoring, and data analysis (Lehnhoff et al., 2012).

The OPC UA standard consists of a series of specifications. The first seven parts define the core specification (overview and concepts, security, address space model, services, information model, mappings and profiles), while the others define data access (events, alarms, conditions, historical access). The address space and

8 https://opcfoundation.org/about/opc-technologies/opc-ua/
9 https://www.onem2m.org/

information model of OPC UA employ a tree-based hierarchical representation and use references to organize nodes and constitute a network (Lam and Haugen, 2019).

Schleipen et al. (2016) describe four different application scenarios for OPC UA, which go from a simple data and information hub, to the most common scenario of monitoring and control, and even a more "futuristic" scenario that demonstrates the ability of OPC UA to be used as a generic interface to orchestrate every component (CPS) in a production cell.

### 2.3.1 *Communication*

OPC UA works primarily as a client-server communication model, where clients and servers implement a set of services to handle communication and the exchange of data (Mathias et al., 2020), but it also implements a publish-subscriber communication model, to allow OPC UA to perform deterministic real-time data transfer over Time Sensitive Networks (TSN) (Zezulka et al., 2019).

The client-server communication model involves the client sending a request to the server, which then responds with the appropriate data or action. In OPC UA, this means a client can send requests to monitor variables in the server, call methods, subscribe to events and alarms, or access historical data.

In the publish-subscribe model, devices and systems can publish data to a topic or channel, and any interested subscribers can receive this data without having to make explicit requests. This approach allows the exchange of data in real-time between devices, and is often used when time constraints need to be met.

### 2.3.2 *Historical Data Access*

OPC UA Historical Data Access (HDA) is a separate specification that describes the access to past values of object instances within an OPC UA server. Under this service, a client is able to query a snapshot of the server within a specified time range (Mathias et al., 2020). The server then provides the historical process data in a standardized manner, which enables applications to retrieve and analyze data from different sources, including legacy systems, and integrate it into a unified system for analysis and visualization.

In an industrial context, this historical process data can be used to identify trends and patterns and provide insights about the performance of industrial processes over time. By analyzing this data, developers understand how different variables affect the process, and how it has changed over time. This information can then be used to optimize processes, improve product quality, reduce energy consumption, etc.

### 2.3.3 *Companion Specifications*

A standardized address space and information model is what allows OPC UA to be used to connect systems from different vendors and foster interoperability. For that reason, beyond the core OPC UA information model, there has been an increasing trend towards the creation of domain-specific information models for different fields in the industry, such as vision systems, robotics, injection machines, etc. These information models are called OPC UA Companion Specifications (CS), and they are created with the collaboration of cross-vendor working groups (Friedl et al., 2020).

The CS derive from OPC UA core model, inheriting its features and hierarchy, while also allowing some modifications. This approach allows third parties to develop their own models, which are suitable for describing new devices and their capabilities (Perzylo et al., 2019). When similar devices from different manufacturers follow the same CS, it increases compatibility, eases integration and improves interoperability.

## 2.4 DIGITAL TWIN

A critical component of both Industry 4.0 and Smart Factories is the Digital Twin, a virtual replica or digital model of a physical object, system, or process. DTs are created by combining data from various sources to create a real-time virtual representation of the physical system (Glaessgen and Stargel, 2012). A DT can be used throughout the lifecycle of a system or product, from design and development, through operation and maintenance, to optimize performance, reduce downtime, and support decision-making (Tao, H. Zhang, et al., 2019).

The development of DTs encompasses several challenges, including model accuracy, security, and the integration of different devices and systems. DTs often require the integration of data from a variety of sources, including sensors, manufacturing systems, and ERP systems. Achieving interoperability and standardization across these disparate systems is a significant challenge (Fuller et al., 2020; Liu et al., 2021; Lu et al., 2020; Tao, Xiao, et al., 2022).

Abdelsattar et al. (2022) proposed an architecture based on OPC UA, which offers the conversion between an existing SCADA system to a DT, without the need for additional hardware. It consists of a single client/gateway device that acts as the point of intersection between industrial controllers, embedded system controllers and cloud services. Each of these components is considered a server node and use OPC UA to connect to the gateway device, which is responsible for gathering and processing all data, and updating the DT. In their case study, a FESTO MPS (Module Production System) processing station was chosen to demonstrate the concept, a PLC and a Raspberry-Pi acted as server nodes, and the Ignition Designer[10] module was used to store data and create SCADA and HMI screens. However, in their case study there is no 3D representation for their DT, and they rely on the SCADA screen for monitoring and visualization.

The architecture proposed by Souza et al. (2019) utilizes a network of IIoT devices to gather data and control a physical process. This IIoT Gateway uses OPC UA to structure this information and send it to an Internal Server, which is responsible for processing this data and updating the DT. Their DT has two access interfaces, a more comprehensive one with an operational approach, and a more condensed one focused on supporting business and management decisions. This concept was used in an experimental application in a didactic assembling plant. As in the previous work, their DT does not provide a 3D representation for visualization. The use of OPC UA is restricted to data gathering and structuring, and there is no data exchange between devices for process control.

The work presented by Protic et al. (2020) implements a bi-directional DT application for a smart cobots assembly cell (SCAC). The SCAC consists of two collaborative robots and a motion control software application responsible for sending them tasks. The 3D virtual representation of the cell is created in Siemens NX, and Ignition SCADA is employed as well to acquire and display real time data from the cobots. All these components are linked through ethernet and communicate via OPC UA. The use of OPC UA methods in the motion control application, and in the cobots' servers for process control, is not explored in this work.

The work by Martins et al. (2020) aims to reduce the commissioning time of automated systems by implementing Virtual Commissioning using the DT as a shared model. The work defines methodologies to integrate legacy and newly released

10 https://inductiveautomation.com/ignition/designer

industrial equipment into a Smart Factory environment, using their DT for design, commissioning, support, and supervision of the real equipment. A set of novel virtual engineering tools based on OPC UA and ABB RobotStudio simulation software was created to achieve this goal, raising the use of simulation to a new level, with the capability to monitor a real asset, while proposing a methodology to make industrial devices OPC UA-enabled for Industry 4.0 and future factories. The guidelines proposed were then implemented in an automation system created within an academic robotics laboratory. However there is no mention in this work about historical process data access or the use of databases connected to the DT. This work served as the basis for the development of the work detailed in the next chapters of this report.

Finally, Perzylo et al. (2019) propose the architecture for a semantic DT, which is a semantic representation of a manufacturing resource's properties, using OWL (Web Ontology Language) ontologies to encode the information models found in OPC UA NodeSet specifications, combined with the resource's geometry and kinematic model, when applicable. This concept should enable the creation of a full-fledged formal representation of hardware and software properties of a manufacturing resource. As in the previous mentioned work, this work also does not explore the concept of historical data access in the DT.

# 3 BUILDING BLOCKS

This chapter describes the development of key block applications using OPC UA, that will be used in the more comprehensive applications described in the next chapters. This generic applications are important because they provide a foundation for building more complex and specific applications that can meet the unique needs of various industries. These key block applications have been designed to be modular, scalable, and interoperable, which allows them to be easily integrated into larger systems and customized to meet specific requirements.

As referred previously, OPC UA is an open source multiplatform service oriented architecture for automation systems. Several implementations exist, and the ones used throughout this work are the OPC UA .NET stack, from the OPC Foundation[1], and the open62541 C stack[2], developed and maintained mainly by a group of companies and institutions.

Developed in C# using the .NET StandardLibrary[3], UA .NET allows the development of applications that run on many common platforms available today, without requiring platform-specific modifications. As OPC Foundation's official stack, UA .NET is well maintained and regularly updated, ensuring that it remains compatible with the latest OPC UA specifications and best practices. This makes it a reliable and future-proof choice for developing OPC UA applications.

Open62541, developed in C99 and C++98, is designed to be a lightweight and modular stack, using less computational resources, and allowing OPC UA applications to run in less powerful platforms, such as microcontrollers. This makes open62541 a great choice for developing OPC UA applications in embedded systems, IoT devices, and other resource-constrained environments. Despite its lightweight nature, open62541 is still a fully compliant OPC UA stack, supporting all the essential features and functionalities required for building secure and reliable OPC UA applications, namely OPC UA client/server communication, subscriptions, method calls and security (with user authentication and encryption). It also includes a range of advanced features, such as support for multi-threading, user-defined data

1 https://github.com/OPCFoundation/UA-.NETStandard
2 https://github.com/open62541/open62541
3 https://learn.microsoft.com/en-us/dotnet/fundamentals/

types, and custom information models, making it a versatile and flexible option for developers.

## 3.1 OPC UA SERVER SMARTCOMPONENT

RobotStudio is a powerful software developed by ABB for programming and simulating robot-based processes. It allows users to program, simulate, and test ABB robots in a virtual environment using the same programs and configuration files used on the shop floor, thus reducing the need for physical prototyping and testing. This not only saves time and resources, but also enables users to optimize robot processes before they are deployed on the shop floor.

RobotStudio (RS) provides a feature called SmartComponent (SC), which are software modules that simulate the behaviour of important components in a robotized system, such as sensors, grippers, conveyors, vision systems, and other mechanisms. By using SCs, users can accurately simulate the interactions between different components and optimize the overall performance of the robotized process. For example, a SC can be created to simulate the behaviour of a conveyor belt, allowing users to test how the robot interacts with the conveyor without the need for a physical conveyor.

RS comes with a variety of built-in SCs for simpler operations, such as linear motion conveyors and mechanism joint position control. These SCs can be easily used to create custom SCs using the SmartComponent Editor, which provides a visual interface for defining the properties and behaviour of the component. For more complex operations, users can also create their own SCs using the RS SDK in C# through Visual Studio. With this SDK, users can create custom scripts that can be run within RS, custom user interfaces, or applications to integrate RS with other software programs.

When creating a SC, users need to define its properties, inputs, and outputs. Properties are the parameters that define the behaviour of the SC, such as the speed of a conveyor belt, or the force exerted by a gripper. Inputs and outputs are the communication channels through which the SC interacts with other components in the RS simulation. For example, an input for an SC that simulates a sensor could be the distance between the robot and the object being sensed, while an output could be a Boolean value indicating whether the object has been detected.

In their work, Martins et al. (2020) have demonstrated the potential of SCs by developing a generic SC that launches an OPC UA server or client for an industrial device, depending on its working mode. The SC works in two modes: monitoring and simulation. In monitoring mode, the SC runs a client that connects to the device's server and subscribes to data, allowing for the DT to be updated in real-time in RS. This allows users to monitor the behaviour of the physical device in RS, without the need for physical access to the device. In simulation mode, the SC runs a server with the same structure as the real device's server and allows the DT to be used to simulate the device's behaviour. This can be useful in situations where the physical device is not available, or for testing and development purposes. Based on their work, and in order to support the development of the DT described later in this report, an OPC UA Server SC was developed using RS SDK, with the structure shown in Fig. 4.

While in the work by Martins et al. (2020) the device's behaviour is also implemented in the SC developed with RS SDK, alongside the implementation of the OPC UA server and client, the SC developed in this work can be used to create an OPC UA server for any existing SC that already implements the behaviour of a device. The developed SC replicates all inputs, outputs and properties of the existing SC, so it can be connected to other components in the RS simulation in the same way. If there is an existing geometry or mechanism linked to the existing SC, the developed SC can carry it and bind it to the existing SC. This approach allows this SC structure to be used to create an OPC UA server for any existing component in the RS simulation with minimal changes to the code, as it eliminates the need to recreate the behaviour of the device from scratch.

The OPC UA server in the SC is implemented with UA .NET and its Address Space can reflect some or all of the properties, inputs and outputs of the existing component, as well as any other additional data, depending on the user's needs related to which information should be available in the device's server. When launched in RS, the SC provides the endpoint of the server that was created.

## 3.2 OPC UA SERVER ON AN ESP32

As previously mentioned, the authors of open62541 developed it to be a lightweight and modular stack implementation of OPC UA, which allows for the creation of OPC UA applications in many types of devices, including microcontrollers. The ESP32 is a low-cost, low-power system-on-a-chip (SoC) microcontroller designed by Espressif

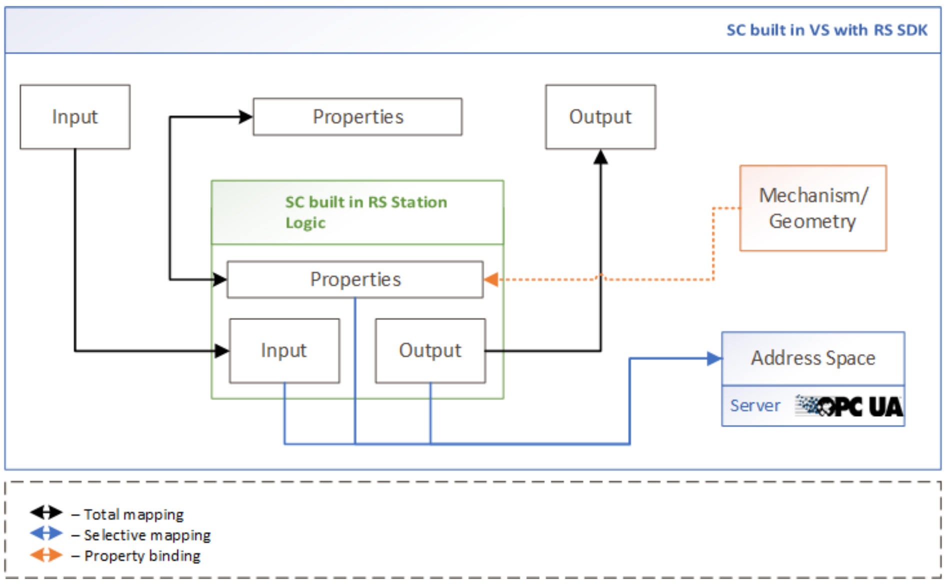


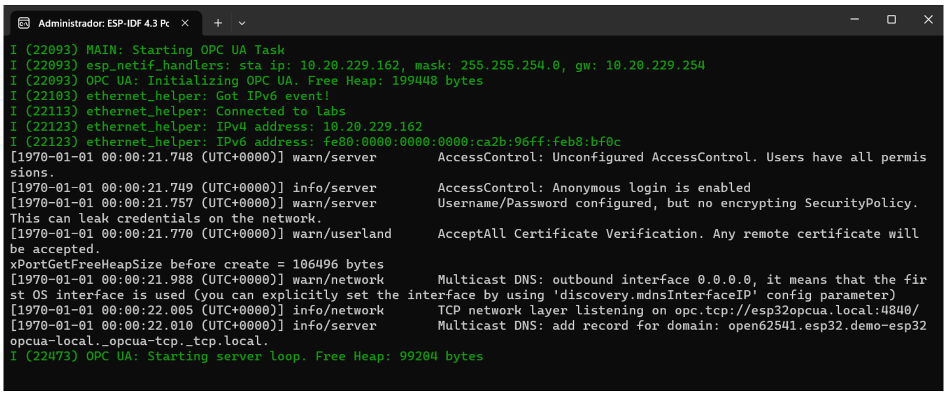
Figure 4: Structure of the OPC UA server SmartComponent.

