الجمهـــوريــــة الجزائـــريــــة الديمقــراطيــــة الشـعبيــــة
RépubliqueAlgérienneDémocratique et Populaire
وزارة التـــعلــيـــــم العـــــالـــــي والبـــحــــث العــلـمـــي
Ministère de l'EnseignementSupérieure et de la RechercheScientifique

جامعة وهران 2 محمد بن أ حمد
Universitéd'Oran 2 Mohamed Ben Ahmed

معهد الصيانة والأمن الصناعي
Institut de Maintenance et de SécuritéIndustrielle

Département de Maintenance en Instrumentation

MÉMOIRE

Pour l'obtention du diplôme de Master

Filière :Génie Industriel
Spécialité : Maintenance des automates et de l'instrumentationindustriels

Thème

Simulation of assembly line and automated storage with Factory IO

Présenté et soutenu publiquement par :

Nom : AYACHI Prénom : Mohammed Aniss

Nom : ETTALEBY Prénom : Yasmine

Devant le jury composéde :

Nom et Prénom	Grade	Etablissement	Qualité
………………………………...	………….	IMSI-Univ. D'Oran2	**Président**
M^{me} AISSANI NASSIMA	MCA	IMSI-Univ. D'Oran2	**Encadreur**
………………………………...	………….	IMSI-Univ. D'Oran2	**Examinateur**

Année 2020/2021

Dedication

We dedicate this work to our parents who supported and encouraged us during these years of study.

May they find here the testimony of our deep gratitude.

To my brother, my grandparents and those who shared with me all the emotional moments while carrying out this work. They have warmly supported and encouraged me throughout my journey.

To my family, loved ones and those who give me love and liveliness.

To all my friends who have always encouraged me, and to whom I wish more success.

Everyone I love.

Appreciation

First, we thank Allah for helping us to accomplish this work.

We would like to express our gratitude to our thesis director, Madam AISSANI Nassima. We thank her for mentoring, guiding, helping and advising us.

We address our sincere thanks to all the professors, speakers and all the people who by their words, their writings, their advice and their criticisms guided our reflections and agreed to meet us and answer our questions during our research.

We thank our thank our families for their encouragement.

Finally, we thank our friends their unconditional support and encouragement has been a great help.

To all of these speakers, I present my thanks, respect and gratitude.

Table of content

Table of Figures

List of Table

Summary

This thesis aim is to automate a process of stocking and counting commodity in warehouses.

To achieve that we use TIA Portal software for programming and FACTORY I/O software for simulation, and S7-1200 Siemens PLC.

Where we use simple and basic instructions to simulate the process and control perfectly.

This software guarantees a very similar representation to the real scene that we aim to study, which gives us the important information that we need to assemble our actual factory and the problems we might face in as a real engineer.

General Introduction

Innovation is making many distribution center cycles more proficient by augmenting the work of humans or, sometimes, automating tedious, freeing up associates to focus on more hard assignments. Distribution center robotization takes many structures, including PLCs, machines and robots that guide laborers with measures identified with stock from when it shows up at the stockroom until it leaves. Use distribution center computerization arrangements can help stockroom increment usefulness and precision, reduces work costs and improve well-being.

That does not mean robots are dominating or taking positions from human laborers. Maybe, organizations are exploiting the accuracy of machines, just as their capacity to turn out constantly for extended periods without stopping, to make warehouse tasks safer and more efficient. Some stockroom mechanization arrangements cover everything from dumping trailers to satisfying requests.

However, people are still essential for the scene. We should investigate the kinds of distribution center computerization, how stockroom mechanization works and the essential advantages distribution centers acquire by carrying out robotization innovation.

I.1.Siemens AG History:

Energy technology and manufacturing company formed in 1966 through the merger of Siemens &Halske AG (founded 1847), Siemens-Schuckertwerke (founded 1903), and Siemens-Reiniger-Werke AG (founded 1932). Operating in more than 200 countries and regions, it engages in a wide range of manufacturing and services in areas such as power generation and transmission, energy management, transportation, telecommunications systems, and medical engineering.

The company invests heavily in research and development and ranks among the largest patent holders in the world. Headquarters are in Munich.

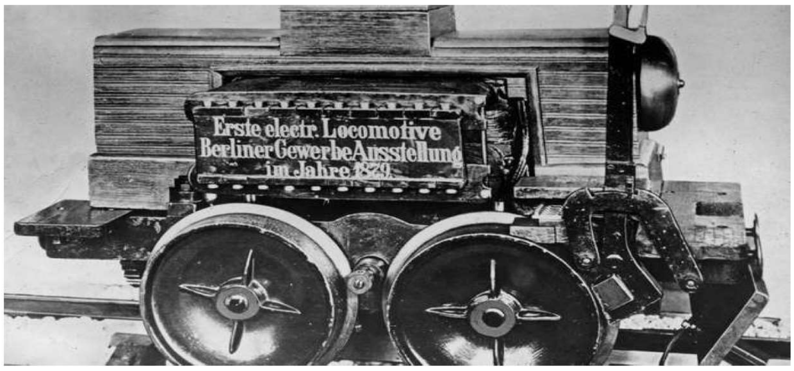


Figure 1. The first electric locomotive, built by the Siemens electric company, 1879

The first Siemens Company, Telegraphen-Bau-Anstalt von Siemens & Halske ("Telegraph Construction Firm of Siemens &Halske"), was founded in Berlin in 1847 by Werner von Siemens (1816–92), his cousin Johann Georg Siemens (1805–79), and Johann Georg Halske (1814–90); its purpose was to build telegraph installations and other electrical equipment.

It soon began spreading telegraph lines across Germany, establishing in 1855 a branch in St. Petersburg for Russian lines and in 1858 a branch in London for English lines, the latter headed by Werner's brother William Siemens (1823–83).

As the firm grew and introduced mass production, Halske, who was less inclined toward expansion, withdrew (1867), leaving control of the company to the four Siemens brothers and their descendants.

Meanwhile, the company's activities were enlarging to include dynamos, cables, telephones, electric power, electric lighting, and other advances of the later Industrial Revolution. In 1890 it became a limited partnership, with Carl Siemens (Werner's brother) and Arnold and Wilhelm Siemens (Werner's sons) as the senior partners; in 1897 it became a limited-liability company, Siemens &Halske AG. [A]

I.2.The first automated storage and retrieval system (AS/RS):

Daifuku developed the first automated storage and retrieval system (AS/RS) in Japan in 1966. The objectives for developing this system included achieving workload reductions and cost savings through (1) the effective use of land, (2) improvements in storage efficiency, (3) personnel saving and labor saving in warehouse work, and (4) improvements in management levels. At the time, single-story warehouses were standard type for warehouses. Logistics capabilities were not as advanced as current levels because loading, unloading, storage work mostly consisted of manual labor, and stored goods were managed through ledgers and slips. The automated warehouse that emerged under these circumstances was a revolutionary logistics technology innovation that overturned the traditional concept of warehousing.

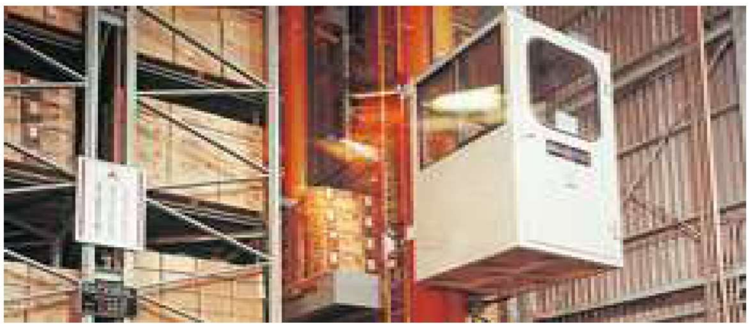


Figure 2. First AS/RS

About forty years since the birth of this technology, automated warehouses have now become more high-capacity and high-performance through the expansion of models, including stacker cranes and peripheral equipment, while its purposes and industries and businesses delivered to have rapidly grown. Here we will introduce our automated warehouse market and technology developments, as well as future trends.

For this purpose, this dissertation will be organized as follow; first chapter will be dedicated to overall generalities, and the common hardware used in automated warehouses. The second chapter will be about the programs used and how they function. The third chapter will talk about hardware and virtual (simulation) materials used, and finally the fourth chapter, will contain the connection between all the hardware and software setup and the main program to achieve the desired ASSEMBLY LINE AND AUTOMATED STORAGE.

Chapter I: Generalities

II.1.Introduction:

In this chapter, we will talk about automation in general and its main component, without the extra details, just an overall scoop of how they function and what we are going to use in our next chapter.

II.2.Automation:

Automation is the use of logical programming commands and mechanized equipment to replace the decision making and manual command-response activities of human beings. Historically, mechanization—such as the use of a timing mechanism to trip a lever or ratchet and pawl—aided humans in performing the physical requirements of a task. Automation, however, takes mechanization a step further, greatly reducing the need for human sensory and mental requirements while simultaneously optimizing productivity.

It is believed that the term automation was first coined in the 1940s by a Ford Motor Company engineer describing various systems where automatic actions and controls were substituted human effort and intelligence. At this time, control devices were electromechanical in nature. Logic was performed by means of relays and timers interlocked with human feedback at decision points. By wiring relays, timers, push buttons, and mechanical position sensors together, simple logical motion sequences could be performed by turning on and off motors and actuators. With the advent of computers and solid-state devices, these control systems became smaller, more flexible, and less expensive to implement and modify. The first programmable logic controllers were developed in the 1970s and 1980s by Modicon in response to a challenge by GM to develop a substitute for hardwired relay logic. As technology improved and more automation companies entered the market, new control products were developed. Today, myriad computerized logic control devices developed by hundreds of different manufacturers exist in the industry. [1]

Automation in manufacturing is categorized into three types: fixed automation, flexible automation and programmable. The association between the three types depends on the variety and quantity of the products as displayed in Figure 1. Fixed automation is generally fully automated systems, while programmable automation is usually half-automated systems and flexible automation is the mixture or combination of the other two types. [2]

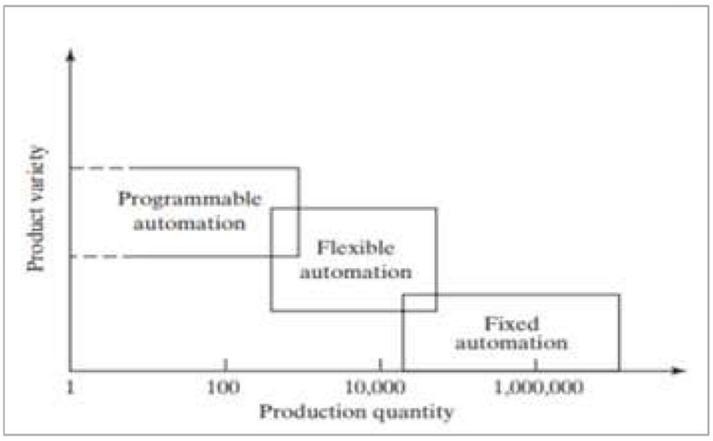


Figure I. 1: *Relation between automation and variety/quantity of product.*

II.2.1.Automation in warehouse:

Storing and retrieving goods in a warehouse or distribution center is a common and traditional concept. Storage/warehouse systems are diverse, but there are two types: manual and automated in terms of operation.

According to the Peerless Research Group, only 4% of them are highly automated (Logistics Management, 2019). The rest are mostly manual and half-automated, where operators move around the facility, lift each product by forklifts and pick up by hand. The main drawback of manual storage systems is that they are dependently labour-intensive and noncomputerized. They involve many moving pieces and heavy machinery other than forklifts controlled by storage/warehouse workers. Inaccurate counts and misplacement might result in time wasting and running out of essential items for new orders. Moreover, there is apossibility that employees encounter injuries as more than 5% of traditional warehouse workers were injured in 2016 (Conveyco, 2019).

