UNIVERSIDAD DE LAS AMÉRICAS PUEBLA

School of Engineering

Department of Computing, Electronics and Mechatronics

OPC UA Standard for IIoT in Industry and Remote Education

Thesis that, for completing the requirements of the Program of Honors is presented by the student

Luis Gerardo Carvajal Fernández

156069

Bachelor of Mechatronics Engineering

Director: PhD. José Luis Vázquez González

San Andrés Cholula, Puebla. Spring 2021

Signature Sheet

Thesis that, for completing the requirements of the Program of Honors is presented by the student Luis Gerardo Carvajal Fernández with ID 156069.

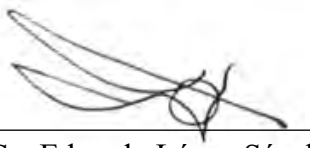
Director of Thesis

PhD. José Luis Vázquez González

President of Thesis

PhD. Gibran Etcheverry Doger

Secretary of Thesis



M. Sc. Eduardo López Sánchez

Acknowledgement

To my mother, Natalia, for her support to follow my path and pursue my dreams, and for always giving me hope. To my father, Edgar, for teaching me perseverance and that there are many ways to reach our destinations. Thank you both for your love and kindliness throughout my life.

To my extended family, for always being there for me and for giving me their encouragement, caring, and company. To my friends, who have been there with me in the toughest times and with whom I share my happiest memories. Thank you for your motivation and life lessons.

To Dr. José Luis Vázquez, for motivating me throughout the career with his lectures and advice, and for being my mentor for this thesis sharing his insight and support. To Dr. Gibran Etcheverry, for his trust and understanding as my tutor and for encouraging me to take steps forward towards my growth. To Dr. Ofelia Cervantes, for sharing her valuable perspectives with all of her students, including myself, and for giving me the opportunity to expand my boundaries to another country. To M.Sc. Eduardo López, for always searching for ways to help students and to innovate their learning experience. To Dr. Pedro Bañuelos, for reminding us of the purpose and passion behind everything we learn.

Abstract

The Fourth Industrial Revolution (also known as Industry 4.0) has the Industrial Internet of Things (IIoT) as one of its principal trends. It offers connectivity and access to information for industrial devices. However, the heterogeneity of communication protocols set by different vendors and the lack of TCP/IP readiness prevent several devices from implementing IIoT. The OPC UA standard can offer a solution due to its client-server communication model and its information architecture, which are-platform independent. By implementing it in industrial devices, the standard can bridge the gap between protocols and help achieve IIoT. The main potential benefits are monitoring and control for industrial processes and remote access to laboratories for education. The goal of the current work is to use OPC UA to connect a Programmable Logic Controller (PLC) and a Virtual Instrument (VI) running as a Web Services application. It uses a SCADA system as a case study; it consists of a bottle production line simulated in FluidSIM. S7-PLCSIM is used to simulate the PLC that controls the system, and LabVIEW to create a VI and export it as an application. NI OPC Servers software is then employed to run an OPC UA server and define the PLC as its client. The results show a successful connection and demonstrate that the OPC UA standard can be used for implementing IIoT by connecting a simulated PLC to a VI for bidirectional information exchange in monitoring and control tasks.

Keywords: Industrial Internet of Things (IIoT), OPC UA, Programmable Logic Controller (PLC), Virtual Instrument (VI).

Index

1. Introduction

The world is experiencing a transition towards the Fourth Industrial Revolution, also known as Industry 4.0. As technologies evolve, the global panorama shows a clear tendency of a digital transformation for society. The Internet is the most evident demonstration of this, and it has served in the last decades to close the gap between devices. Its impact is undeniable: it provides near-instantaneous connectivity and data exchange worldwide.

Now, one of the leading trends of Industry 4.0 promises to use the Internet as a way to communicate remotely with industrial machinery. That trend is the Industrial Internet of Things (IIoT), the industrial version of the Internet of Things (IoT). It allows sensors, controllers, and even entire machines in production processes to share data via the Internet. The data includes relevant information about the state of the devices and their production, which can be used for analysis, monitoring, and control.

However, there is a challenge that limits the scope that IIoT can have in the industry: vendors have traditionally developed communication protocols that are specific to their products. As a consequence, it becomes a challenge to configure communication between devices from different vendors. Moreover, many of them do not have the capabilities to directly connect to the Internet via the TCP/IP protocol suite.

An additional challenge occurs in the training of technicians and engineers in industry fields. The tools that are used traditionally to instruct them are devices at laboratories and software simulators. Nonetheless, the use of laboratories is limited to those who attend them in person. There is currently a limitation for remote access to those devices, and simulators

alone do not offer the required experience level. The above restrict learning opportunities for students. Therefore, implementing IIoT for laboratory settings is a potential solution that benefits education.

There is a potential solution to solve the challenges that limit the implementation of IIoT in devices used for industry and education. It is the OPC UA standard, which consists of an architecture that offers platform-independent communication between devices. The standard provides an information model that can be implemented in several programming languages, eliminating the barrier of protocol heterogeneity. It also presents a communication model based on a client-server architecture, which is straightforward to implement on any operating system.

In this thesis, the use of the abovementioned standard is explored by working on a case study. It consists of two industrial processes at the end of a bottle production line, controlled by a Programmable Logic Controller (PLC). The document proposes a methodology for connecting the PLC to a Virtual Instrument (VI) accessible by a web browser to monitor and control the processes. The goal is to demonstrate that OPC UA can bridge the gap that limits the devices' transition towards IIoT in industry and academy.

2. Justification

The Fourth Industrial Revolution is on the rise. It comes with an increasing set of technological breakthroughs that are enabling the transformation of production, management, and governance systems. This innovation can lead to a supply-side increase in efficiency and productivity while opening new markets and economic growth possibilities (Schwab, 2016). The current industrial revolution is also called the "digital revolution" and "Industry 4.0", in which digital technology has become present in every aspect of modern life.

In the current context, the COVID-19 global pandemic has increased the transition towards digital technology, as it has served as an element to prevent the disconnection of multiple sectors: business, commercial, social, educational, and medical, among others (Wright, 2020). As Agrawal *et al.* (2021) mention, the technologies associated with Industry 4.0 have been critical to the response of companies towards the crisis. More specifically, the priority use cases for digital technologies are the ones that enable remote work, that connect the end-to-end value chain for better visibility and control, and that offer connectivity and data visualization for transparency in operational performance. Moreover, the level of implementation of such technologies was related to the ability of companies to respond to the crisis, as shown in Figure 1.

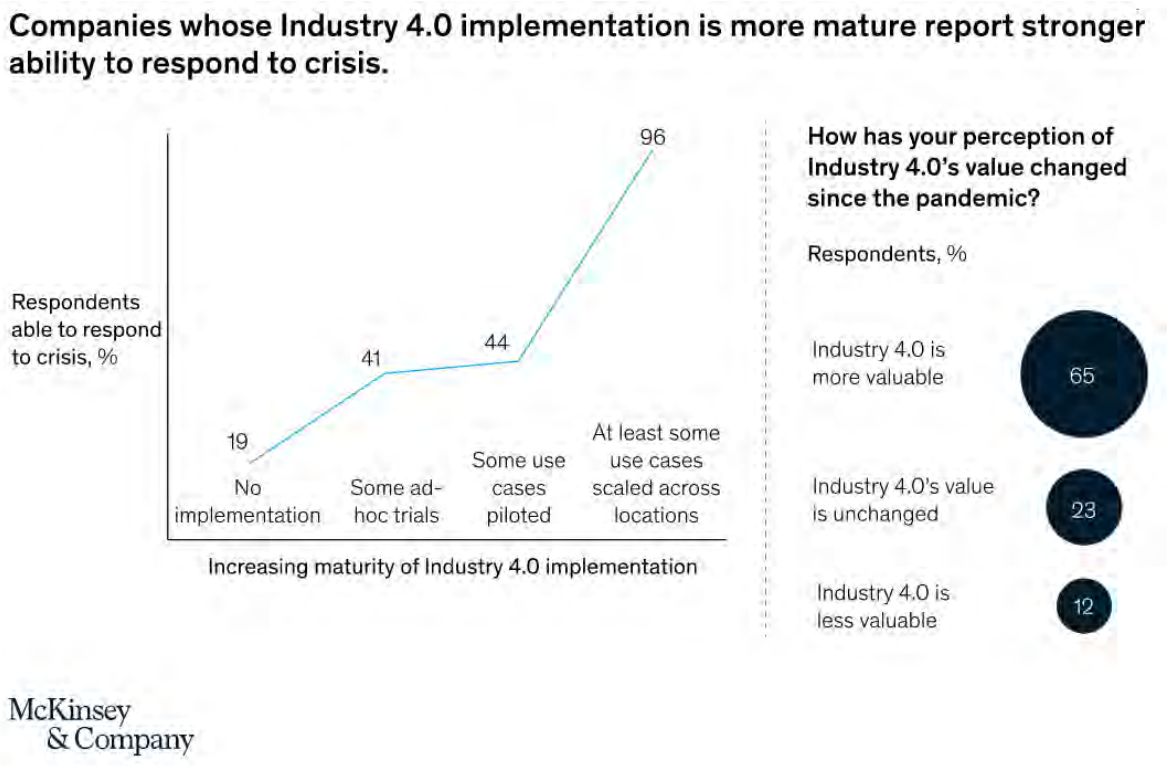


Figure 1. Maturity of Industry of 4.0 implementation compared to the ability of companies to respond to the COVID-19 pandemic crisis. (Agarwal, Dutta, Kelly, & Millán, 2021)

The pandemic has also impacted the education sector. Students worldwide have been forced to continue their studies at home due to lockdown preventive measures. The adaptation requires them to take advantage of technologies such as the Internet of Things (IoT) to continue with their education; however, this approach is generally limited to online meetings and assignments (Darma, Ilmi, Darma, & Syaharuddin, 2020). In this context, there is an increased challenge in disciplines that require technical skills, such as STEM (Science, Technology, Engineering, and Mathematics) and medicine. A potential way of learning those skills is by having online access to real environments where students can remotely manipulate

devices in a way similar to an in-person approach. By doing so, they could test their projects in physical devices and monitor real behaviors to validate them against analytical and simulation-based results. For instance, a paper presented by Vazquez-Gonzalez *et al.* (2018) describes a teaching strategy that leverages current technologies to work with industrial automation laboratories that can be accessed in physical, virtual, and mixed environments.

The transition towards Industry 4.0 technologies demands the connection of devices to the Internet in both of the previously presented cases (industry and education). To be connected enables access to a wide range of digital technologies (Machine Learning, Cloud Computing, among others) and ensures safe, remote access to machines and devices for monitoring and control. According to a report by MarketsandMarkets (2019), Industrial Internet adoption is recognized as one of the three key elements for market growth. The two other elements are: efficiency enhancement of machinery and systems, and reduced production costs.

The Industry 4.0 market had an estimated value of USD 71.7 billion in 2019, with expected growth to USD 156.6 billion by 2024 and a CAGR (Compound Annual Growth Rate) of 16.9% in the 2019-2024 period (see Figure 2). Among the top market players are ABB, Mitsubishi, Yaskawa, KUKA, FANUC, General Electric, IBM, Cisco, Microsoft, Stratasys, Google, Intel, HP, and Siemens (MarketsandMarkets, 2019).

Figure 2. Expected market growth of Industry 4.0, highlighting its trends and focus. (MarketsandMarkets, 2019)

Statistics from Microsoft (2019), show that 38% of companies have not adopted IoT due to complexity and technical challenges. To overcome this issue, several vendors currently offer IoT solutions in the shape of products and services. However, significant investment costs are required and thus limit the implementation based on available resources, with 29% of companies not being able of doing so because of a lack of budget and staff resources (Microsoft, 2019).

The challenge comes when trying to connect devices not adapted for IoT or its industrial equivalent, IIoT (Industrial Internet of Things). Many of them are designed for communications in a low-level setting e.g. M2M (Machine to Machine) and/or not configured for the Internet protocol suite (TCP/IP). In industrial devices used in factories and education, the technical challenges and cost limitations for IIoT adoption can be overcome

by the use of standards. That is the case of OPC UA, a communication and interoperability standard based on OPC Classic that is secure to be used as a foundation for IIoT applications (Kominek & Brubaker, 2018). A five-step process for achieving an IIoT adoption while increasing the ROI (Return of Investment) is shown in Figure 3.

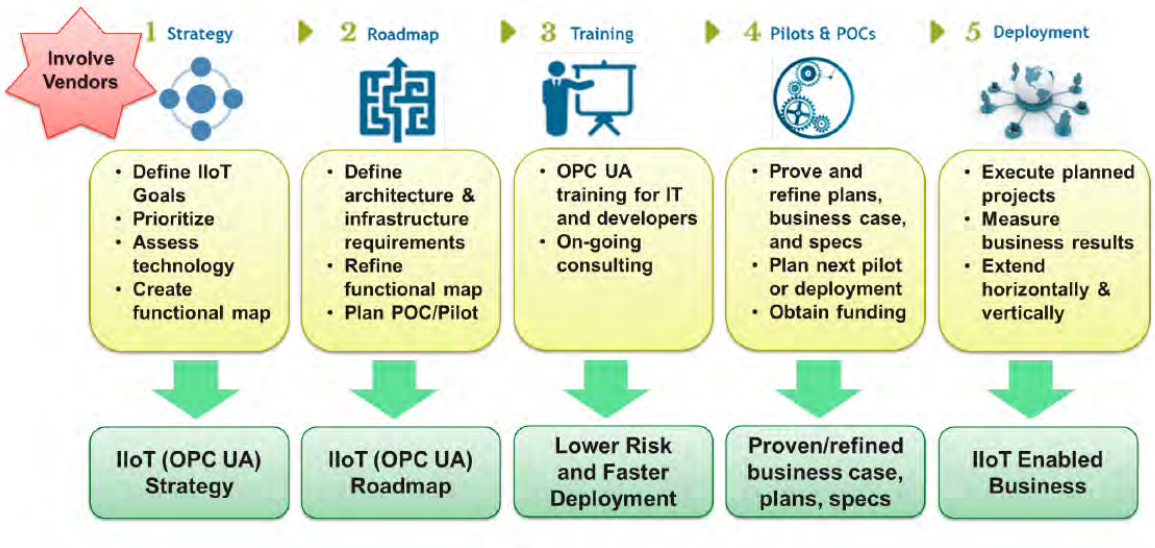


Figure 3. Systematic IIoT strategy to achieve increased ROI. (Kominek & Brubaker, 2018)

Emerson (2019) highlights five key advantages of using OPC UA for IIoT from a technical level, which are:

1. Security of information via authentication, authorization, encryption, and data integrity.
2. Platform independence and scalability, providing data exchange regardless of the operating system and allowing a chain of systems that integrates with OPC UA.

3. Object-oriented approach for formulating an information model according to the specific needs of the system.
4. Server redundancy for high availability of information, providing multiple paths for communication and reducing probabilities of failure.
5. Support of inner loop (deterministic control) and outer loop (non-deterministic advice) control strategies.

The potential benefits for industrial end-users of using OPC UA are based on the advantages listed above. Furthermore, the standard provides a way of integrating new machinery and legacy systems, while contributing to reducing the complexity of communication. The former allows companies to reduce operational expenses and increase their profits (Exor International, 2019).

OPC UA has been traditionally used for the industry, but it has potential in academic institutions (universities, technical capacitation institutes, etc.) because it offers a path for students to access devices such as Programmable Logic Controllers (PLCs) to monitor data and control processes in an educational setting, without the need of physically being there. The above permits a more accessible education that accounts for the needs of students and enables online education of technical skills via the remote use of physical devices.

The work presented in this thesis aims to implement an OPC UA server for integration between a simulated plant, a PLC, a Virtual Instrument (VI), and Web Services, to demonstrate that communication can be achieved at a low cost while offering relevant functionalities for the industrial and educational sectors.

3. Objectives

The following section presents the general and specific objectives, accompanied by the project hypothesis.

3.1. General Objective

To apply the OPC UA standard for communication between a Virtual Instrument and a simulated Programmable Logic Controller that interacts with electromechanical processes in a simulator, to implement IIoT via a Web Services application for control and variable monitoring.

3.2. Specific Objectives

1. Design electromechanical processes in simulation software.
2. Develop a PLC program to monitor and control the simulated processes.
3. Create a Visual Instrument as a front panel to interact with the PLC.
4. Mount an OPC UA server to communicate the PLC and the Virtual Instrument.
5. Launch the Virtual Instrument as a Web Services application to use in a web browser.
6. Validate the data exchange between the application, Virtual Instrument, PLC, and simulated processes.

3.3. Hypothesis

In conformity with the proposed objective and considering the issues and context highlighted in the justification, the following hypothesis is formulated: the use of the OPC UA standard for connecting a PLC and a Virtual Instrument enables low-cost, platform-independent IIoT for industrial and academic applications.

4. Background Knowledge

4.1. First, Second and Third Industrial Revolutions

Humanity has gone through various transition periods, called industrial revolutions, in which quality of life and the way of doing work have significantly shifted. According to the business version of the Cambridge Dictionary (2021), an industrial revolution can be defined as "any period of time during which there is a lot of growth in industry or in a particular industry." Koc and Teker (2019) mention that modern society unquestionably owes its existence and developments to industrial revolutions, in which technological developments made an impact on the economy of countries and the lifestyle of their societies.

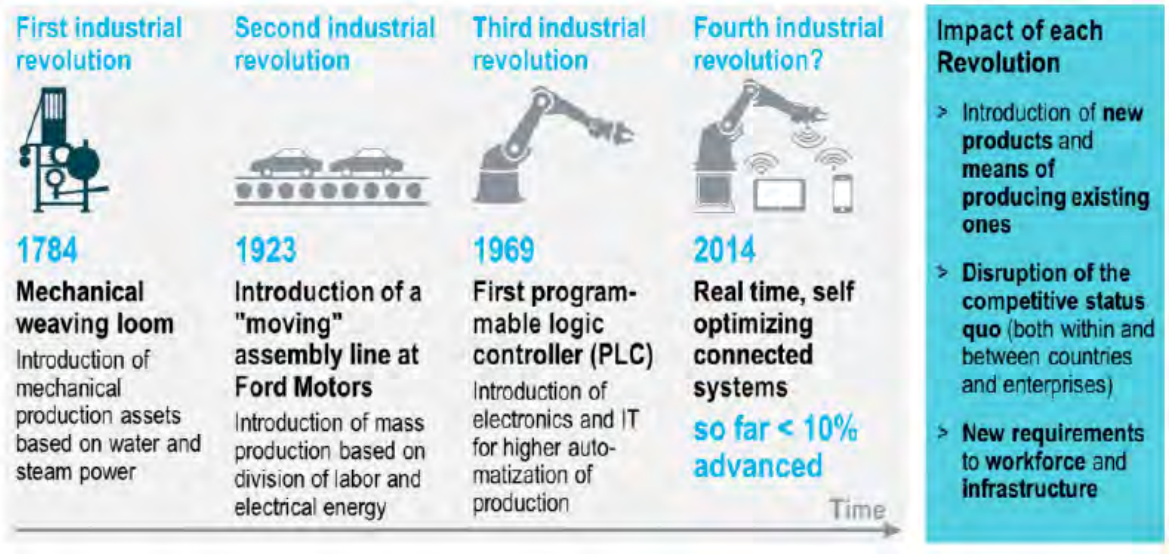


Figure 4. Industrial revolutions: timeline and main characteristics. (Aulbur, CJ, & Bigghe, 2016)

History recognizes four industrial revolutions, as seen in Figure 4. Three of them correspond to the past, and the Fourth Industrial Revolution is happening today. Vickers and

Ziebarth (2019) consider the following periods for the former industrial revolutions: "(1) the First Industrial Revolution (1770-1830) focusing on Britain; (2) the Second Industrial Revolution (1860-1930) focusing on the U.S.; and (3) the Information and Communications Technology (ICT) Revolution (1970-2000) focusing on western countries." (p. 3). From a technological perspective, the First Industrial Revolution is characterized by the introduction of mechanical production powered by steam (Koc & Teker, 2019). The development and implementation of the steam engine changed the perception of work and society by establishing the importance of machines in the modern world, making an energetical transition towards a coal-based society, and laying the foundation of the industrial society: the factory (Ferrell, 2000).

Then, the Second Industrial Revolution leaped forward by the introduction of electrical energy and production assembly lines (Koc & Teker, 2019). The shift towards electricity required facilities to adjust their infrastructure to benefit from this form of energy. It also meant a breakthrough for innovation, allowing numerous new technologies to emerge (Vickers & Ziebarth, 2019). "Electricity is a high quality energy carrier – more productive and flexible than other energy vectors, with zero pollution at the end use point." (Stern, Burke, & Bruns, 2019, p. 8). Nowadays, electricity continues to be the main energy carrier used for controlling processes in the industry, and the concept of assembly lines is used in a wide number of sectors. As Marcin (n.d.) mentions, "Though the assembly line's history is not very long, the world can hardly function without it now. The ease and uniformity it produces have helped manufacturers provide standardized products for their customers, and has made constructing replacement parts very simple."

The elements that made the Second Industrial Revolution possible became the backbone of the next industrial revolutions. Nevertheless, electricity began to be used in a novel way: the development of transistor-based electronics and communication technologies enabled industrial processes to be controlled by computer systems, which became the basis of the Third Industrial Revolution (Vickers & Ziebarth, 2019). In that context, the concept of "automation" emerged and became a standard, enhancing already-existing machinery and assembly lines, and allowing new technologies to be developed. Automation can be defined as "[...] the use of logical programming commands and mechanized equipment to replace the decision making and manual command-response activities of human beings." (Lamb, 2013, p. 1).

By incorporating automation, manufacturing systems can become flexible: they can be reprogrammed to repetitively perform different operations. One of the main motivations for doing so is that productivity increases while reducing production costs. The former also allows companies to respond quickly to market shifts. Additionally, computer-based process control improves the quality and reliability of products, by maintaining a constant accuracy of machines (Helfgott, 1986).

4.2. Programmable Logic Controllers (PLCs)

A central component for automation in industry is the Programmable Logic Controller (PLC), which was first developed in the 1970-1980 period by Modicon to substitute relay-based control logic. It is a digital computer used for the control of electromechanical processes. In contrast with general-purpose computers, a PLC has features that allow it to interact with machines in an industrial environment: multiple inputs and outputs, and

increased resistance to thermal, mechanical, and noise perturbations (Lamb, 2013, pp. 2; 62-63). Figure 5 shows a block diagram of the main components of a PLC.

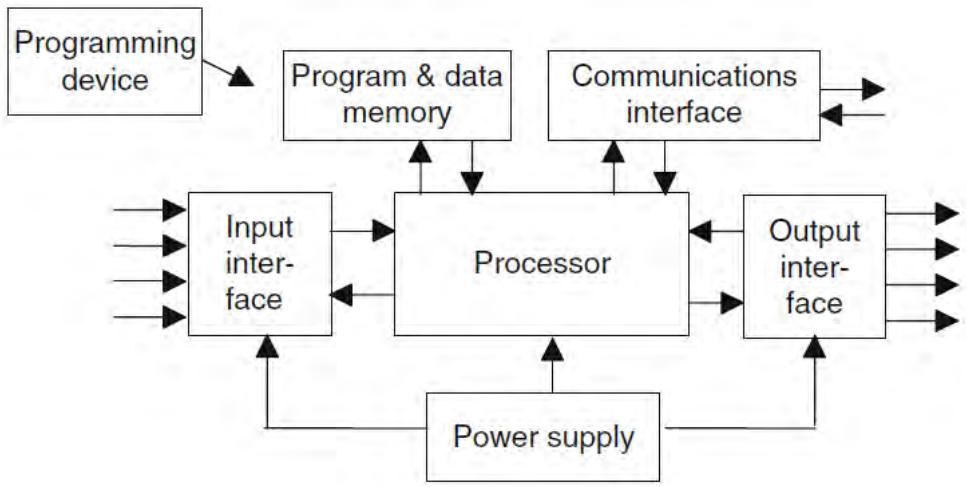


Figure 5. Block diagram of a Programmable Logic Controller. (Bolton, 2009)

A key advantage of a PLC is its flexibility: the same controller can be used in a wide range of applications by modifying the program (set of instructions) loaded into the device, allowing changes in functionality without the need to rewire. A program is designed to implement a specific logic to control a process or machine, and it can perform functions such as sequencing, timing counting, and arithmetic. The most used format for writing PLC programs is ladder programming, which has been adopted by most PLC manufacturers (Bolton, 2009). The international standard IEC 61131-3 normalizes programming languages for PLCs, including the Ladder Diagram (LD) and Function Block Diagram (FBD) as graphic languages, along with two textual languages: Instruction List (IL) and Structured Text (ST).