Systems[4]. It integrates Wi-Fi and Bluetooth connectivity, and includes a variety of peripheral interfaces, such as Universal Asynchronous Receiver/Transmitter (UART), I2C and SPI. Due to their low cost and energy efficiency, there has been a recent trend towards using microcontrollers like the ESP32 in small-scale industrial applications (Gatial et al., 2020).

ESP-IDF[5] is Espressif's official open-source IoT Development Framework for the ESP32 series of SoCs. It provides an SDK for generic application development on those platforms, using programming languages such as C and C++. It supports many software components, including Real Time Operating System (RTOS), which is used for real-time computing applications that process data and events with critically defined time constraints. To achieve these constraints, in an RTOS application, repeated tasks are performed within a tight timeframe, unlike in a general-purpose operating system. The most used RTOS for microcontrollers such as ESP32 is FreeRTOS[6].

An application that uses an RTOS can be structured as a set of tasks, where each task executes within its own context, with no dependency on other tasks. Therefore it is possible to develop an OPC UA server in ESP32 using open62541 and FreeRTOS, by creating a task that implements the server's configuration, context and main loop iteration. Fig. 5 shows the output of a server running on ESP32, visualized with the ESP-IDF monitor function, which relays the logging output of

4 https://www.espressif.com/en
5 https://www.espressif.com/en/products/sdks/esp-idf
6 https://www.freertos.org/



```
I (22093) MAIN: Starting OPC UA Task
I (22093) esp_netif_handlers: sta ip: 10.20.229.162, mask: 255.255.254.0, gw: 10.20.229.254
I (22093) OPC UA: Initializing OPC UA. Free Heap: 199448 bytes
I (22103) ethernet_helper: Got IPv6 event!
I (22113) ethernet_helper: Connected to labs
I (22123) ethernet_helper: IPv4 address: 10.20.229.162
I (22123) ethernet_helper: IPv6 address: fe80:0000:0000:0000:ca2b:96ff:feb8:bf0c
[1970-01-01 00:00:21.748 (UTC+0000)] warn/server        AccessControl: Unconfigured AccessControl. Users have all permis
sions.
[1970-01-01 00:00:21.749 (UTC+0000)] info/server        AccessControl: Anonymous login is enabled
[1970-01-01 00:00:21.757 (UTC+0000)] warn/server        Username/Password configured, but no encrypting SecurityPolicy.
This can leak credentials on the network.
[1970-01-01 00:00:21.770 (UTC+0000)] warn/userland      AcceptAll Certificate Verification. Any remote certificate will
be accepted.
xPortGetFreeHeapSize before create = 106496 bytes
[1970-01-01 00:00:21.988 (UTC+0000)] warn/network       Multicast DNS: outbound interface 0.0.0.0, it means that the fir
st OS interface is used (you can explicitly set the interface by using 'discovery.mdnsInterfaceIP' config parameter)
[1970-01-01 00:00:22.005 (UTC+0000)] info/network       TCP network layer listening on opc.tcp://esp32opcua.local:4840/
[1970-01-01 00:00:22.010 (UTC+0000)] info/server        Multicast DNS: add record for domain: open62541.esp32.demo-esp32
opcua-local._opcua-tcp._tcp.local.
I (22473) OPC UA: Starting server loop. Free Heap: 99204 bytes
```

Figure 5: Output of OPC UA task running on ESP32.

the applications running in ESP32, through the serial connection. This development was based on the project available in https://github.com/Pro/open62541-esp32.

## 3.3 OPC UA SERVER WITH HISTORICAL ACCESS

As mentioned in Chapter 2, OPC UA defines in its specification a standard to store and retrieve historical process data. The HDA enables the clients to access and analyze the historical data of a server.

Open62541 provides a plugin[7] to support the access to historical data. This plugin contains three main elements, as shown in Fig. 6:

- HistoryDatabase, which contains the main interface between the server and the plugin. It is responsible for handling the requests made by clients to access historical data.
- HistoryDataBackend, which implements the integration with a specific database. It provides the implementation of the callbacks used to retrieve data from the database. The HistoryDataBackend is responsible for translating the requests made by the client into queries that can be executed by the database. It also provides a mechanism for storing data in the database.
- HistoryDataGathering, which encapsulates the gathering and storage of data. It is responsible for periodically collecting the data from the process and storing it in the database by calling the methods defined in the HistoryDataBackend, so it can be retrieved by clients later. Each node in a server that provides HDA needs to be registered for the gathering of historical data.

7 https://blog.basyskom.com/2019/initial-support-for-servers-with-historical-data-access-in-open62541/, accessed in 23/03/2020

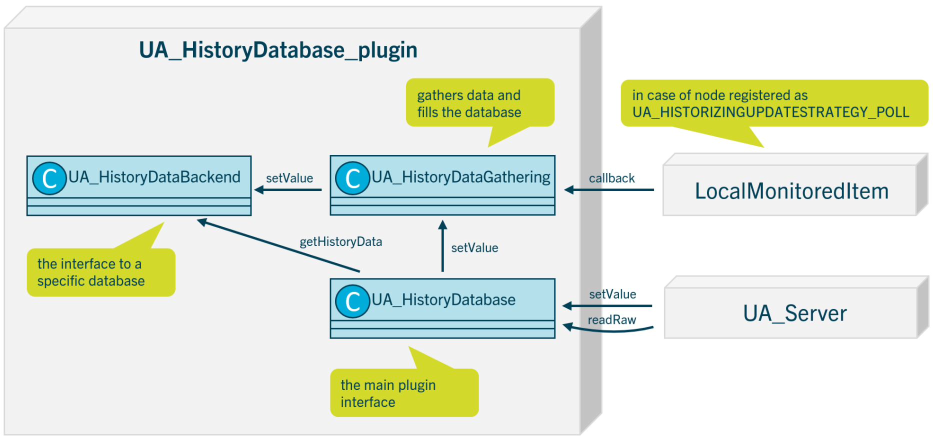

UA_HistoryDatabase_plugin
gathers data and fills the database
in case of node registered as UA_HISTORIZINGUPDATESTRATEGY_POLL
C UA_HistoryDataBackend
setValue
C UA_HistoryDataGathering
callback
LocalMonitoredItem
getHistoryData
setValue
the interface to a specific database
C UA_HistoryDatabase
setValue
readRaw
UA_Server
the main plugin interface

Figure 6: Open62541 HDA plugin[7].

Open62541 provides a sample HistoryDataBackend that implements an in-memory database, which is suitable for testing and development purposes, but not for production environments where data should persist in the long term. In an industrial context, where process information is mostly stored in a local network database, a different backend should be implemented. The implementation of a custom backend involves creating a database connection, defining the necessary queries to insert and retrieve data, and implementing the HistoryDataBackend interface methods to perform the required operations.

The example provided in https://github.com/nicolasr75/open62541_sqlite demonstrates a custom implementation of a HistoryDataBackend which allows connection to an SQLite[8] database. Unlike traditional client-server database management systems, SQLite operates on a local file system and does not require a separate server process. Instead, it reads and writes directly to local disk files, using Structured Query Language (SQL) commands. SQLite is a popular choice for embedded systems and mobile devices due to its lightweight nature, zero-configuration setup, and ACID (Atomicity, Consistency, Isolation, Durability) compliance.

Based on the SQLite backend example above, a new HistoryDataBackend was developed, creating a connection to a MySQL database. MySQL[9] is an open-source Relational Database Management System (RDBMS) that is widely used for managing and organizing data in web and server applications. It is one of the most popular database management systems in use today[10], particularly for web-based applications and websites. It is also based on the SQL standard, which is used to manage and manipulate data in relational databases. It can be used to create and

8 https://www.sqlite.org/index.html
9 https://www.mysql.com/
10 https://db-engines.com/en/ranking

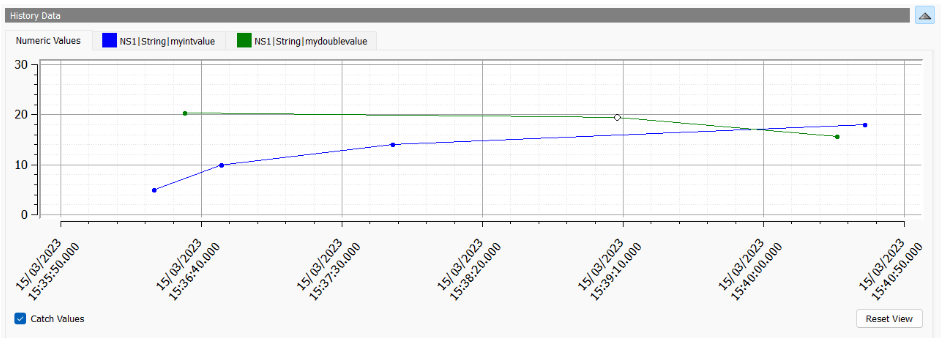


Figure 7: History data request in UaExpert.

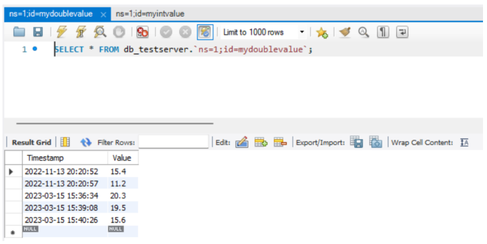
manage databases, tables, and other structures, as well as to insert, update, and retrieve data.

In this project, the MySQL backend was developed in C++, using the MySQL Connector/C++[11] library. When compared to the SQLite backend, the first improvement was the implementation of the `serverSetHistoryData`, which is a method in the HistoryDataBackend called by the HistoryDataGathering component to store new data in the database. This method was not implemented in the SQLite example. Additionally, the MySQL backend was developed to be more generic, as it is prepared to handle database interactions for variable nodes of string, integer and float datatypes. In the database, the historical data of each node is stored in individual tables, and the table names are a string composed by the node's namespace and id in the OPC UA server. Each table stores the timestamp and value of the variable node.

Fig. 7 shows the result of a historical data access request to the server using UaExpert[12], which is an OPC UA client application from Unified Automation. The client requested the historical data of two variables in the server in a certain timeframe, and the History Data View in UaExpert allows for the plot visualization of these numeric values. Fig. 8 shows how the same data is stored in the database (shown here using MySQL Workbench[13] viewer).

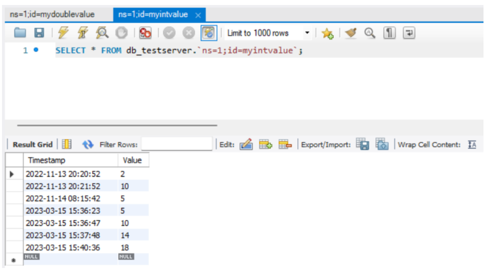
The use of MySQL in this application allows users to store and retrieve OPC UA historical data in a widely used and reliable RDBMS. The MySQL database can be hosted locally or remotely, and can handle large amounts of data, depending on the database server hardware, making it suitable for industrial applications that require long-term storage of process data.

11 https://dev.mysql.com/doc/dev/connector-cpp/8.0/
12 https://www.unified-automation.com/products/development-tools/uaexpert.html
13 https://www.mysql.com/products/workbench/



```sql
SELECT * FROM db_testserver.`ns=1;id=mydoublevalue`;
```

| Timestamp | Value |
|---|---|
| 2022-11-13 20:20:52 | 15.4 |
| 2022-11-13 20:20:57 | 11.2 |
| 2023-03-15 15:36:34 | 20.3 |
| 2023-03-15 15:39:08 | 19.5 |
| 2023-03-15 15:40:26 | 15.6 |
| NULL | NULL |



```sql
SELECT * FROM db_testserver.`ns=1;id=myintvalue`;
```

| Timestamp | Value |
|---|---|
| 2022-11-13 20:20:52 | 2 |
| 2022-11-13 20:21:52 | 10 |
| 2022-11-14 08:15:42 | 5 |
| 2023-03-15 15:36:23 | 5 |
| 2023-03-15 15:36:47 | 10 |
| 2023-03-15 15:37:48 | 14 |
| 2023-03-15 15:40:36 | 18 |
| NULL | NULL |

Figure 8: Historical data stored in MySQL database.

## 3.4 OPC UA CLIENT-SERVER APPLICATION

In a Smart Factory environment where every device communicates over OPC UA, a situation might happen that a single application needs to gather data from many devices, as a client, and also have data or services available, as a server. A simple way to create such an application is to use open62541.