Manual storage/warehouse systems, despite their simplicity and low-cost investments, cannot optimize the space and capacity of the warehouse regarding height due to limited reach of forklifts, thus occupying the considerably large land area. In manual storage systems, numerous products are usually stacked up one on top of another, which might reduce the quality of some at the bottom. Because of the lack of automation and computerization, it is very challenging to track the inflow and outflow of each product and its quantity. Therefore, each product transaction needs to be monitored continuously to maintain its availability and stocking levels. The tracking and monitoring information are not shared within any automated systems or programs, thus accumulating the wastes of time and labour and making the whole process more cumbersome.

II.2.2.Levels of automation:

Device level:

This is the lowest level in the automation hierarchy. It includes the actuators, sensors, and other hardware components that comprise the machine level. The devices are combined into the individual control loops of the machine, for example, the feedback control loop for one axis of a CNC machine or one joint of an industrial robot.

Machine level:

Hardware at the device level is assembled into individual machines. Examples include CNC machine tools and similar production equipment, industrial robots, powered conveyors, and automated guided vehicles. Control functions at this level include performing the sequence of steps in the program of instructions in the correct order and making sure that each step is properly executed.

Cell or system level:

This is the manufacturing cell or system level, which operates under instructions from the plant level. A manufacturing cell or system is a group of machines or workstations connected and supported by a material handling system, computer, and other equipment appropriate to the manufacturing process. Production lines are included in this level. Functions include part dispatching and machine loading, coordination among machines and material handling system, and collecting and evaluating inspection data.

Plant level:

This is the factory or production systems level. It receives instructions from the corporate information system and translates them into operational plans for production. Likely functions include order processing, process planning, inventory control, purchasing, material requirements planning, shop floor control, and quality control.

Enterprise level:

This is the highest level, consisting of the corporate information system. It is concerned with all of the functions necessary to manage the company: marketing and sales, accounting, design, research, aggregate planning, and master production scheduling. The corporate information system is managed using Enterprise Resource Planning. [3]

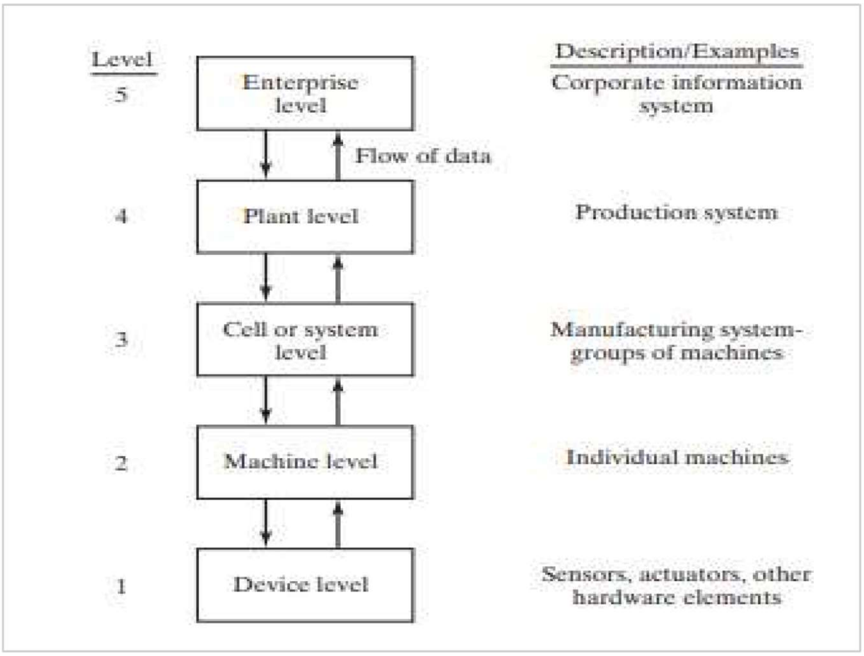


Figure I. 2: *Five levels of automation and control in manufacturing.*

II.3. Programmable Logic Controllers (PLCs):

II.3.1. Definition:

A programmable logic controller is a solid-state system designed to perform the logic functions previously accomplished by components such as electromechanical relays, drum switches, mechanical timers/counters, etc., for the control and operation of manufacturing process equipment and machinery. Even though the electromechanical-relay (control relays, pneumatic timer relays, etc.) has served well for many generations, often, under adverse conditions, the ever-increasing sophistication and complexity of modern processing equipment requires faster acting, more reliable control functions than electromechanical relays and/or timing devices can offer. Relays have to be hardwired to perform a specific function, and when the system requirements change, the relay wiring has to be changed or modified. In extreme cases, such as in the auto industry, complete control panels had to be replaced since it was not economically feasible to rewire the old panels with each model changeover. [5]

II.3.2. PLC history:

In the late 1960's PLCs were first introduced. The primary reason for designing such a device was eliminating the large cost involved in replacing the complicated relay based machine control systems. Bedford Associates (Bedford, MA) proposed something called a Modular Digital Controller (MODICON) to a major US car manufacturer. Other companies at the time proposed computer based schemes, one of which was based upon the PDP-8. The MODICON 084 brought the world's first PLC into commercial production.

When production requirements changed so did the control system. This becomes very expensive when the change is frequent. Since relays are mechanical devices, they also have a limited lifetime, which required strict adhesion to maintenance schedules. Troubleshooting was also quite tedious when so many relays are involved. Now picture a machine control panel that included many, possibly hundreds or thousands, of individual relays. The size could be mind-boggling. How about the complicated initial wiring of so many individual devices! These relays would be individually wired together in a manner that would yield the desired outcome. Were there problems? You bet!

These "new controllers" also had to be easily programmed by maintenance and plant engineers. The lifetime had to be long and programming changes easily performed. They also had to survive the harsh industrial environment. That's a lot to ask! The answers were to use a programming technique most people were already familiar with and replace mechanical parts with solid-state ones.

In the mid70's the dominant PLC technologies were sequencer state-machines and the bit-slice based CPU. The AMD 2901 and 2903 were quite popular in Modicon and A-B PLCs. Conventional microprocessors lacked the power to quickly solve PLC logic in all but the smallest PLCs. As conventional microprocessors, evolved, larger and larger PLCs were being based upon them. However, even today some are still based upon the 2903.[5]
Modicon has yet to build a faster PLC than their 984A/B/X, which was based upon the 2901. Communications abilities began to appear in approximately 1973.

The first such system was Modicon's Modbus. The PLC could now talk to other PLCs and they could be far away from the actual machine they were controlling. They could also now be used to send and receive varying voltages to allow them to enter the analog world. Unfortunately, the lack of standardization coupled with continually changing technology has made PLC communications a nightmare of incompatible protocols and physical networks. Still, it was a great decade for the PLC!

The 80's saw an attempt to standardize communications with General Motor's manufacturing automation protocol (MAP). It was also a time for reducing the size of the PLC and making them software programmable through symbolic programming on personal computers instead of dedicated programming terminals or handheld programmers. Today the world's smallest PLC is about the size of a single control relay!

The 90's have seen a gradual reduction in the introduction of new protocols, and the modernization of the physical layers of some of the more popular protocols that survived the 1980's. The latest standard (IEC 1131-3) has tried to merge PLC programming languages under one international standard.

We now have PLCs that are programmable in function block diagrams, instruction lists, C and structured text all at the same time! PC's are also being used to replace PLCs in some applications. The original company who commissioned the MODICON 084 has actually switched to a PC based control system.

II.3.3. Programming languages:

The programming language is a set of elements, blocks and standardized rules that allows the programming of processor to perform defined functions. In other words, the programmer writes a set of instructions that the PLC must perform by means of default programming language.

Generally, the programmers use high-level programming language, while the one of PLC processor is of low level. The task, that allows conversion from a high-level programming language to a low-level one, is called compilation and compiler performs it.
In other words, the compiler changes the program language by moving from a high-level programming language to a low-level syntax.

All programming languages constitutes of a "set of operations" that the programmer must use to implement any function regardless of its complexity.

Each programming language has its own set of operations and therefore the solution of a single problem can be implemented in a different way; indeed, it is not possible to switch from one programming language to another automatically. Generally, the set of operations are elementary functions that come in logical operations, counts, compare, or timers.

Programming languages are divided on the basis of the visual representation of the functions in two categories: graphical programming languages and literal programming languages.

Graphical programming languages are characterized by the use of graphical symbols, while literal programming languages use mnemonic literal codes.
The programming languages can follow the school of American thought and the school of German thought.

The programming languages of the American school are:

• Ladder diagram, a language based on graphic symbols (figure 3.1);
• Boolean keys, a language based on “mnemonic” literal codes (figure 3.2);

• Functional block, a language based on functional blocks containing an instruction (figure 3.3);
• High-level language (HLL), a language that can also be used by computers (figure 3.4).
The programming languages of the German school were created by Siemens and they are:

- • KOP, a contact programming language based on graphical symbols corresponding to the Ladder diagram (figure 3.1);
- • FUP or FBS, defined as a logical scheme corresponding to the functional block (figure 3.3). It is a graphical language because it uses logical blocks as symbols;
- • AWL, a list of instructions (STL), which could also be called a symbolic language because desired instructions are written using mnemonic literals that identify the set of operations functions such as counters, timers and logical functions (AND, OR, non, NAND, etc.). It corresponds to the Boolean status keys (figure 3.2).
- • Structured Control Language (SCL), a high-level structured programming language. It is at the level of HLL language of American origin (figure 3.5)

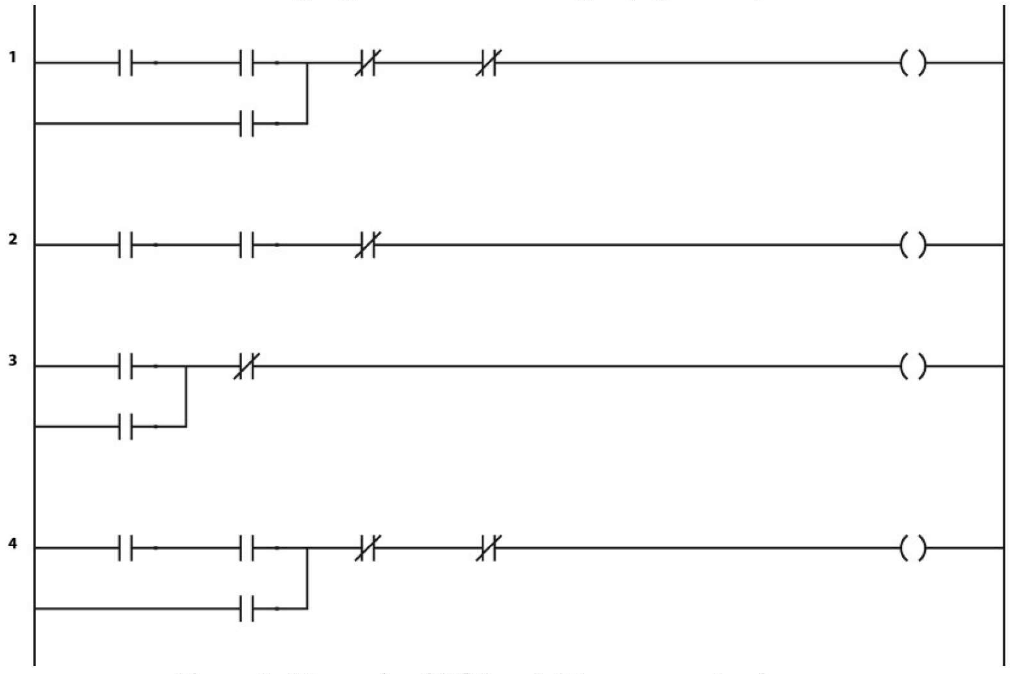


Figure I. *3Example of KOP or LAD programming language.*

```
000    L I0.1
001    O Q0.1
002    ANQ0.2
003    A I0.0
004    = Q0.1
005    L I0.2
006    O Q0.2
007    ANQ0.1
008    A I0.0
009    = Q0.2
```