It also defines the Sequential Function Chart (SFC) programming methodology for structuring and organizing a program (International Electrotechnical Commission, 2003).

The PLC operates by what is known as a scan cycle. It is a repetitive process in which inputs are read, the program is executed, diagnostic and communication tasks are performed and, finally, all outputs are updated; the process is illustrated in Figure 6. In a general case, a factory that has automated processes will use PLCs, but the way a device is used will vary according to the application. There are three general cases: (1) single-ended, where a PLC controls one process; (2) multitask, where one PLC controls several processes; and (3) control management, in which multiple PLCs control several others (Petruzella, 2011).

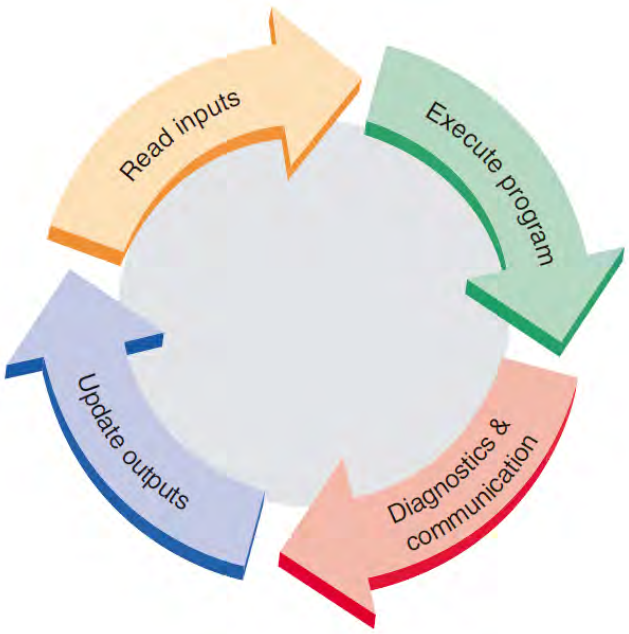


Figure 6. Typical PLC scan cycle. (Petruzella, 2011)

Regardless of which of the applications mentioned above is implemented in a factory, the information used to control processes remains at a local level and there is hardly any access to it except for *in situ*[1] displays such as HMIs (Human-Machine Interfaces). This is a notorious limitation faced by the technology used in the Third Industrial Revolution, especially in the Information Age, which is defined by the Merriam-Webster Dictionary (n.d.) as "the modern age regarded as a time in which information has become a commodity that is quickly and widely disseminated and easily available especially through the use of computer technology." Considering the former definition, the lack of rapid accessibility and use of the information gathered in industrial processes presents both a challenge and an opportunity.

4.3. Fourth Industrial Revolution (Industry 4.0)

Here is where the current period, the Fourth Industrial Revolution, has its origin. When it started, the technological transition in communication technologies (including the Internet) had already happened as part of the Third Industrial Revolution, but those technologies were only at the edge of the industrial sector, providing a solution to connect with partners and customers in the supply chain (Chou, 2019). The abovementioned information gap meant a continuous loss of potential opportunities and of the capacity of predicting failures on time to prevent damage to industrial machinery. As Chou (2019) mentions "Data collected intermittently during production processes or at later stages are insufficient and can lead to serious production delays and cost overruns." (p. 109).

[1] Latin for "on-site".

The Fourth Industrial Revolution closes the information gaps by blurring the lines between the physical, digital, and biological spheres. It is characterized by rapidly developed technological breakthroughs that are disrupting almost every industry. It has made possible affordable access to the digital world, causing major shifts in both the supply and demand sides of the economy (Schwab, 2016). As a consequence, the manufacturing industry has been transformed, denoted as "Industry 4.0". It is based on what are known as cyber-physical production systems, in which the real and digital worlds are connected and interact continuously (Aulbur, CJ, & Bigghe, 2016).

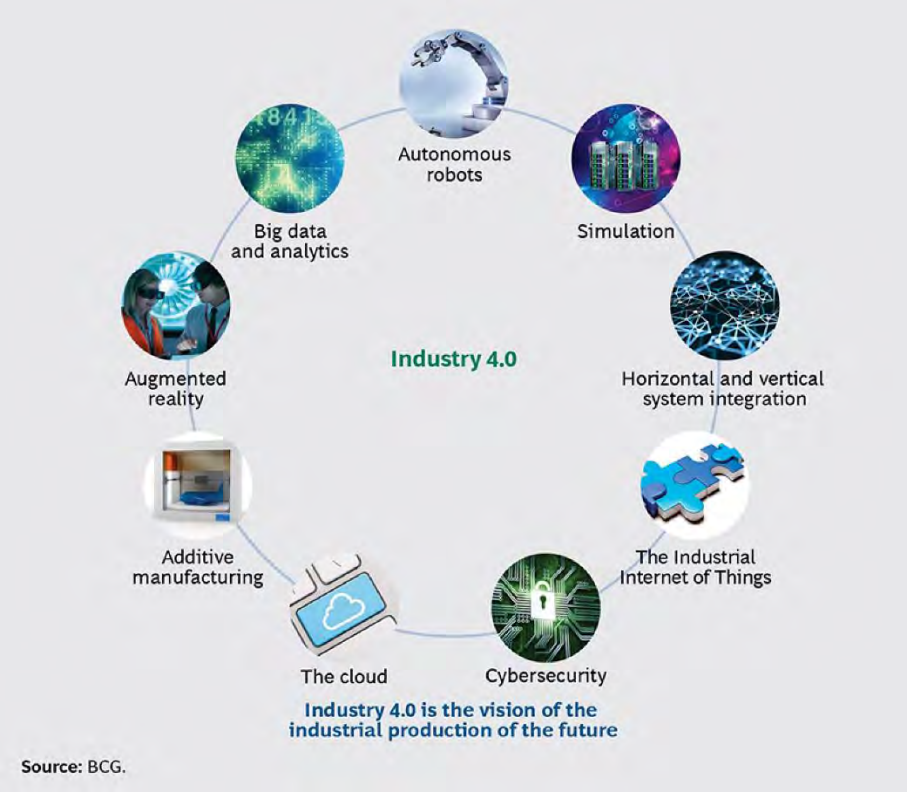


Figure 7. Nine technologies that are transforming industrial production in Industry 4.0. (Lorenz, et al., 2015)

4.4. Industrial Internet of Things (IIoT)

According to Lorenz *et al.* (2015), nine technology trends are enabling the transformation towards Industry 4.0 (see Figure 7). These will transform production by integrating isolated manufacturing cells into an automated and optimized production flow. Among these trends, the Industrial Internet of Things (IIoT) is of special interest because it develops as "[...] a natural extension of prior levels of connectivity achieved by the Internet and World Wide Web (WWW) to include physical objects and systems." (Chou, 2019, p. 107). IIoT is based upon the Internet of Things (IoT), which emerged as a network that enables machines to communicate among each other (Koc & Teker, 2019). IoT became important because commonly used objects (appliances, vehicles, among others) started sharing and collecting data with minimal human intervention; this makes them able to record, monitor, and adjust interactions (Oracle, n.d.).

The functionality provided by IoT is adapted to an industrial level with IIoT. According to the World Economic Forum (2015), IIoT offers key opportunities in four major areas:

1. Improved operational efficiency through predictive maintenance and remote management.
2. The emergence of an outcome-based economy based on the increased visibility of products, processes, partners, and customers.
3. Connected ecosystems through software platforms, overcoming traditional industry boundaries.
4. Human-machine collaboration, resulting in increased productivity.

The features that allow the implementation of IIoT evolve from legacy monitoring systems used in industries that used PLCs for control, along with Supervisory Control And Data Acquisition (SCADA) systems. The difference between legacy and IIoT systems is the connectivity of the latter over an Internet Protocol (IP) network structure (Zhou, Damiano, Whisner, & Reyes, 2017). Therefore, to fully understand IIoT, it is important to develop on the topics of SCADA systems and IP networks.

4.5. SCADA systems

A SCADA system consists of software and hardware elements that allow industries to remotely control and monitor industrial processes in real-time, allowing operators to interact with them through devices such as Human-Machine Interfaces (HMIs). Their architecture makes use of PLCs and data acquisition components such as Remote Terminal Units (RTUs), as illustrated in Figure 8. Modern communication protocols did not exist when SCADA first appeared in the mid-20th century. In the last two decades of the century, SCADA evolved to use Local Area Networking (LAN) to connect similar systems; however, a disadvantage of these first interconnected systems is that most protocols were unique to specific vendors, therefore devices from different brands were not able to communicate (Inductive Automation, 2018).

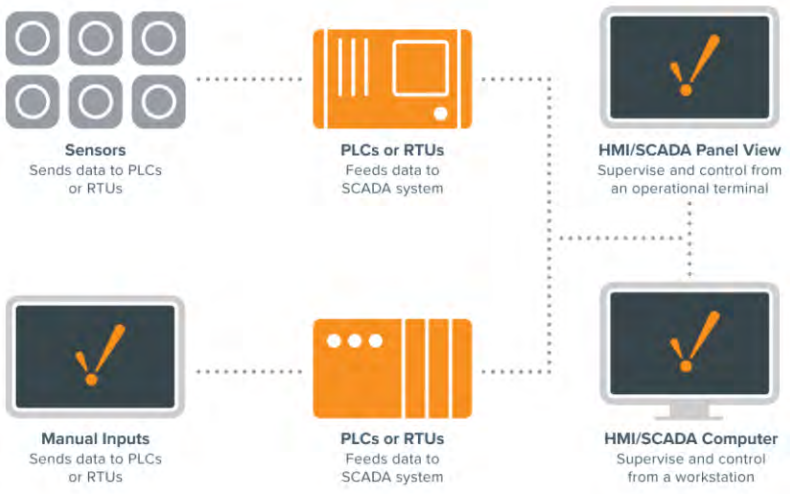


Figure 8. A basic SCADA diagram. (Inductive Automation, 2018)

4.6. Network Models

The interconnection of systems, as shown in the case of SCADA, is an important step to take in any context where various devices (computers or other hardware devices) have relevant information that would be valuable to communicate. In such contexts, establishing a network allows them to exchange information and cooperate. Networking is a complex subject because technologies defined by different groups interoperate to accomplish the overall goal of network communication. For that reason, models have been created to describe the different layers in the network; the most common general model is the Open Systems Interconnection (OSI) reference model (Kozierok, 2015). It was defined by the International Standard Organization (ISO) with the purpose of "[...] open communication between different systems without requiring changes to logic of the underlying hardware and software" (Jasud, 2017, p. 1).

This model divides the networking functions into a set of seven conceptual layers, each representing an abstraction level, i.e. its relationship with the actual hardware; they are presented in Figure 9. As the layers increase, the interaction with hardware becomes less direct, and the interaction with software extends. The layers can be divided into two groupings: the first one is denoted as the "lower layers" and comprises layers 1-4. It is concerned with data formatting, encoding, and transmission over the network. The second grouping is entitled the "upper layers", comprising layers 5-7. It is focused on the interaction with the user and the implementation of applications that run on the network (Kozierok, 2015).

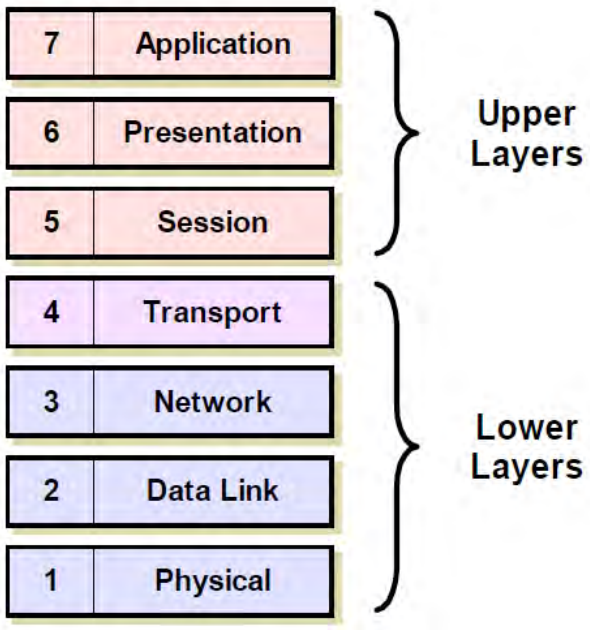


Figure 9. OSI reference model layers. (Kozierok, 2015)

A brief overview of each layer is presented below:

1. Physical layer: responsible for the physical network structures, including mechanical and electrical specifications, to transmit individual bits (represented as 0 and 1) from one node to another.
2. Data Link layer: responsible for organizing bits from the Physical Layer into logical groups of information, named frames. In this layer, frames are exchanged between two nodes within the same logical network using methods.
3. Network layer: in charge of the logical addressing and routing of information in the form of "packets" between independent networks.
4. Transport layer: ensures the delivery of individual packets to a destination, using error and flow control methods.
5. Session layer: allows two devices to maintain ongoing communications by an active session across a network, in which they can exchange data or messages.
6. Presentation layer: concerned with formatting the data transferred in the lower layers, focusing on syntax and semantics so that the sending and receiving applications can interpret the information.
7. Application layer: enables the user (human or digital) to access the network, allowing the visualization and interaction with the information received by the application.

(Jasud, 2017).

The lower layers in the OSI model (Physical, Data Link, Network, and Transport layers) are useful to visualize the process that data follows to be transmitted and received.

On top of that, the higher layers (Session, Presentation, and Application layers) are helpful to analyze how data interacts with the user, and they are mostly implemented via software running on computers or other hardware devices (Kozierok, 2015).

A different architectural model to represent the functional division of a network is the TCP/IP model, which consists of four layers that approximately match six layers of the OSI model (Kozierok, 2015).

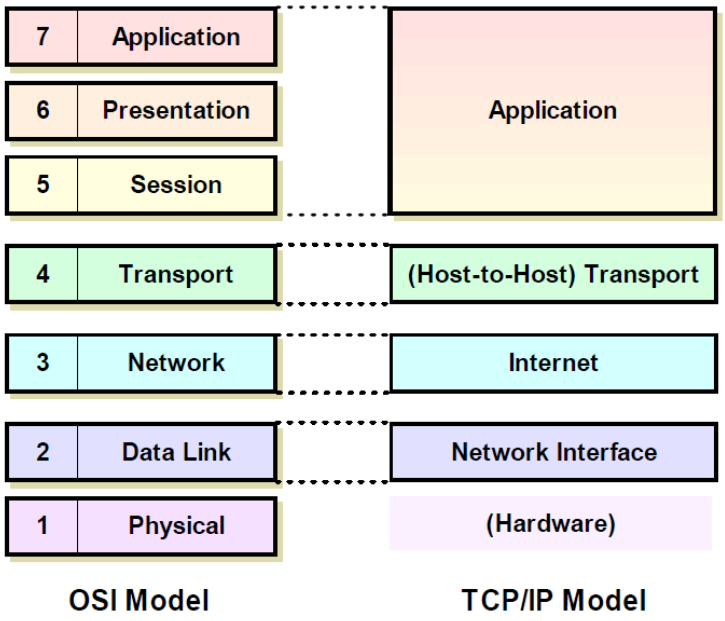


Figure 10. Comparison between the OSI and TCP/IP models. (Kozierok, 2015)

As seen in Figure 10, the TCP/IP model does not take into consideration the first layer of the OSI model; additionally, the TCP/IP Application layer encompasses OSI layers 5-7. The remaining layers are represented in both models with different terminologies. The architecture of the TCP/IP model eases the understanding of the protocol with the same name,

described by Blank (2004) as "[...] a strong, fast, scalable, and efficient suite of protocols." (p. 1). It consists of a collection of network protocols, among which the most important ones are the Transmission Control Protocol (TCP) and the Internet Protocol (IP). IP operates on the OSI Network layer and provides functions such as addressing and datagram routing; on the other side, TCP acts upon the OSI Transport layer, allowing the establishment of connections and management of data transport between application (software) processes (Kozierok, 2015).

The relevance of TCP/IP is such that it is referred to as "the language of the Internet" because it is the most widely used protocol for communicating on it. Moreover, the origin of the Internet is set in the creation of ARPAnet, a super-network which was developed by ARPA[2] in the United States in 1969, initially using a protocol called Network Control Protocol (NCP), but it showed to not be robust enough for the network. TCP/IP was created to overcome the limitations imposed by NCP, and it became the standard language of ARPAnet; it is now an essential part of the Internet, and it has evolved to meet its changing requirements (Blank, 2004).

4.7. Internet & IP Protocol

As mentioned in the SCADA systems and observed on the Internet with TCP/IP, protocols are crucial for exchanging information among devices. Kozierok (2015) denotes that "a protocol represents communication between logical or physical devices at the same layer of the model [...]" (p. 153). The protocol that a network uses depends on the application. A useful classification distinguishes between protocols used for Information Technology (IT)

[2] Advanced Research Projects Agency.

and Operational Technology (OT); while IT refers to business and office networks, OT is applied to industrial processes and factories (Bloem, et al., 2014).

The largest network used by IT devices is the Internet. It consists of a global network of devices that enables resource sharing and general-purpose communication. The basis of the modern Internet is packet switching because it allows multiple senders to transmit data over a shared network in the form of packets (small blocks of data), using a process called statistical multiplexing. Packet switching technologies are classified according to their area coverage. This classification includes:

- Local Area Network (LAN), spanning a single room or a single building.
- Metropolitan Area Network (MAN), spanning a major city or metropolitan area.
- Wide Area Network (WAN), spanning multiple cities.

(Comer, 2015).

To connect between devices, each of them is given a unique address called Media Access Control (MAC) address, which is standardized by the Institute for Electrical and Electronics Engineers (IEEE). Even though each device has its unique MAC address, it is not enough since the Internet includes different network technologies that define their own addresses. To guarantee uniform addressing, the Internet Protocol (IP) defines a method for addressing that does not depend on the network hardware or MAC addresses. IP specifies an address for each host in a network. There are currently two main versions of the protocol being used: IPv4, which uses 32-bit addresses and is the most commonly used; and IPv6, which uses 128-bit addresses (Comer, 2015).

The IP addresses in IPv4 are commonly expressed in what is known as dotted decimal notation because it consists of four sets of decimal numbers separated by periods. When interpreted by a machine, each set of numbers is translated into binary octets (groups of 8 bits). IP addresses can be classified into five classes under the "classful" IP addressing scheme, denoted as Classes A, B, C, D, and E. An address can be identified as part of a class according to the value of its first numbers (called the prefix) (Blank, 2004). Table 1 shows an illustrative table for identifying the class of an address.

Table 1. IP Address Class Bit Patterns, First-Octet Ranges and Address Ranges. (Kozierok, 2015)

IP Address Class	First Octet of IP Address	Lowest Value of First Octet (binary)	Highest Value of First Octet (binary)	Range of First Octet Values (decimal)	Octets in Network ID / Host ID	Theoretical IP Address Range
Class A	**0**xxx xxxx	**0**000 0001	**0**111 1110	1 to 126	1 / 3	**1**.0.0.0 to **126**.255.255.255
Class B	**10**xx xxxx	**10**00 0000	**10**11 1111	128 to 191	2 / 2	**128.0**.0.0 to **191.255**.255.255
Class C	**110**x xxxx	**110**0 0000	**110**1 1111	192 to 223	3 / 1	**192.0.0**.0 to **223.255.255**.255
Class D	**1110** xxxx	**1110** 0000	**1110** 1111	224 to 239	–	224.0.0.0 to 239.255.255.255
Class E	**1111** xxxx	**1111** 0000	**1111** 1111	240 to 255	–	240.0.0.0 to 255.255.255.255

The IP protocol is especially relevant for communicating between the independent networks that make up the Internet, and for the communication of devices inside a local network. Each IP address is made up of two portions: the network portion and the host portion. For that reason, the TCP/IP uses a subnet mask, which consists of a sequence of bits with the same length as the IP address, to distinguish between the bits used for the network portion of the address (value = 1), and the ones used to identify a device/host (value = 0).

Once IP identifies the network address, it determines whether the destination host is inside the same local network or on a remote network. Then, the Address Resolution Protocol (ARP), which is part of the TCP/IP protocol collection, sends an ARP broadcast to the local network or to a router to locate the destination host. In case it is located on a different network, the router sends it to the next network on its path to the appropriate destination network (Blank, 2004). It is important to distinguish between private and public IP network addresses because the former operate inside a private network where its owner has control of the address assignment for all devices, but the latter require management mechanisms given that they can be reached by the Internet; therefore, overlapping must be avoided (Kozierok, 2015).

The protocols, addresses, and technologies that constitute the Internet are the groundwork for achieving a true Industry 4.0 with IIoT, by using communication standards that enable devices to communicate with each other locally and on the Internet. Besides, the Wireless Local Area Network (WLAN) concept is relevant for a modern implementation of IIoT, using standards such as Wi-Fi – defined by IEEE 802.11 specification, and Bluetooth – based on IEEE 802.15.1 specification, among others. These technologies provide the means for wireless communication between industrial devices (machine-to-machine, or M2M) and between them and a user application to provide scenarios such as remote monitoring and predictive maintenance (Sultanow & Chircu, 2019).

4.8. Industrial Communications

Nevertheless, is crucial to take into account that industrial communications tend to differ according to the application and the brands of components, as mentioned while describing

SCADA systems. Industrial protocols are central for automation and control applications in the context of the Third and Fourth Industrial Revolutions. In the early 1980s, vendors and end-users were demanding to establish a global protocol standard. As a result, a wide range of standards was created, including IEC 61158 which covers nine protocol profiles: FOUNDATION Fieldbus, CIP, PROFIBUS/PROFINET, P-NET, WorldFIP, INTERBUS, CC-Link, HART, and SERCOS (Knapp & Langill, 2015).

Wired industrial communications are the most common means for exchanging information in an industrial environment. Due to the different requirements, industrial communication systems are classified into four different types: (1) sensor/actuator networks – field level; (2) Fieldbus systems – collection and distribution of data, and communication between field devices and controllers; (3) controller networks – controller level; (4) Wide Area Networks – enterprise level, for networked segments of an automation system. The challenge comes when there is the need to communicate devices with different protocol suites. For instance, In Fieldbus systems, PROFIBUS is used by Siemens, WordFIP by Schneider, and INTERBUS by Phoenix Contact. Each protocol contains unique communication models, transmission technologies, and formats (Pereira & Neumann, 2009).

In the context of IIoT, a crucial challenge is to offer a machine-to-machine (M2M) communication that can align and converge different protocols into a unified scheme. It also is to provide an industry-level integration between OT and IT (see Figure 11), which have traditionally been managed and controlled independently from one another (Bloem, et al., 2014).

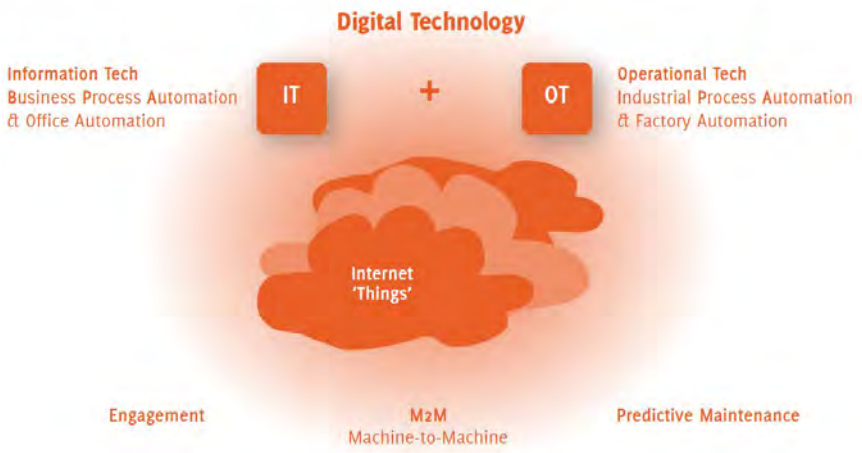


Figure 11. Interaction between IT and OT through Internet. (Bloem, et al., 2014)

4.9. OPC Standard

The abovementioned challenges can be resolved with a well-designed solution that ensures data transfer regardless of protocol variety; an important and attractive approach for achieving that goal is called OPC. It originally was an acronym for OLE (Object Linking and Embedding) for Process Control, and nowadays OPC stands for Open Platform Communications. The OPC standard (now known as OPC Classic) was released in 1996 with the purpose of abstracting PLC-specific protocols into an interface that would convert generic requests (read/write) into device-specific requests, and vice-versa. A more recent version of the standard, called OPC UA (Unified Architecture), was developed to address the needs of service-oriented architectures in manufacturing systems, such as security and data modeling. OPC has been adopted in multiple industries including manufacturing, oil and gas, building automation, and renewable energy & utilities (OPC Foundation, n.d.-a).