The OPC UA client implementation included with open62541, does not yet implement a background thread or a main loop, which means that the client will not perform any actions automatically in the background. Therefore, in a new client application's main loop, one needs to periodically call a function that keeps the connection to the server established. Without this, the client may lose its connection to the server, leading to data loss or system downtime.

On the other hand, the open62541 server's implementation allows running the server's main loop by calling a function only once. This function starts the server, and the server remains active until it is shut down. However, similar to the client's implementation, it is also possible to periodically call a function that executes one iteration of the server's main loop. This provides the flexibility to integrate the server into an existing application and execute other code simultaneously.

Therefore, by combining these two periodical functions in a loop, it is possible to develop an application that implements both a server and a client. Listing 1 shows an excerpt of this implementation.

Listing 1: Open62541 client-server implementation overview.

```c
int main(void) {
    //Create server's object
    UA_Server* server = UA_Server_new();
    /*
    * Perform server's configuration here
    *
    */
    //Create client's object
    UA_Client* client = UA_Client_new();
    /*
    * Perform client's configuration here
    *
    */
    while(running) {
        UA_Server_run_iterate(server, true);
        UA_Client_run_iterate(client, 0);
    }
}
```

# 4 INTELITEK PALLET CONVEYOR OPC UA SERVER

This chapter describes the methodology used for the implementation of an OPC UA server for a pallet conveyor using ESP32 and open62541. The objective of this implementation is to showcase the integration of a legacy device into Industry 4.0 by adopting the OPC UA standard for data exchange. By using the ESP32, the implementation benefits from its low power consumption, built-in Wi-Fi capability, and robustness, making it well-suited for industrial environments. Moreover, by running the OPC UA server within ESP32, no major physical modifications will be necessary around the existing pallet conveyor, as ESP32 is such a small piece of hardware.

In their work, Martins et al. (2019) described the virtualization of a production system, created in the Advanced Robotics and Smart Factories laboratory of the Polytechnic of Leiria, mainly used for research, development and education, composed by several devices, including an Intelitek pallet conveyor. The device is a rounded corner rectangle, continuous loop, pallet conveyor that belongs to an earlier CIM (Computer Integrated Manufacturing) system from Intelitek, and it is controlled by a PLC using a list of commands (strings) sent and received through RS232 serial communication.

It contains four pallet transporters that move on the conveyor and can be stopped at three different stations. Each station has a set of sensors that capture the passage of a transporter, and a message (string) is sent by the conveyor to indicate these passages when they happen. Each transporter has an unique identification (1-4), and the user can send a command to stop a certain transporter at a certain station. This list of strings (messages and commands) was obtained by applying some reverse engineering to understand how the PLC exchanged data.

As part of the virtualization of the production system, a RS SC was developed using RS SDK to replicate the geometry and behaviour of the conveyor. The same list of strings were used for the component in simulation, but TCP/IP communication was used for outside interaction with the conveyor, instead of serial communication, in order to provide improved integration within the laboratory systems network. Fig. 9 shows the conveyor at the laboratory, and Fig. 10 shows the conveyor in RS.

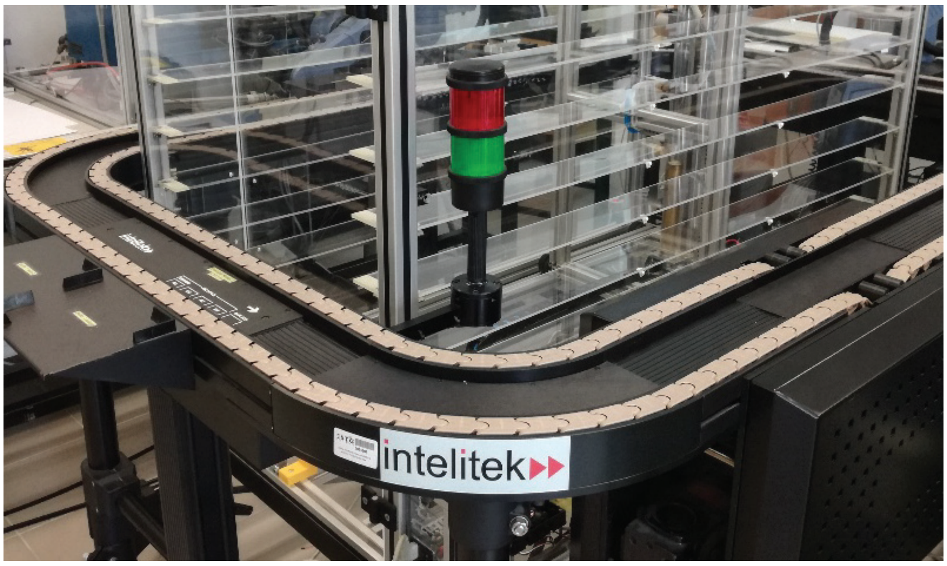


Figure 9: Intelitek pallet conveyor (Martins et al., 2019).

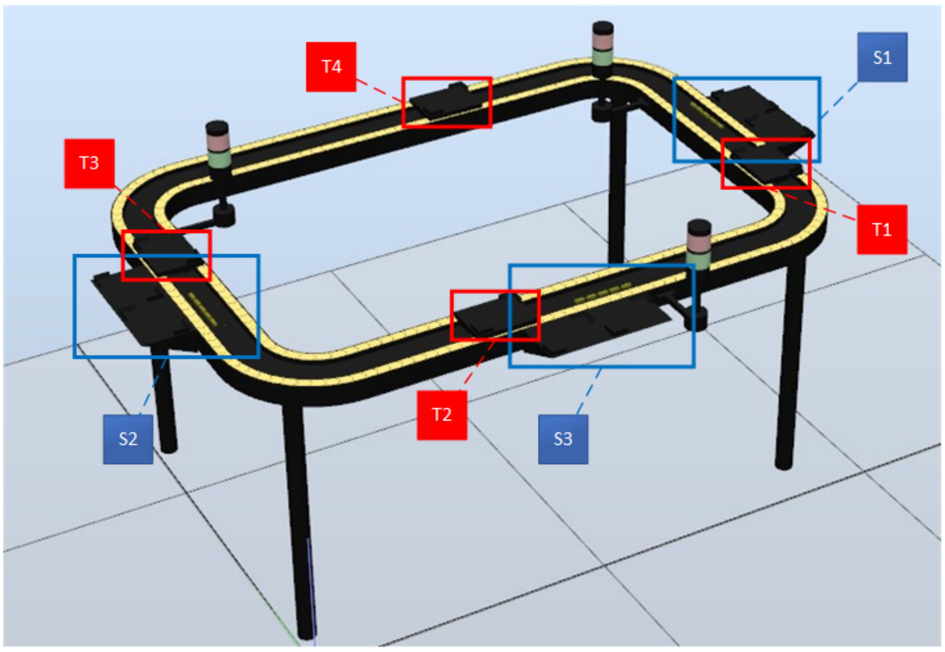


Figure 10: Intelitek conveyor in RobotStudio with stations (S) and transporters (T) indications.

Regarding the project requirements for this development, the control of the conveyor will continue to be performed by the PLC it is connected to, therefore there are no strict time constraints. For the OPC UA communication of the conveyor with the remaining devices in the laboratory, it is important that the conveyor is able to respond to commands within a timeframe of 500 milliseconds. This is so because the commands to the conveyor are expected to be done early, before the parts reach their goal destination, or while the parts are stopped, and because the interaction of external devices with the parts on the conveyor are always done with the parts at rest.

## 4.1 SMARTCOMPONENT UPDATE

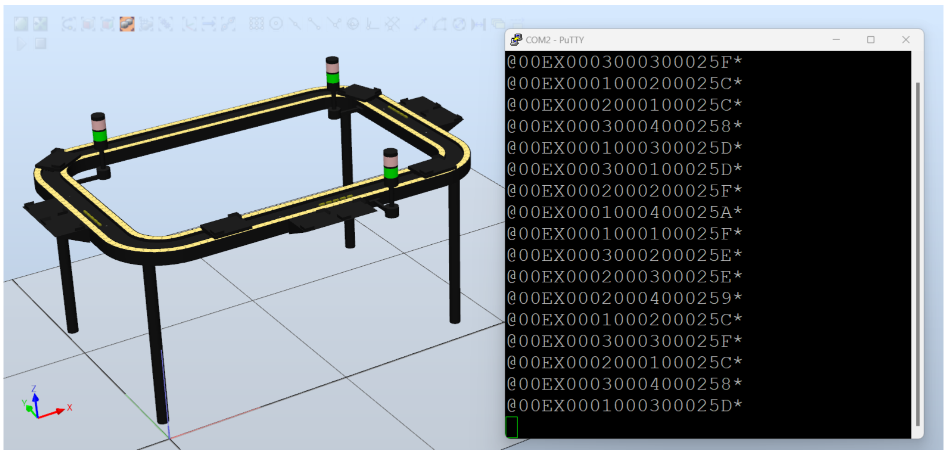
In order to develop an OPC UA server in an ESP32 that can be employed in the real pallet conveyor at the laboratory, the PLC that controls the conveyor needs to able to communicate with the ESP32 through RS232. To facilitate the development of the ESP32 application in this scenario, the virtual conveyor running in RS was used as a DT. This allows for the testing and validation of the ESP32 application's functionality without requiring the physical conveyor, thus reducing the development time.

To be able to use the simulated conveyor as the real one, the conveyor SC was updated to implement serial communication. Then, all development was done using the virtual component running in RS, by connecting an USB-RS232 converter to the computer and allowing the use of RS232 for outside interaction with the virtual conveyor, as it would be done with the real one.

Fig. 11 shows the strings sent by the conveyor through the serial connection, as each transporter passes by the stopping stations. The serial port to be used should be indicated by the user in the SC properties in RS.

## 4.2 RS232-UART INTERFACE

As previously mentioned, the ESP32 is a SoC microcontroller that integrates Wi-Fi and Bluetooth communication, and includes a variety of peripheral interfaces, such as UART, which is used for asynchronous serial communication. By leveraging the UART interface, the ESP32 can be easily used to communicate with legacy devices that may not support more recent communication protocols.



```
@00EX0003000300025F*
@00EX0001000200025C*
@00EX0002000100025C*
@00EX00030004000258*
@00EX0001000300025D*
@00EX0003000100025D*
@00EX0002000200025F*
@00EX0001000400025A*
@00EX0001000100025F*
@00EX0003000200025E*
@00EX0002000300025E*
@00EX00020004000259*
@00EX0001000200025C*
@00EX0003000300025F*
@00EX0002000100025C*
@00EX00030004000258*
@00EX0001000300025D*
```

Figure 11: Strings sent from conveyor SC through serial connection.

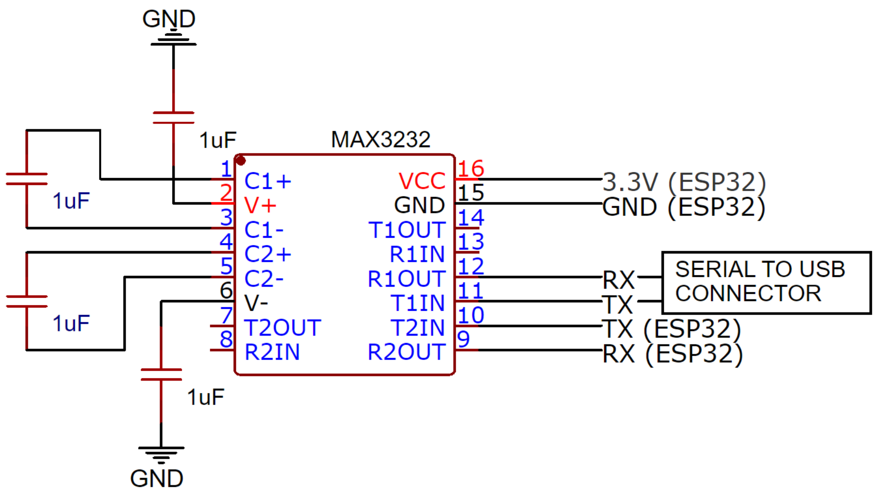


Figure 12: MAX3232-based circuit used in RS232-UART conversion.

RS232 and UART are both serial communication protocols, but they use different signal voltage levels, which means that they require a transducer to enable communication between them. Such interface can be created using a simple circuit based on the MAX3232[1], thus providing the necessary voltage level conversion and signal conditioning to enable reliable communication between RS232 and UART devices.

Therefore, the data exchanged between the virtual conveyor running in RS (through the USB-RS232 interface) and the ESP32 (through the UART interface) can be converted using a circuit with MAX3232. Fig. 12 shows the schematics of this circuit connecting the Tx (transmitter) and Rx (receiver) from the USB-RS232 converter to the Rx and Tx of ESP32's UART interface. The circuit also uses four electrolytic capacitors and it is powered by the 3.3V source pin in ESP32.

1 https://www.ti.com/lit/ds/symlink/max3232.pdf?ts=1679554852103

## 4.3 OPC UA SERVER

As described in Chapter 3, an OPC UA server can be developed for an ESP32 using FreeRTOS and open62541. This allows the ESP32 to act as a powerful and secure industrial automation device, capable of exchanging data and information with other devices and systems that support the OPC UA protocol.

The address space for the conveyor's server contains one variable, which displays the messages received from the conveyor through the UART interface, and one method, which provides one input argument for the user to send a command to the conveyor. Even though the same type of messages are used to control and communicate with the conveyor, this implementation provides a first step to enable this legacy device to comprise with the Industry 4.0 requirements through the OPC UA protocol.