Figure I. 4*Example of STL programming language.*

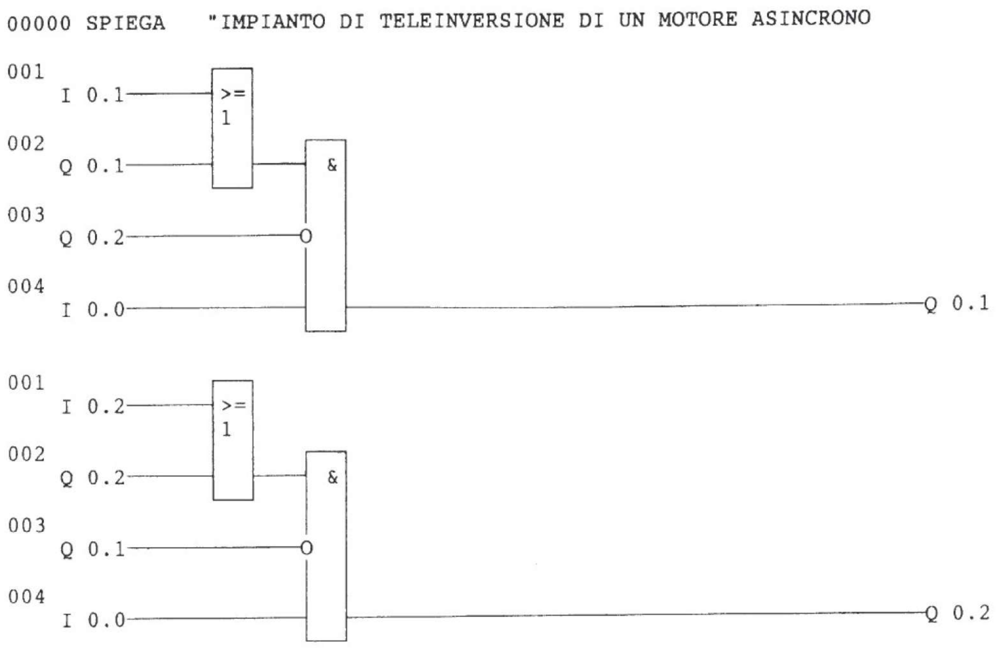
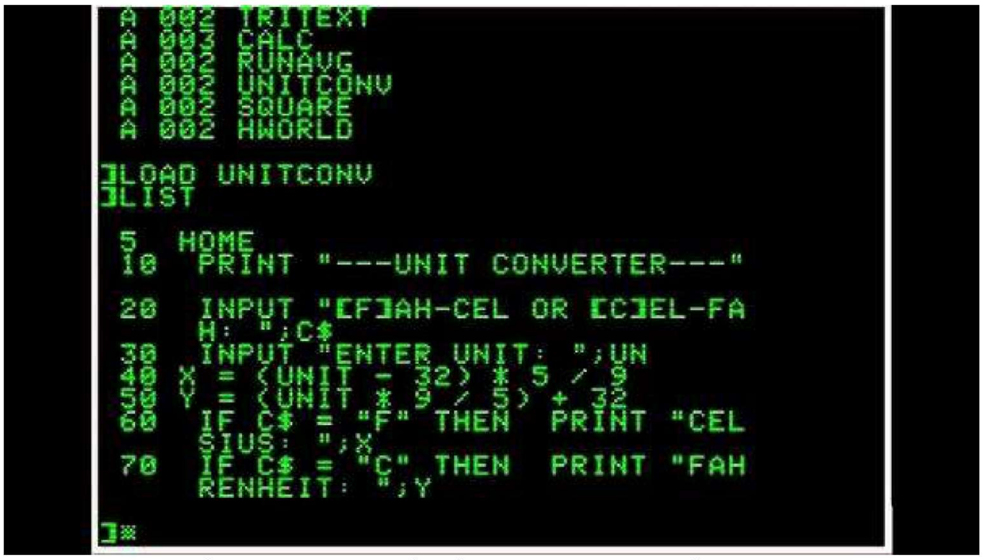
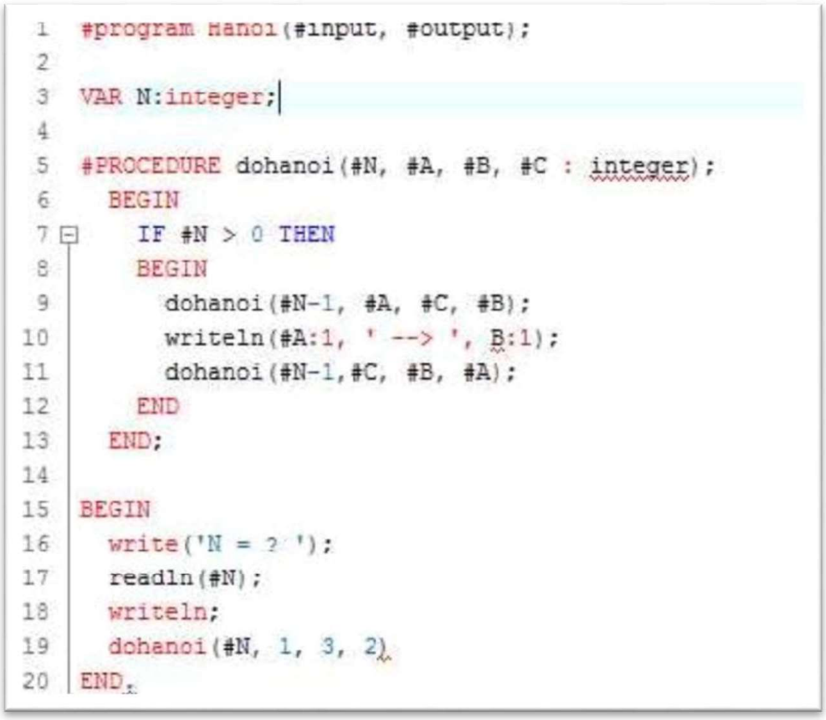


Figure I. 5: *Example of FUP or FBS programming language.*



```
 A 002 TRITEXT
 A 003 CALC
 A 002 RUNAVG
 A 002 UNITCONV
 A 002 SQUARE
 A 002 HWORLD

]LOAD UNITCONV
]LIST

 5  HOME
 10  PRINT "---UNIT CONVERTER---"

 20  INPUT "[F]AH-CEL OR [C]EL-FA
     H: ";C$
 30  INPUT "ENTER UNIT: ";UN
 40 X = (UNIT - 32) * 5 / 9
 50 Y = (UNIT * 9 / 5) + 32
 60  IF C$ = "F" THEN  PRINT "CEL
     SIUS: ";X
 70  IF C$ = "C" THEN  PRINT "FAH
     RENHEIT: ";Y

]※
```

Figure I. 6: *Example of HLL programming language.*



```
#program Hanoi(#input, #output);

VAR N:integer;

#PROCEDURE dohanoi(#N, #A, #B, #C : integer);
  BEGIN
    IF #N > 0 THEN
    BEGIN
      dohanoi(#N-1, #A, #C, #B);
      writeln(#A:1, ' --> ', B:1);
      dohanoi(#N-1,#C, #B, #A);
    END
  END;

BEGIN
  write('N = ? ');
  readln(#N);
  writeln;
  dohanoi(#N, 1, 3, 2)
END.
```

Figure I. 7: *Example of SCL programming language.*

Nowadays, Structured Control Language (SCL) is widely used by programmers, because it eases some functions and it conforms to the PLC standard (IEC 61131).

A high-level programming language is based on the Pascal programming language. It allows structured programming using top-level elements and simple elements such as input, output, timers, counters, etc. In other words, it integrates and expands the functions of other lower-level programming languages such as the KOP.

In conclusion, PLC programming languages let the programmer create a complex software by means of basic functions in a simple and intuitive way.

II.3.4. Hardware:

Typically, a PLC system has the basic functional components of processor unit, memory, power supply unit, input/output interface section, communications interface, and the programming device. Figure 3 shows the basic arrangement.

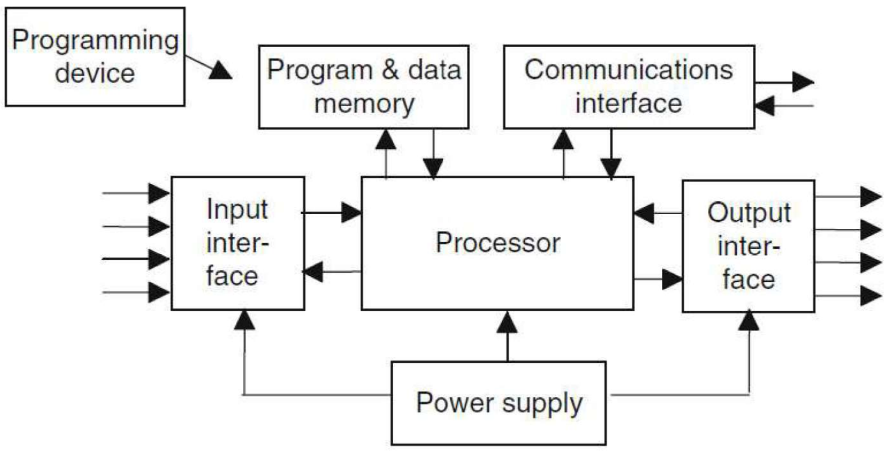


Figure I. 8: *The PLC system.*

• The processor unit or central processing unit (CPU) is the unit containing the microprocessor. This unit interprets the input signals and carries out the control actions according to the program stored in its memory, communicating the decisions as action signals to the outputs.

• The power supply unit is needed to convert the mains AC voltage to the low DC voltage (5 V) necessary for the processor and the circuits in the input and output interface modules.

• The programming device is used to enter the required program into the memory of the processor. The program is developed in the device and then transferred to the memory unit of the plc.

• The memory unit is where the program containing the control actions to be exercised by the microprocessor is stored and where the data is stored from the input for processing and for the output.

• The input and output sections are where the processor receives information from external devices and communicates information to external devices. The inputs might thus be from switches, with the automatic drill, or other sensors such as photoelectric cells, as in the counter mechanism temperature sensors, flow sensors, or the like. The outputs might be to motor starter coils, solenoid valves, or similar things. Input and output devices can be classified as giving signals that are discrete, digital or analog.

Devices giving discrete or digital signals are ones where the signals are either off or on. Thus a switch is a device giving a discrete signal, either no voltage or a voltage. Digital devices can be considered essentially as discrete devices that give a sequence of on/off signals. Analog devices give signals of which the size is proportional to the size of the variable being monitored. For example, a temperature sensor may give a voltage proportional to the temperature.

• The communications interface is used to receive and transmit data on communication networks from or to other remote PLCs. It is concerned with such actions as device verification, data acquisition, synchronization between user applications, and connection management. [4]

II.3.5. PLC Functioning:

Basis of a PLC function is continual scanning of a program. Under scanning, we mean running through all conditions within a guaranteed period. Scanning process has three basic steps:

- Step 1.

Testing input status. First, a PLC checks each of the inputs with intention to see which one of them has status ON or OFF.

In other words, it checks whether a sensor or a switch etc. connected with an input is activated or not. Information that processor thus obtains through this step is stored in memory in order to be used in the following step.