As the OPC Foundation (n.d.-b) describes it, “The OPC Classic specifications are based on Microsoft Windows technology using the COM/DCOM (Distributed Component Object Model) for the exchange of data between software components. The specifications provide separate definitions for accessing process data, alarms and historical data.” The aforementioned definitions are:

1. OPC Data Access (OPC DA): exchange of data, which may include values, time, and quality information.
2. OPC Alarms & Events (OPC AE): exchange of event and alarm type messages, along with state variables for state management.
3. OPC Historical Data Access (OPC HDA): query methods and analytics for historical data.

(OPC Foundation, n.d.-b).

The OPC architecture specifies interfaces for communication among different elements. Its communication structure consists of various OPC clients communicated through one or more OPC servers. Each server vendor supplies a code that determines the data and devices to which a server has access, along with information on how it physically accesses that data (OPC Foundation, 2003). The client/server relationship is shown in Figure 12.

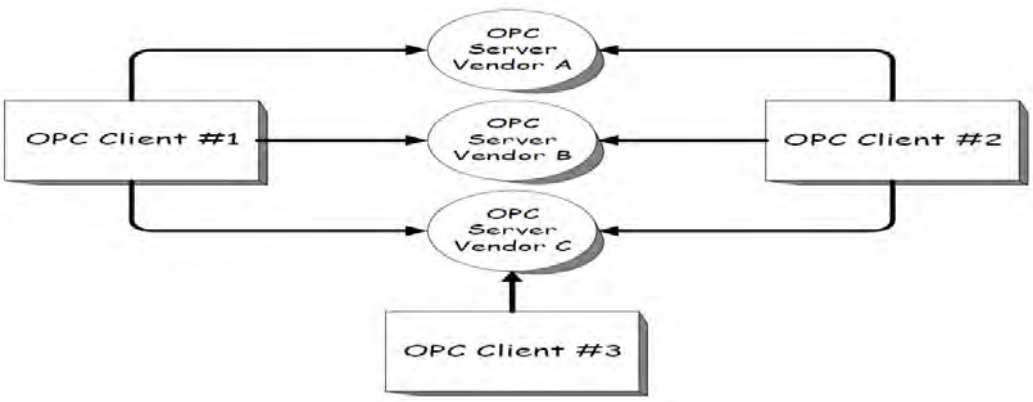


Figure 12. OPC client/server relationship. (OPC Foundation, 2003)

An OPC server is comprised of various objects: the server, groups, and items. The OPC server object holds information about itself and contains the OPC group objects. Each OPC group object keeps the information about the group, and it contains and organizes OPC items locally. Finally, OPC items represent connections to data sources inside of the server. All the access to OPC items is done via an OPC group, within which they are defined and contained (OPC Foundation, 2003). The abovementioned relationship is illustrated in Figure 13.

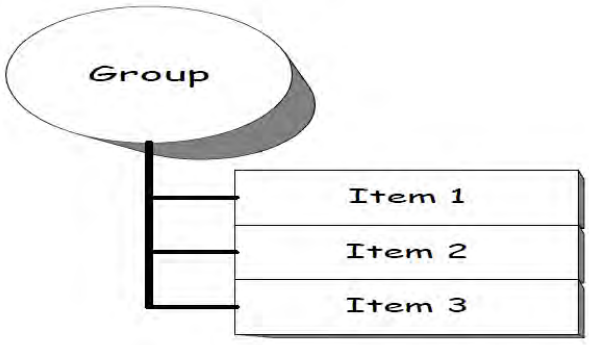


Figure 13. OPC group/item relationship. (OPC Foundation, 2003)

The OPC interfaces allow OPC servers to be implemented in various applications, such as collecting raw data from physical devices, transferring that data into a SCADA system, or from such system into a client application (OPC Foundation, 2003). Figure 14 illustrates these possible interactions.

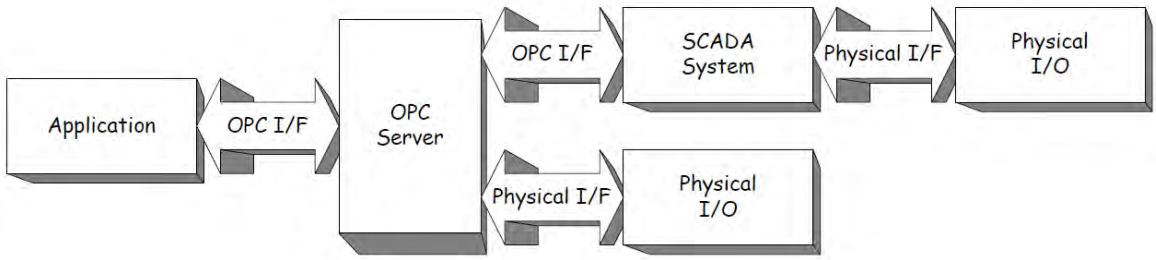


Figure 14. OPC interfaces. (OPC Foundation, 2003)

It is relevant to remark that the OPC specification states what the interfaces are, but it does not specify how they are implemented. There are two sets of interfaces defined by the OPC specification: OPC Custom and OPC Automation interfaces, which are for general use and for facilitating the use of automation-enabled products (such as Visual Basic), respectively. The OPC server is responsible for providing interfaces and for managing the OPC objects (OPC Foundation, 2003). A typical OPC architecture implementation that uses both interfaces is shown in Figure 15.

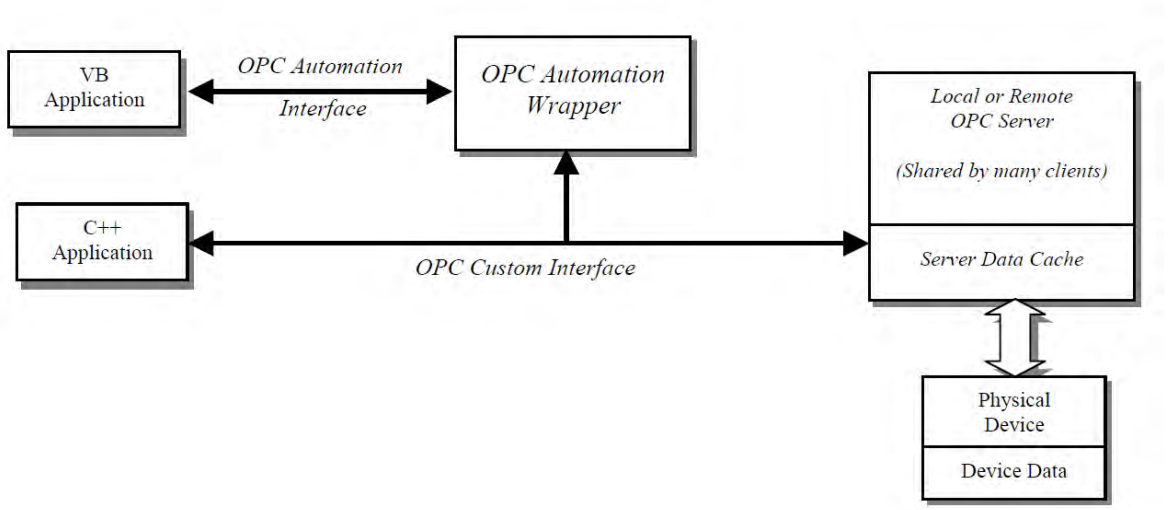


Figure 15. Typical OPC architecture. (OPC Foundation, 2003)

The implementation of an OPC server defines the types of data sources available for communication (read/write). An OPC client communicates with the server through interfaces, providing functionality to create and manipulate OPC group objects. The former are used to organize the data that a client wants to access while giving it a way to "subscribe" to the list of items and receive notifications as they change. An important aspect to consider is that OPC itself does not offer methods for ensuring synchronization of values and attributes; therefore, additional aspects such as "handshaking" and flag passing (to signal the state of data transfer) are needed. Additionally, serialization (the way for a client to control the order of writing values) is not a part of OPC specifications but is strongly recommended (OPC Foundation, 2003). The data types that can be transferred using OPC DA are summarized in Table 2.

Table 2. OPC data types. (Matrikon, 2020)

Data Type	Description
VT_EMPTY	Default/Empty (No data)
VT_I2	2 byte signed integer
VT_I4	4 byte signed integer
VT_R4	4 byte real
VT_R8	8 byte real
VT_CY	Currency
VT_DATE	Date
VT_BSTR	Text
VT_ERROR	Error code
VT_BOOL	Boolean (TRUE = -1, FALSE = 0)
VT_I1	1 byte signed character
VT_UI1	1 byte unsigned character
VT_UI2	2 byte unsigned character
VT_UI4	4 byte unsigned character
VT_ARRAY	Array of values

The prefix “VT” used in the data type names of Table 2 corresponds to the “VARIANT” type from Windows COM Automation for representing data values. Most data types can be converted into Object elements that can be processed by the Microsoft .NET tools and languages. A successful conversion is guaranteed by using Common Language Specification (CLS), which contains rules and restrictions for cross-language compatibility (OPC Labs, n.d.).

A challenge that must be considered when implementing an OPC Classic server is the data transfer frequency to physical devices, considering that non-sharable communication paths exist. In addition, an OPC client can specify an update rate for each group, or even for individual items within a group; the OPC server must not send data at a rate faster than requested. In case that an item has a faster sampling rate than its group or if the value changes

more often than the update rate, the server will buffer the data and pass it to the client in the scheduled callback; the size of the buffer depends on the server (OPC Foundation, 2003).

Based on the fundaments of OPC Classic, the OPC Unified Architecture (UA) specification "[...] fulfills a technology shift from the retiring COM/DCOM technology to a service-oriented architecture providing data in a platform-independent manner via Web Services or its own optimized TCP-based protocol." (Mahnke, Leitner, & Damm, 2009, p. ix). The fundamental components of OPC UA are shown in Figure 16.

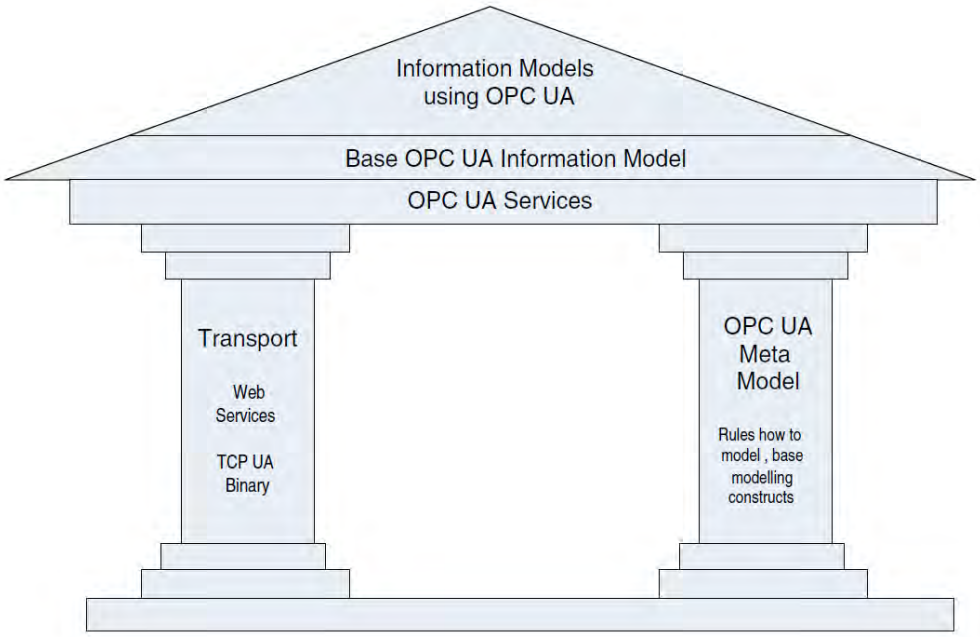


Figure 16. The foundation of OPC UA. (Mahnke, Leitner, & Damm, 2009)

As mentioned above, OPC UA defines an optimized TCP protocol aimed at intranet communication. It also handles the mapping to accepted internet standards such as HTTP, Web Services, and XML for internet communication. OPC UA uses a client-server concept

similar to OPC Classic. It defines a server as an application that exposes its information to other applications; inversely, a client is an application that consumes information from others (Mahnke, Leitner, & Damm, 2009).

The platforms for implementing OPC UA expand in contrast with OPC Classic: the former can be implemented in any platform and Operating System (OS) including embedded processors and Personal Computers (PCs) and servers, while OPC Classic only runs on Windows platforms. The ease of implementation simplifies the integration of multi-vendor systems. Also, OPC UA supports dynamically-updated Object-Oriented information modeling and provides predefined data structures (Kominek & Brubaker, 2018).

The purpose of using OPC in IIoT is to offer a robust and efficient data exchange between protocols and systems. It provides a single Application Programming Interface (API) for communicating with systems from many vendors. A benefit of its client-server paradigm is that multiple clients can be connected to any server at a particular time, providing efficient use of data within a system (Figure 17). Likewise, multiple servers can be connected to a client (Figure 18), so that it handles data processing while each server application connects to a different data source (Janke, 2000).

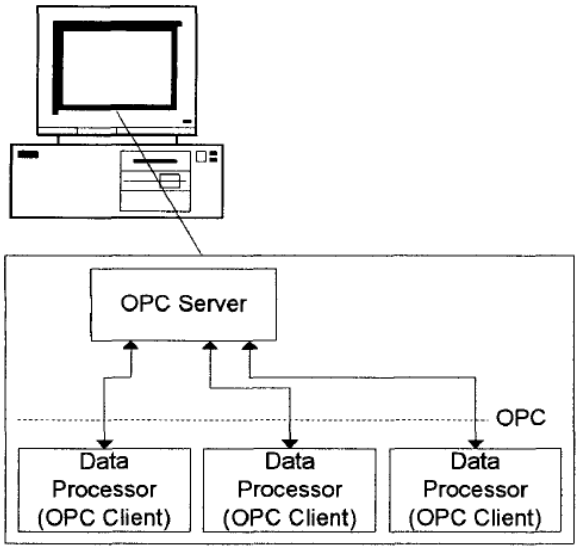


Figure 17. Multiple OPC clients connected to an OPC server. (Janke, 2000)

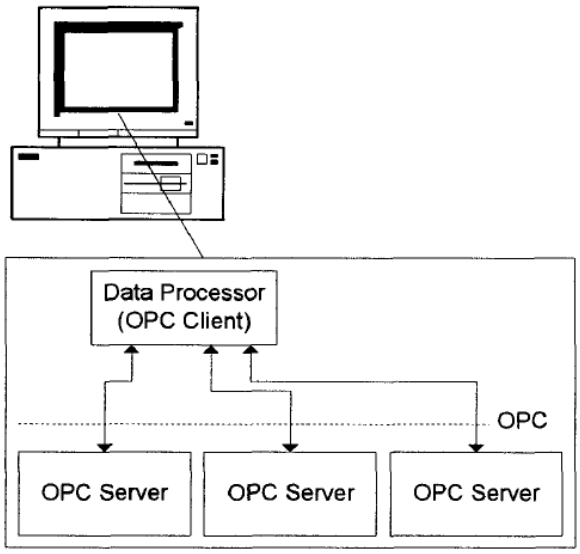


Figure 18. OPC client connected to multiple OPC servers. (Janke, 2000)

IIoT can be implemented using an OPC architecture, where the OPC server acts as a gateway for sharing data on the Internet. It offers an intermediary role for vertical integration, allowing data transfer from the device level to the application level (Ghazivakili, Demartini, & Zunino, 2018).

There are currently several implementations of OPC for IIoT. An example is the work done by Kim & Sung (2017), who proposed the implementation and evaluation of the PLCopen OPC-UA software in industrial control systems. This software was released by the PLCopen organization for defining the information model for connecting PLCs with OPC UA servers. Another application comes from the work done by Lin & Hwang (2019), where OPC UA interacts with SOAP (Simple Object Access Protocol). The former is based on XML and used as a backbone for Web Services. By the above-mentioned interaction, a Stabuli robotic arm is connected via SOAP to the OPC UA server, which connects to the Windows Azure Web Application for Smart Manufacturing to upload the robot trajectory and related parameters to the cloud. Furthermore, smart devices known as SmartBoxes have been developed by various researchers, such as the effort done by Torres *et al.* (2019) in which each SmartBox (based on a Raspberry Pi hardware platform) can be used for real-time M2M communications using OPC-UA, along with MQTT (Message Queuing Telemetry Transport). In a case study, a SmartBox was set-up in a clothing label factory, where predictive maintenance functionalities were provided, along with a connection to an IIoT cloud platform, with the physical architecture shown in Figure 19.

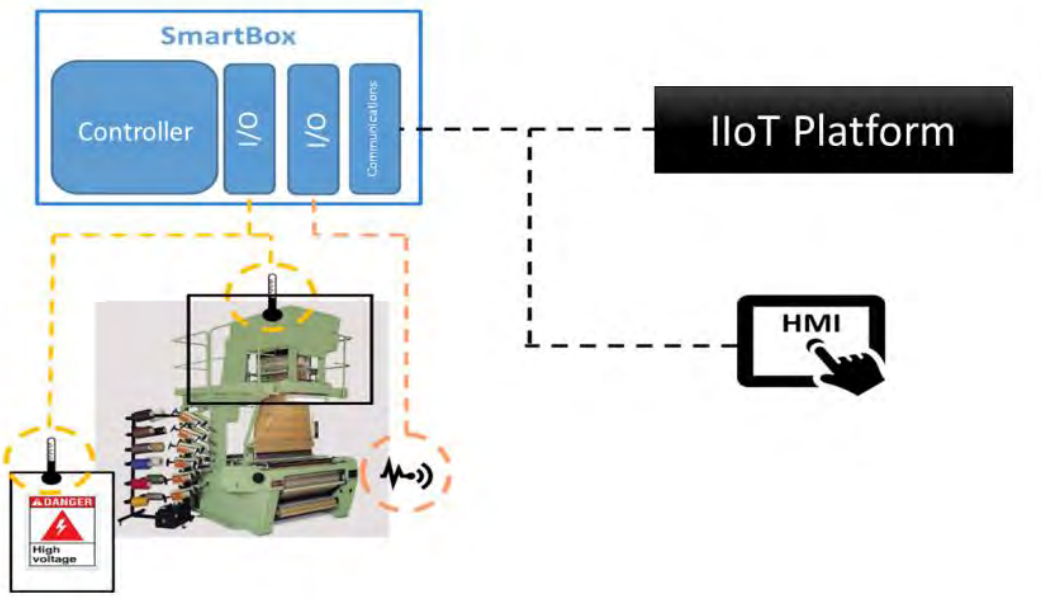


Figure 19. Physical structure of a SmartBox-based system on a label factory. (Torres, et al., 2019)

The infrastructure proposed by Vazquez-Gonzalez *et al.* (2018) offers various approaches for promoting the learning of industrial automation systems, using OPC servers to exchange digital and analog data that students could use to verify the programming of a PLC. One of the proposed approaches is a remote laboratory where a physical lab could be accessed via Internet, using a web server. The proposal is shown in Figure 20.

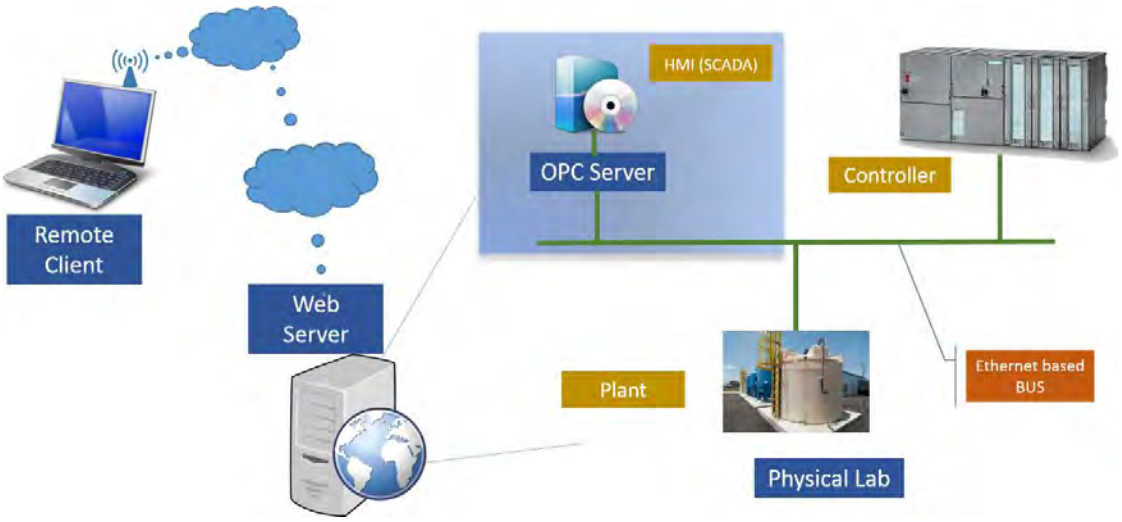


Figure 20. Proposal of a remote laboratory that uses the OPC standard for data exchange, along with a web server. (Vazquez-Gonzalez et al., 2018)

5. Methodology

The goal of this section is to describe the procedure followed to implement a simulated plant, design a program for control and monitoring with a PLC, mount an OPC UA server, create a Virtual Instrument (VI) to interact through the server with the PLC, and publish a Web Services application that runs the VI. The description of each part of the process is accompanied by an explanation of the steps to follow, the software tools used, and relevant considerations.

5.1. Case Study

An industrial application was chosen as a case study for the abovementioned procedure. It corresponds to a SCADA system with two processes used as a final stage in the manufacturing of plastic bottles. The first process monitors and counts the number of bottles to be packed inside a box, passing by a sensor. It uses an industrial traffic light that changes according to the counted bottles; if the number exceeds a critical value, an alarm is activated and continues active up to two seconds after the value is no longer critical. The second process activates the electric motor of a conveyor band that transports bottles and sends information about the expected position of a bottle being transported by the band, for an operator to visualize as an animation. The specifications for the case study are listed below:

- Only one process can be in execution at once.
- The process is selected by a physical switch at the plant.
- The processes include a VI that takes the role of the interface between the process and the user, where control actions and relevant information are displayed.

- If a process is selected, it can be activated by the START button and deactivated by the STOP button, either physical or digital (in the VI).
- The VI must include START and STOP buttons, along with indicators for the selected and active processes.
- If the switch for process selection changes while a process is active, it remains active until stopped. Pressing the START button again activates the currently selected process.
- Process 1 counts bottles increasingly or decreasingly, according to a switch. If it is closed, the count increases; conversely, it decreases.
- The sensor for counting bottles is modeled as a switch.
- If the number of pieces is greater than 1 and lower than 5, a green light is activated; if it is greater to or equal than 5 and lower than 10, a yellow light is activated; finally, if it is greater than 10, a red light is activated. Likewise, in this case, an acoustic alarm must be activated for 2 seconds; if the condition remains after the time has finished, it activates again.
- Process 2 consists of a conveyor band. The motor must be activated along with a 200 ms timer associated to a counter.
- Every 200 ms, the value of the counter must increase. The counter is used to animate the movement of a bottle on the VI.
- When the counter reaches a value of 10, it must reset to 0.
- The display section of the VI for Process 1 must include an industrial traffic light and a field that shows the current value of the counter.

- The display section of the VI for Process 2 must include two animations: one for the bottle moving along the conveyor band, and the other showing the rotation of the motor.

5.2. Simulated Plant

The lockdown procedures due to the COVID-19 pandemic did not allow to have access to university facilities. Therefore, the implementation of the plant and the consequent stages of the methodology were done entirely using simulators and other software tools, all running on Windows. The program used for creating the simulated plant was Festo FluidSIM® 4.2p Pneumatics. It was selected because, as Festo (2021) describes it, “For more than 20 years, FluidSIM® has been the world’s leading circuit diagram design and simulation program for pneumatics, hydraulics, and now also for electrical engineering.” Additionally, it offers input and output ports that can communicate via an OPC server.

The system was divided into input and output sections. In the input section (Figure 21), a switch was used to model the sensor used to detect bottles in the first process. Additional functionality was added by adding switches for selecting the active process, starting or stopping it, and signaling whether the counter value increases or decreases when detecting a bottle.