Internally, the variable in the server implements a function that periodically reads from the UART and checks for new messages. In a similar way, the method implements a function to write in the UART when executed by a client.

5

# DIGITAL TWIN FOR A QUALITY CONTROL CELL

This chapter describes the methodology used for the implementation of a Digital Twin for a Quality Control cell. The development of DTs has gained significant attention in recent years due to their potential to enhance efficiency and productivity in various industries, including manufacturing. The DT described in this work was built in RS, using some of the building blocks described earlier, with the communication between the devices in the cell being supported by OPC UA.

QC is the process of evaluating a product or service to ensure that it meets the desired level of quality. In production and manufacturing, QC involves verifying a product's conformity to its original design by using techniques such as measuring, examining, testing, or gauging (Babic et al., 2021). This is an important process, as it helps to identify and address any defects or issues in the manufacturing process that could lead to product failures, safety hazards, or customer dissatisfaction.

By implementing rigorous QC processes, manufacturers can ensure that their products meet the required quality and safety standards. In addition, effective QC can help improve the efficiency and productivity of production processes, as it can be used to identify areas for optimization and improvement.

## 5.1 THE MANUFACTURING PROCESS

The QC cell outlined in this work is part of a broader production process in the automotive industry that aims to manufacture plastic components with a metallic appearance. The component in question is formed by the assembly of two different parts (top and bottom), as shown in Fig. 13.

These parts come from the injection process and are stored in the factory for 24 hours before heading to the QC cell for analysis. This waiting period is necessary as certain defects in the injection process may not be immediately noticeable and only manifest after a period of time.

In the QC cell, the parts are subjected to two types of analysis to ensure their conformity to the required standards. The first one is a dimensional analysis, which

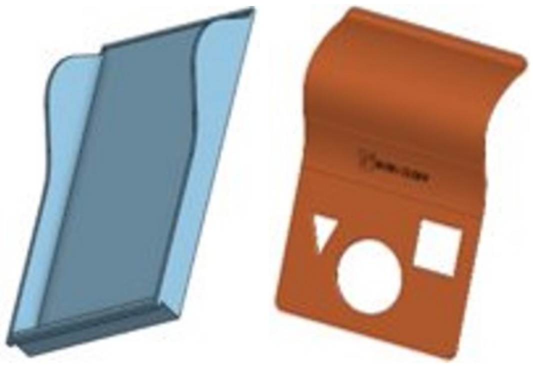


Figure 13: Top and bottom parts of the component.

is carried out using a laser profiler to scan the parts. It uses laser triangulation technology to scan and digitally reconstruct the part as a 3D point cloud. The second type of analysis is a surface (texture) analysis, which is performed by processing high-resolution images of the parts captured by a set of cameras. It uses the deflectometry technique, which consists of creating various scenarios by varying the intensity and angle of incidence of light on the part, and the position of the cameras in relation to the part, in order to enhance its surface for defects detection, such as scratches or material changes. The QC cell is linked to a database where the results of all analyses are stored. As the two types of parts have different QC requirements, the top part needs to undergo both dimensional and surface analysis, while the bottom part will only undergo the dimensional analysis.

In addition to performing QC analysis, the QC cell also has a traceability function. This is achieved through the use of a label maker machine that is capable of printing and applying labels to the parts that have been approved in the QC process. This step is important as it could allow for the development of self-aware smart products, as the product itself carries information which allows access to its own manufacturing process data.

After leaving the QC cell, the parts go back to storage and later are directed to the laser welding station, where they are assembled. Fig. 14 illustrates the manufacturing process.

## 5.2 QC CELL STRUCTURE AND REQUIREMENTS

As mentioned before, the QC cell in the outlined manufacturing process needs to be able to manipulate, analyze, and label two types of parts. To achieve this goal, the cell must be carefully designed and optimized to ensure its ability to handle different

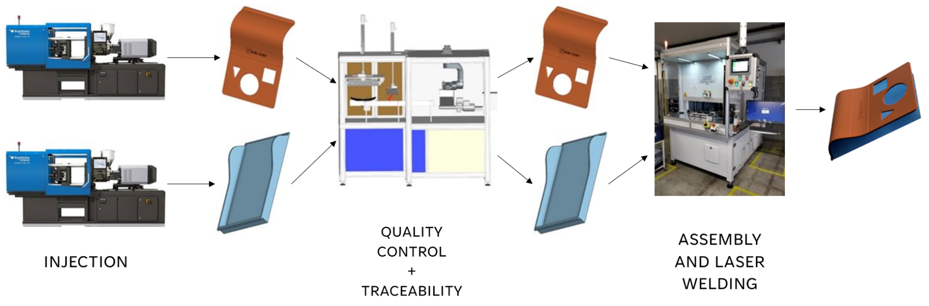


Figure 14: The manufacturing process of the component.

products in a flexible and efficient manner. This requires careful consideration of the layout of the cell and the movement of the parts through the various stages of analysis and labeling. The equipment used in the cell must also be carefully chosen and configured to ensure the cell is capable of processing the parts accurately and efficiently.

The QC cell can be divided in two areas, as shown in Fig. 17:

- The manipulation area, composed by:
  - an **entry conveyor**, where parts will be introduced in the QC process.
  - a **discard conveyor**, for parts which do not meet the quality requirements.
  - an **out conveyor**, for parts which meet the requirements to exit the cell and be picked up for storage.
  - a Hermes+ **label maker** with applicator, to tag the approved parts.
  - a **robot manipulator** to move parts inside the cell, in this case, an Epson N2.
- The Quality Control area, composed by:
  - a Gocator, which is a **3D laser profiler** that will be used to take measurements of both types of parts, to check if they meet the dimensional criteria.
  - Genie Nano **high resolution cameras**, which will be used to check if the parts (top type) meet the surface (texture) quality criteria.
  - a **linear rail with a support structure** where the robot will place the parts to be analyzed.

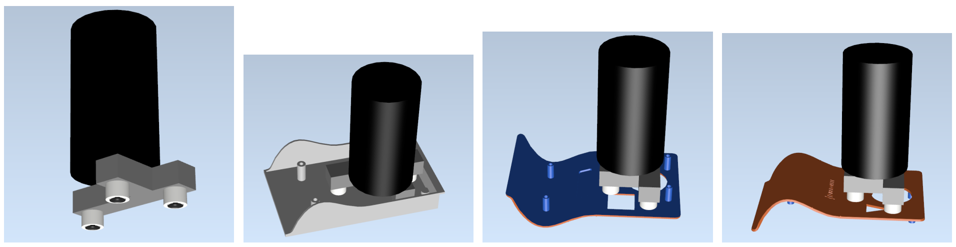


Figure 15: Gripper designed to fit the parts.

In this cell, the devices in the manipulation area are controlled independently. In the quality control area, the laser profiler and cameras are directly connected to a computer that runs the image processing software and provides the analyses results. Furthermore, the interaction between the various systems is done at rest, that is, the parts images are captured with the parts stopped and the robots pick the parts when they are stopped. The only dynamic situation is during the dimensional analysis, given that the part is translated while a scan is performed. However, that is considered as a single system from the implementation point of view, integrating the linear actuators, encoder and the Gocator, which is modelled here at a higher-level, given that the implementation of that system in practice was not part of this work. Therefore, there are no strict time constrains for the cell and, as such, we specified a maximum response of 500 milliseconds for the various OPC UA systems. The only restriction in place is the total amount of time available to process a single part, from input to output of the cell, which should be lower or equal to 90 seconds.

To support the development of the DT, the whole structure of the cell was imported to RS. The structure geometry came from the 3D model CAD files provided by the company. Additionally, the geometry of a gripper with three vacuum cups was created to fit the dimensions of both parts that will be manipulated by the robot, as shown in Figs. 15.

To move the parts within the QC process, two ABB IRB1200 robot manipulators were added outside of the cell. One will move the parts from the input box to the entry conveyor, and the other will move the approved parts from the out conveyor to another (output) box. Both of these robots are also equipped with the gripper designed to fit the parts. Figs. 16 and 17 show the cell built in RS.

## 5.3 DEVICES BEHAVIOUR SIMULATION

To support the DT simulation in RS, the behaviour of each device in the cell needs to be implemented. In RS there is a type of component called Mechanism, which is

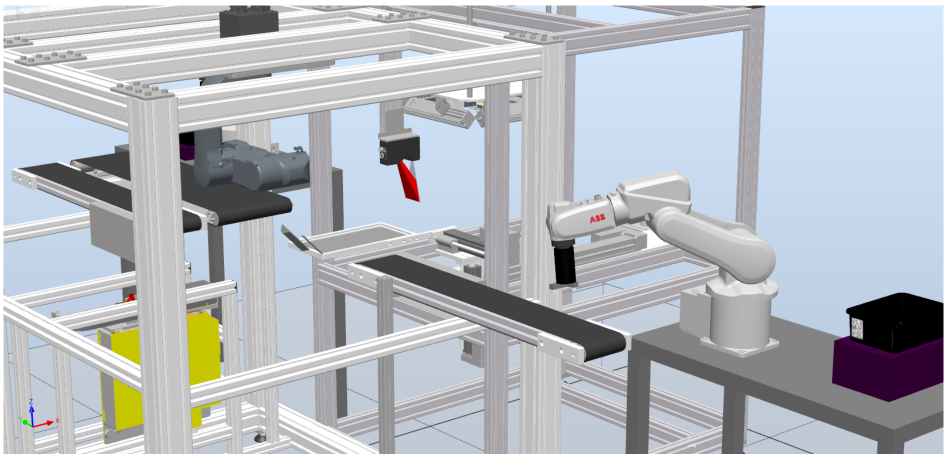


Figure 16: Quality Control Cell in RobotStudio (perspective overview).

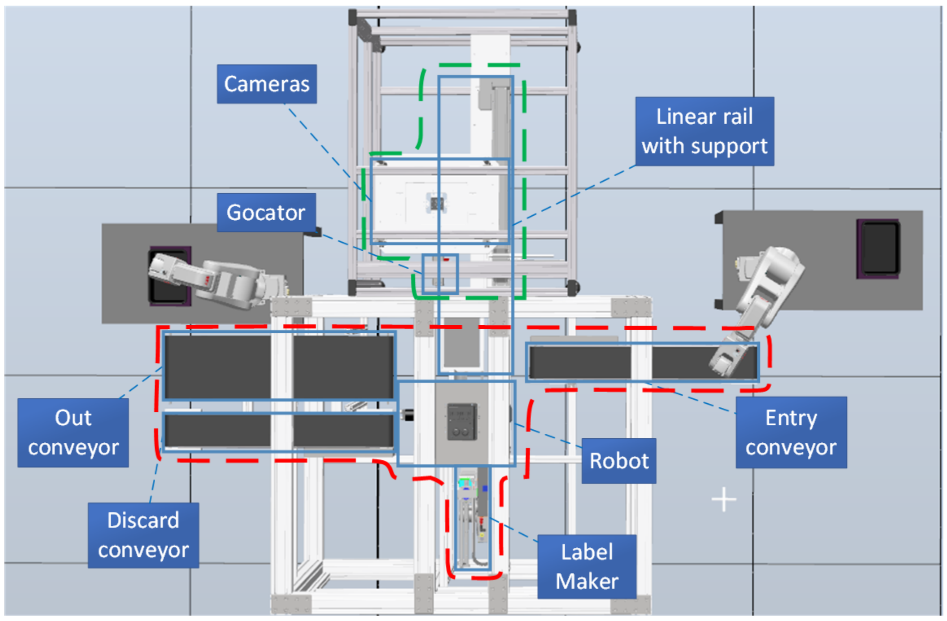


Figure 17: Quality Control Cell in RobotStudio (top view with the identification of each component).

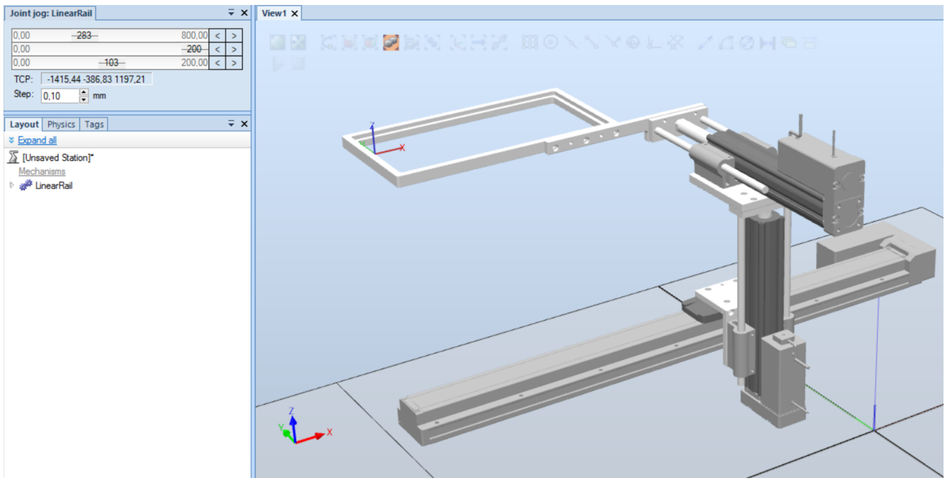
used to define the physical characteristics and behaviour of a robot or a machine, from the definition of its links and joints. Three mechanisms were created in RS: the Epson robot manipulator, the label maker and the linear rail. These mechanisms allow the motion of the devices when the joint values are changed. As an example, Fig. 18 shows the jog joint window for the linear rail mechanism in RS.

To control these mechanisms and the behaviour of the other devices in the cell, SCs were designed in RS Station Logic, using the SmartComponent Editor. Each SC contains the geometry and/or mechanism of the equipment, and implements the device's operating logic. Inputs and outputs were defined for each SC to control the device and to allow them to interact in the cell. Table 2 details the five SCs that were created.