- Step 2.

Program execution. Here a PLC executes a program, instruction by instruction. Based on a program and based on the status of that input as obtained in the preceding step, an appropriate action is taken.

This reaction can be defined as activation of a certain output, or results can be put off and stored in memory to be retrieved later in the following step.

- Step 3.

Checkup and correction of output status. Finally, a PLC checks up output status and adjusts it as needed.

Change is performed based on the input status that had been read during the first step, and based on the results of program execution in step two.

Following the execution of step 3 PLC returns to the beginning of this cycle and continually repeats these steps.

Scanning time is defined by the time needed to perform these three steps, and sometimes it is an important program feature.

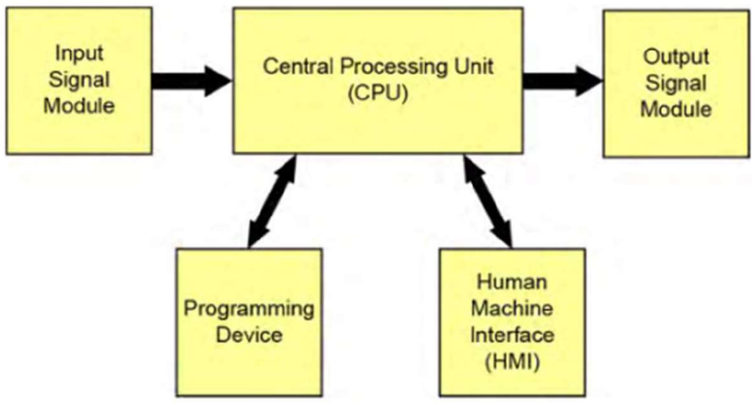


Figure I. 9: *Simple Example of PLC Function.*

II.3.6. Addressing:

Each module has pins to which external devices, such as sensors and actuators, are connected. In the case of input, the signals come from the outside, while in the case of output the CPU sends the signals. In both cases, you need an interface that permits a connection between hardware (physical pin) and software (logical signal value).

The companies of PLC impose to each bit an address that allows creation of this connection.

Particularly, each digital module has 8 pins which are divided into 2 groups of 4 pins each. Each pin is associated with one bit, and each module has a byte, because a byte corresponds with 8 bits.

The byte address is the number that marks a module, instead the single pin is marked with the bit address from 0 to 7, so the address of the first bit of any byte is 0, and the address of the last bit of any byte is 7.

In addition, the input pins are indicated by capital letter "I", while the output pins are indicated with capital letter "Q".

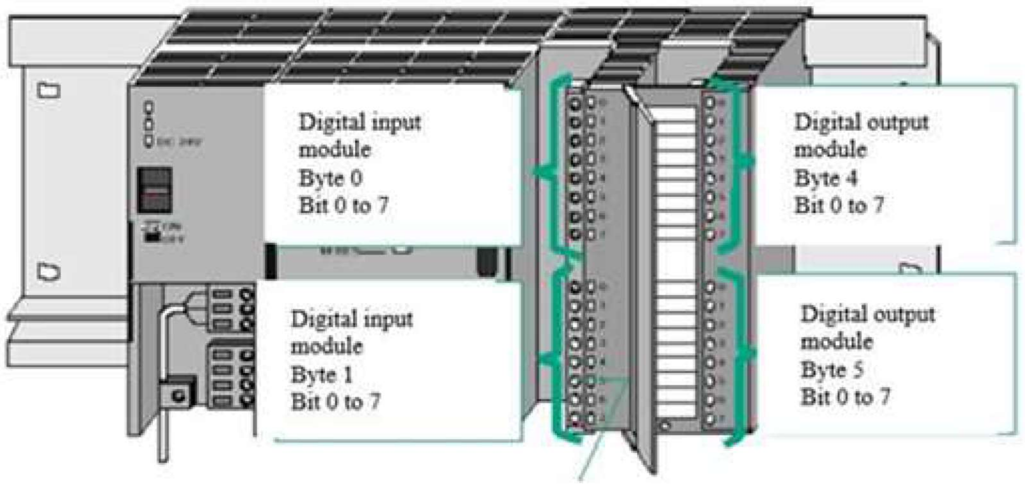

Digital input module Byte 0 Bit 0 to 7
Digital input module Byte 1 Bit 0 to 7
Digital output module Byte 4 Bit 0 to 7
Digital output module Byte 5 Bit 0 to 7

Figure I. 10: *Scheme of Siemens CPU with digital input module 0 and 1, and digital output module 4 and 5.*

Consequently, complete address of input pin is *IX.Y.*Where *X* is the byte address and *Y* is the pin address.

For example, given the PLC shown in pervious Fig, the address of the fifth input from above is *I0.4* because the byte address is 0 and the bit address is 4.

While type of output pin complete address is *QX.Y.*Where X is the byte address and Y is the pin address.

For example, the address of the last exit is *Q5.7* because the byte address is 5 and the bit address is 7.

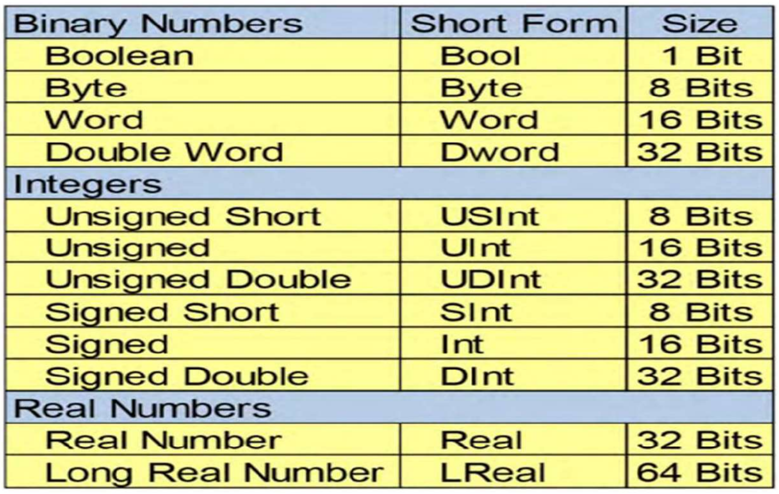
This type of addressing applies to both analog and digital signals. Specifically, an analog signal's address indicates a word and not a single bit because analog signals are words.

Finally, we say that addressing is activity that permits translation of a physical signal into a virtual logical signal.

II.3.7. Data Types:

Binary bits can be used independently or as a group. When used as a group, they are used to represent numerical values as well as other types of data.

In order to know how a bit or bit string will be interpreted by a PLC, you must know the data type, which is the PLCs description of the data. PLC data types of various lengths are specified for binary numbers, integers (also called whole numbers), real numbers (also called floating-point numbers), date and time, characters, parameters, system data, and other types of data. Because the number of data types has increased overtime, not all data types are available for all SIMATIC S7 PLCs.



Binary Numbers	Short Form	Size
Boolean	Bool	1 Bit
Byte	Byte	8 Bits
Word	Word	16 Bits
Double Word	Dword	32 Bits
Integers		
Unsigned Short	USInt	8 Bits
Unsigned	UInt	16 Bits
Unsigned Double	UDInt	32 Bits
Signed Short	SInt	8 Bits
Signed	Int	16 Bits
Signed Double	DInt	32 Bits
Real Numbers		
Real Number	Real	32 Bits
Long Real Number	LReal	64 Bits

Table 1. Data Types 1.

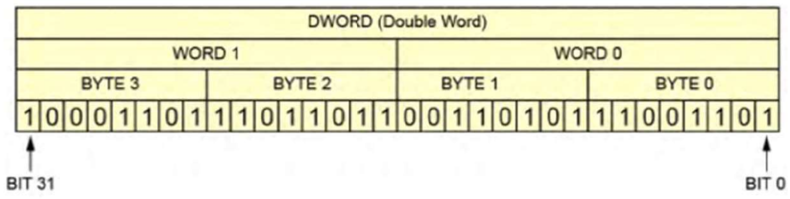


Table 2. Data types 2.

II.4. HMI:

II.4.1. Definition:

A Human-Machine Interface (HMI) is a user interface or dashboard that connects a person to a machine, system, or device. While the term can technically be applied to any screen that allows a user to interact with a device, HMI is most commonly used in the context of an industrial process.

Although HMI is the most common term for this technology, it is sometimes referred to as Man-Machine Interface (MMI), Operator Interface Terminal (OIT), Local Operator Interface (LOI), or Operator Terminal (OT). HMI and Graphical User Interface (GUI) are similar but not synonymous: GUIs are often leveraged within HMIs for visualization capabilities.

In industrial settings, HMIs can be used to:

- Visually display data
- Track production time, trends, and tags
- Oversee KPIs
- Monitor machine inputs and outputs…

II.4.2. Common Uses of HMI:

HMIs communicate with Programmable Logic Controllers (PLCs) and input/output sensors to get and display information for users to view. HMI screens can be used for a single function, like monitoring and tracking, or for performing more sophisticated operations, like switching machines off or increasing production speed, depending on how they are implemented.

HMIs are used to optimize an industrial process by digitizing and centralizing data for a viewer. By leveraging HMI, operators can see important information displayed in graphs, charts, or digital dashboards, view and manage alarms, and connect with SCADA and MES systems, all through one console.

II.5. Conclusion:

In this chapter, we learned about automation and what are the different types of hardware used in this domain in general way, while we take a small scoop of what a PLC is and how it works.

In the next chapter, we talk about TIA Portal program, so we can go further into programming a SIEMENS PLC. And on top of that we also introduce Factory IO Software that allows us to simulate in real time.

Chapter II:
Tia Portal

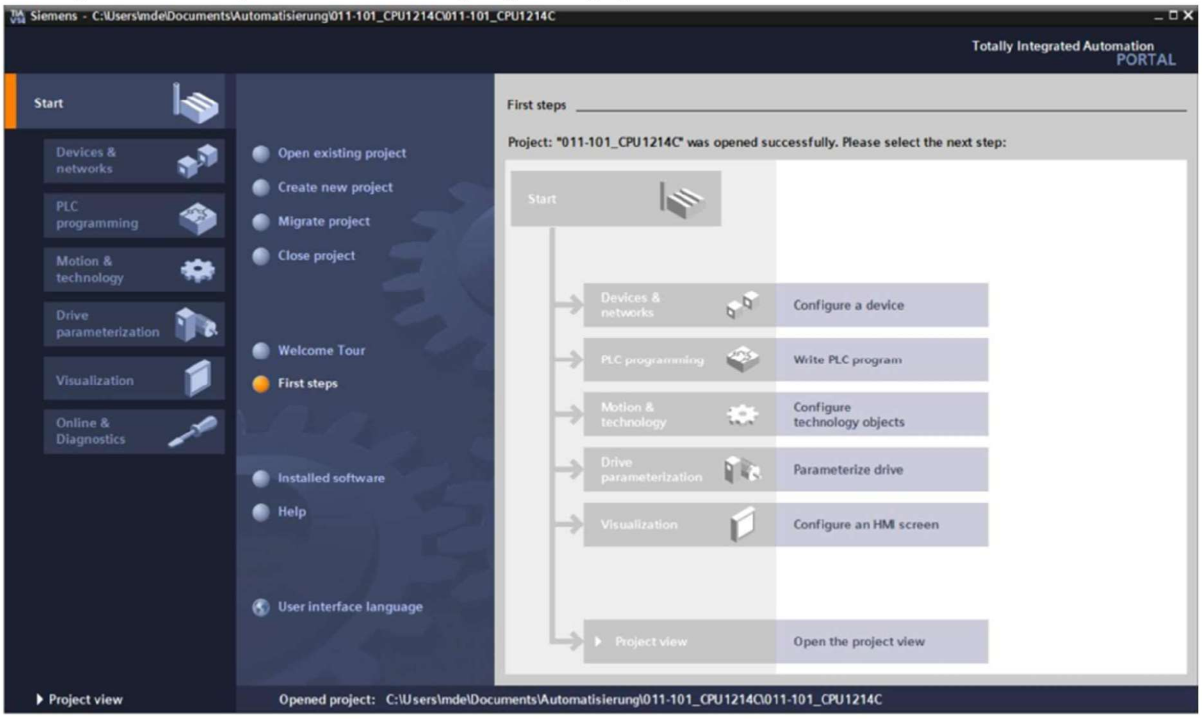
This chapter contains a tutorial on TIA Portal and Factory IO, in the Tia portal tutorial we see how different programming blocks function and we go further to the basic instructions that we are using in our last chapter.