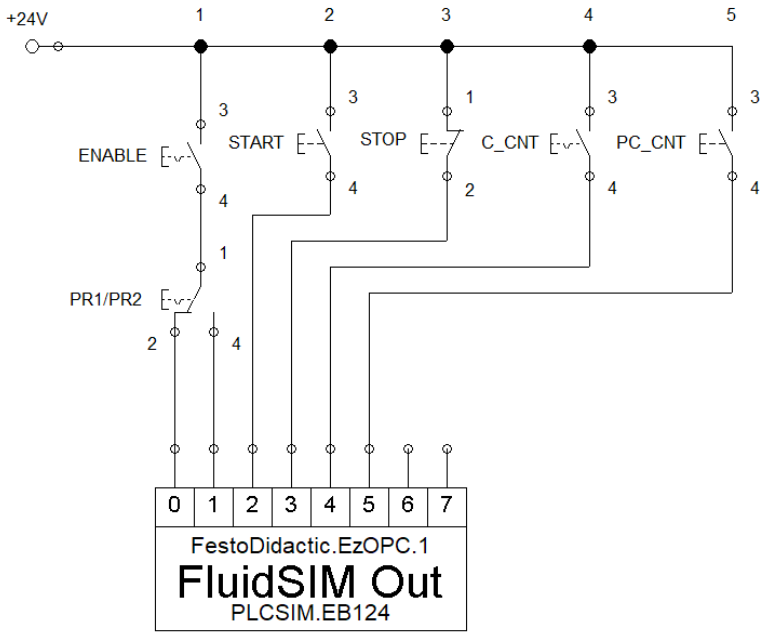


Figure 21. Input section of the simulated plant in FluidSIM.

The output section (Figure 22) consisted of three indicator lights that represent the states of the industrial traffic light (green, yellow, and red), an acoustic indicator for the alarm, and an electric motor for the conveyor band.

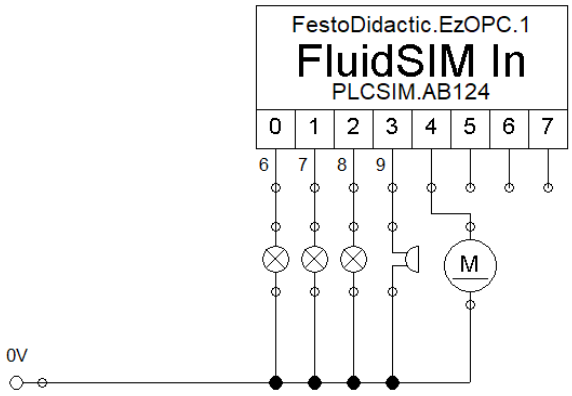


Figure 22. Output section of the simulated plant in FluidSIM.

As it can be observed, both sections are connected to ports called “FluidSIM Out” and “FluidSIM In”, which correspond to the output and input of the software towards other programs, respectively. FluidSIM allows this interaction with the use of EasyPort, OPC, or DDE; in this case, OPC is used. For that purpose, the software Festo Didactic EzOPC V5.6 was selected. It uses the OPC Data Access (Classic) standard and allows connectivity between process simulations (in CIROS/COSIMIR, FluidSIM, or other sources via EasyPort) and controllers (S7-PLCSIM, CoDeSys, PLC via EasyPort, or FluidSIM controllers). Since the simulated plant is meant to be controlled with a PLC, the data flow was configured to connect FluidSIM with S7-PLCSIM; further details of the latter program are covered in Section 5.2 of the present document. The “Overview” panel of EzOPC shows the aforesaid data flow, as seen in Figure 23.

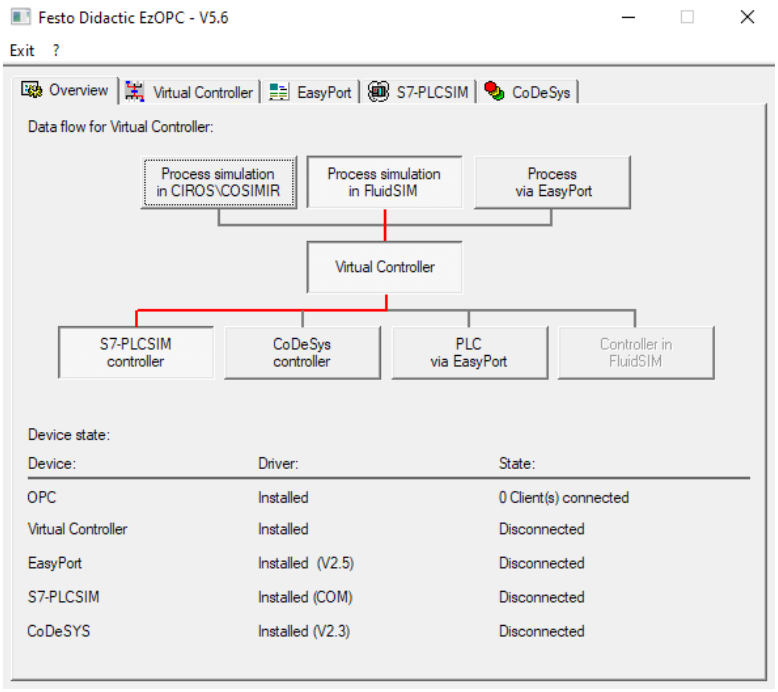


Figure 23. "Overview" panel of EzOPC.

The next step after configuring the connection was to establish the range of input and output bytes of the PLC to be shared via OPC with FluidSIM. The addresses need to be associated with reachable memory areas of the PLC; therefore, it was important to consider a specific PLC model. The selected device was a Siemens Simatic S7-300/CPU 312C. Its inputs cover the byte address range 124-126, and its outputs, the range 124-125. The specifications of the device are covered in Section 5.2. Knowing the IO (Input/Output) range, it was configured in the "S7-PLCSIM" panel of EzOPC (Figure 24). Since the program does not allow to have different ranges for inputs and outputs, the byte address range configured was 124-126; the ranges for analog inputs and outputs were left unchanged because they were not used.

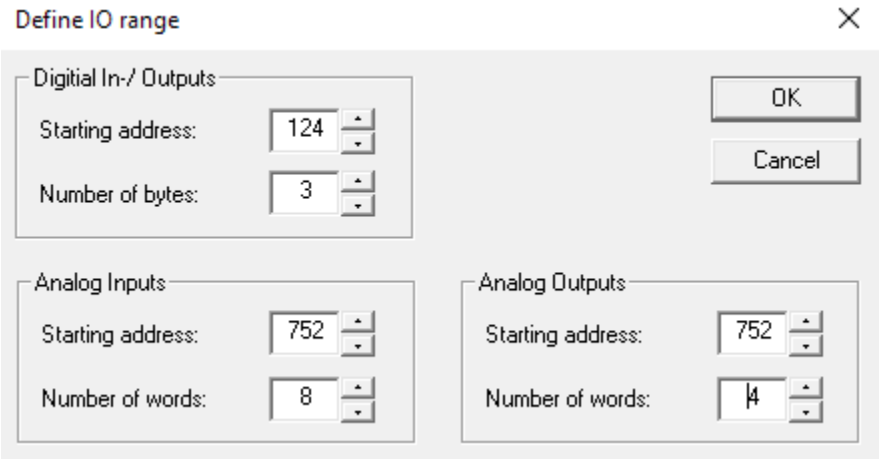


Figure 24. Window for defining the IO range for the bytes being used.

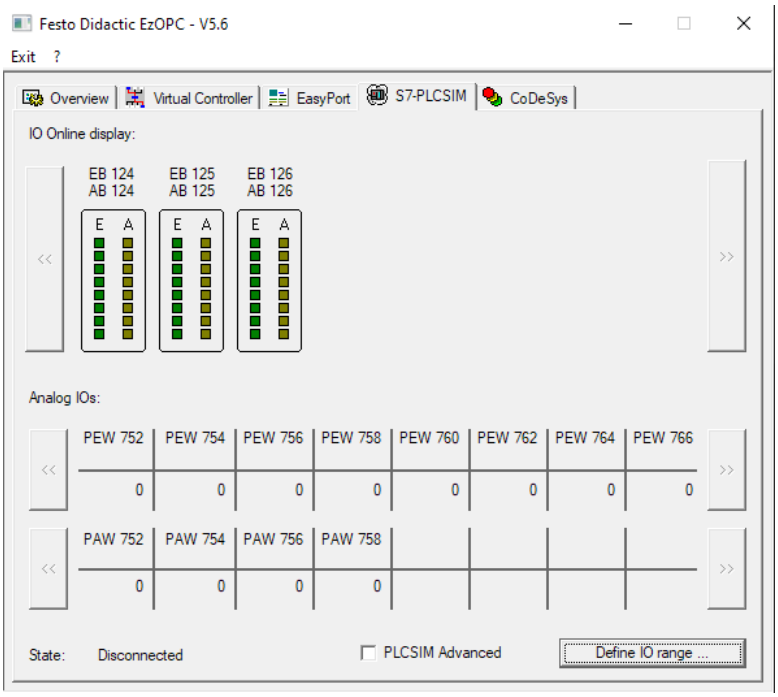


Figure 25. "S7-PLCSIM" panel of EzOPC, after configuring the IO range.

With EzOPC configured (Figure 25), the ports “FluidSIM In” and “FluidSIM Out” needed to be assigned to a specific OPC server and a defined byte address; these actions were done by opening a configuration window by right-clicking on a port and selecting “Properties”. It was necessary to click on the “Select...” button next to each parameter to open a selection window to choose the OPC server (Figure 26) and the address (Figure 27).

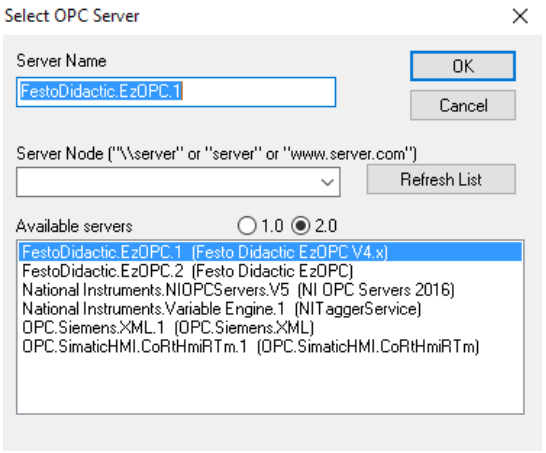


Figure 26. Window to select the OPC server to use in a FluidSIM port.

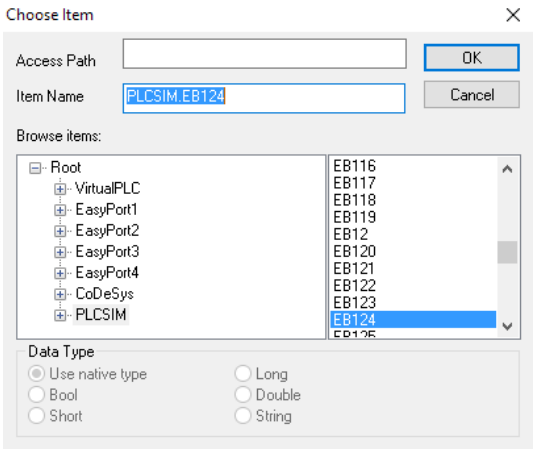


Figure 27. Window to select the address to be accessed by a FluidSIM port.

After configuring the ports, the plant in FluidSIM was completely defined and ready to run in a simulation. The following part of the procedure consisted in creating the simulated PLC and writing a program for it to interact with the plant.

5.3. Programmable Logic Controller (PLC)

As mentioned in the previous section, the PLC selected to control the plant was the Siemens Simatic S7-300/CPU 312C. The specific model is 6ES7 312-5BF04-0AB0. Its characteristics are specified by a datasheet, and the most relevant ones are listed below:

- 10 digital inputs
- 6 digital outputs
- 2 high-speed counters
- MPI communication interface
- 64 kb of work memory

(Siemens, 2021a).

The selected PLC model has the required number of digital inputs and outputs to use with the plant. Additionally, it is part of the S7-300 series, which can be simulated using the S7-PLCSIM software. Recalling the previous section, EzOPC was configured for data exchange between FluidSIM and a controller via S7-PLCSIM.

The entire procedure of creating, programming, and simulating the PLC was done with the software suite TIA (Totally Integrated Automation) Portal, version V15. It includes various programs for programming PLCs, designing Human-Machine Interfaces (HMIs), controlling motion, parametrizing, among other features (Siemens, 2021c). For the current

work, the relevant software contained in TIA Portal is SIMATIC STEP 7 – to program the PLC, and S7-PLCSIM – to simulate it.

The initial step was to create a new project in TIA Portal and select the previously described PLC. Because the focus was on later using OPC for communicating the PLC with a Web Services application based on a VI, it was necessary to also define a communications module connected to the PLC to enable information exchange. The module was a CP 343-1 communications processor, which offers connectivity via Industrial Ethernet, PROFINET, TCP/IP, among others (Siemens, 2021b). The device configuration of the PLC and the module, both mounted on a virtual rail, is shown in Figure 28.

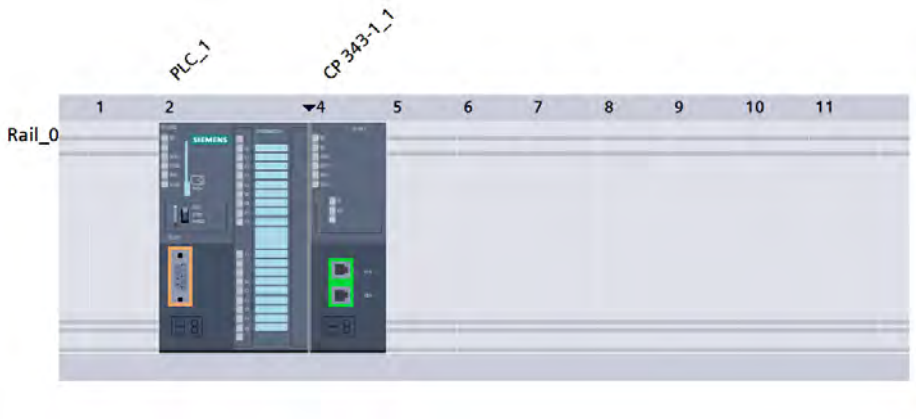


Figure 28. PLC and communications processor, mounted on a virtual rail in TIA Portal.

The following part of the procedure was defining the tags for the inputs, outputs, counters, timers, and memory addresses used in the program. The assignment was done considering the bits used for the input and output ports in FluidSIM, to preserve consistency. The memory addresses were defined to support the internal logic of the program and to

receive control signals from the VI. Furthermore, the purposes of the timers were: (1) to keep the alarm on for two seconds after the deactivation condition occurs, and (2) to set the framerate for the animation shown in the VI. In addition, the counters were defined to: (1) count the bottles in Process 1, and (2) set the current frame shown in the animation of Process 2. The resulting tag table is presented in Figure 29.

PLC tags

Name	Data type	Address	Retain	Accessible from HMI/OPC UA	Writable from HMI/OPC UA	Visible in HMI engineering	Comment
PR1_EN	Bool	%I124.0		True	True	True	Enable Process 1
PR2_EN	Bool	%I124.1		True	True	True	Enable Process
START	Bool	%I124.2		True	True	True	Start (button)
STOP	Bool	%I124.3		True	True	True	Stop (button, NC)
C_CNT	Bool	%I124.4		True	True	True	Choose to count down/up
PC_CNT	Bool	%I124.5		True	True	True	Piece Count
G_LIGHT	Bool	%Q124.0		True	True	True	Green Light
Y_LIGHT	Bool	%Q124.1		True	True	True	Yellow Light
R_LIGHT	Bool	%Q124.2		True	True	True	Red Light
ALARM	Bool	%Q124.3		True	True	True	Alarm for excess of pieces (10+)
MOTOR	Bool	%Q124.4		True	True	True	Conveyor Band Motor
PR1	Bool	%M0.0		True	True	True	Process 1
PR2	Bool	%M0.1		True	True	True	Process 2
PIECE_CNT	Counter	%C0		True	True	True	Piece counter
CNT0_VAL	Int	%MW1		True	True	True	Value for piece counter
Timer_Alarm	Timer	%T0		True	True	True	Timer for alarm (turns itself off)
TMR0_VAL	Int	%MW3		True	True	True	Value for TMR0
TMR_0	Bool	%M0.2		True	True	True	Timer for alarm
Timer_Bottle	Timer	%T1		True	True	True	Timer for animation transition
TMR_1	Bool	%M0.3		True	True	True	Timer for bottle animation
BOTTLE_CNT	Counter	%C1		True	True	True	Bottle counter
CNT1_VAL	Int	%MW5		True	True	True	Value for bottle movement counter
START_HMI	Bool	%M0.4		True	True	True	Start (VI)
STOP_HMI	Bool	%M0.5		True	True	True	Stop (VI)

Figure 29. Tag table for the PLC.

As it can be observed, the tag table contains two columns that represent the permission of tags to be accessible from/written by the OPC UA standard or an HMI. All tags were given the permissions above, because of the goal of connecting the PLC with the VI via OPC UA.

The program for the PLC was created after having the tags defined, using the Ladder Logic (LAD) programming language. It was divided into eleven networks, which can be classified into six functional groups. The first group (Figure 30, encompassing Networks 1 and 2) is used for starting, interlocking, and stopping Processes 1 and 2, respectively.

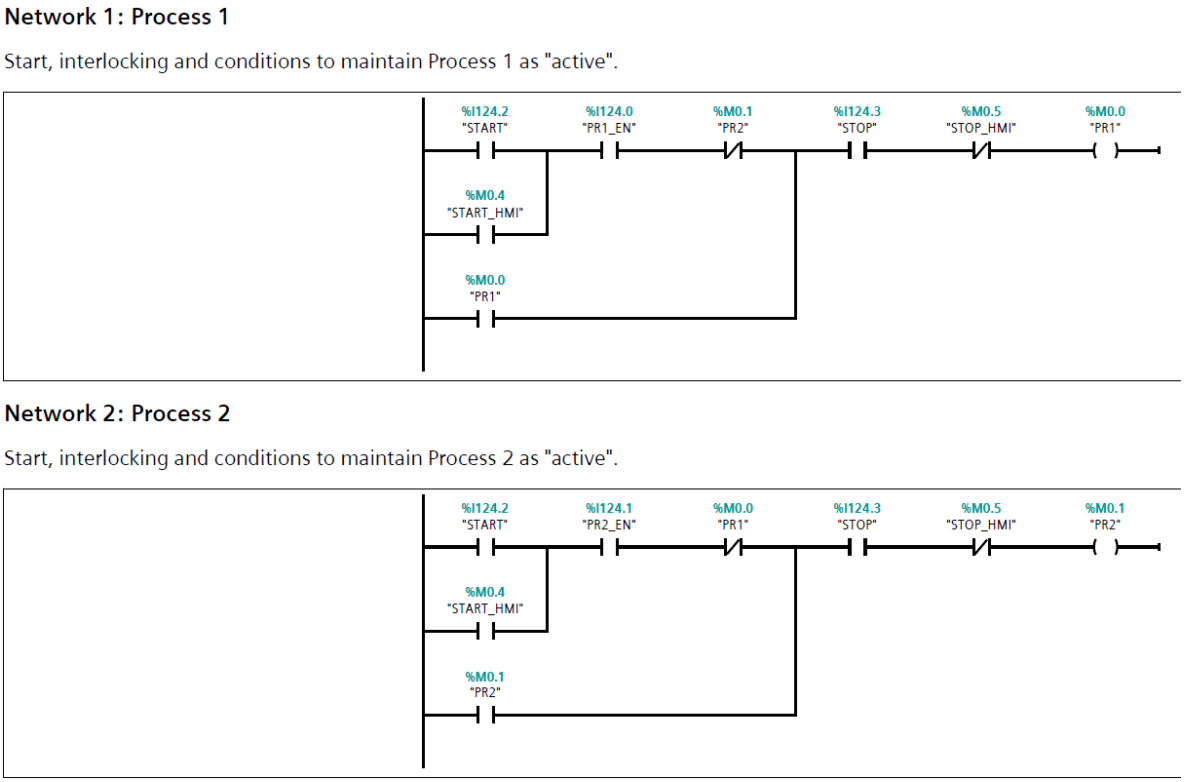


Figure 30. Networks 1 and 2 of the PLC program.

The second group consists only of Network 3 (Figure 31), and its purpose is to perform the counting process, with its value increasing or decreasing according to the piece-counting sensor, “PC_CNT” and the state of the input “C_CNT”, which corresponds to a selection mode switch in the simulated plant.

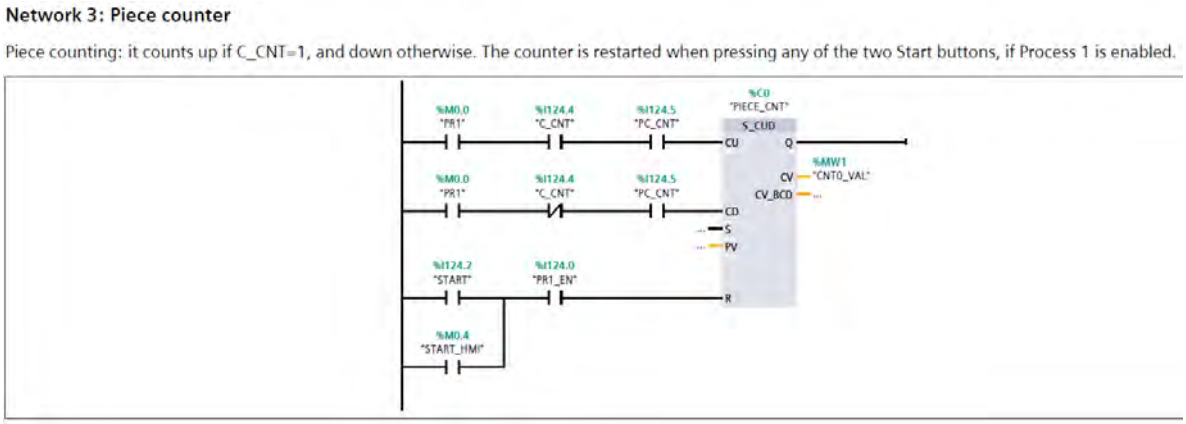


Figure 31. Network 3 of the PLC program.

The third group (Figure 32, including Networks 4-6) corresponds to the three indicator lights of the industrial traffic light used in Process 1. Each network contains conditions that depend on the number of counted bottles: the green light (Network 4) turns on if the value of the counter is $0 < CNT0\_VAL < 5$, the yellow light (Network 5) is active if $5 \leq CNT0\_VAL < 10$, and the red light (Network 6) operates when $CNT0\_VAL \geq 10$.

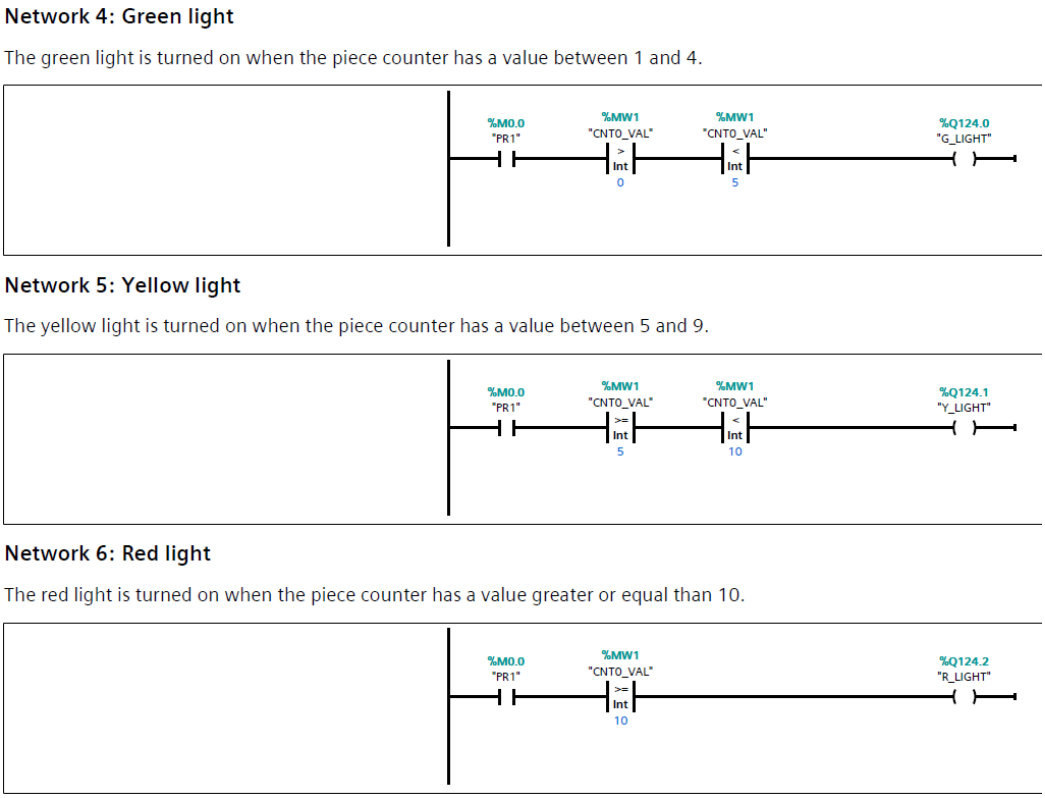


Figure 32. Networks 4-6 of the PLC program.