The GripperControl SC implements the gripper's behaviour to attach and detach parts. It has one digital input, to open and close the griper, and one digital output to indicate its status. The RobotControl SC receives the desired joint values and execution time, and there is an input to execute the robot's motion. As the robot is implemented as a Mechanism (RS only includes ABB robots) it is not possible to create targets and paths graphically in RS to control its motion. This SC encapsulates the GripperControl SC, as the gripper is used as the robot's tool, so the GripperControl SC inputs and outputs are replicated in the RobotControl SC. There is also a digital output indicating if the robot is busy (i.e. performing any motion).

The ConveyorControl SC is used to control the behaviour of the three conveyors in the QC station. It simulates two sensors, one at the beggining of the conveyor and one at the end, to identify the presence of parts and move them along path. Every time a part is detected at the end of the conveyor, it is stopped until the part is picked up from the conveyor. If there are no parts in the conveyor, it also stops. The SC has a digital output that indicates the presence of a part waiting to be picked up, and another digital output to indicate if the conveyor is running. The conveyor's speed can be changed in the SC properties.

The LabelMakerControl SC implements the behaviour of applying a label to a part. There are two joint positions stored in the label maker mechanism: one is used when the applicator is retracted, and the other is used when it moves forward to the position to apply a label. The SC has a digital input that triggers the motion between these two positions and back, to simulate the application of the label. It also has an output to indicate if the device is busy.



0,00 283 800,00
0,00 200
0,00 103 200,00
TCP: -1415,44 -386,83 1197,21
Step: 0,10 mm

Figure 18: Mechanism jog joint for the linear rail mechanism.

The VisionSystemControl SC encapsulates the linear rail, the Gocator and the cameras, and it is used to implement and control the behaviour of the QC area as a whole. The SC has two digital inputs, each to trigger one type of analysis. When an analysis is triggered, the linear rail mechanism moves to an already stored joint position (which is different for each analysis), waits for a determined amount of time, and returns to the initial position. The SC has outputs to indicate if a part was approved (true) or not (false). At this stage of development of the DT, the parts are simulated as OK or NOK (i.e. Not OK) through a random process implemented in the SC, in which there is a 10% probability that the part will be considered NOK.

Additionally, the behaviour of the two IRB1200 manipulators was programmed with RAPID, which is a high-level programming language used to control ABB robots. Fig. 19 shows the targets and path that implement the robot's motion to take a part from the input box and place it on the entry conveyor.

## 5.4 QC CELL OPERATION

Fig. 20 shows the flowchart created for determining the operating logic of the QC cell. The process begins when the operator scans a box of parts, with the part type contained in the box being transmitted to the QC cell controller. If the part is a bottom one, it undergoes dimensional analysis using the Gocator. Upon approval, a label is generated and affixed to the part, which then proceeds to the out conveyor for storage.

Table 2: Details and descriptions of each device SC implemented in RS.

| **SmartComponent** | **Properties** | **Input** | **Output** | **Behaviour** |
|---|---|---|---|---|
| GripperControl | Gripper geometry | diClose | doClosed | Attaches and detaches parts to the gripper. |
| RobotControl | Robot Mechanism<br>Jointvalues[] (degrees)<br>Time (s) | diExecMove<br>diOpenGripper<br>diCloseGripper | doRobotBusy<br>doGripperOpened<br>doGripperClosed | Moves the robot mechanism to a certain position, based on the entered joint and time values. |
| ConveyorControl | AddedPart geometry<br>Speed (m/s) | diPartAdded | doPartAvailable<br>doConveyorRunning | Moves the parts added in the beginning of the conveyor geometry to the end. |
| LabelMakerControl | Label maker Mechanism | diApplyTag | doTagApplied<br>doBusy | Moves the label maker applicator to the tagging position and back. |
| VisionSystemControl | Linear rail Mechanism | diExecDA<br>diExecSA | doBusy<br>doDAOK<br>doDANOK<br>doSAOK<br>doSAOK | Moves the linear rail to the designated position for analysis and back, simulating the results. |

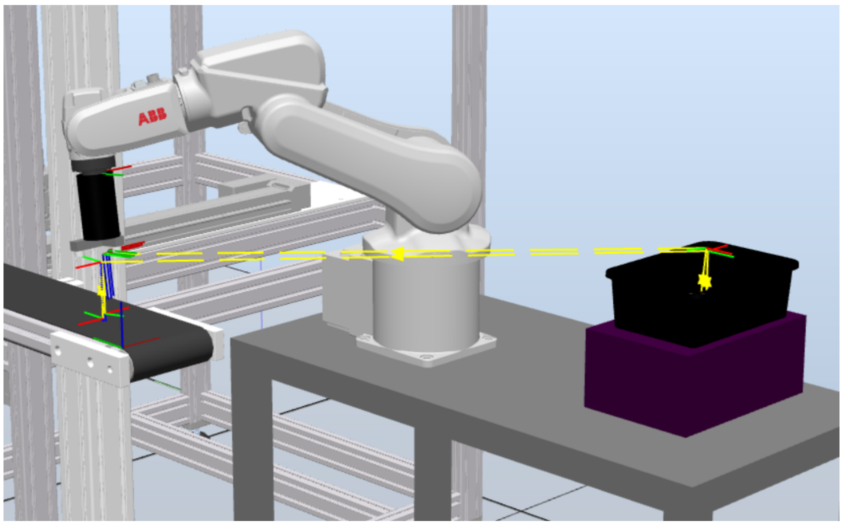


Figure 19: Targets and path for ABB IRB1200.

In the case of a top part, it must undergo both dimensional and surface analysis, requiring the part to be rotated, as these tests are performed on different faces of the part. To aid the process of rotating the part, a 45 degree support was added to the linear rail, designed to allow the robot to place the part on one side and pick it from the other. Only if both tests are OK, will the top part receive a label and exit the QC cell. Any bottom or top part that fails a test is discarded and not labeled. The interaction between all components that implement the cell's operating logic is done through OPC UA and it is detailed in Fig. 21.

### 5.4.1 *Individual devices servers*

To support the DT simulation, four OPC UA servers were created, namely, one for the label maker, one for the robot, one for the entry conveyor, and one for the vision system. These servers run within RS and were implemented using the RS and UA .NET SDKs, as described in Chapter 3.

Tables 3 and 4 present the structure of the servers' address spaces, regarding the variables and methods available in each one, respectively. These structures were defined based on each device's behaviour implementation (described in the previous section) and the cell's operating logic, through the identification of the necessary sensors and controls for the cell, and the determination of which information each equipment should provide.

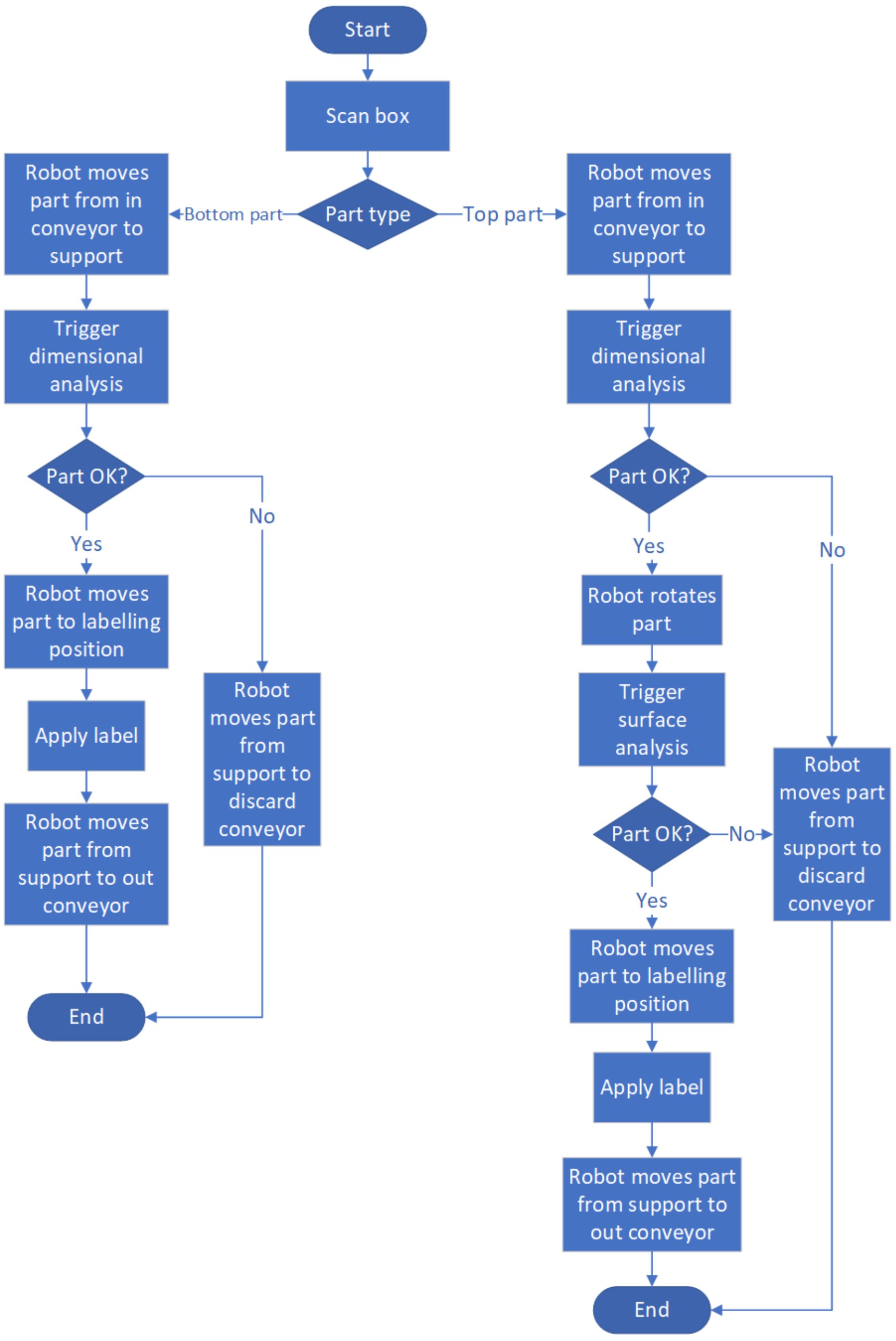

Start
Scan box
Part type
Bottom part
Top part
Robot moves part from in conveyor to support
Trigger dimensional analysis
Part OK?
Yes
No
Robot moves part to labelling position
Apply label
Robot moves part from support to out conveyor
Robot moves part from support to discard conveyor
End
Robot moves part from in conveyor to support
Trigger dimensional analysis
Part OK?
Yes
No
Robot rotates part
Trigger surface analysis
Part OK?
No
Yes
Robot moves part from support to discard conveyor
Robot moves part to labelling position
Apply label
Robot moves part from support to out conveyor
End

Figure 20: Flowchart of the cell operation.

Table 3: OPC UA Variables available in each individual equipment's server.

| **Equipment** | **Variable** | **Type** | **Description** | **Is monitored?** |
|---|---|---|---|---|
| Entry Conveyor | *SensorOutState* | Bool | Indicates if there is a part at the end of the conveyor ready to be picked up. | Yes |
| Robot | *J1 - J6* | Double[] | Joint values. | No |
| | *RobotBusy* | Bool | Indicates if the robot is moving. | Yes |
| Label Maker | *Label* | String | Shows the text of the last printed tag. | No |
| Vision System | *DAresult* | Bool | Indicates the result of the last dimensional analysis performed. | Yes |
| | *SAresult* | Bool | Indicates the result of the last surface analysis performed. | Yes |

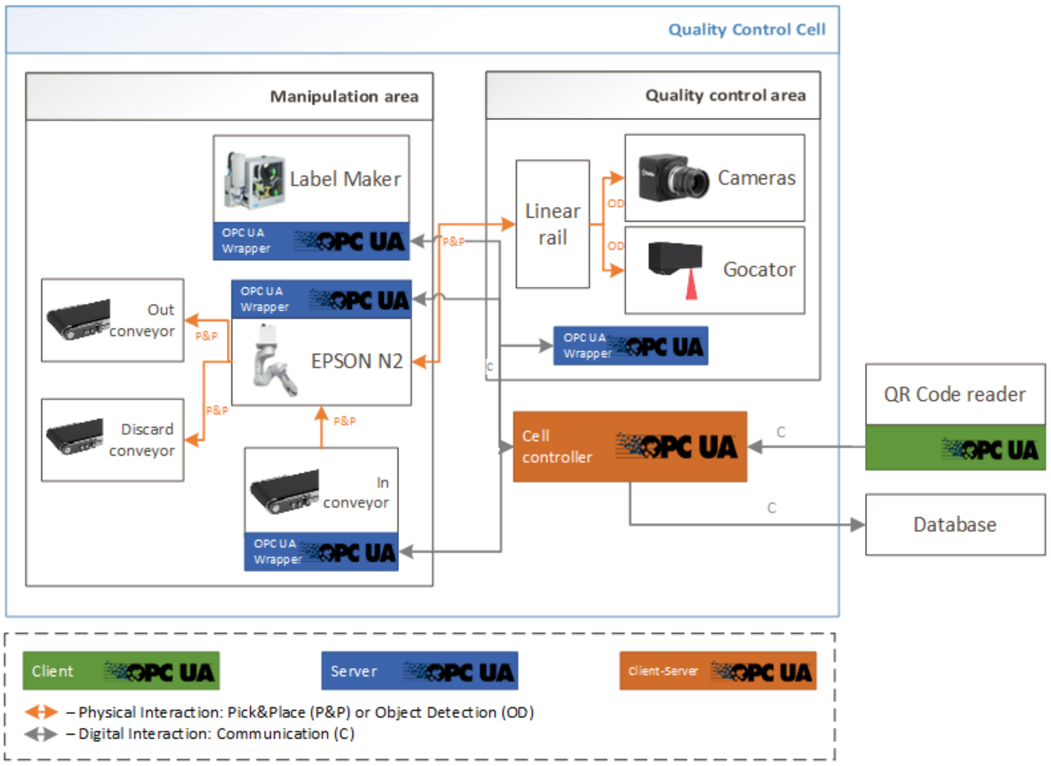


Figure 21: Block diagram of component interaction.