III.1. Totally Integrated Automation Portal (TIA Portal):

Siemens Totally Integrated Automation Portal (TIA Portal) is engineering software that is used in all phases of the design, operation, and maintenance of systems that can include Siemens PLCs, HMIs, PCs, electronic drives, and related devices.

TIA Portal combines the software editors needed for these various tasks in one engineering tool with a common layout and navigation design. This integrated, intuitive approach speeds the learning process and allows experienced users to function more efficiently at the start of a project, the user can choose between the portal view, which guides the user through each engineering step, and the project view, which offers fast access to all the relevant tools. With one click, a user can toggle between views.

The portal view is a good place to start for a new user or anyone who wants to logically proceed with the development of a new project or continue with the development of an existing project. The available tasks are clearly identified, such as "Create new project," Configure a device," or "Write PLC program".[B]


Siemens - C:\Users\mde\Documents\Automatisierung\011-101_CPU1214C\011-101_CPU1214C

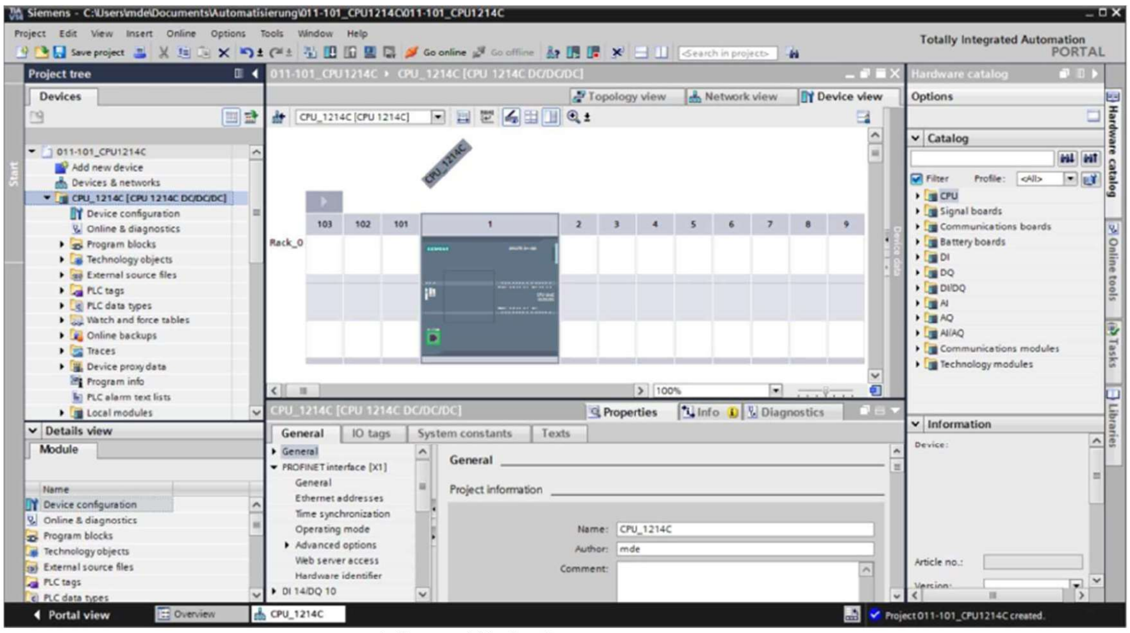
Project: "011-101_CPU1214C" was opened successfully. Please select the next step:

Opened project: C:\Users\mde\Documents\Automatisierung\011-101_CPU1214C\011-101_CPU1214C

Figure II. 1: Portal View.

The project view, as shown below, it is used for hardware configuration, programming, creation of the visualization and many other tasks.

If an element (for example, the device configuration) is selected in the project tree, it is displayed in the center and can be worked on there.



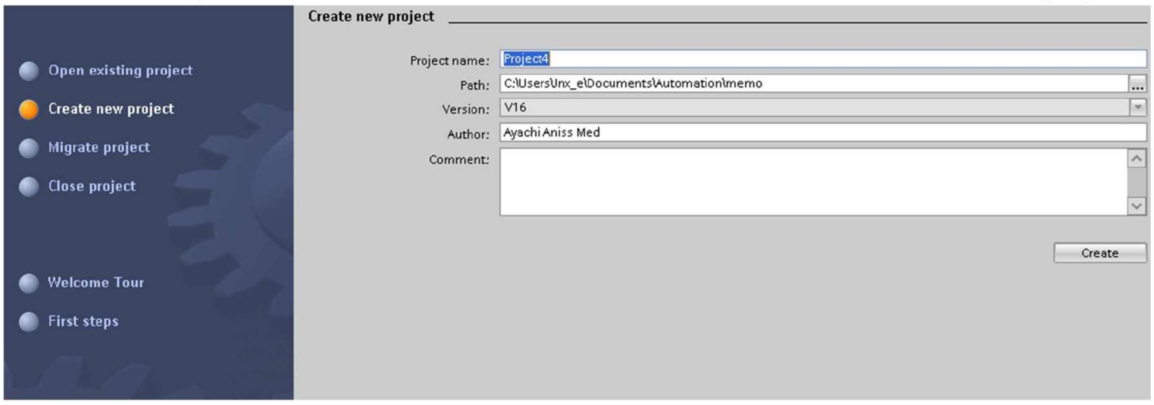
Name: CPU_1214C
Author: mde
Comment:

Figure II. 2: Project view.

III.1.1. TIA Portal guide

III.1.1.1. Create new project:

In the portal view under the "Start" menu, select the command "Create new project".



Create new project

Project name: Project4
Path: C:\Users\nx_e\Documents\Automation\memo
Version: V16
Author: Ayachi Aniss Med
Comment:

Figure II. 3: Portal View.

Modify the project name, path, author ,and comment if there is. Then press, “Create”.

III.1.1.2. Insert the CPU: After creating a project, go to “Configure a device”.

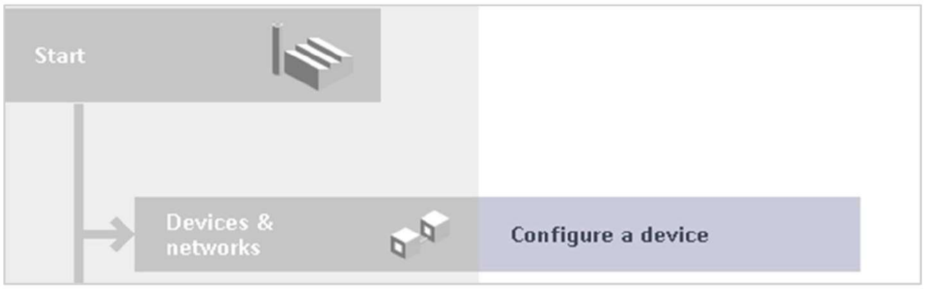
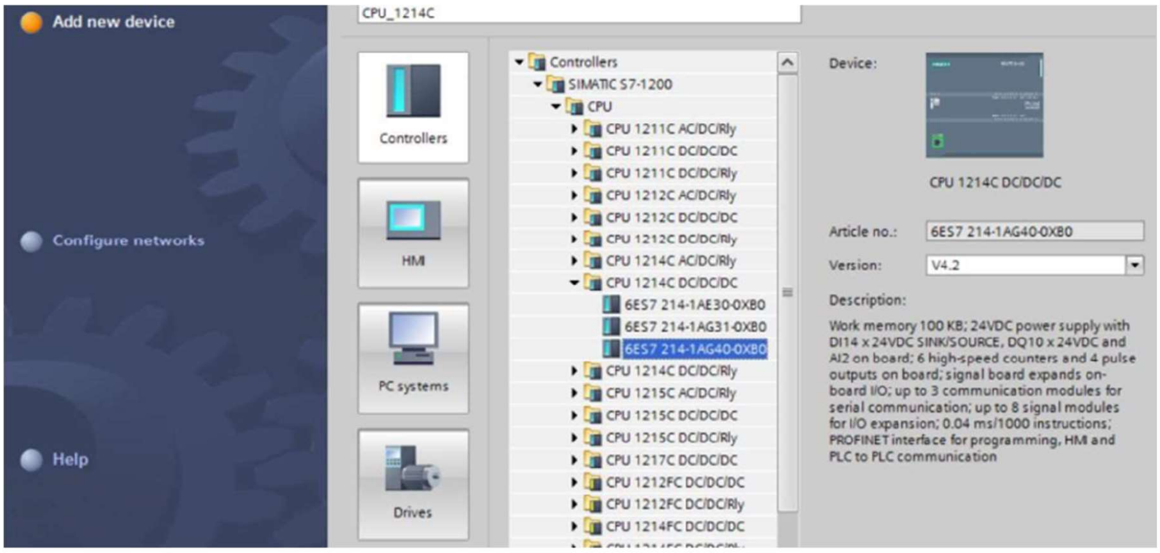


Figure II. 4: Configure a device.

Then “Add new device” and make sure you select controllers, then read your device model and reference and click “Add”.



- Controllers
  - SIMATIC S7-1200
    - CPU
      - CPU 1211C AC/DC/Rly
      - CPU 1211C DC/DC/DC
      - CPU 1211C DC/DC/Rly
      - CPU 1212C AC/DC/Rly
      - CPU 1212C DC/DC/DC
      - CPU 1212C DC/DC/Rly
      - CPU 1214C AC/DC/Rly
      - CPU 1214C DC/DC/DC
        - 6ES7 214-1AE30-0XB0
        - 6ES7 214-1AG31-0XB0
        - 6ES7 214-1AG40-0XB0
      - CPU 1214C DC/DC/Rly
      - CPU 1215C AC/DC/Rly
      - CPU 1215C DC/DC/DC
      - CPU 1215C DC/DC/Rly
      - CPU 1217C DC/DC/DC
      - CPU 1212FC DC/DC/DC
      - CPU 1212FC DC/DC/Rly
      - CPU 1214FC DC/DC/DC