Group 4 (Figure 33, consisting of Networks 7 and 8) is responsible for maintaining the alarm active up to two seconds after the value of the counter stops being in a critical condition, i.e. greater or equal than 10 (Network 7), and activating it when the counter condition is met (Network 8).

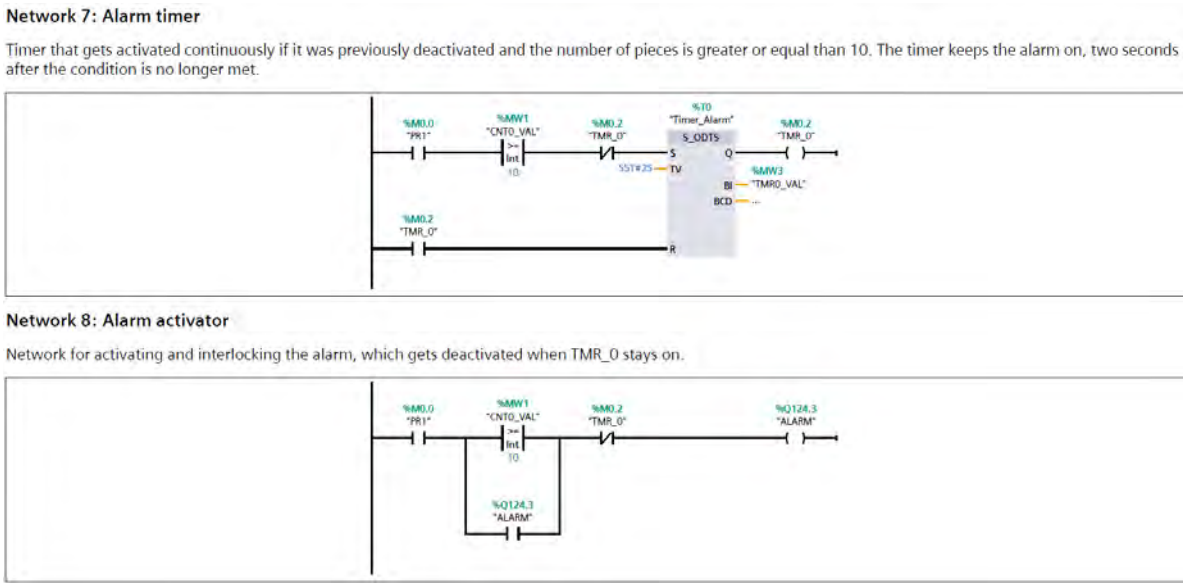


Figure 33. Networks 7 and 8 of the PLC program.

The following group consists only of Network 9 (Figure 34), and its goal is to activate the electric motor of the conveyor band while Process 2 is active.

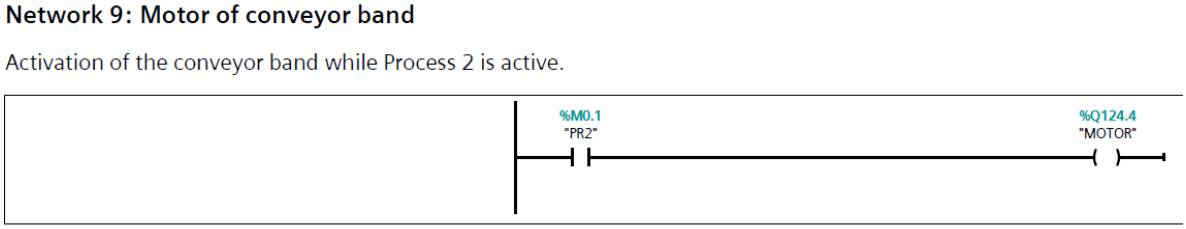


Figure 34. Network 9 of the PLC program.

The purpose of the last group (Figure 35, comprehending Networks 10 and 11) is to control the flow of the animation that gets visualized on the VI to show the position of a bottle moving on the conveyor band. Network 10 consists of a timer that generates a signal

every 200 milliseconds; the signal is received as an increase of the counter in Network 11, and its value determines the current frame of the animation. When the counter reaches 10, it resets to zero.

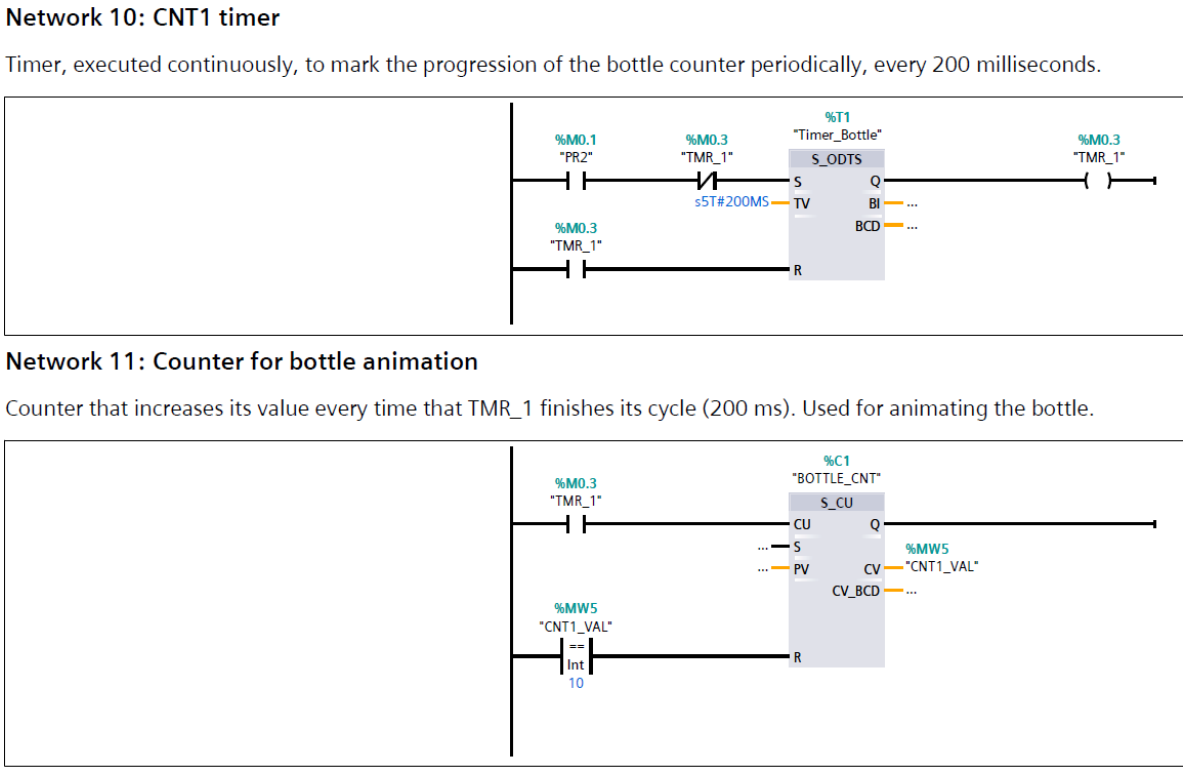


Figure 35. Networks 10 and 11 of the PLC program.

The program presented above was compiled successfully in TIA Portal. After saving the project, S7-PLCSIM was executed and the program was uploaded into it. Since EzOPC was already configured, a test was performed by running the simulation in FluidSIM and setting the PLC in “RUN-P” mode, which enabled program execution and read/write access from the programming interface.

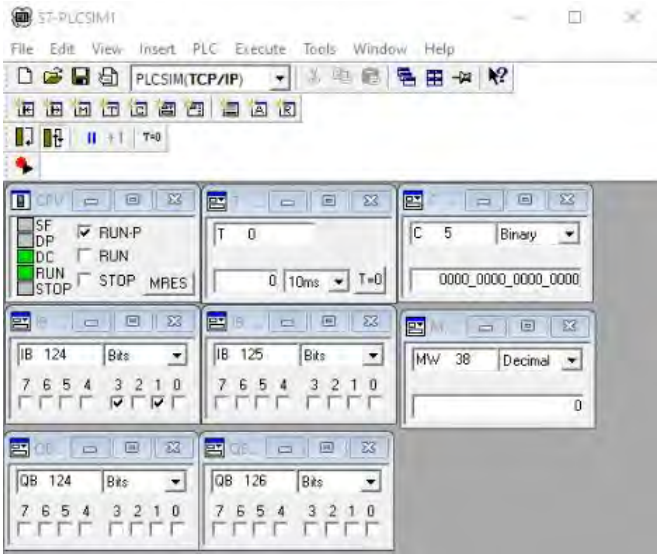


Figure 36. S7-PLCSIM in "RUN-P" mode, running the simulated PLC.

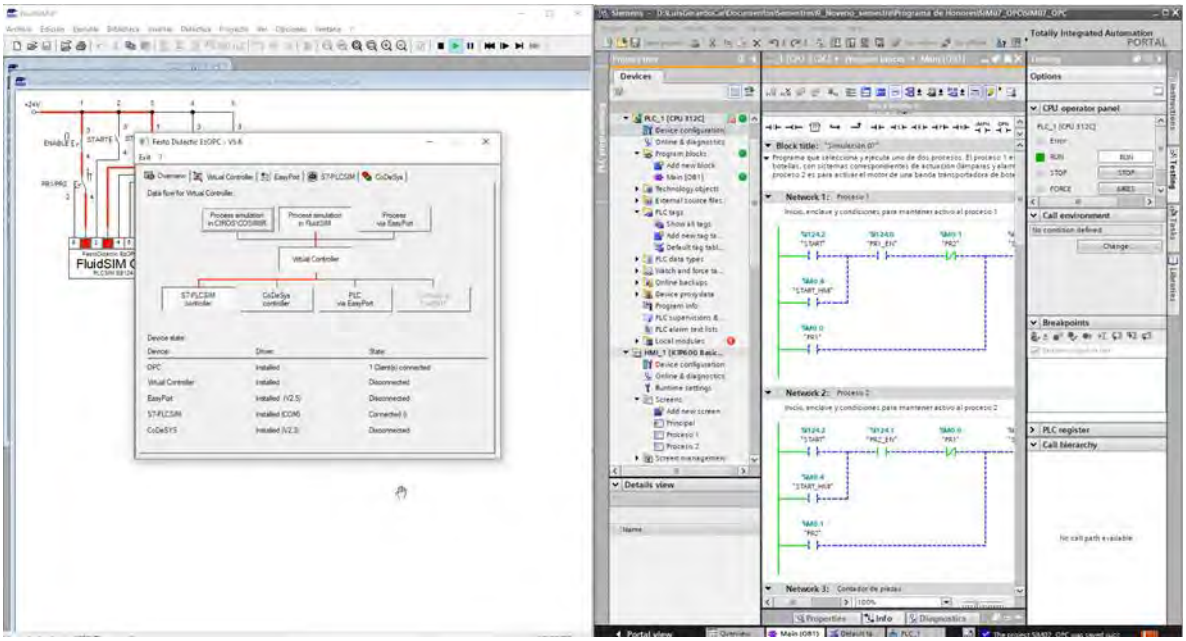


Figure 37. Connection test between FluidSIM (left, with EzOPC superimposed) and TIA Portal (right).

Figure 36 shows the window of S7-PLCSIM, simulating the PLC with the uploaded program in “RUN-P” mode. As it can be observed, the input byte (IB) 124 was receiving signals in bits 1 and 3, which correspond to the signals being sent by FluidSIM, as observed in the left side of Figure 37. Besides, the right side of the same figure shows the PLC program in TIA Portal, with the “Monitoring” function enabled to view the current state of the networks and their tags.

The demonstration validated the successful connection and data exchange between FluidSIM, TIA Portal, and S7-PLCSIM using EzOPC. The focus of the next part of the methodology was to create the OPC UA server to connect S7-PLCSIM with the VI. For that purpose, it was important to define an IP address for the PLC. It was done by accessing the configuration of the communications module CP 343-1. The IP address and subnet masks were kept with their default values, as noticed in Figure 38, and recorded for their use in later steps.

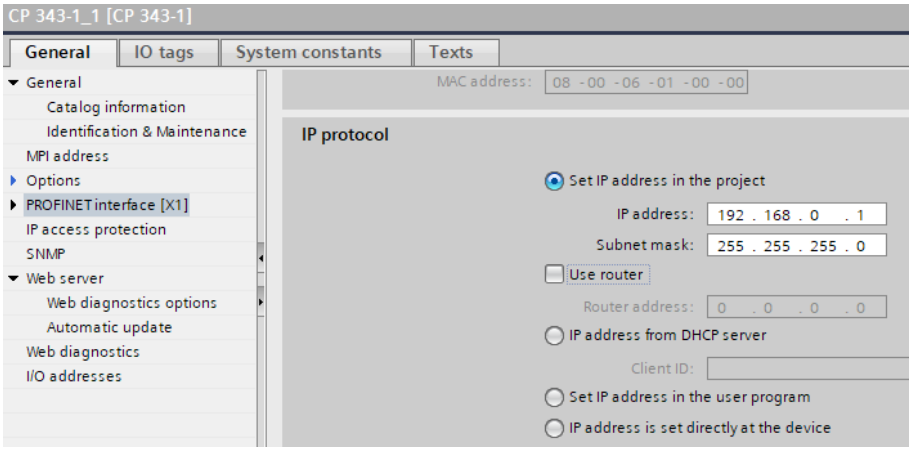


Figure 38. Configuration of the IP address and subnet mask for the PLC.

5.4. OPC UA Server

The purpose of the OPC UA server, as mentioned in previous sections, is to allow data exchange between the simulated PLC in S7-PLCSIM and the VI, created using the software LabVIEW from National Instruments (NI). For that reason, software from NI was used to implement the server, as it offers modules for creating, configuring, and managing OPC UA clients and servers.

Following the instructions provided by National Instruments (2020a; 2020b), the LabVIEW Datalogging and Supervisory Control (DSC) 2020 module and the NI OPC Servers 2016 add-on were installed. The DSC module includes tools for real-time monitoring and for networking LabVIEW elements and OPC devices; NI OPC Servers offers functionality for creating and managing channels and devices in an OPC UA server, along with providing a client that connects to LabVIEW applications.

The NI OPC Servers software was used to add the PLC to the server. It is relevant to highlight that the OPC server starts automatically when the software is active. Thus, after opening the software a new channel was created by using the creation wizard found in the "Edit" menu. After defining a name for the channel, the device driver was selected as "Siemens TCP/IP Ethernet" (see Figure 39) to use with the IP address defined in the previous section for the PLC.

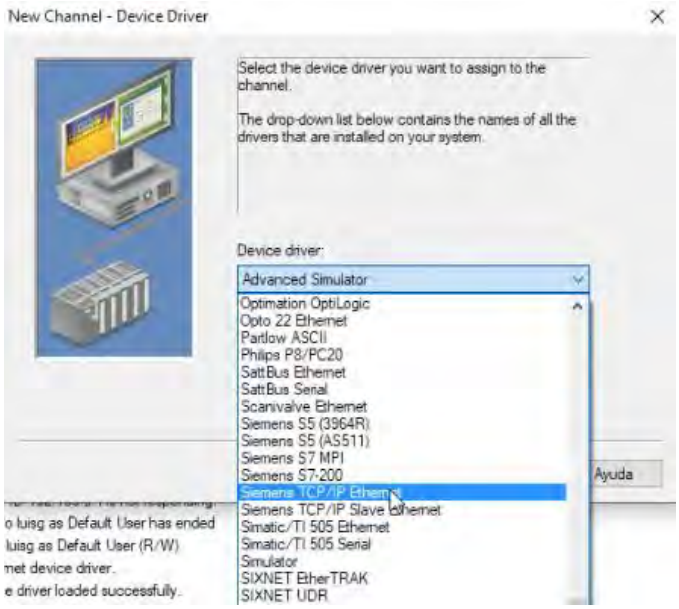


Figure 39. Selection of the device driver for the new channel in NI OPC Servers.

The remaining steps of the wizard were followed leaving all parameters as default. The summary of the channel creation is shown in Figure 40.

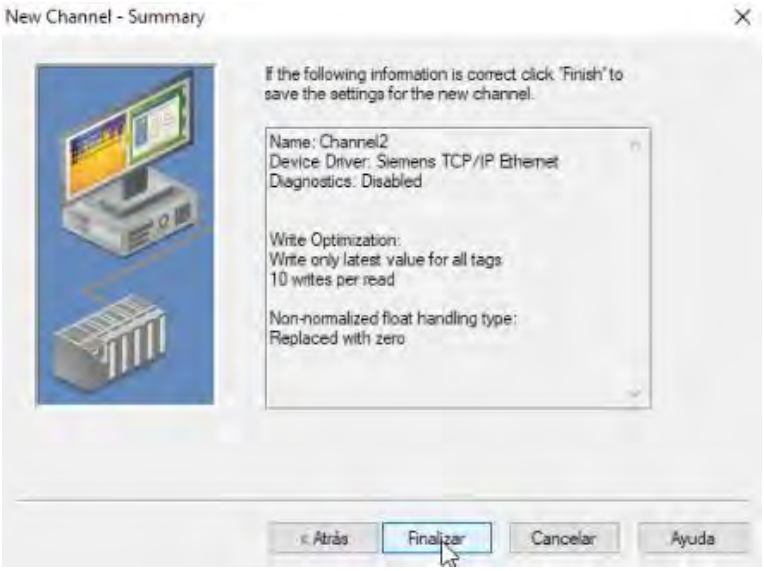


Figure 40. Summary of the settings for the new channel in NI OPC Servers.

Consequently, a new device was added to the channel by right-clicking on it and selecting “New device”. Following the wizard, the device model was selected to coincide with the PLC model (S7-300), as seen in Figure 41.

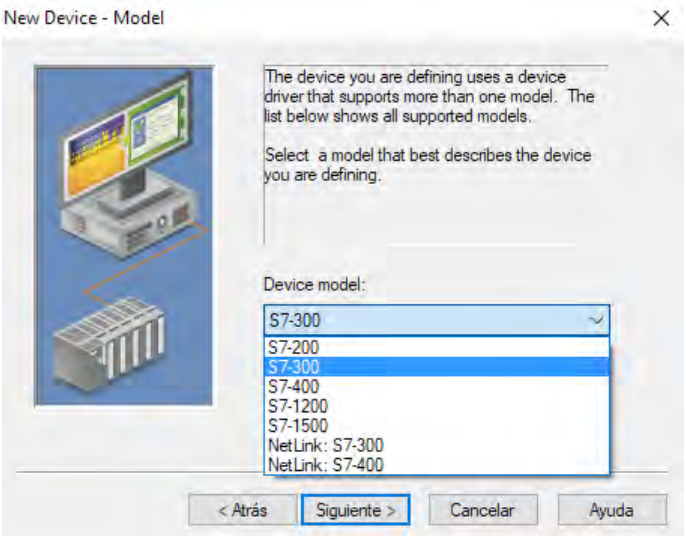


Figure 41. Selection of the model of the new device, created for the OPC server channel.

The next step of the wizard requested an ID (IP address) for the model; a temporary address without physical significance (1.1.1.1) was assigned since an additional step, presented further below, is required to use the PLC IP address with NI software. The TCP/IP port number was also left in its default value (102). The summary of the new device is presented in Figure 42.

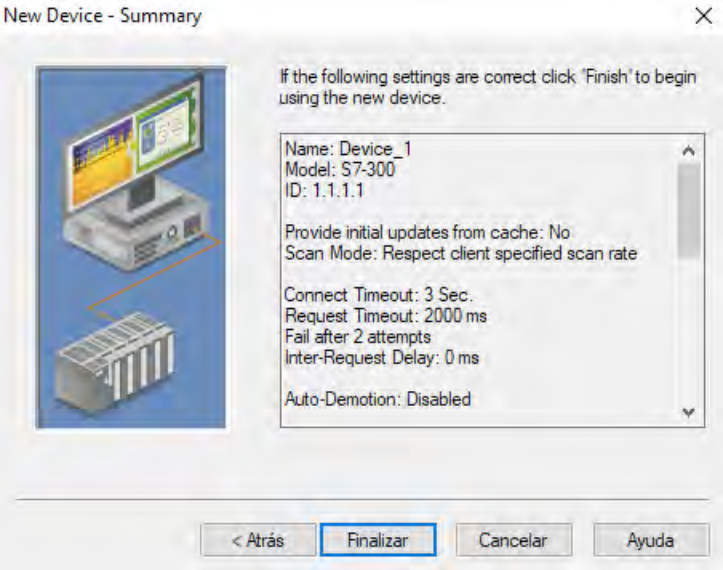


Figure 42. Summary of the settings for the new device in NI OPC Servers.

As mentioned previously, an additional step was needed because the IP address used in S7-PLCSIM is private and cannot be directly used with external applications such as the NI OPC server. The tool NetToPLCsim was used for that purpose; it acts as a network extension of the PLC simulation via TCP/IP (Wiens, n.d.). Figure 43 illustrates the functionality provided by the tool.

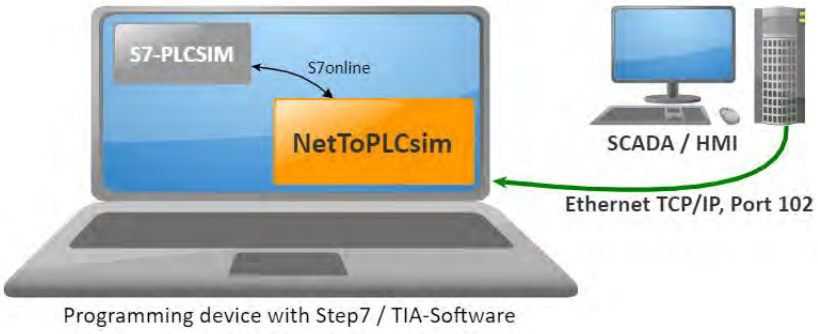


Figure 43. Network functionality provided by NetToPLCsim. (Wiens, n.d.)

NetToPLCsim requires only two parameters to function: the PLC IP address, and the destination IP address to which it is redirected. To ensure a correct address redirection, the IP address of the computer used for programming was obtained by using the "ipconfig" command in Windows Command Prompt (Figure 44).

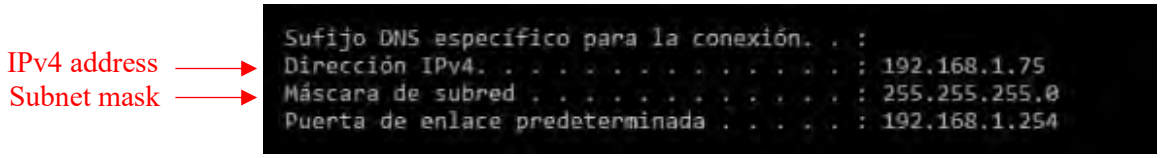


Figure 44. IPv4 address and subnet mask of the computer (Windows Command Prompt).

The IP address to redirect the PLC in the network was defined after obtaining the IP address (IPv4) of the computer, considering that the subnet mask is 255.255.255.0. The resulting address was 192.168.1.74, created to be located in the same subnet as the computer but with a different value to avoid interference. The resulting configuration of NetToPLCsim is presented in Figure 45.