Table 4: OPC UA Methods available in each individual device's server.

| **Equipment** | **Method** | **Input** | **Description** |
|---|---|---|---|
| Entry Conveyor | - | - | - |
| Robot | *MoveMethod* | Double[]: Joint values<br>Double: Speed | Moves the robot to the desired joint values<br>at a certain speed (degrees/s). |
| | *GripperMethod* | Int: 0(open)/1(close) | Opens or closes gripper. |
| Label Maker | *LabelMethod* | String: Label | Triggers label maker to print and apply a label. |
| Vision System | *DAmethod* | String: Part type | Triggers vision system for dimensional |
| | *SAmethod* | String: Part type | analysis. Triggers vision system for<br>surface analysis. |

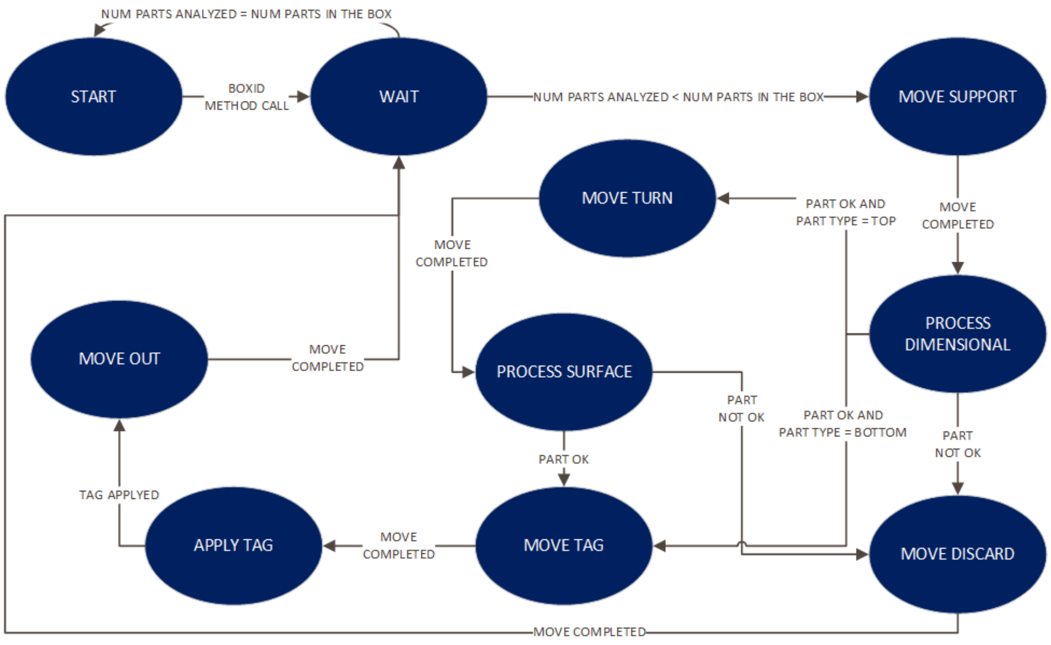


Figure 22: Finite state machine that implements cell's operating logic.

### 5.4.2 *Cell controller*

The cell controller application, developed in C++ using open62541, serves as the central hub for initiating the QC process, and implements the operating logic of the cell, as shown in Fig. 20 in the form of a Finite State Machine (FSM), pictured in Fig. 22. To do that, this application needs the information from the individual device servers so, four OPC UA clients are implemented. The reason one needs each individual client for every server is because, in open62541, once a client connects to an endpoint, it is not possible to switch to another server. The last column in Table 3 indicates which variables are monitored by the cell controller application in each of the servers.

Additionally, the cell controller application implements an OPC UA server for any outside interaction with the cell (by a MES or an ERP, for instance). The server's address space is structured as shown in Table 5. There are four OPC UA variables, which essentially replicate from the individual servers key information about the cell that should be available for an outside operator. This provides a single point of access to the cell where all important information is available, eliminating the need for a user to connect to each individual device server to follow the QC process.

The OPC UA method present in the cell server is used to start the process of QC to a certain box of parts. This method takes as argument the label of the box to be

Table 5: Quality Control cell server's Address Space structure.

| **Type** | **Name** | **Description** |
|---|---|---|
| Variable | DAResult | Indicates the result of the last dimensional analysis performed. |
| | SAResult | Indicates the result of the last surface analysis performed. |
| | ProcessedBoxLabel | Shows the label of the last box of parts that entered the cell. |
| | PrintedLabel | Shows the last printed label in the label maker. |
| Method | Box ID | Receives the box label as an input argument and iniciates the QC process. |

analyzed, which consists of a hyphenated string indicating the box's id, the type of part inside, and the number of parts. This information is saved in the memory of the application, as it is crucial for the operation of the developed FSM.

As the MoveMethod in the robot's server receives as argument a vector with the joint values and a speed value (which will be converted to a time value to enter the RobotControl SC, based on the maximum distance between the desired and current joint values), each MOVE state in the FSM is associated with a list of joint values vectors that describes the path that the robot needs to perform. Usually, when working in RS with ABB manipulators, these paths are described by a set of targets that indicate position, orientation and configuration, and the robot controller performs the kinematics to control the robot's motion. However, the robot manipulator in this cell was implemented in RS as a Mechanism, and it does not have an ABB virtual robot controller to perform its kinematics calculations. Therefore the construction of each path was manually done in RoboDK[1], which is a software for industrial robot simulation and programming that has a library of robots from multiple manufacturers. The main targets for each path were created in RS, where all the cell's geometry is represented, and then replicated in RoboDK, where some intermediate targets were added and the paths were created and tested. All the targets that formed a path were then converted to joint values vectors to compose the lists of targets in each MOVE state. Fig. 23 shows the targets created in RS and RoboDK.

1 https://robodk.com/index

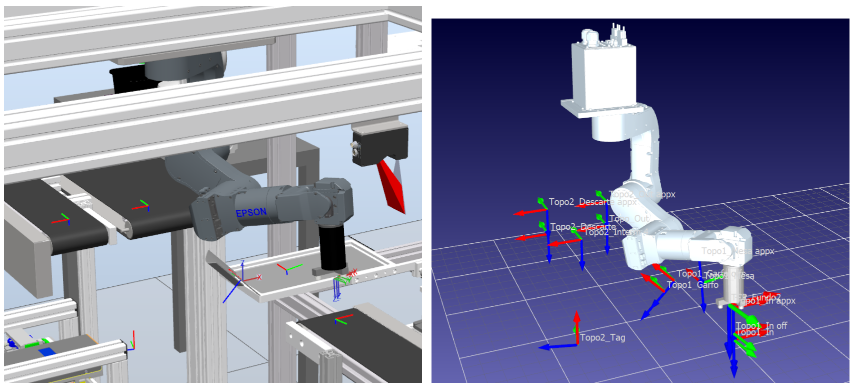


Figure 23: Targets in RS (left) and RoboDK (right).

Finally, the cell controller application implements a connection to a MySQL database, to enable the historical access of the four available variables. This HDA implementation was detailed in Chapter 3. Additionally, this connection to the database also allows for the storage of the process data in a separate table, which concatenates the results of the analysis for each part in a readable manner, and make it available to be accessed by other processes within the factory.

### 5.4.3 *QR Code client*

To interact with the cell server and send the trigger for the start of the QC process, an OPC UA client application was created using *open62541* . This program uses OpenCV[2] to, using a camera, identify a QR code representing the label of the box to be processed by the cell, and transmit this information through OPC UA to the controller. As detailed before, the box label is a hyphenated string, and this application, upon identifying the QR code, calls the BoxId method in the cell server parsing this string as an argument. This client application was developed to be easily integrated in the QC process as an HMI tool and allow operators to quickly interact with the cell.

2 https://www.opencv.org/

# 6 TESTS AND RESULTS

This chapter presents several tests and results obtained from the developments described in this report.

## 6.1 INTELITEK PALLET CONVEYOR OPC UA SERVER

As described in Chapter 4, the development of the OPC UA server for the pallet conveyor was entirely made using the virtual conveyor running as a SC in RS. After updating the conveyor's SC, its behaviour is identical to the real conveyor, including the RS232 communication, and it can be considered as a DT of the real device.

To connect the virtual conveyor to the ESP32 running the server, a USB-RS232 converter is used in the computer running the RS simulation, attached to a serial cable with DB9 connectors. In the other end of the cable, the DB9 connector was removed to expose the wires from TX, RX, and GND, so they could be connected in the RS232-UART interface using the MAX3232 circuit in the breadboard. The circuit is then connected to the ESP32, as shown in Fig. 24 (following the schematics shown in 12).

Once the simulation is running in RS, and the server is running on ESP32, it is possible to successfully connect to the server using UaExpert to monitor and control the virtual conveyor. Fig. 25 shows the variable holding the strings sent by the conveyor being monitored. Fig. 26 shows a method call in which the user sent a string to stop the transporter number one at the first station. The transporter will be stopped at the station until another string is sent to let it go.

The results obtained are an example of the power of the DT technology, as it enables the development of software and hardware applications for industrial devices, even if the physical asset is inaccessible. It is also important to note the use of ESP32 to implement the integration of a legacy device in an Industry 4.0 environment through the use of OPC UA.

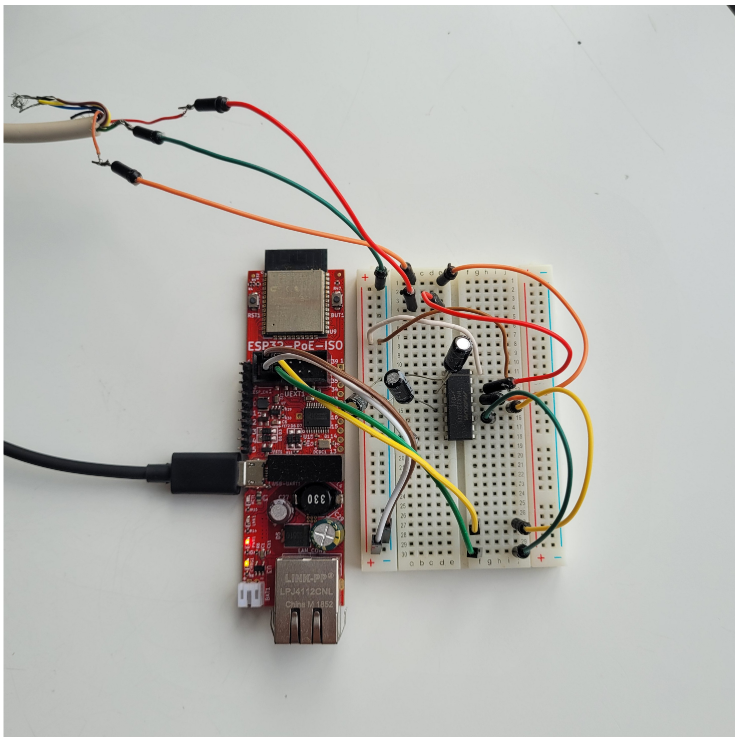
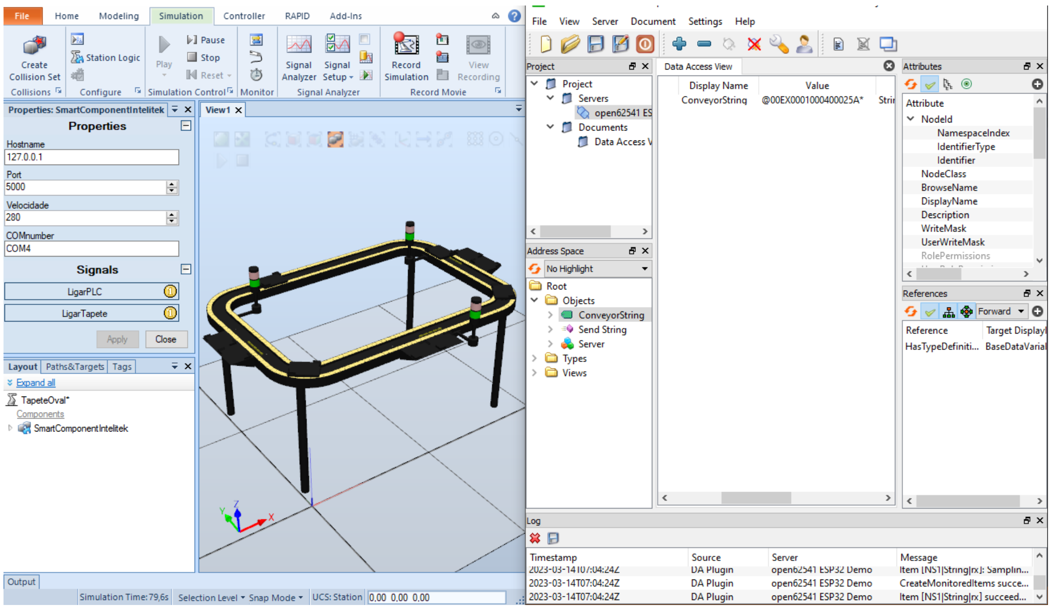


Figure 24: ESP32 connected to RS232-UART circuit.