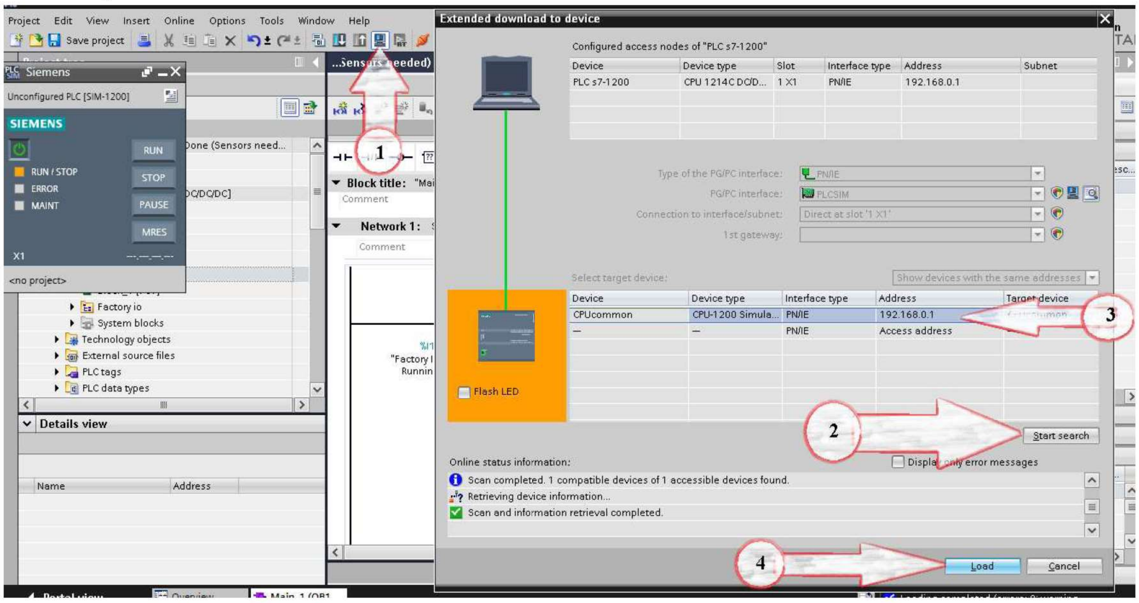
Device:
CPU 1214C DC/DC/DC
Article no.: 6ES7 214-1AG40-0XB0
Version: V4.2
Description:
Work memory 100 KB; 24VDC power supply with DI14 x 24VDC SINK/SOURCE, DQ10 x 24VDC and AI2 on board; 6 high-speed counters and 4 pulse outputs on board; signal board expands on-board I/O; up to 3 communication modules for serial communication; up to 8 signal modules for I/O expansion; 0.04 ms/1000 instructions; PROFINET interface for programming, HMI and PLC to PLC communication

Figure II. 5:*Adding a controller.*

III.1.1.3. PLCSIM:

To simulate the program with PLCSIM after adding the material and making the program

1) Click on Start simulation.
2) Then search for the CPU used (wait *for the PLCSIM to load and be configured)*
3) Select the CPU and click Load



Extended download to device

Configured access nodes of "PLC s7-1200"

Device	Device type	Slot	Interface type	Address	Subnet
PLC s7-1200	CPU 1214C DC/D...	1 X1	PN/IE	192.168.0.1	

Type of the PG/PC interface: PN/IE
PG/PC interface: PLCSIM
Connection to interface/subnet: Direct at slot '1 X1'
1st gateway:

Select target device: Show devices with the same addresses

Device	Device type	Interface type	Address	Target device
CPUcommon	CPU-1200 Simula...	PN/IE	192.168.0.1	CPUcommon
—	—	PN/IE	Access address	

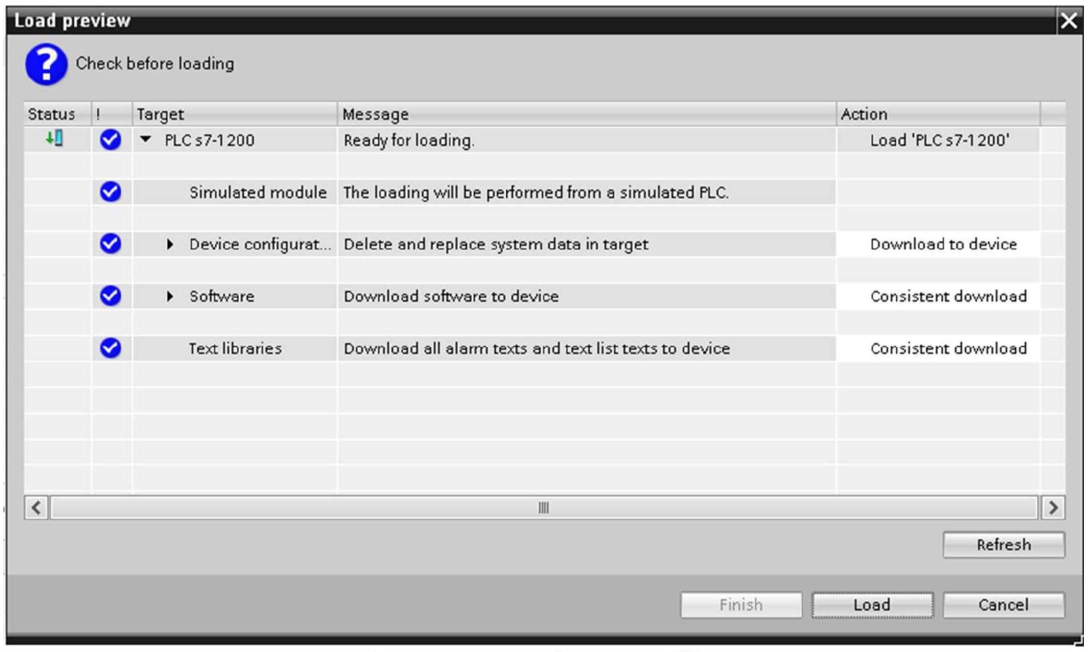
Start search

Online status information:
Display only error messages
Scan completed. 1 compatible devices of 1 accessible devices found.
Retrieving device information...
Scan and information retrieval completed.

Load
Cancel

Figure II. 6:*PLCSIM simulation.*

4) Wait for the program to compile check for errors if there is none, click Load (Fig below)



Load preview

Check before loading

Status	!	Target	Message	Action
	✓	PLC s7-1200	Ready for loading.	Load 'PLC s7-1200'
	✓	Simulated module	The loading will be performed from a simulated PLC.	
	✓	Device configurat...	Delete and replace system data in target	Download to device
	✓	Software	Download software to device	Consistent download
	✓	Text libraries	Download all alarm texts and text list texts to device	Consistent download

Refresh

Finish Load Cancel

***Figure II.** 7:Loading to PLCSIM.*

5) When the program is loaded Click RUN in PLCSIM, wait for the green LED to turn to green without blinking.
6) Click on monitoring to monitor the changes in your program.

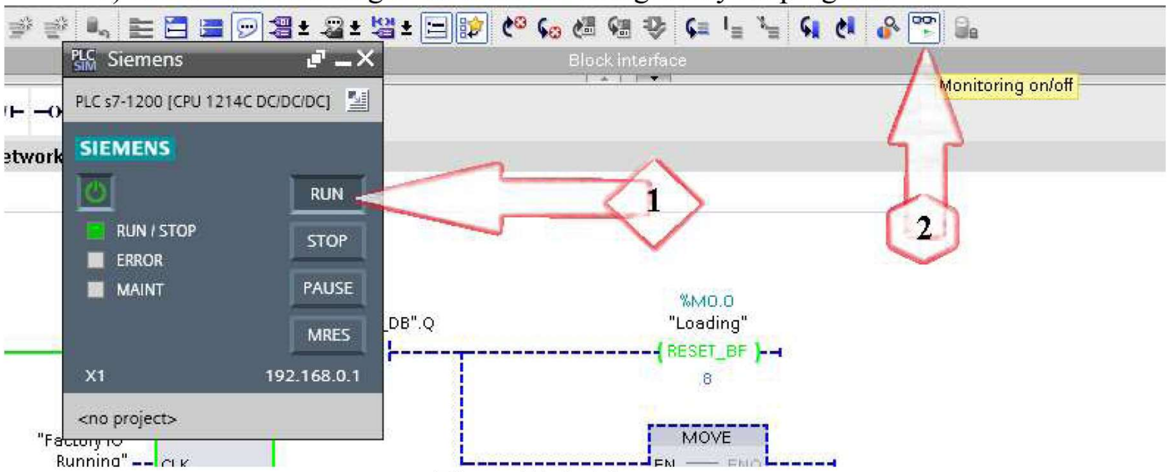


***Figure II. 8:** Monitoring.*

III.1.1.4. Program Blocks:

An S7 PLC program can include data blocks (DBs) and three types of program blocks: organization blocks (OBs), function blocks (FBs), and functions (FCs). Program blocks are blocks that include instructions.

To add any program block in the project we click on "Add new block"

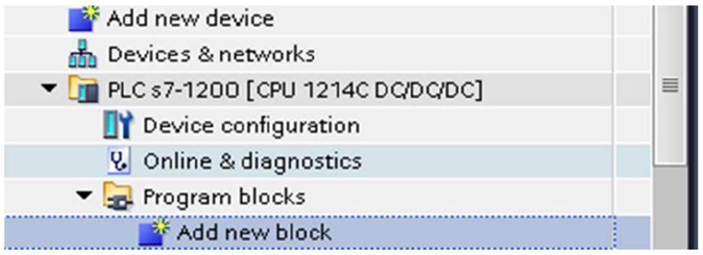


Figure II. 9: *Adding a new block.*

Data blocks (DBs):are used to store data. A global DB stores data that can be used by any program block, while an instance DB stores data for a specific function block (FB).

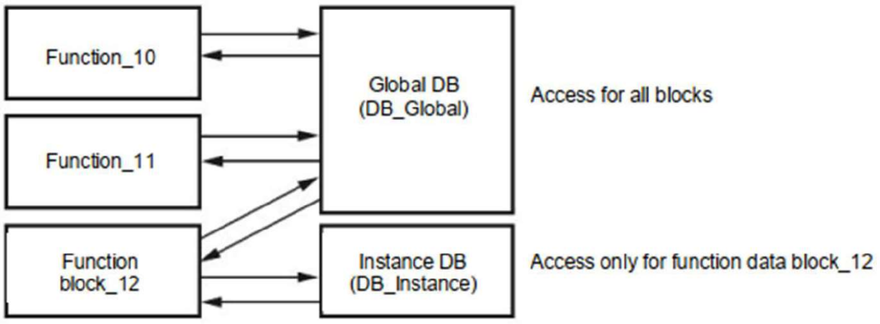


Figure II. 10: *Data Blocks.*

Organization blocks (OBs) define the structure of the program. Every program must have at least one OB. If it has only one OB, that block is identified as OB 1. OB 1 is the main program block, and it controls the execution of the user program. There is many more OBs and every OB has its specific job, it all depends on what you want to work with in your program.