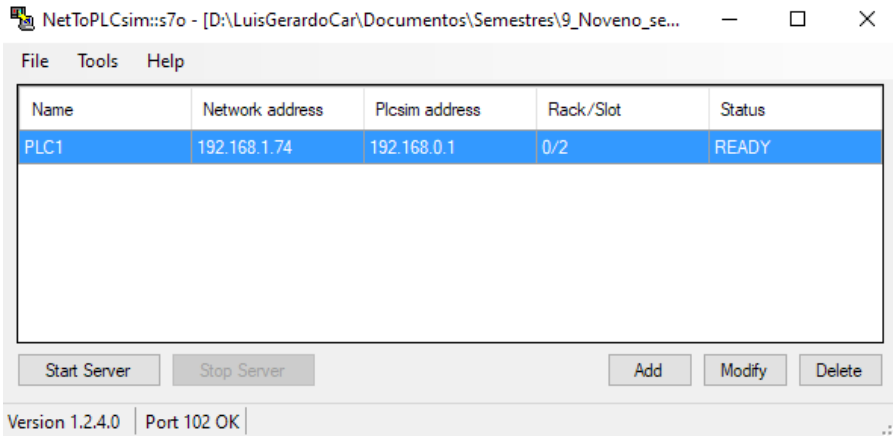


Figure 45. NetToPLCsim, after configuring the IP addresses for the network and S7-PLCSIM, respectively.

With the address defined, NetToPLCsim was started using the “Start Server” button. Next, the properties panel of the recently created device in NI OPC Servers was opened, and the network IP address was set for the “ID” parameter (Figure 46).

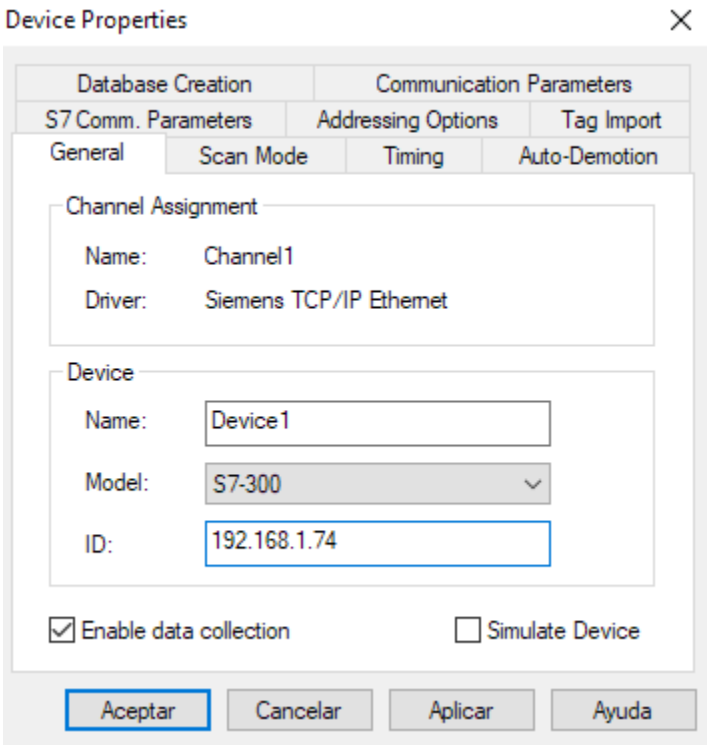


Figure 46. “General” tab of the device properties panel.

Afterward, the tags for the variables to share between the PLC and the VI were defined. A tag defined in NI OPC Servers is interpreted as an item in the OPC UA standard. For that purpose, the “New Tag” option, found when right-clicking on the device in NI OPC Servers, was selected. The window to create a new tag requests relevant parameters of the tag such as its name, address, description (optional), data type, access (read, write,

read/write), and scan rate (set as 100 milliseconds by default). An example of the tag properties definition for the variable “PR1_EN” is presented in Figure 47.

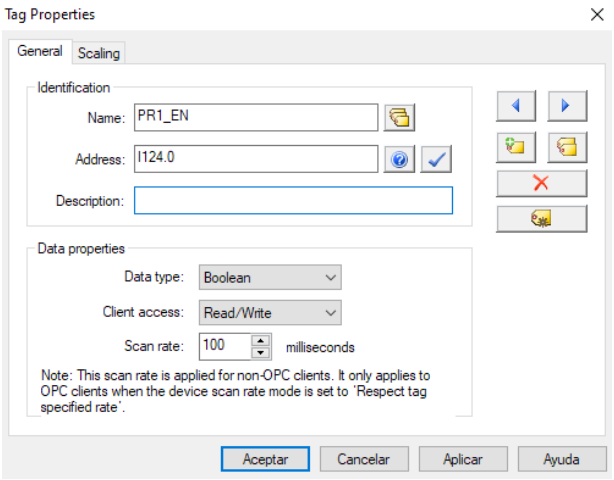


Figure 47. Example of the creation of a tag in NI OPC Servers.

Since the tag table for the PLC was already defined, the procedure for creating the tags for NI OPC Servers implicated selecting the tags that were relevant to share via OPC UA with the VI. The resulting tag table for the OPC server is shown in Figure 48.

Tag Name	Address	Data Type	Scan Rate	Scaling
PR1_EN	I124.0	Boolean	100	None
PR2_EN	I124.1	Boolean	100	None
START	I124.2	Boolean	100	None
STOP	I124.3	Boolean	100	None
C_CNT	I124.4	Boolean	100	None
PC_CNT	I124.5	Boolean	100	None
PR1	M0.0	Boolean	100	None
PR2	M0.1	Boolean	100	None
START_HMI	M0.4	Boolean	100	None
STOP_HMI	M0.5	Boolean	100	None
CNT0_VAL	MW1	Word	100	None
TMR0_VAL	MW3	Word	100	None
CNT1_VAL	MW5	Word	100	None
G_LIGHT	Q124.0	Boolean	100	None
Y_LIGHT	Q124.1	Boolean	100	None
R_LIGHT	Q124.2	Boolean	100	None
ALARM	Q124.3	Boolean	100	None
MOTOR	Q124.4	Boolean	100	None

Figure 48. Tag table for the NI OPC server, sorted by address.

The definition of tags was the last step to establish the OPC server for data exchange between the PLC and LabVIEW applications (in this case, the VI). Before proceeding to create the VI, a connection test was performed to verify that the OPC server was connected to the simulated PLC in S7-PLCSIM. For that, a monitoring client called OPC Quick Client was launched; it is contained in NI OPC Servers and can be launched from the "Tools" menu. Once opened, the client shows information about the items associated with tags: item ID, data type, value, timestamp, quality, and update count (number of times the item has changed value). Figure 49 shows an example of the items monitored by OPC Quick Client, and Figure 50 presents a test performed between FluidSIM, S7-PLCSIM, and the NI OPC server to verify the successful connection between programs.

Item ID	Data Type	Value	Timestamp	Quality	Update Count
Channel1.Device1.MOTOR	Boolean	0	18:16:08.152	Good	17
Channel1.Device1.PC_CNT	Boolean	0	16:31:40.159	Good	5
Channel1.Device1.PR1	Boolean	0	16:32:13.149	Good	7
Channel1.Device1.PR1_EN	Boolean	0	20:42:55.574	Good	11
Channel1.Device1.PR2	Boolean	0	18:16:08.152	Good	17
Channel1.Device1.PR2_EN	Boolean	1	20:42:55.574	Good	9
Channel1.Device1.R_LIGHT	Boolean	0	16:31:40.159	Good	5
Channel1.Device1.START	Boolean	0	16:31:40.159	Good	5
Channel1.Device1.START_HMI	Boolean	0	16:32:12.150	Good	9
Channel1.Device1.STOP	Boolean	1	16:32:08.151	Good	6
Channel1.Device1.STOP_HMI	Boolean	0	18:16:09.149	Good	13
Channel1.Device1.TMR0_VAL	Word	0	16:31:40.159	Good	5
Channel1.Device1.Y_LIGHT	Boolean	0	16:31:40.159	Good	5

Figure 49. Example of item monitoring using OPC Quick Client.

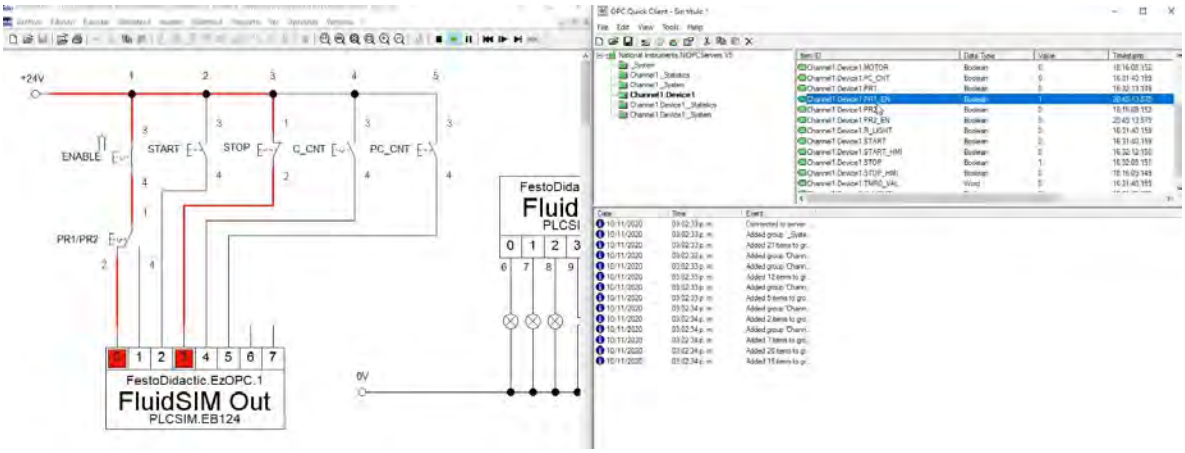


Figure 50. Connection test between FluidSIM (left) and a client of the NI OPC server (right).

5.5. Virtual Instrument (VI)

The purpose of the Virtual Instrument is to serve as an interface for an operator or administrator to visualize information regarding the processes clearly and concisely, and to control their start/stop state. As mentioned in previous sections, the software used to develop the VI is NI LabVIEW (2020, 32-bit version). Described by National Instruments (n.d.), it is a "[...] systems engineering software for applications that require test, measurement, and control with rapid access to hardware and data insights."

A blank LabVIEW project was first created. Consequently, the LabVIEW software automatically generated a front panel (containing the interactive visual elements of the VI) and a block diagram (comprising the interactions and functional behavior of the elements in the Front Panel). The front panel was initially developed as its elements need to be defined prior to establishing their functionality. The elements used in the panel were added considering the variables that were defined in the NI OPC server. Additionally, it was

important to focus the design towards the actual processes meant to be visualized in the VI: the counting of plastic bottles and their transport in a conveyor band.

Taking the previous considerations into account, indicator lights were added to the panel to display the process selected by the switches in FluidSIM, the active process, the industrial traffic light associated with the value of the bottle counter, and the alarm used in Process 1. A text indicator was included on top of the industrial traffic light to show the current value of the counter. Also, “Start” and “Stop” buttons were added to remotely control the activation/deactivation of the currently selected process. Finally, “Picture Ring” elements were introduced to create animations showing the movement of a bottle in a conveyor band, and the rotation of the associated motor. The elements added up to this point can be observed in Figure 51.

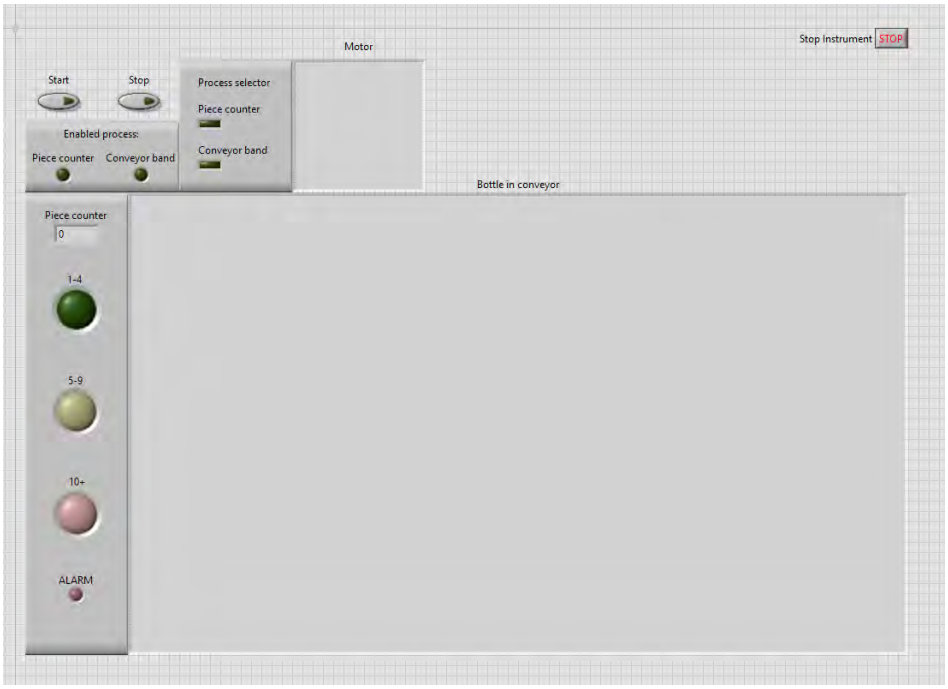


Figure 51. Elements of the Front Panel of the VI before adding animation frames.

The process of creating the animations involved using the blank “Picture Ring” objects under the “Motor” and “Bottle in conveyor” titles in Figure 51. An animation consists of various frames (individual images) switching in a determined order at constant time intervals. Thus, the image edition software Photoshop was used to create each of the frames used for both animations. For the animation of the bottle moving in a conveyor band, eleven frames were created to coincide with the number of values delivered by the counter that was previously defined when programming the PLC; on each frame, the bottle was displaced to the right at regular intervals. In the case of the animation of the motor, the drawing of a gear was rotated proportionally to generate 14 frames. The resulting frames are shown in Figures 52 and 53.

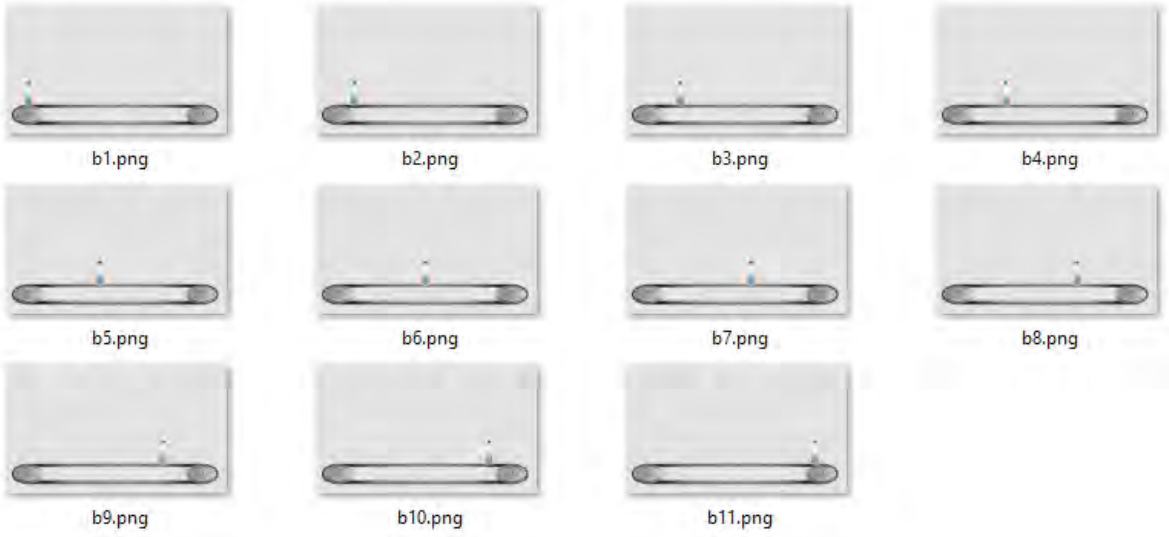


Figure 52. Collection of frames created for the “Bottle in conveyor” animation.

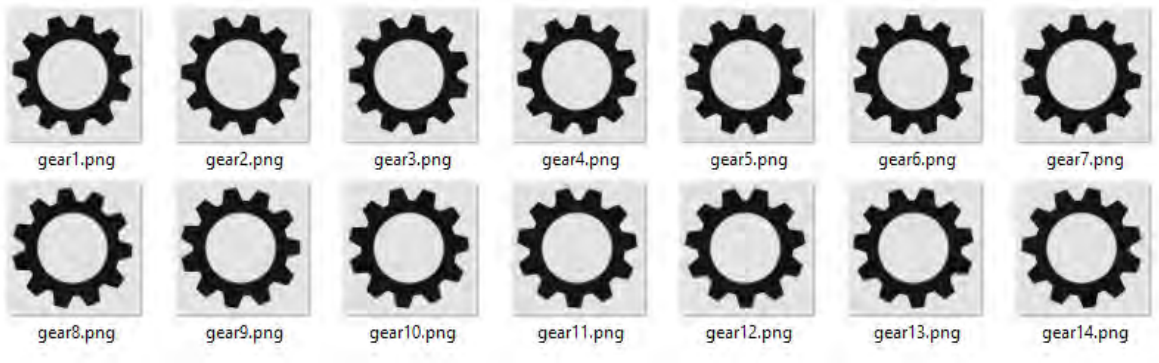


Figure 53. Collection of frames created for the "Motor" animation.

The instructions provided by Travis & Kring (2007, pp. 857-860) were used to import the previously created frames to the "Picture Ring" elements in order, by dragging and selecting "Insert Picture After" in the right-click menu for each frame. The resulting front panel for the VI is shown in Figure 54.

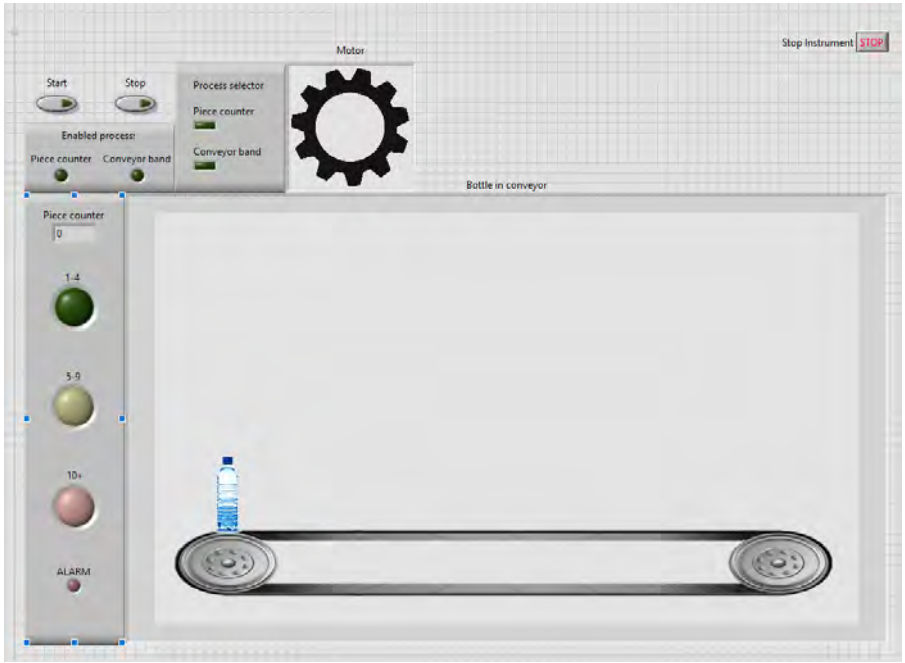


Figure 54. Elements of the Front Panel of the VI after adding animation frames.

The next part of the process was to import the tags from the NI OPC server into variables that could be used by LabVIEW. For that, an OPC Client I/O sever needed to be defined in the VI. Figure 55 illustrates the data flow that occurs after configuring the client with the already-existing server.

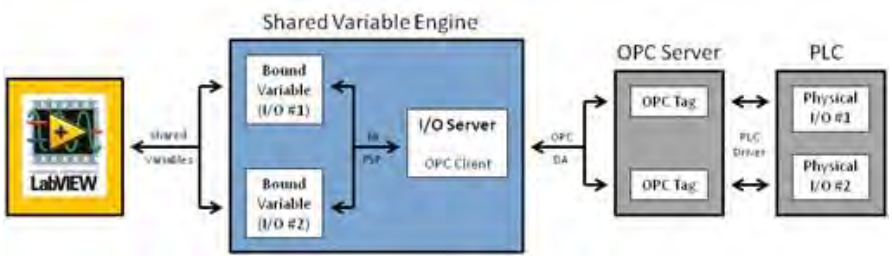


Figure 55. Data flow between LabVIEW and a PLC using an I/O server. (National Instruments, 2019)

The I/O server was created by right-clicking “My Computer” on the project menu in LabVIEW, and selecting “New” → “I/O Server”. Then, the type of the server was set as “OPC Client”. In the configuration window, the server “National Instruments.NIOPCServers.V5” was selected; it corresponds to the NI OPC server configured in the previous section. The update rate was set to 100 milliseconds, to coincide with the scan rate of the tags in the server. The resulting configuration is shown in Figure 56.

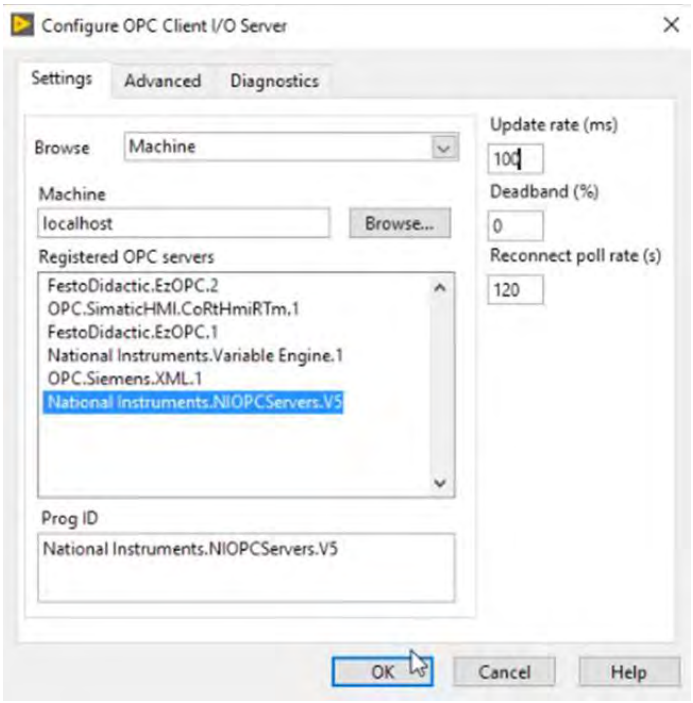


Figure 56. Configuration of the OPC Client I/O Server for the VI in LabVIEW.

Subsequently, a new library was added to the project. It was used to import the OPC server tags as variables, by right-clicking on the library and selecting “Create Bound Variables...”. In the window that appeared, the desired tags were selected from a browser and added to LabVIEW (Figure 57).

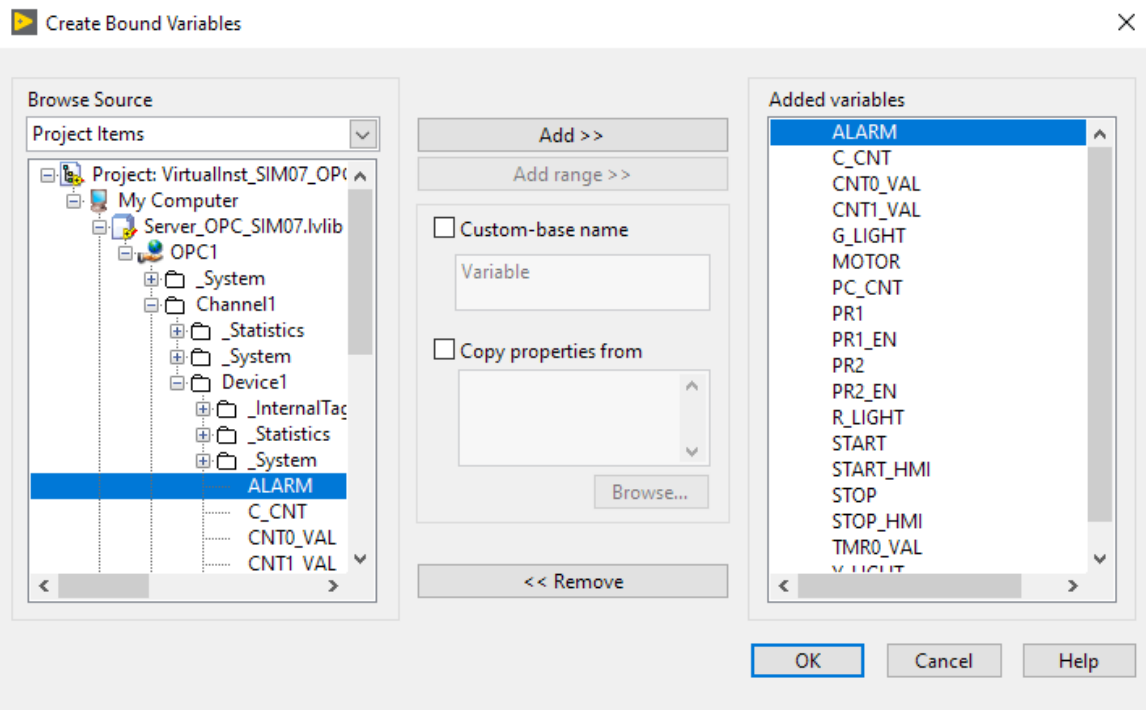


Figure 57. Creation of bound variables in LabVIEW from tags in the NI OPC server.