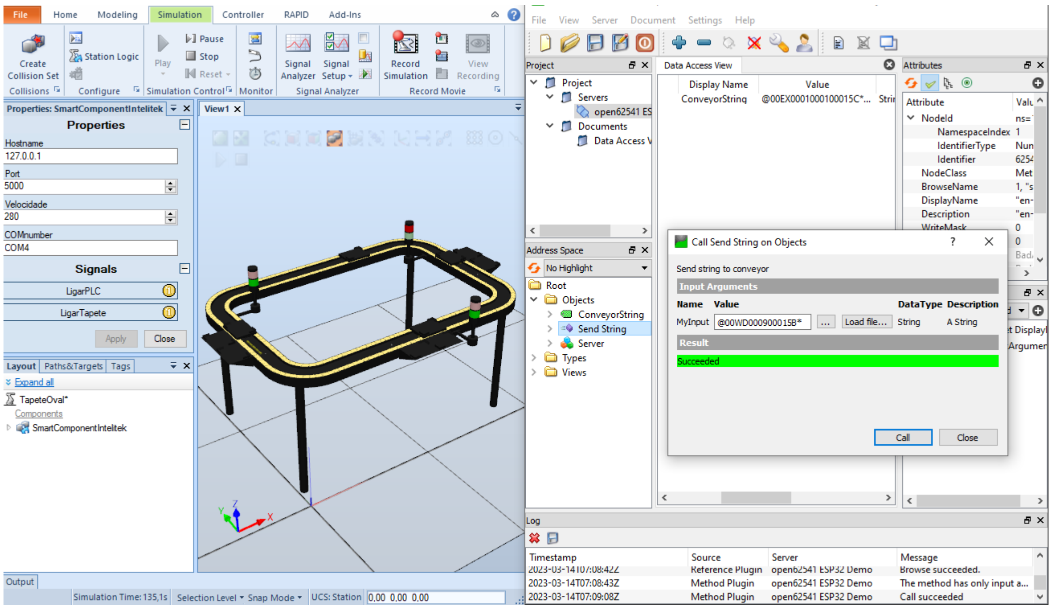
Properties: SmartComponentIntelitek

Hostname: 127.0.0.1
Port: 5000
Velocidade: 280
COMnumber: COM4

| Display Name | Value |
|---|---|
| ConveyorString | @00EX0001000400025A* |

```
Timestamp             Source     Server                Message
2023-03-14T07:04:24Z  DA Plugin  open62541 ESP32 Demo  Item [NS1|String|rx]: Samplin...
2023-03-14T07:04:24Z  DA Plugin  open62541 ESP32 Demo  CreateMonitoredItems succe...
2023-03-14T07:04:24Z  DA Plugin  open62541 ESP32 Demo  Item [NS1|String|rx] succeed...
```

Figure 25: Variable in ESP32 server showing the strings sent by the conveyor.



Properties: SmartComponentIntelitek

Properties

Hostname: 127.0.0.1
Port: 5000
Velocidade: 280
COMnumber: COM4

Signals

LigarPLC
LigarTapete

TapeteOval*
Components
SmartComponentIntelitek

Simulation Time: 135,1s

| Display Name | Value |
|---|---|
| ConveyorString | @00EX0001000100015C*... |

Call Send String on Objects

Send string to conveyor

Input Arguments

| Name | Value | DataType | Description |
|---|---|---|---|
| MyInput | @00WD000900015B* | String | A String |

Result

Succeeded

```
Timestamp             Source            Server                   Message
2023-03-14T07:08:42Z  Reference Plugin  open62541 ESP32 Demo     Browse succeeded.
2023-03-14T07:08:43Z  Method Plugin     open62541 ESP32 Demo     The method has only input a...
2023-03-14T07:09:08Z  Method Plugin     open62541 ESP32 Demo     Call succeeded
```

Figure 26: Method in the ESP32 server to send a string to the conveyor.

## 6.2 DIGITAL TWIN FOR A QUALITY CONTROL CELL

Many tests were executed in the DT to evaluate the performance of the RS simulation, the success of the communication between the cell controller application and the individual servers running in RS, the implementation of the cell's operating logic, the QC cell's historical data access, and the efficiency of the QR Code reader OPC UA client implementation to trigger the process. The steps to run the DT are:

- Run the RS simulation.
- Run the cell controller application.
- Run the QR code client application, and scan the QR code in the box, as in Fig. 27.

When all these steps run successfully, the DT functions as expected. A video of the simulation is available at https://www.youtube.com/watch?v=_E5botn8A_I. It demonstrates an effective and seamless OPC UA communication between all devices and the cell controller, which is successful in performing the proposed operating logic for the cell, with the FSM-based implementation. The highest processing time achieved for a single part was 72 seconds, fulfilling the project requirements. The QR code client application is also able to successfully identify the text and call the method in the cell's server that triggers the start of the process.

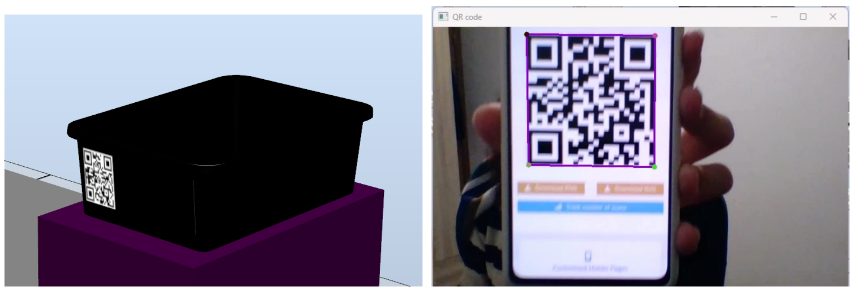
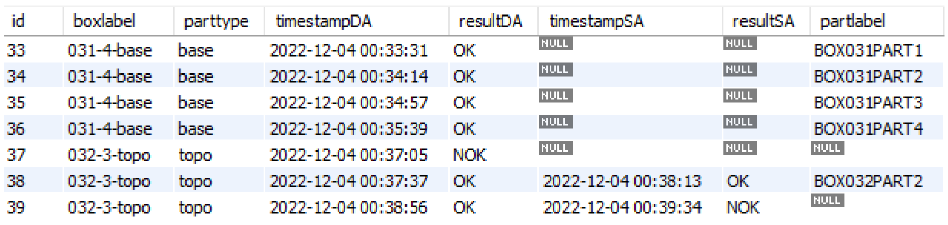


Figure 27: Box with QR code pictured in RS (left) and test performed in QR code client (right).

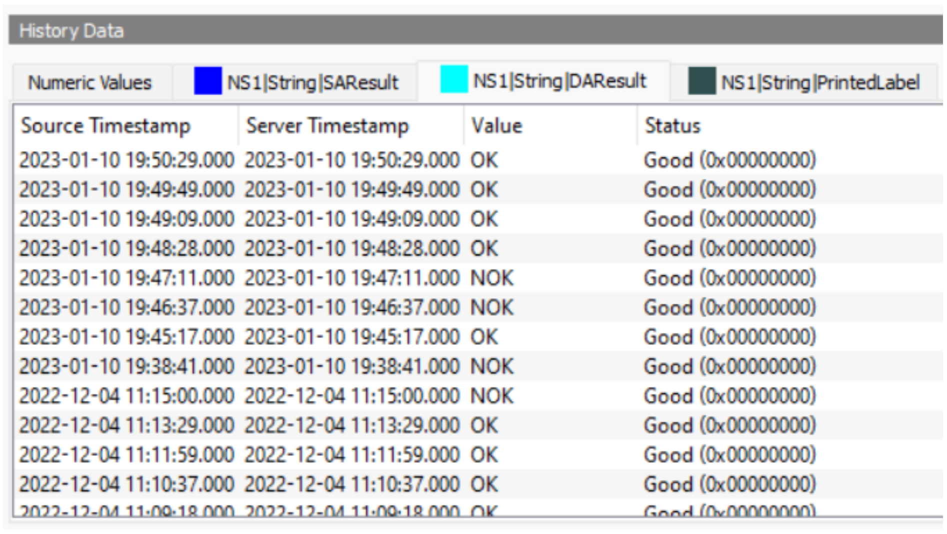


| id | boxlabel | parttype | timestampDA | resultDA | timestampSA | resultSA | partlabel |
|---|---|---|---|---|---|---|---|
| 33 | 031-4-base | base | 2022-12-04 00:33:31 | OK | NULL | NULL | BOX031PART1 |
| 34 | 031-4-base | base | 2022-12-04 00:34:14 | OK | NULL | NULL | BOX031PART2 |
| 35 | 031-4-base | base | 2022-12-04 00:34:57 | OK | NULL | NULL | BOX031PART3 |
| 36 | 031-4-base | base | 2022-12-04 00:35:39 | OK | NULL | NULL | BOX031PART4 |
| 37 | 032-3-topo | topo | 2022-12-04 00:37:05 | NOK | NULL | NULL | NULL |
| 38 | 032-3-topo | topo | 2022-12-04 00:37:37 | OK | 2022-12-04 00:38:13 | OK | BOX032PART2 |
| 39 | 032-3-topo | topo | 2022-12-04 00:38:56 | OK | 2022-12-04 00:39:34 | NOK | NULL |

Figure 28: Results stored in the database for two runs.

Fig. 28 shows the results of the process data stored in the database, resulting from two sequential runs of the DT simulation. First, a box of four bottom components underwent analysis, and all parts were approved and tagged with the labels shown in the last column of the DB table. Next, a box of three top parts was analyzed. Only one part passed both dimensional and surface analysis, while another was discarded immediately after being processed by the Gocator. The last part was approved in terms of dimensions, but failed the surface control. The database also records the label of the box and the analysis' timestamps.

Finally, Fig. 29 shows the result of a request for historical data access of one of the variables available in the QC cell's server. The request was made using UaExpert, and it is attached to a time frame that the server will use as a parameter to retrieve data from the database.



| Source Timestamp | Server Timestamp | Value | Status |
|---|---|---|---|
| 2023-01-10 19:50:29.000 | 2023-01-10 19:50:29.000 | OK | Good (0x00000000) |
| 2023-01-10 19:49:49.000 | 2023-01-10 19:49:49.000 | OK | Good (0x00000000) |
| 2023-01-10 19:49:09.000 | 2023-01-10 19:49:09.000 | OK | Good (0x00000000) |
| 2023-01-10 19:48:28.000 | 2023-01-10 19:48:28.000 | OK | Good (0x00000000) |
| 2023-01-10 19:47:11.000 | 2023-01-10 19:47:11.000 | NOK | Good (0x00000000) |
| 2023-01-10 19:46:37.000 | 2023-01-10 19:46:37.000 | NOK | Good (0x00000000) |
| 2023-01-10 19:45:17.000 | 2023-01-10 19:45:17.000 | OK | Good (0x00000000) |
| 2023-01-10 19:38:41.000 | 2023-01-10 19:38:41.000 | NOK | Good (0x00000000) |
| 2022-12-04 11:15:00.000 | 2022-12-04 11:15:00.000 | NOK | Good (0x00000000) |
| 2022-12-04 11:13:29.000 | 2022-12-04 11:13:29.000 | OK | Good (0x00000000) |
| 2022-12-04 11:11:59.000 | 2022-12-04 11:11:59.000 | OK | Good (0x00000000) |
| 2022-12-04 11:10:37.000 | 2022-12-04 11:10:37.000 | OK | Good (0x00000000) |
| 2022-12-04 11:09:18.000 | 2022-12-04 11:09:18.000 | OK | Good (0x00000000) |

Figure 29: Result of a historical data access request in the cell's server using UaExpert.

# 7 CONCLUSION AND FUTURE WORK

This report first presented the development of four generic and reusable applications using OPC UA: a SmartComponent that implements an OPC UA server for a simulated device in RS, an OPC UA server application to run in an ESP32, the implementation of a HistoryDataBackend to allow the use of OPC UA HDA using MySQL, and a single C++ application capable of running both an OPC UA server and a client. All these implementations can be useful in the development of more complex applications involving OPC UA, RobotStudio, embedded systems, and database servers.

These building blocks were essential for the development of the remainder of the work detailed in this report, as they require minimal changes to customize and replicate, allowing for a shorter development time. Moreover, the use of different SDKs and platforms shows the flexibility and scalability of the OPC UA standard.

The report also described the development of an OPC UA server in an embedded system to elevate a legacy device to an Industry 4.0 level, enabling it to communicate seamlessly with other devices and systems in an industrial context using an Ethernet-based network. This development was entirely done using the DT of the device, which shows how the DT technology can be used to facilitate and speed up development. By using the DT, it was easier to perform tests and validate the developed work.

Future work for this development should be focused on updating the server's address space to improve the interaction of the user with the pallet conveyor, changing the string commands for a more intuitive and accessible logic. It will also be useful to design a PCB (Printed Circuit Board) to take the circuit with the MAX3232 out of the breadboard, and create a case for the ESP32 to make the hardware more resilient and usable in an industrial setting. Furthermore, the hardware should be tested and deployed in the real pallet conveyor.

Finally, the report has presented the development of a Digital Twin for a quality control cell using RobotStudio, OPC UA and MySQL. The use of RS in this development provides a realist virtual environment for the cell, allowing for an accurate simulation of the QC process. The use of OPC UA for the communication between individual devices and the cell controller allows for a standardized centralized point of access for the operator to interact with the cell, while keeping less relevant information still available in the individual servers. By connecting the cell to a RDBMS like MySQL, the results of the quality control process are registered in a straightforward and readable manner, and become available to be accessed by other processes within the factory. The connection to the database also makes it possible for the cell's server to offer HDA to the variables available in the address space. The use of the OPC UA servers for the DTs, offering the same address space, variables and methods as the physical system, allows for easier interchange use of the DT with the real system.