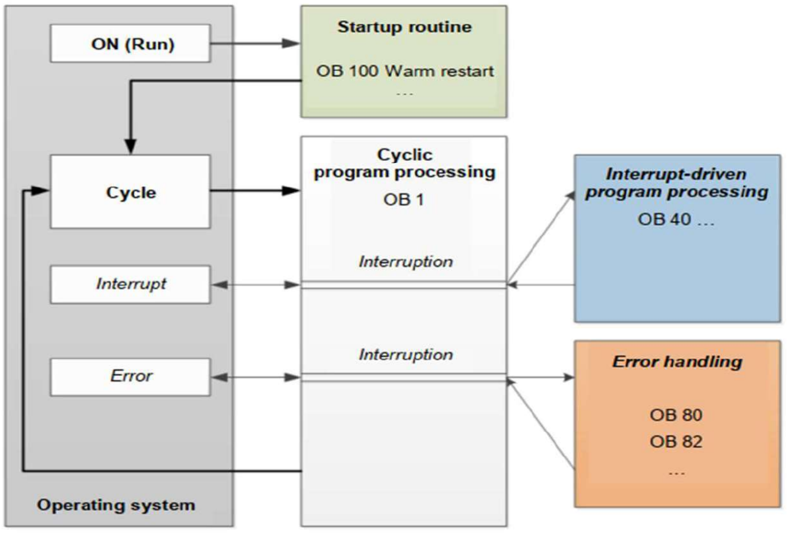


Figure II. 11:*Organization Blocks (OBs).*

FCs and FBs: FCs and FBs contain the program code that performs specific tasks. An FB uses an associated data block, called an instance DB, that stores data for that FB. Another type of data block, called a global data block, contains data that is available to any program block. An FC does not have an instance data block, and the output data values from the FC must be written to a memory address or to a global DB. [B]

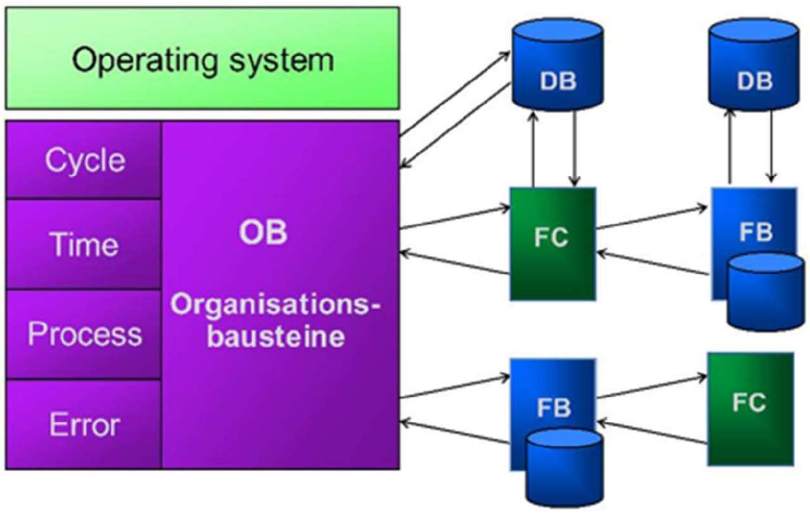


Figure II. 12: *Example of program blocks.*

III.1.1.5. Basic instructions**:**

- Normally open contact: ---| |---
  The activation of the normally open contact depends on the signal state of the associated operand. When the operand has signal state "1", the normally open contact closes and the signal state at the output is set to the signal state of the input. When the operand has signal state "0", the normally open contact is not activated and the signal state at the output of the instruction is reset to "0". [C]
  Example:

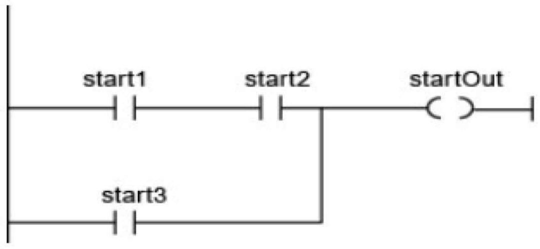


Figure II. 13: *Normally open contact.*

  In this example our Output will be set to 1 in case of both Start1 AND Start 2 Are ON, OR Start 3 is ON.

- Normally closed contact: ----| / |---

This instruction is the opposite of the previous one. Meaning if the Input is 1 then the Output is 0.

- Assignment: ---( )---
  You can use the "Assignment" instruction to set the bit of a specified operand. If the result of logic operation (RLO) at the input of the coil has signal state "1", the specified operand is set to signal state "1". If the signal state is "0" at the input of the coil, the bit of the specified operand is reset to "0".
  The instruction does not influence the RLO. The RLO at the input of the coil is sent directly to the output. [C]
- Set output: ---(S)---
  This instruction is only executed if the result of logic operation (RLO) at the input of the coil is "1". If power flows to the coil (RLO = "1"), the specified operand is set to "1". If the RLO at the input of the coil is "0" (no signal flow to the coil), the signal state of the specified operand remains unchanged. [C]
  Example:

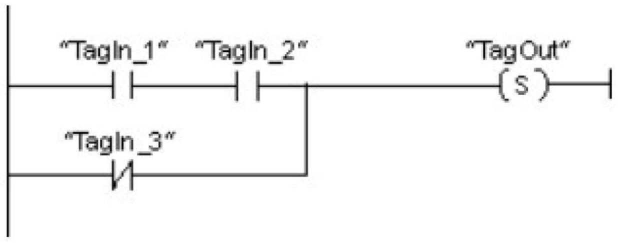


***Figure II. 14:** Set Output.*

In this example "TagOut" will be set to 1 if "TagIn_1" AND "TagIn_2" are ON in the same time, OR "TagIn_3" is OFF. No matter how long they stay ON, "TagOut" will remain ON.

- Set bit field: SET_BF
  This instruction is used to set several bits starting from a certain address.
  Example: Setting ON %M0.0 to %M0.4

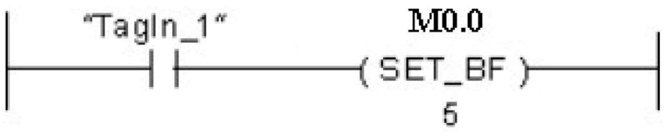


***Figure II. 15:** Set bit filed.*

- Reset output: ---(R)---
  The instruction is only executed if the result of logic operation (RLO) at the input of the coil is "1". If power flows to the coil (RLO = "1"), the specified operand is reset to "0". If the RLO at the input of the coil is "0" (no signal

flow to the coil), the signal state of the specified operand remains unchanged. Example: Resetting "TagOut" from "Fig II. 16"

- Reset bit field: RESET_BF
  This instruction is to reset several bits starting from a certain address.

- Set/reset flip-flop: SR
  This instruction is used to Set an Output to 1 if S=1 And Reset the same Output to 0 if R1=1. If S=1 AND R1=1 then the priority goes to R1. [C]
- Reset/set flip-flop: RS
  It's the same as the previous one, except the priority goes to S1 if they are both on.
  Example:

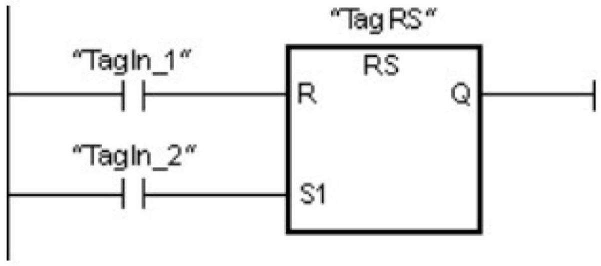


Figure II. 17: *Reset/Set.*

- Detect positive signal edge: R_TRIG
  With the "Detect positive signal edge" instruction, you can detect a state change from "0" to "1" at the CLK input. The instruction compares the current value at the CLK input with the state of the previous query (edge memory bit) that is saved in the specified instance. If the instruction detects a state change at the CLK input from "0" to "1", a positive signal edge is generated at the Q output, i.e., the output has the value TRUE or "1" for exactly one cycle.
  In all other cases, the signal state at the output of the instruction is "0".[C]
  Example:

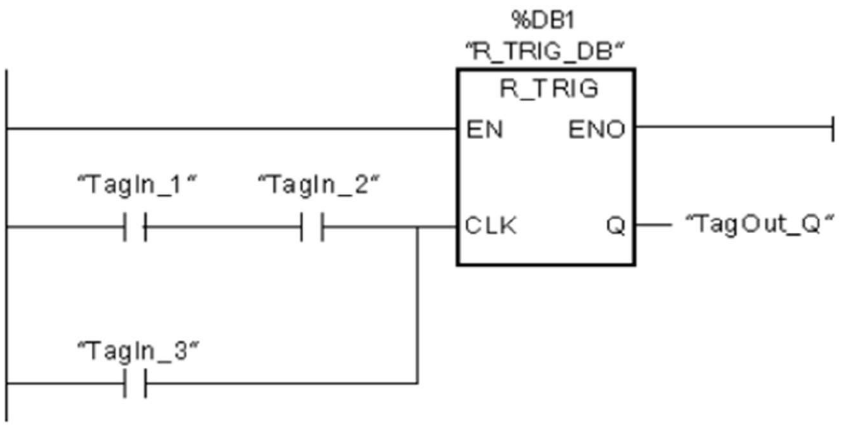


Figure II. 18: *R_Trig.*

"TagOut_Q" is set 1 for one cycle if "TagIn_1" AND "TagIn_2" change their stat from 0 to 1, OR if "TagIn_3" change from 0 to 1.
You can also use this form below which is the same as the one above.

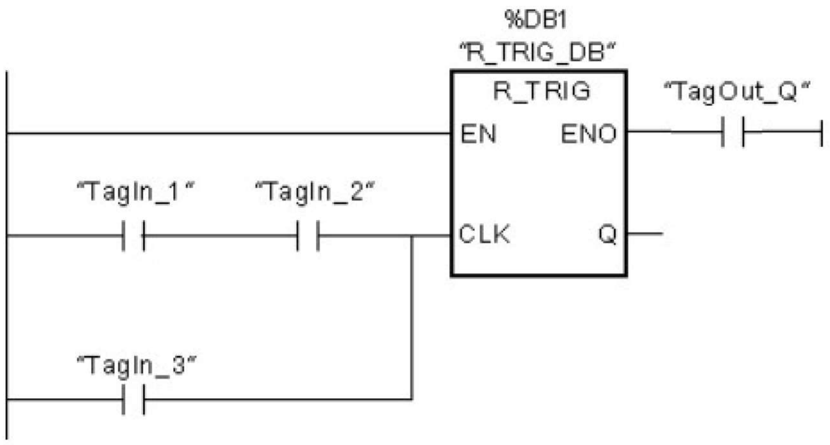


Figure II. 19: *R_Trig.*

- <u>Detect negative signal edge:</u> F_TRIG
  This instruction is the opposite of the previous one, where this one detects the change from "1" to "0".
  Example:

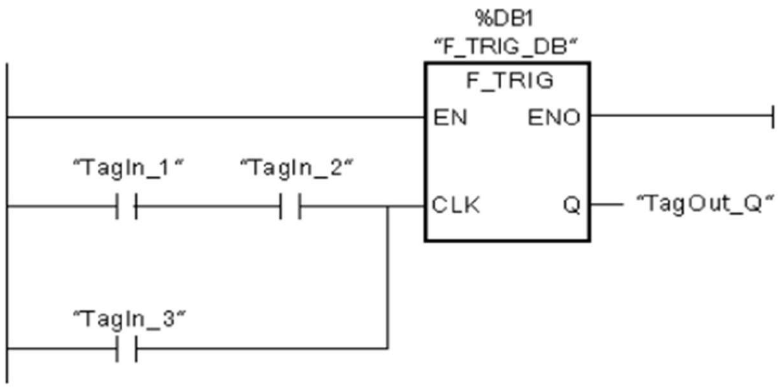


Figure II. 20: *F_Trig.*

"TagOut_Q" is set to 1for one cycle if "TagIn_1" AND "TagIn_2" change from 1 to 0, OR "TagIn_3" change stat from 1 to 0.

- <u>Move value:</u> MOVE

  You use the "Move value" instruction to transfer the content of the operand at the IN input to the operand at the OUT1 output. The transfer is always made in the direction of ascending address. [C]

  Example:

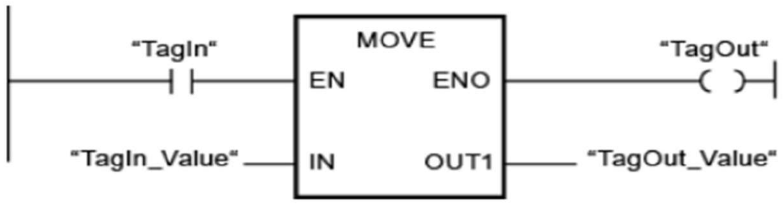


Figure II. 21: *Move.*

When "TagIn" is 1 the IN Value goes to OUT1, So that "TagIn_Value" is equal to "TagOut_Value" while "TagOut" is set to 1.