The bound variables were available for use after being added. Thus, the next step was to use the block diagram window to assign variables to the previously defined elements in the front panel. The process consisted in dragging each variable from the project menu to the block diagram and then connecting it to its associated element with a wire. All variables were added to the diagram as inputs by default; so, for the cases where the variables were used as outputs, their access mode was redefined from “Read” to “Write” in their right-click menu.

As National Instruments (2020b) indicates, an additional stage was needed to make the VI function with the bound variables. It was done by enclosing all elements in the block diagram inside a while loop structure. Then, a Stop control was added to the loop by right-clicking on the loop condition (red square) and selecting “Create Control”; this allows a user

of the VI to stop it when needed. Afterward, a “Wait Until Next ms Multiple” element was added to the block diagram inside the while loop. Its function is to set the iteration rate for the loop, and it was assigned a value of 100 ms to coincide with the update rate of the tags in the OPC server.

A control structure was designed inside the while loop to manage the animations shown in the front panel. The “Picture Ring” elements were placed inside a case structure, which acted like a conditional block that was enabled only when the variable “MOTOR” was active, i.e. when Process 2 is active. The value of the bottle counter was directly connected to the “Bottle in conveyor” animation. In contrast, the “Motor” animation was connected to the result of a mathematical operation that obtained the residue of the division between the loop iteration count value (represented by the letter “i” inside a blue box) and a constant. The operation allowed the animation to switch frames continuously at regular intervals, to give the appearance of a rotating motor. A detailed view of the control structure is shown in Figure 58, and the complete block diagram of the VI is presented in Figure 59.

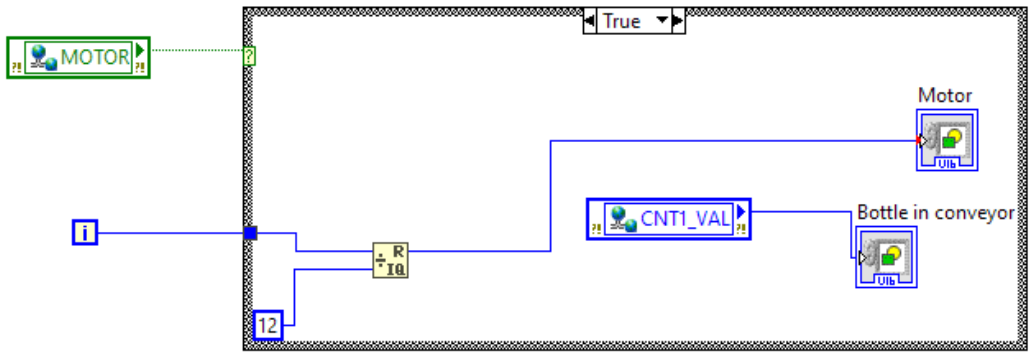


Figure 58. Control structure for showing animations in the VI.

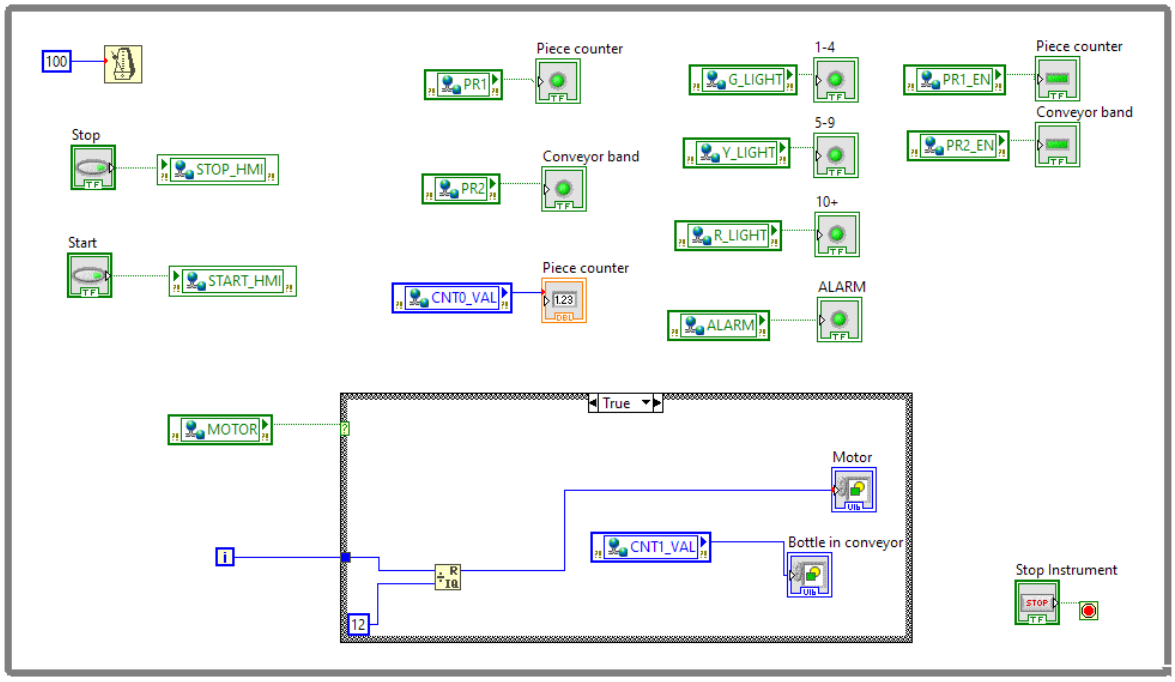


Figure 59. Block diagram of the VI.

Configuring the animations was the final part of the development process for the VI. A test was performed by running the VI to verify that the bound variables and the block diagram were operating correctly (see Figure 60). This test followed the same procedure of the tests presented in previous sections and validated the connection between FluidSIM, the simulated PLC, and the VI through the use of OPC server.

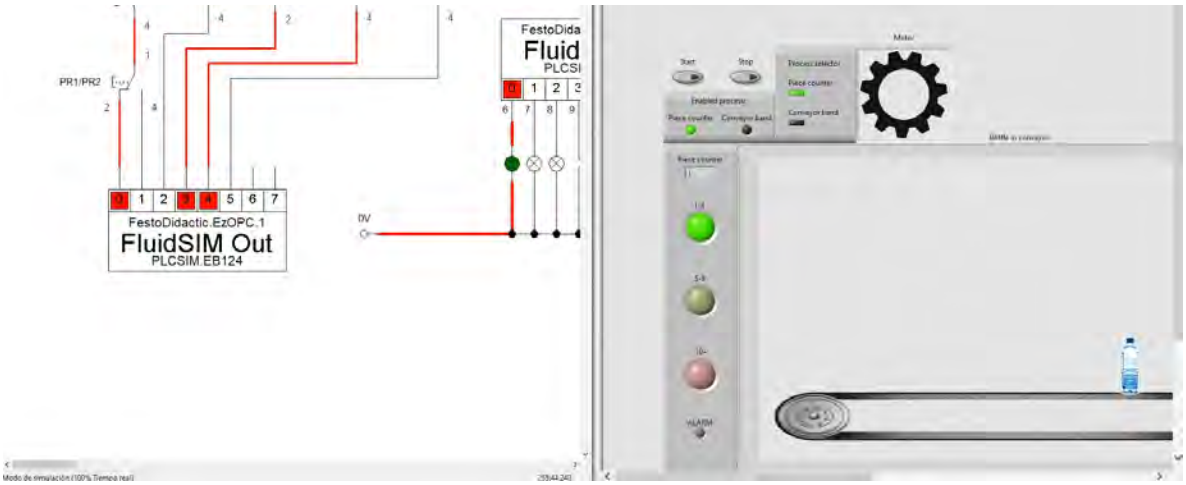


Figure 60. Connection test between FluidSIM (left) and the VI (right).

5.6. Web Services application

The concluding part of the methodology was the implementation of the VI as a Web Services application. It was carried out by using a tool provided by LabVIEW, called Web Publishing Tool, which can be accessed from the "Tool" tab in the project browser. Once opened, a wizard appears and provides three options (or viewing modes) for publishing: (1) embedded, for running a web browser version of the VI, (2) snapshot, for a single static image of the VI that only gets updated after refreshing the page, and (3) monitor, for an image of the VI that updates periodically, but does not allow user interaction. The first option (embedded) was selected, to achieve the goal of monitoring and controlling the processes. The window with the aforesaid configuration is presented in Figure 61.

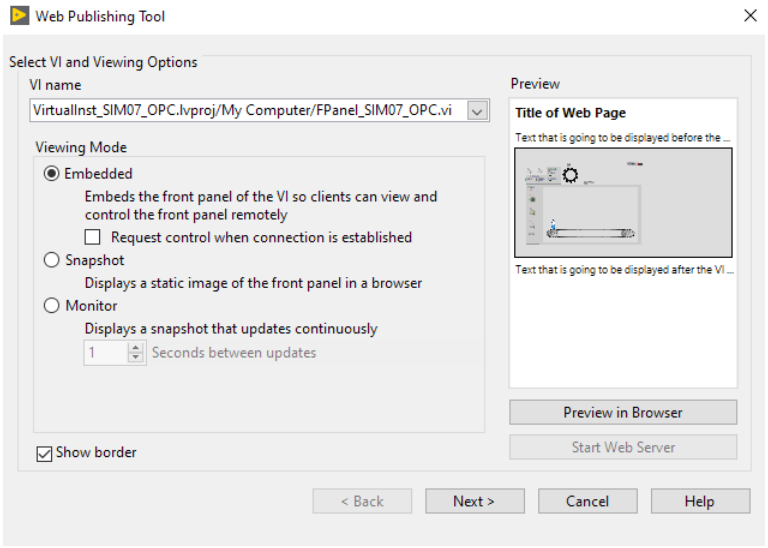


Figure 61. Viewing Mode selection menu of the Web Publishing Tool.

Once the viewing mode was selected, the next menu of the wizard (Figure 62) contained three fields to write the page title, header, and footer of the HTML output; each field was filled with descriptive information related to it.

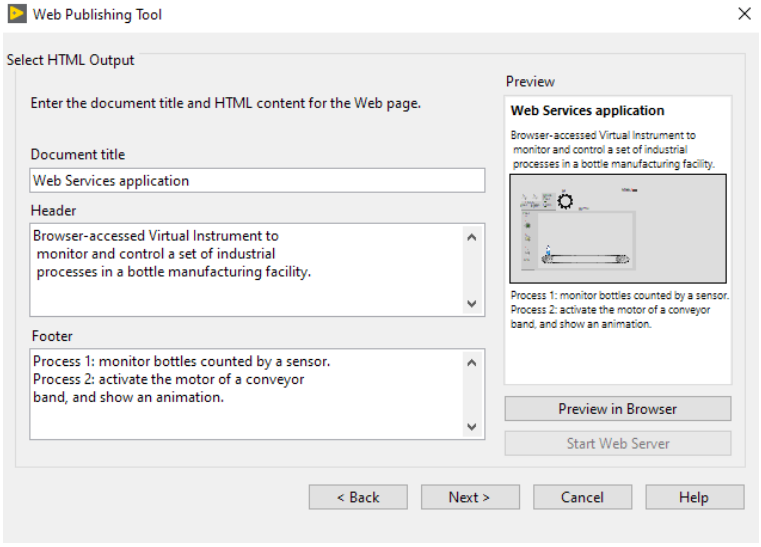


Figure 62. HTML output menu of the Web Publishing Tool.

The last step of the wizard (Figure 63) requested a local directory to save the web page and a filename that corresponds to the last part of the web address (URL). After setting both fields, the web page was saved to the local directory as an HTML file.

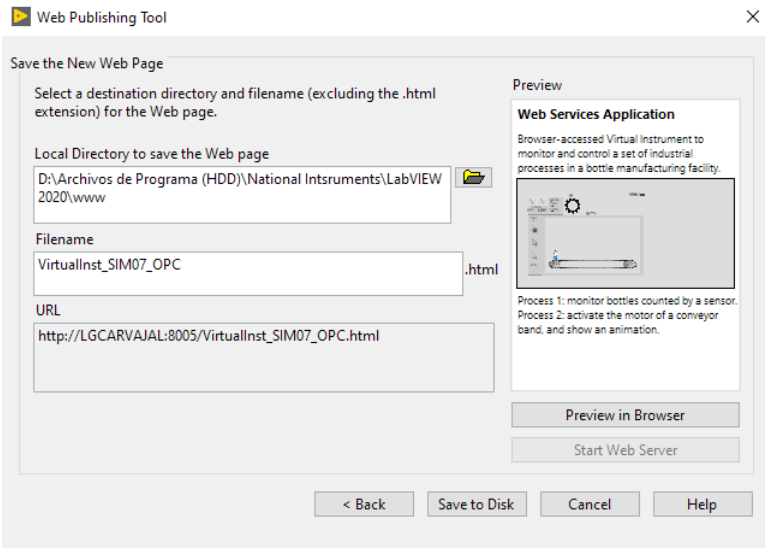


Figure 63. New Web Page menu of the Web Publishing Tool.

The Web Services application could theoretically be accessed by writing the URL in any web browser. However, most of them showed an error when trying to access the page. The title, header, and footer appear, but the VI application did not load, as seen in Figure 64.

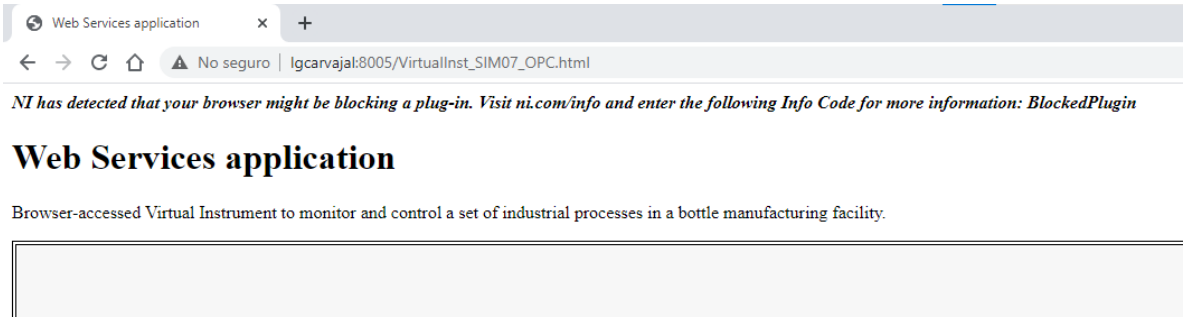


Figure 64. BlockedPlugin error after trying to access the web page from Google Chrome.

The only web browser that successfully deployed the application was Internet Explorer, due to legacy features. Additionally, local antimalware software had to be configured to allow LabVIEW and its modules through firewalls. By doing the above, the VI application was adequately deployed in the web browser, as Figure 65 shows.

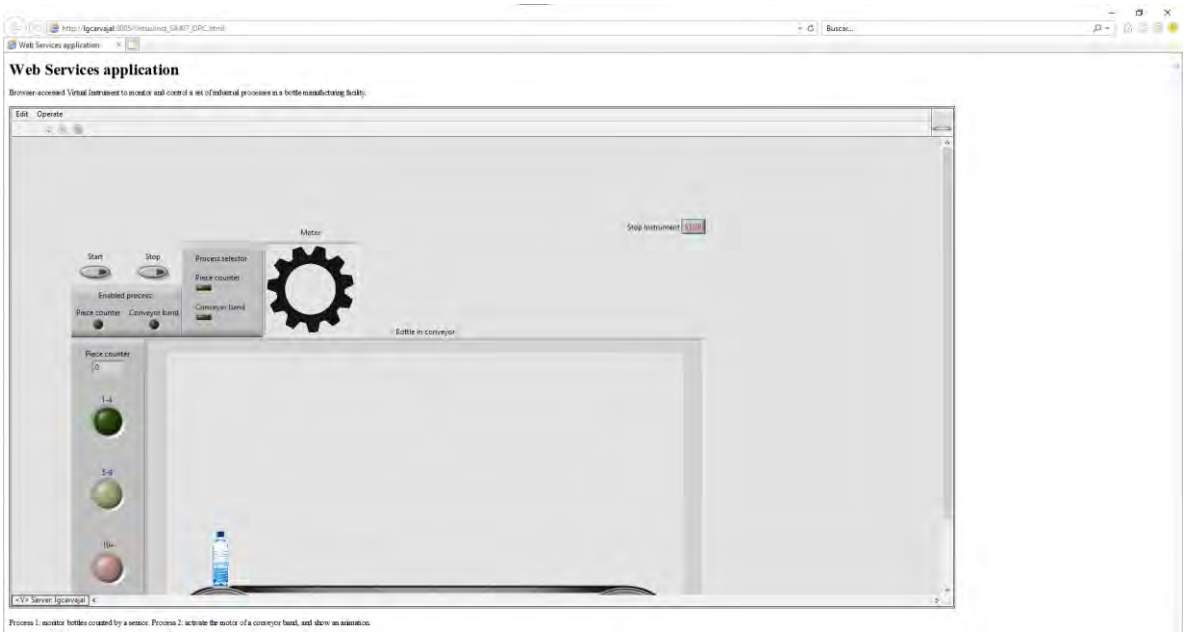


Figure 65. Virtual Instrument deployed as a Web Services application in Internet Explorer.

6. Results and Discussion

The methodology successfully allowed to connect all the intended elements: the simulated plant in FluidSIM was connected with the simulated PLC in S7-PLCSIM via EzOPC; the latter became an OPC client of the NI OPC server and was able to exchange information with the VI in LabVIEW. Furthermore, the VI was executed as a Web Services application in Internet Explorer.

Images are presented below to demonstrate the abovementioned; each one shows the simulated plant in FluidSIM on the left and the VI as a Web Services application in Internet Explorer on the right. Figures 66-69 illustrate Process 1 (counting bottles and activating the industrial traffic light and the alarm according to the value); Figures 70 and 71 evidence the execution of animations in Process 2 (activating the motor of a conveyor band to transport bottles).

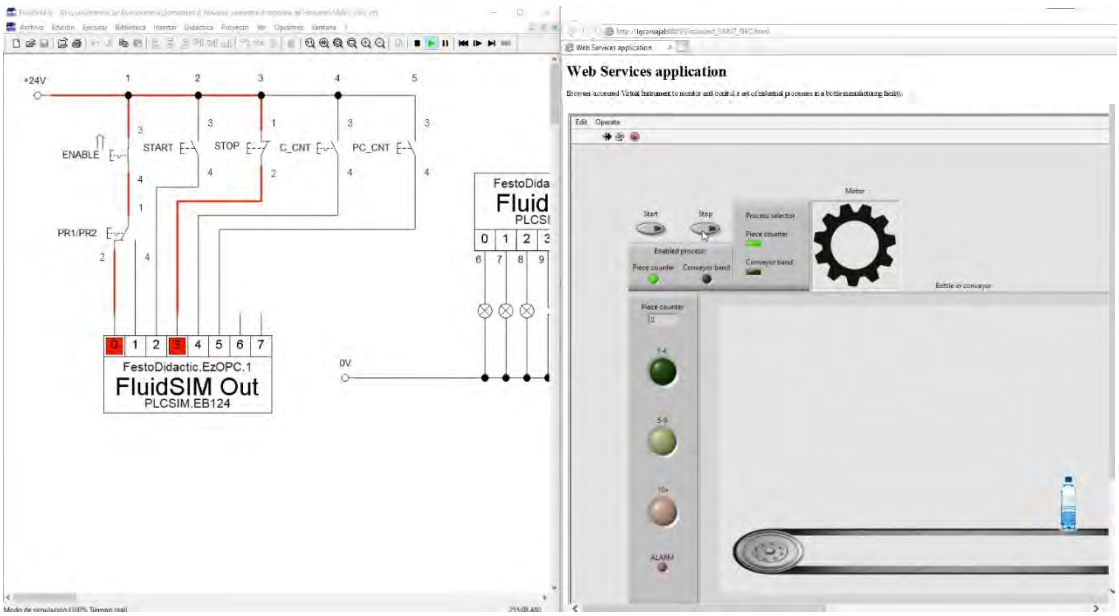


Figure 66. Execution of Process 1 in the plant and application: no bottles counted.

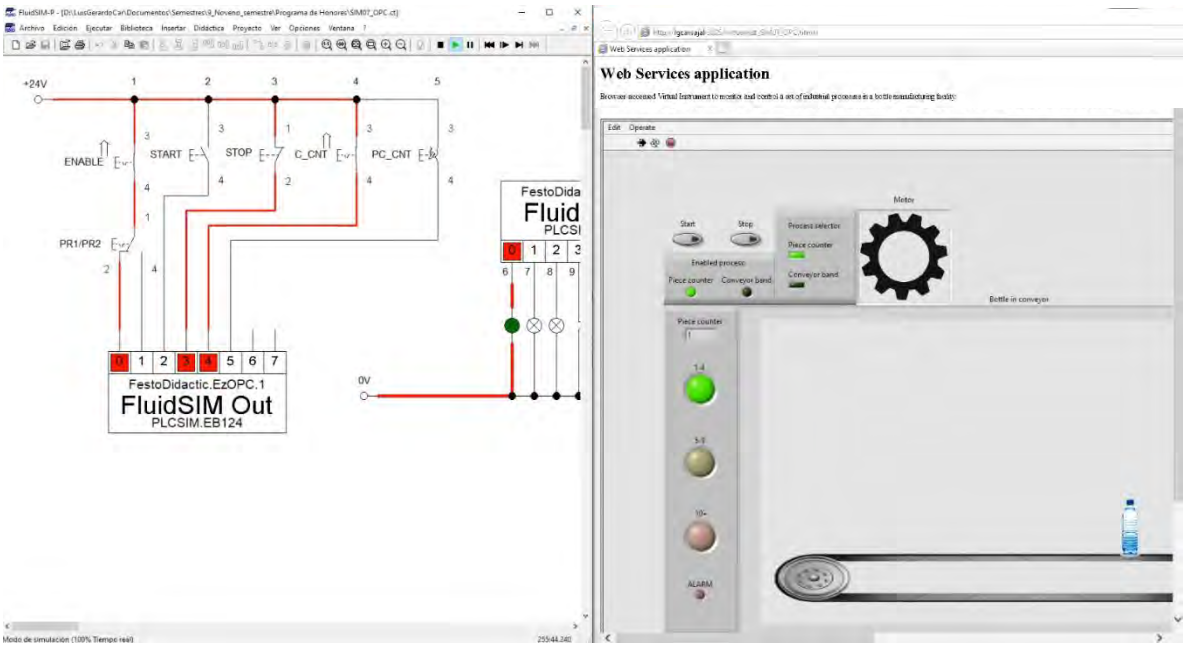


Figure 67. Execution of Process 1 in the plant and application: 1-4 bottles counted.