The tests performed in the DT showed the success of this development regarding the cell's operation and control logic and the OPC UA communication, complying with the project's requirements. Additionally, the client application with the QR Code reader can be easily integrated in the process as an HMI tool. Future work of this development will be focused on connecting the DT with the real QC cell at the shop floor, and updating the Vision System server to hold detailed information about the measurements done by the laser profiler and the cameras, beyond the true or false results. Additionally, the database should be updated to store these detailed results.

Overall, all the OPC UA server applications detailed in this work can be improved by the use of OPC UA security mechanisms and Companion Specifications. Secure communication is an important aspect of OPC UA server applications, as it ensures the confidentiality, integrity, and the authenticity of the data being transferred over the network. OPC UA provides different security mechanisms, such as encryption, user authentication, and authorization, to protect the communication between clients and servers.

The CS, on the other hand, provide a standard way of describing the information models of devices and systems for different fields of the industry. In this work, the OPC UA servers for the robot manipulator, and the vision system in the QC cell' DT, should be updated to follow existing Robotics and Vision Systems CS. This would be an additional step towards the system's interoperability in the future.

# BIBLIOGRAPHY

Abdelsattar, Ahmad, Edward J. Park, and Amr Marzouk (July 2022). “An OPC UA Client/Gateway-Based Digital Twin Architecture of a SCADA System with Embedded System Connections”. In: *2022 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. ISSN: 2159-6255, pp. 798–803. DOI: 10.1109/AIM52237.2022.9863367.

Andrei, G. et al. (2020). “Industrial Messaging Middleware: Standards and Performance Evaluation”. In: *2020 IEEE 14th International Conference on Application of Information and Communication Technologies (AICT)*, pp. 1–6. DOI: 10.1109/AICT50176.2020.9368846.

Babic, Milica, Mojtaba A. Farahani, and Thorsten Wuest (Jan. 2021). “Image Based Quality Inspection in Smart Manufacturing Systems: A Literature Review”. en. In: *Procedia CIRP*. 9th CIRP Global Web Conference – Sustainable, resilient, and agile manufacturing and service operations : Lessons from COVID-19 103, pp. 262–267. ISSN: 2212-8271. DOI: 10.1016/j.procir.2021.10.042. URL: https://www.sciencedirect.com/science/article/pii/S2212827121008830 (visited on 01/29/2023).

Balador, A., N. Ericsson, and Z. Bakhshi (2017). “Communication middleware technologies for industrial distributed control systems: A literature review”. In: *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1–6. DOI: 10.1109/ETFA.2017.8247730.

Carlsson, Thomas (May 2022). *Industrial networks keep growing despite challenging times*. URL: https://www.hms-networks.com/news-and-insights/news-from-hms/2022/05/02/industrial-networks-keep-growing-despite-challenging-times (visited on 03/28/2023).

Céspedes Cubides, Andres Sebastian and Eduardo Barrera Gualdrón (Nov. 2020). “Implementación de SCADA a través del protocolo MQTT”. In: *2020 IX International Congress of Mechatronics Engineering and Automation (CIIMA)*, pp. 1–5. DOI: 10.1109/CIIMA50553.2020.9290302.

Ch, G. D. Salazar et al. (Nov. 2018). “Open Middleware proposal for IoT focused on Industry 4.0”. In: *2018 IEEE 2nd Colombian Conference on Robotics and Automation (CCRA)*, pp. 1–6. DOI: 10.1109/CCRA.2018.8588117.

Friedl, Sebastian et al. (Apr. 2020). “Generation of OPC UA Companion Specification with Eclipse Modeling Framework”. In: *2020 16th IEEE International Conference on Factory Communication Systems (WFCS)*, pp. 1–7. DOI: 10.1109/WFCS47810.2020.9114448.

Fuller, Aidan et al. (2020). “Digital Twin: Enabling Technologies, Challenges and Open Research”. In: *IEEE Access* 8. Conference Name: IEEE Access, pp. 108952–108971. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2998358.

Gatial, Emil, Zoltán Balogh, and Ladislav Hluchý (July 2020). “Concept of Energy Efficient ESP32 Chip for Industrial Wireless Sensor Network”. In: *2020 IEEE 24th International Conference on Intelligent Engineering Systems (INES)*. ISSN: 1543-9259, pp. 179–184. DOI: 10.1109/INES49302.2020.9147189.

Glaessgen, Edward and David Stargel (Apr. 2012). “The digital twin paradigm for future NASA and U.S. air force vehicles”. In: ISBN: 978-1-60086-937-2. DOI: 10.2514/6.2012-1818.

Hermann, Mario, Tobias Pentek, and Boris Otto (Jan. 2016). “Design Principles for Industrie 4.0 Scenarios”. In: *2016 49th Hawaii International Conference on System Sciences (HICSS)*. ISSN: 1530-1605, pp. 3928–3937. DOI: 10.1109/HICSS.2016.488.

Karaagac, Abdulkadir, Niels Verbeeck, and Jeroen Hoebeke (2019). “The Integration of LwM2M and OPC UA: An Interoperability Approach for Industrial IoT”. In: *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, pp. 313–318. DOI: 10.1109/WF-IoT.2019.8767209.

keophilavong, T., Widyawan, and M. N. Rizal (July 2019). “Data Transmission in Machine to Machine Communication Protocols for Internet of Things Application: A Review”. In: *2019 International Conference on Information and Communications Technology (ICOIACT)*, pp. 899–904. DOI: 10.1109/ICOIACT46704.2019.8938420.

Lam, An Ngoc and Øystein Haugen (Oct. 2019). “Implementing OPC-UA services for Industrial Cyber-Physical Systems in Service-Oriented Architecture”. In: *IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society*. Vol. 1. ISSN: 2577-1647, pp. 5486–5492. DOI: 10.1109/IECON.2019.8926972.

Lehnhoff, Sebastian et al. (June 2012). “OPC Unified Architecture: A Service-oriented Architecture for Smart Grids”. In: *2012 First International Workshop on Software Engineering Challenges for the Smart Grid (SE-SmartGrids)*, pp. 1–7. DOI: 10.1109/SE4SG.2012.6225723.

Leiba, Barry (Jan. 2008). “An Introduction to Internet Standards”. In: *IEEE Internet Computing* 12.1. Conference Name: IEEE Internet Computing, pp. 71–74. ISSN: 1941-0131. DOI: 10.1109/MIC.2008.2.

Li, Xiaomin et al. (Jan. 2017). “A review of industrial wireless networks in the context of Industry 4.0”. en. In: *Wireless Networks* 23.1, pp. 23–41. ISSN: 1572-8196. DOI: 10.1007/s11276-015-1133-7. URL: https://doi.org/10.1007/s11276-015-1133-7 (visited on 04/14/2021).

Liu, Mengnan et al. (Jan. 2021). “Review of digital twin about concepts, technologies, and industrial applications”. en. In: *Journal of Manufacturing Systems*. Digital Twin towards Smart Manufacturing and Industry 4.0 58, pp. 346–361. ISSN: 0278-6125. DOI: 10.1016/j.jmsy.2020.06.017. URL: https://www.sciencedirect.com/science/article/pii/S0278612520301072 (visited on 02/02/2023).

Lu, Yuqian et al. (Feb. 2020). “Digital Twin-driven smart manufacturing: Connotation, reference model, applications and research issues”. en. In: *Robotics and Computer-Integrated Manufacturing* 61, p. 101837. ISSN: 0736-5845. DOI: 10.1016/j.rcim.2019.101837. URL: https://www.sciencedirect.com/science/article/pii/S0736584519302480 (visited on 02/02/2023).

Martins, André, Hugo Costelha, and Carlos Neves (Apr. 2019). “Shop Floor Virtualization and Industry 4.0”. In: *2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pp. 1–6. DOI: 10.1109/ICARSC.2019.8733657.

— (Apr. 2020). “Supporting the Design, Commissioning and Supervision of Smart Factory Components through their Digital Twin”. In: *2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pp. 114–119. DOI: 10.1109/ICARSC49921.2020.9096072.

Mathias, Selvine G., Sebastian Schmied, and Daniel Grossmann (Jan. 2020). “An Investigation on Database Connections in OPC UA Applications”. en. In: *Procedia Computer Science*. The 11th International Conference on Ambient Systems, Networks and Technologies (ANT) / The 3rd International Conference on Emerging Data and Industry 4.0 (EDI40) / Affiliated Workshops 170, pp. 602–609. ISSN: 1877-0509. DOI: 10.1016/j.procs.2020.03.132. URL: https://www.sciencedirect.com/science/article/pii/S1877050920305706 (visited on 03/18/2023).

Perzylo, Alexander et al. (Sept. 2019). “OPC UA NodeSet Ontologies as a Pillar of Representing Semantic Digital Twins of Manufacturing Resources”. In: *2019 24th IEEE International Conference on Emerging Technologies and Factory*

*Automation (ETFA)*. ISSN: 1946-0759, pp. 1085–1092. DOI: 10.1109/ETFA.2019.8868954.

Profanter, Stefan et al. (Feb. 2019). "OPC UA versus ROS, DDS, and MQTT: Performance Evaluation of Industry 4.0 Protocols". In: *2019 IEEE International Conference on Industrial Technology (ICIT)*. ISSN: 2643-2978, pp. 955–962. DOI: 10.1109/ICIT.2019.8755050.

Protic, A. et al. (Dec. 2020). "Implementation of a Bi-Directional Digital Twin for Industry 4 Labs in Academia: A Solution Based on OPC UA". In: *2020 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pp. 979–983. DOI: 10.1109/IEEM45057.2020.9309953.

Rojko, Andreja (July 2017). "Industry 4.0 Concept: Background and Overview". en. In: *International Journal of Interactive Mobile Technologies (iJIM)* 11.5, pp. 77–90. ISSN: 1865-7923. URL: https://online-journals.org/index.php/i-jim/article/view/7072 (visited on 04/12/2021).

Schleipen, Miriam et al. (Jan. 2016). "OPC UA & Industrie 4.0 - Enabling Technology with High Diversity and Variability". en. In: *Procedia CIRP*. Factories of the Future in the digital environment - Proceedings of the 49th CIRP Conference on Manufacturing Systems 57, pp. 315–320. ISSN: 2212-8271. DOI: 10.1016/j.procir.2016.11.055. URL: https://www.sciencedirect.com/science/article/pii/S2212827116312094 (visited on 03/17/2023).

Souza, Vinicius et al. (Jan. 2019). "A Digital Twin Architecture Based on the Industrial Internet of Things Technologies". In: *2019 IEEE International Conference on Consumer Electronics (ICCE)*. ISSN: 2158-4001, pp. 1–2. DOI: 10.1109/ICCE.2019.8662081.

Tao, Fei, Bin Xiao, et al. (July 2022). "Digital twin modeling". en. In: *Journal of Manufacturing Systems* 64, pp. 372–389. ISSN: 0278-6125. DOI: 10.1016/j.jmsy.2022.06.015. URL: https://www.sciencedirect.com/science/article/pii/S0278612522001108 (visited on 01/29/2023).

Tao, Fei, He Zhang, et al. (Apr. 2019). "Digital Twin in Industry: State-of-the-Art". In: *IEEE Transactions on Industrial Informatics* 15.4. Conference Name: IEEE Transactions on Industrial Informatics, pp. 2405–2415. ISSN: 1941-0050. DOI: 10.1109/TII.2018.2873186.

Tao, Fei and Meng Zhang (2017). "Digital Twin Shop-Floor: A New Shop-Floor Paradigm Towards Smart Manufacturing". In: *IEEE Access* 5. Conference Name: IEEE Access, pp. 20418–20427. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2017.2756069.

Tran, D. L., T. Yu, and M. Riedl (2020). "Integration of IIoT Communication Protocols in Distributed Control Applications". In: *IECON 2020 The 46th*

*Annual Conference of the IEEE Industrial Electronics Society*, pp. 2201–2206. DOI: 10.1109/IECON43393.2020.9254220.

Vitturi, S., C. Zunino, and T. Sauter (June 2019). “Industrial Communication Systems and Their Future Challenges: Next-Generation Ethernet, IIoT, and 5G”. In: *Proceedings of the IEEE* 107.6, pp. 944–961. ISSN: 1558-2256. DOI: 10.1109/JPROC.2019.2913443.

Wilamowski, Bogdan M. and J. David Irwin (Apr. 2016). *Industrial Communication Systems*. en. Google-Books-ID: gJbLBQAAQBAJ. CRC Press. ISBN: 978-1-4398-0282-3.

Willner, Alexander et al. (2017). “Semantic communication between components for smart factories based on oneM2M”. In: *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1–8. DOI: 10.1109/ETFA.2017.8247690.

Wollschlaeger, Martin, Thilo Sauter, and Juergen Jasperneite (Mar. 2017). “The Future of Industrial Communication: Automation Networks in the Era of the Internet of Things and Industry 4.0”. In: *IEEE Industrial Electronics Magazine* 11.1. Conference Name: IEEE Industrial Electronics Magazine, pp. 17–27. ISSN: 1941-0115. DOI: 10.1109/MIE.2017.2649104.

Zezulka, F. et al. (Jan. 2019). “Time-Sensitive Networking as the Communication Future of Industry 4.0”. en. In: *IFAC-PapersOnLine*. 16th IFAC Conference on Programmable Devices and Embedded Systems PDES 2019 52.27, pp. 133–138. ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2019.12.745. URL: https://www.sciencedirect.com/science/article/pii/S2405896319326941 (visited on 04/14/2021).

Zhang, Caiming et al. (June 2021). “Industry 4.0 and its Implementation: a Review”. en. In: *Information Systems Frontiers*. ISSN: 1572-9419. DOI: 10.1007/s10796-021-10153-5. URL: https://doi.org/10.1007/s10796-021-10153-5 (visited on 03/05/2023).