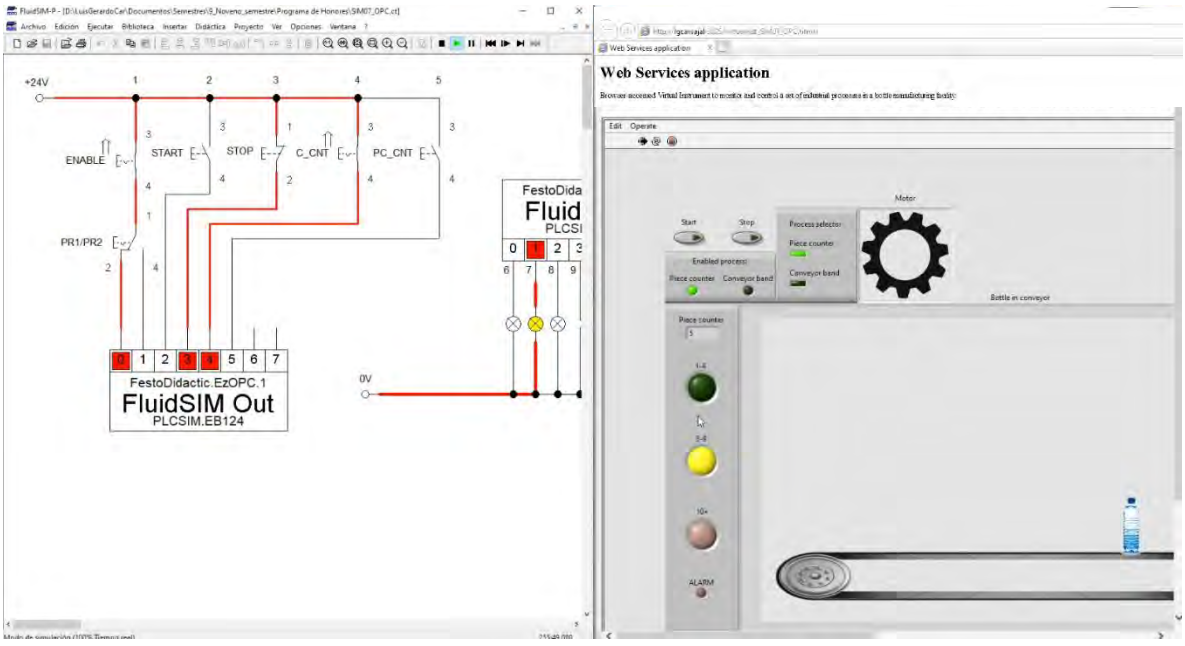


Figure 68. Execution of Process 1 in the plant and application: 5-9 bottles counted.

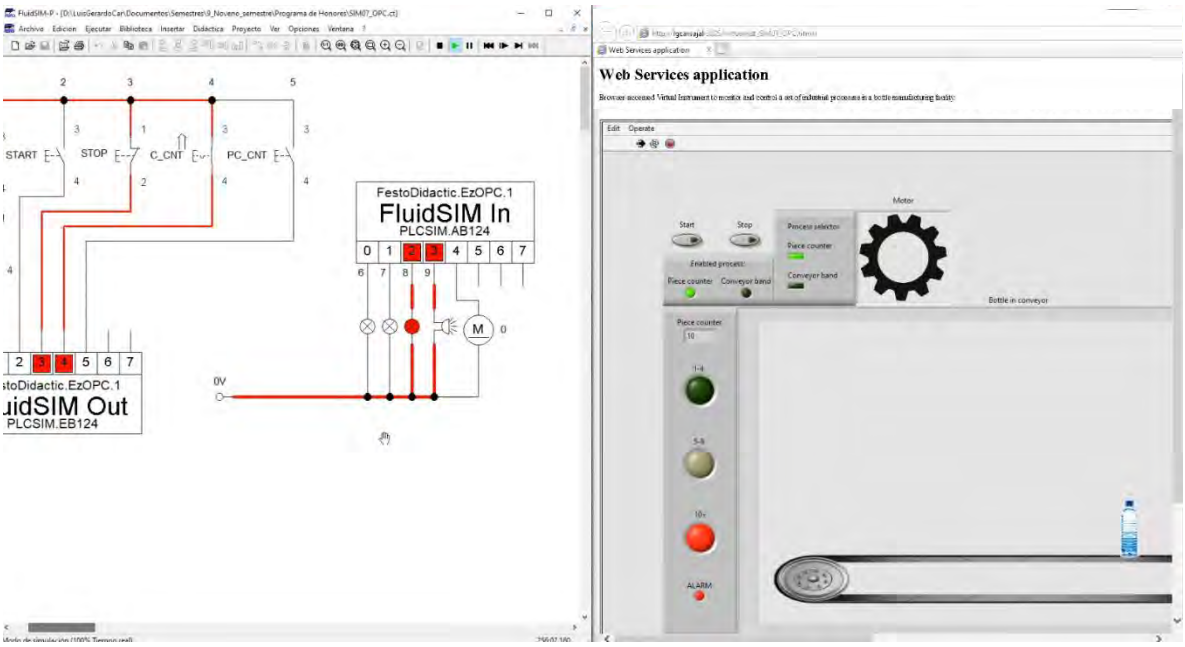


Figure 69. Execution of Process 1 in the plant and application: 10 or more bottles counted.

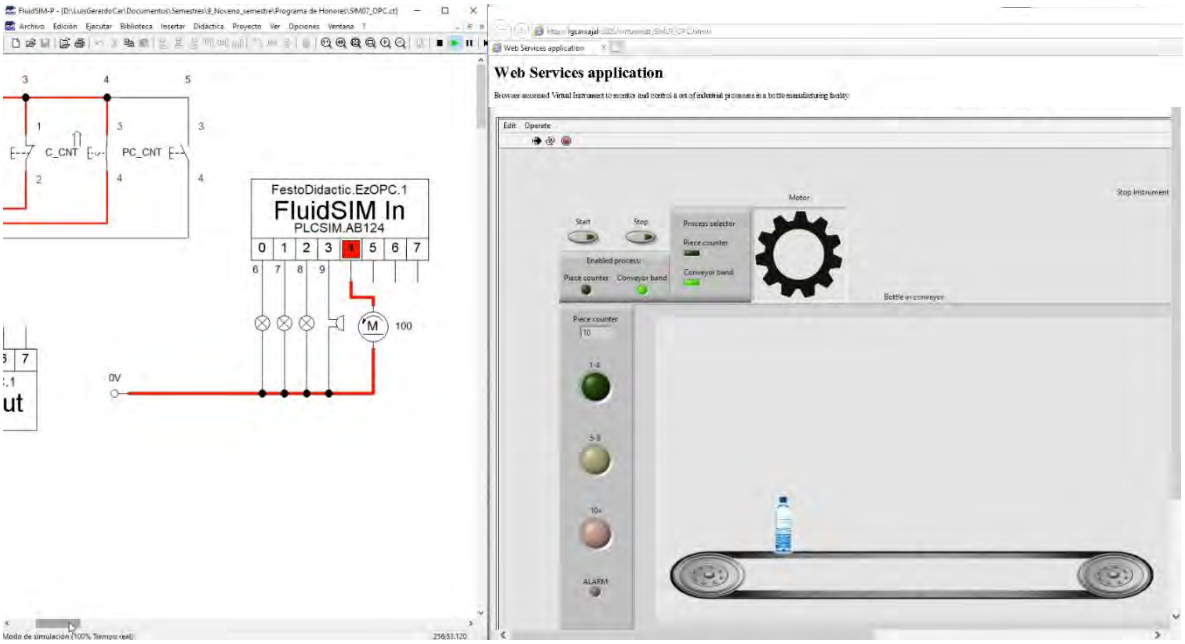


Figure 70. Execution of Process 2 in the plant and application: bottle at the start of the conveyor band (position 1).

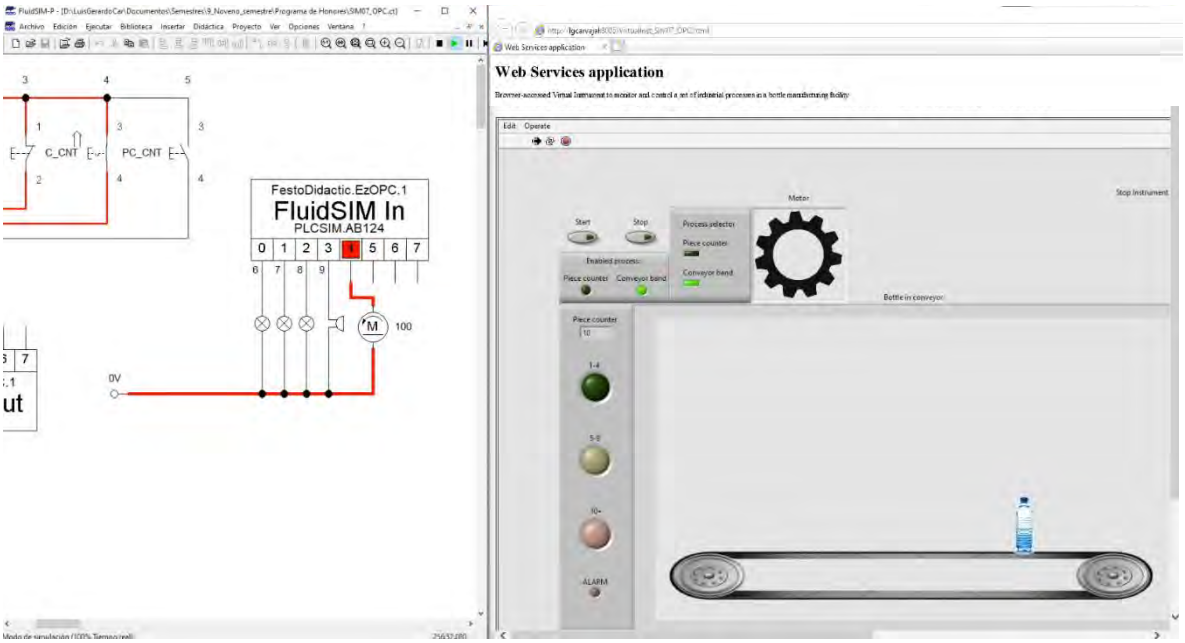


Figure 71. Execution of Process 2 in the plant and application: bottle being transported through the conveyor band (position 2).

A demonstrative video was created in addition to the previous images. It shows the connection between programs during runtime, along with their behavior when interacting with elements in the plant and the application. It is relevant to mention that it was recorded in Spanish. The link is: http://bit.ly/thesisLGCarvajal.

The results validate its use for control and monitoring purposes, enabling data transfer from/to the plant. Since all of the elements are connected and can exchange information for control and monitoring processes, they constitute a SCADA system. Moreover, the system corresponds to an implementation of IIoT because the communication structure allows network access to a plant via a web browser.

Even though the implementation procedure may be perceived as extensive, it can be done straightforward and without issues by having the appropriate software installed; the steps that were presented in the methodology can then be replicated to achieve communication using the OPC UA standard. Once configured, the resulting system can be executed continuously while the OPC server remains active.

Minor technical difficulties occurred during the last part of the methodology when attempting to run the Web Services application; it only worked after deactivating antimalware software and using Internet Explorer as the web browser. The fact that only a legacy browser can run the published Web Service and that antimalware software blocks its execution shows a weakness of the publishing tool provided by National Instruments in LabVIEW.

A connection test in a real environment with physical devices was not possible because of the nature of the current situation regarding COVID-19 lockdown policies. However, the results can be extrapolated to *in situ* applications both in industrial and academic environments. It would only be needed to replace FluidSIM with a physical plant and the simulated PLC with its real equivalent. Besides, the current implementation is already useful for educative purposes in a distance learning approach because students could do course practice of related topics in a simulated environment.

7. Conclusions and Recommendations

The work presented above demonstrates that it is possible to use the OPC UA standard to communicate an industrial process controlled by a PLC with a VI that runs on a Web Services application, achieving an IIoT implementation of a SCADA system. The solution offers a lower cost than alternatives in the IIoT service market; the reason is the relatively low cost of the program licenses. It also enables platform independence since the software can operate on Windows directly or through a Virtual Machine (VM).

The results indicate that it is feasible to replicate the methodology to industrial processes to achieve their remote monitoring and control. The same methodology could also be applied to academic institutions for students to interact remotely with PLCs and industrial plants. By doing so, they would be able to monitor variables and test the execution of programs developed during class. The above would result in a tangible benefit to their learning experience, at the same time that they would practice in a physical environment without the need of going to a laboratory.

Among the benefits of implementing the proposed methodology is that managers and supervisors can eliminate the need to go to a plant in person to monitor the state of a machine, its produced units, and its real-time operation. The aforementioned helps to avoid risks and extended working hours while it favors flexibility and management of production resources.

Another benefit is that it removes the need for students and professors to go to a laboratory. As a result, they can remotely work on projects and validate their functionality. Furthermore, it bridges the learning gap for students who have online study programs,

allowing them to get quality results that exceed the capabilities of simulators; the same applies to students who are currently in situations such as the lockdown due to the pandemic.

It is important to recall that the methodology should be followed in the same order of steps to achieve a successful connection and the desired behavior. Additionally, it is recommended to verify that there is no additional local network service interfering with EzOPC or NI OPC servers. It is good practice to perform a test after each network configuration step to verify a successful bidirectional data transfer.

An area of improvement is the web browser compatibility of Virtual Instruments. For that reason, a recommendation for further work is to develop a software tool that can export a VI to a Web Services application that runs on current web browsers, either on desktop or on mobile devices. A further area of opportunity is the development of comprehensive software that manages the entire process of connecting industrial devices through the OPC UA standard to offer a ready-to-use IIoT solution.

8. Bibliography

Agarwal, M., Dutta, S., Kelly, R., & Millán, I. (2021, January 15). *COVID-19: An inflection point for Industry 4.0*. Retrieved from McKinsey & Company: https://www.mckinsey.com/business-functions/operations/our-insights/covid-19-an-inflection-point-for-industry-40

Aulbur, W., CJ, A., & Bigghe, R. (2016). *Skill Development for Industry 4.0.* Roland Berger GMBH. Retrieved from http://www.globalskillsummit.com/Whitepaper-Summary.pdf

Blank, A. G. (2004). *TCP/IP Foundations.* Alameda: Sybex.

Bloem, J., Van Doorn, M., Duivestein, S., Excoffier, D., Maas, R., & Van Ommeren, E. (2014). The Fourth Industrial Revolution. Things to Tighten the Link Between IT and OT. *VINT research report*(3), 1-39.

Bolton, W. (2009). *Programmable Logic Controllers* (Fifth ed.). Burlington: Elsevier.

Cambridge University Press. (2021). *Meaning of the Industrial Revolution in English*. Retrieved from Cambridge Dictionary: https://dictionary.cambridge.org/us/dictionary/english/industrial-revolution

Chou, S.-Y. (2019). The Fourth Industrial Revolution: Digital Fusion with Internet of Things. *Journal of International Affairs*, 107-120.

Comer, D. E. (2015). *Computer Networks and Internets* (Sixth ed.). Essex: Pearson.

Darma, D. C., Ilmi, Z., Darma, S., & Syaharuddin, Y. (2020). COVID-19 and its Impact on Education: Challenges from Industry 4.0. *Aquademia, 4*(2). doi:10.29333/aquademia/8453

Emerson. (2019, October 4). *Why OPC UA is the IIOT Protocol of Choice.* Retrieved from IndustryWeek: https://www.industryweek.com/white-papers/whitepaper/22028338/why-opc-ua-is-the-iiot-protocol-of-choice

Exor International. (2019, January 17). *What is OPC UA and why will it continue to grow in use?* Retrieved from EXOR: https://www.exorint.com/en/blog/2019/01/17/what-is-opc-ua-and-why-will-it-continue-to-grow-in-use

Ferrell, K. (2000). The Steam Engine Powers the Industrial Revolution. In N. Schlager, & J. Lauer, *Science and Its Times* (Vol. 4, pp. 399-403). Farmington Hills: Gale Group.

Festo. (2021). *FluidSIM® 5*. Retrieved from Festo Didactic: https://www.festo-didactic.com/int-en/learning-systems/digital-learning/other-training-software/fluidsim/fluidsim-5.htm

Ghazivakili, M., Demartini, C., & Zunino, C. (2018). Industrial Data-Collector by enabling OPC-UA standard for Industry 4.0. *2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)* (pp. 1-8). Imperia: IEEE. doi:10.1109/WFCS.2018.8402364

Helfgott, R. B. (1986). America's Third Industrial Revolution. *Challenge*, 41-46.

Inductive Automation. (2018, September 12). *What is SCADA?* Retrieved from Inductive Automation: https://www.inductiveautomation.com/resources/article/what-is-scada

International Electrotechnical Commission. (2003). *International Standard IEC 61131-3* (Second ed.). Geneva: International Electrotechnical Commission.

Janke, M. (2000). OPC - Plug and Play Integration to Legacy Systems. *2000 Annual Pulp and Paper Industry Technical Conference* (pp. 68-72). Atlanta: IEEE.

Jasud, P. V. (2017). The OSI Model: Overview on the Seven Layers of Computer Networks. *International Journal for Innovative Research in Science & Technology, 4*(3), 116-124.

Kim, W., & Sung, M. (2017). Poster Abstract: OPC-UA Communication Framework for PLC-Based Industrial IoT Applications. *2017 IEEE/ACM Second International Conference on Internet-of-Things Design and Implementation (IoTDI)* (pp. 327-328). Pittsburgh: IEEE.

Knapp, E. D., & Langill, J. T. (2015). *Industrial Network Security. Securing Critical Infrastructure Networks for Smart Grid, SCADA, and Other Industrial Control Systems* (Second ed.). Waltham: Elsevier.

Koc, T. C., & Teker, S. (2019). Industrial Revolutions and its effects on quality of life. *PressAcademia Procedia (PAP), 9*, 304-311.

Kominek, D., & Brubaker, S. (2018). *OPC UA vs. Classic OPC. Why Switch and How To Get Started.* Matrikon Inc. & Beeond, Inc.

Kozierok, C. M. (2015). *The TCP/IP Guide.* San Francisco: No Starch Press.

Lamb, F. (2013). *Industrial Automation Hands-On.* New York: McGraww-Hill Education.

Lin, H.-I., & Hwang, Y.-C. (2019). Integration of Robot and IIoT over the OPC Unified Architecture. *2019 International Automatic Control Conference (CACS)* (pp. 1-6). Keelung: IEEE. doi:10.1109/CACS47674.2019.9024732

Lorenz, M., Rüßmann, M., Waldner, M., Engel, P., Harnisch, M., & Justus, J. (2015, April 9). *ANSFORMATION, INDUSTRY 4.0*. Retrieved from BCG: https://www.bcg.com/en-mx/publications/2015/engineered_products_project_business_industry_4_future_productivity_growth_manufacturing_industries

Mahnke, W., Leitner, S.-H., & Damm, M. (2009). *OPC Unified Architecture.* Ladenburg: Springer.

Marcin, M. (n.d.). *The History of the Assembly Line*. Retrieved from Crest Capital: https://www.crestcapital.com/tax/history_of_the_assembly_line

MarketsandMarkets. (2019, December). *Industry 4.0 Market by Technology (IoT, Artificial Intelligence, Industrial Metrology, Industrial Robotics, AR & VR, Blockchain, 3D Printing, Digital Twin, and 5G – Offering, Application, and End Users) and Geography- Global Forecast to 2024*. Retrieved from MarketsandMarkets: https://www.marketsandmarkets.com/Market-Reports/industry-4-market-102536746.html

Matrikon. (2020, October 13). *What are the OPC Datatypes?* Retrieved from OPC Support: https://www.opcsupport.com/s/article/What-are-the-OPC-Datatypes

Merriam-Webster. (n.d.). *Information Age*. Retrieved from Merriam-Webster.com dictionary: https://www.merriam-webster.com/dictionary/Information%20Age

Microsoft. (2019). *IoT Signals. Summary of Research Learnings.* Retrieved from Microsoft Azure: https://azure.microsoft.com/mediahandler/files/resourcefiles/iot-signals/IoT-Signals-Microsoft-072019.pdf

National Instruments. (2019, March 5). *How LabVIEW Uses I/O Servers*. Retrieved from National Instruments: https://www.ni.com/en-us/innovations/white-papers/12/how-labview-uses-i-o-servers.html

National Instruments. (2020a, November 24). *Adding a PLC to NI OPC Servers*. Retrieved from National Instruments: https://knowledge.ni.com/KnowledgeArticleDetails?id=kA03q000000YIRNCA4&l=en-US

National Instruments. (2020b, December 21). *Connect LabVIEW to Any PLC Using OPC*. Retrieved from National Instruments: https://knowledge.ni.com/KnowledgeArticleDetails?id=kA03q000000x0MPCAY&l=en-US

National Instruments. (n.d.). *What Is LabVIEW?* Retrieved from National Instruments: https://www.ni.com/en-us/shop/labview.html

OPC Foundation. (2003). *OPC Data Access Custom Interface Specification 3.0.* Austin: OPC Foundation.

OPC Foundation. (n.d.-a). *What is OPC?* Retrieved from OPC Foundation: https://opcfoundation.org/about/what-is-opc/

OPC Foundation. (n.d.-b). *Classic*. Retrieved from OPC Foundation: https://opcfoundation.org/about/opc-technologies/opc-classic/

OPC Labs. (n.d.). *Data Types in OPC Classic*. Retrieved from QuickOPC User's Guide and Reference: http://opclabs.doc-that.com/files/onlinedocs/QuickOpc/Latest/User%27s%20Guide%20and%20Reference-QuickOPC/Data%20Types%20in%20OPC%20Classic.html

Oracle. (n.d.). *What Is IoT?* Retrieved from Oracle: https://www.oracle.com/internet-of-things/what-is-iot/

Pereira, C. E., & Neumann, P. (2009). Industrial Communication Protocols. In S. Y. Nof, *Springer Handbook of Automation* (pp. 981-999). West Lafayette: Springer.

Petruzella, F. D. (2011). *Programmable Logic Controllers* (Fourth ed.). New York: McGraw-Hill.

Schwab, K. (2016, January 14). *The Fourth Industrial Revolution: what it means, how to respond*. Retrieved from World Economic Forum: https://www.weforum.org/agenda/2016/01/the-fourth-industrial-revolution-what-it-means-and-how-to-respond/

Siemens. (2021a, March 12). *6ES7312-5BF04-0AB0 Data sheet.* Retrieved from Siemens Industry Mall: https://mall.industry.siemens.com/mall/en/es/Catalog/Product/6ES7312-5BF04-0AB0

Siemens. (2021b, February 11). *6GK7343-1GX31-0XE0 Data sheet.* Retrieved from Siemens Industry Mall: https://mall.industry.siemens.com/mall/en/mx/Catalog/Product/6GK7343-1GX31-0XE0

Siemens. (2021c). *Software in TIA Portal*. Retrieved from Siemens: https://new.siemens.com/global/en/products/automation/industry-software/automation-software/tia-portal/software.html

Stern, D. I., Burke, P. J., & Bruns, S. B. (2019). The Impact of Electricity on Economic Development: A Macroeconomic Perspective. *UC Berkeley: Center for Effective Global Action*, 1-42. Retrieved from https://escholarship.org/uc/item/7jb0015q

Sultanow, E., & Chircu, A. (2019). A Review of IoT Technologies, Standards, Tools, Frameworks and Platforms. In Z. Mahmood, *The Internet of Things in the Industrial Sector. Security and Device Connectivity, Smart Environments, and Industry 4.0* (pp. 3-34). Cham: Springer.

Torres, P., Dionísio, R., Malhão, S., Neto, L., Ferreira, R., Gouveia, H., & Castro, H. (2019). Cyber-Physical Production Systems supported by Intelligent Devices (SmartBoxes) for Industrial Processes Digitalization. *2019 IEEE 5th International*

forum on Research and Technology for Society and Industry (RTSI) (pp. 73-78). Florence: IEEE. doi:10.1109/RTSI.2019.8895553

Travis, J., & Kring, J. (2007). *LabVIEW for Everyone* (Third ed.). Upper Saddle River: Prentice Hall.

Vazquez-Gonzalez, J. L., Barrios-Aviles, J., Rosado-Muñoz, A., & Alejos-Palomares, R. (2018). An Industrial Automation Course: Common Infrastructure for Physical, Virtual and Remote Laboratories for PLC Programming. *International Journal of Online and Biomedical Engineering (iJOE), 14*(8), 4-19. doi:https://doi.org/10.3991/ijoe.v14i08.8758

Vickers, C., & Ziebarth, N. L. (2019). Lessons for Today from Past Periods of Rapid Technological Change. *Department of Economic & Social Affairs (United Nations)*, 1-38.

Wiens, T. (n.d.). *NetToPLCsim - Network extension for Plcsim*. Retrieved from NetToPLCsim: http://nettoplcsim.sourceforge.net

World Economic Forum. (2015, January). *Industrial Internet of Things: Unleashing the Potential of Connected Products and Services.* Retrieved from World Economic Forum: http://www3.weforum.org/docs/WEFUSA_IndustrialInternet_Report2015.pdf

Wright, C. C. (2020, August). COVID-19 Pandemic and the Digital Revolution. *ASA Monitor, 84*, 30. doi:10.1097/01.M99.0000695184.52961.46

Zhou, C., Damiano, N., Whisner, B., & Reyes, M. (2017). Industrial Internet of Things: (IIoT) applications in underground coal mines. *Min Eng*, 50-56. doi:10.19150/me.7919