- <u>Generate on-delay:</u> TON

  You can use the "Generate on-delay" instruction to delay setting of the Q output by the programmed time PT. The instruction is started when the result of logic operation (RLO) at input IN changes from "0" to "1" (positive signal edge). The programmed time PT begins when the instruction starts. When the time PT has elapsed, the output Q has the signal state "1". Output Q remains set as long as the start input is still "1". When the signal state at the start input changes from "1" to "0", the Q output is reset. The timer function is started again when a new positive signal edge is detected at the start input. [C]

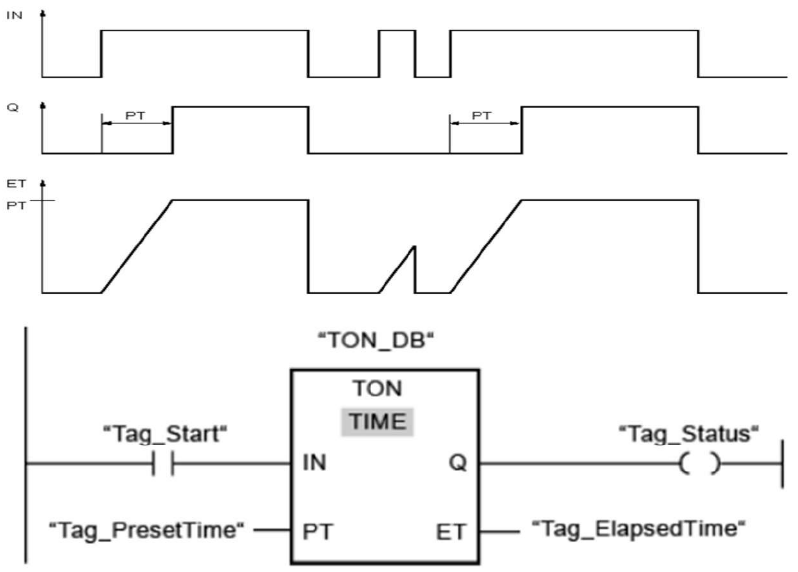


Figure II. 22: TON.

- Generate off-delay: TOF

  You can use the "Generate off-delay" instruction to delay resetting of the Q output by the programmed time PT. The Q output is set when the result of logic operation (RLO) at input IN changes from "1" to "0" (negative signal edge). When the signal state at input IN changes back to "1", the programmed time PT starts. Output Q remains set as long as the time duration PT is running. When the PT time duration expires, the Q output is reset. If the signal state at input IN changes to "1" before the PT time duration expires, the timer is reset. The signal state at the output Q continues to be "1".[C]

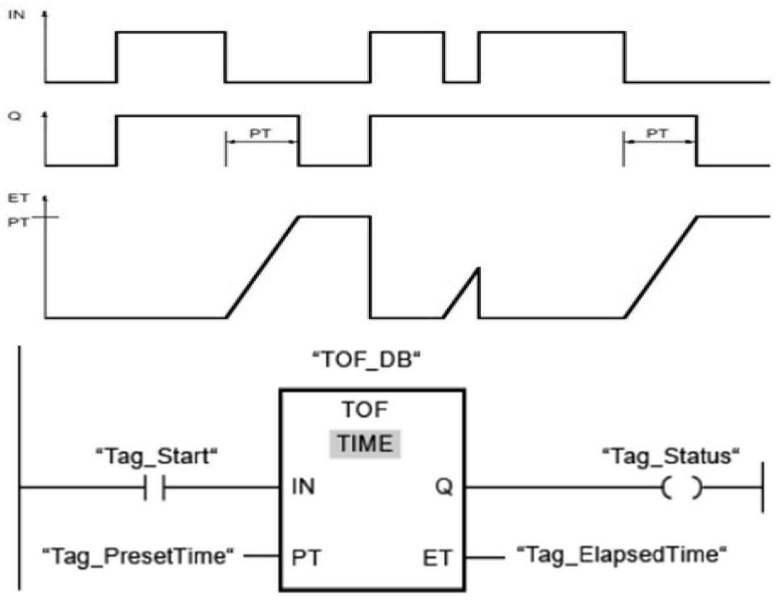


Figure II. 23: TOF.

- Comparator operations:

  All Comparator Instructions work with the same concept.

  Example: *CMP ==: Equal*

  You can use the "Equal" instruction to determine if a first comparison value (<Operand1>) is equal to a second comparison value (<Operand2>).

  If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

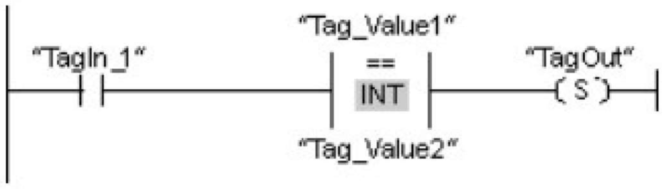


Figure II. 24: *CMP.*

  If "Tag_Value1" = ”Tag_Value2"then “TagOut” is set to 1 whenever “TagIn_1” is 1.

- Count up and down: CTUD

  You can use the "Count up and down" instruction to increment and decrement the counter value at the CV output.CU for countering up and CD to count down, if they both go to “1” then “0” then CV “counter value” will remain unchanged.

  When the signal state at the LD input changes to "1", the counter value at the CV output is set to the value of the PV parameter. As long as the LD input has signal state "1", the signal state at the CU and CD inputs has no effect on the instruction.

  The counter value is set to zero when the signal state at the R input changes to "1". As long as the R input has signal state "1", a change in the signal state of the CU, CD and LD inputs has no effect on the instruction.

  You can query the status of the up counter at the QU output. If the current counter value is greater than or equal to the value of the PV parameter, the QU output is set to signal state "1". In all other cases, the QU output has signal state "0".

  You can query the status of the down counter at the QD output. If the current counter value is less than or equal to zero, the QD output is set to signal state "1". In all other cases, the QD output has signal state "0".[C]

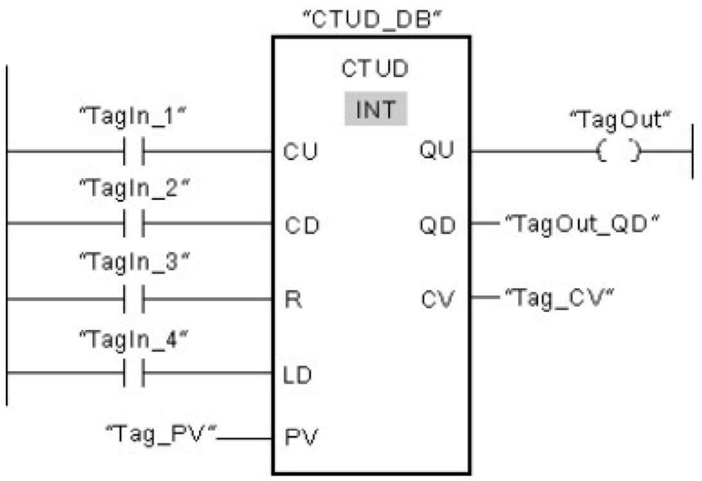


***Figure II.* 25:** *CTUD.*

III.2. Factory I/O:

III.2.1. Definition:

FACTORY I/O is the new generation of Operative Parts simulation software developed by Real Games for learning automation. FACTORY I / O allows you to easily and quickly design your production line by simply assembling the subsystems available in the library provided.

The simulated systems interface very easily with real PLCs (SCHNEIDER, SIEMENS, etc.) or simulated using the many communication drivers available to you (DAQ 4750, DAQ 4704, MODBUS and OPC, PLCSIM ...). With FACTORY I/O, we can have a "virtual" factory with digital and analog I/O.[D]

III.2.2. Creating a scene:

Factory I/O includes a large selection of parts inspired by the most common industrial equipment.

You create a virtual factory by placing and arranging these parts together. By following the next steps, you will learn to create a simple sorting system.

- Click on *File*; choose New (*Ctrl + N*) to create an empty scene.
- On the Toolbar, select the Orbit camera - you should use the Orbit camera when editing. If the Palette is not visible click on the Palette icon.

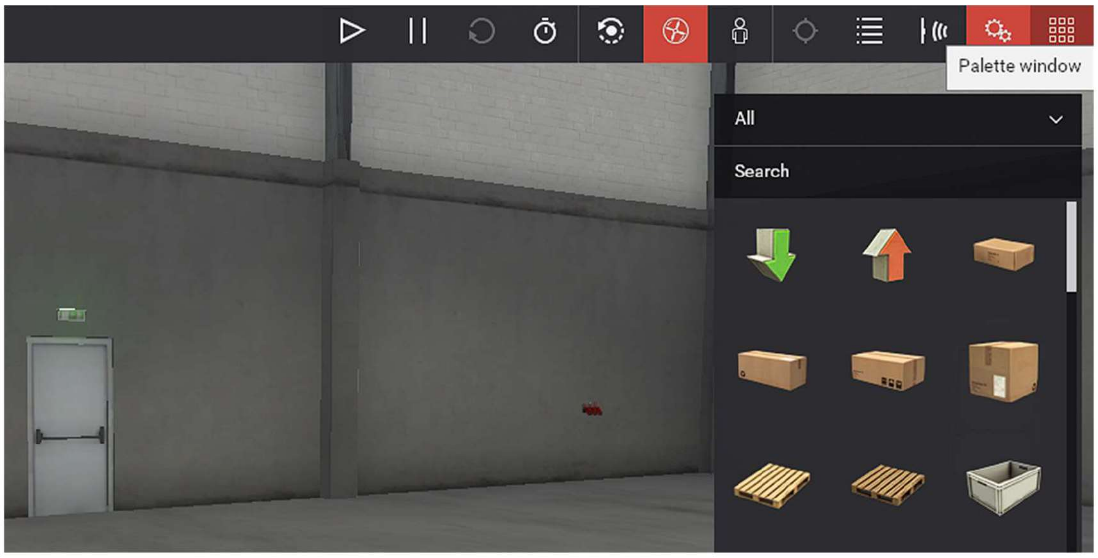
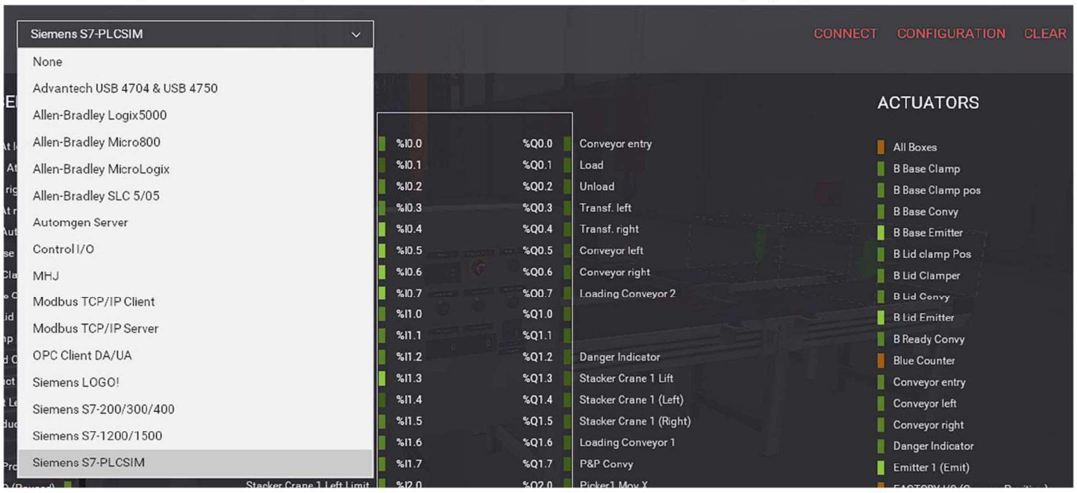


Figure II. 26: Creating scene.

- Drag and drop what items you need for you scene.

III.2.3. Controlling With a PLC:

- Open the Drivers window by clicking on *File* and next on *Drivers (F4)*. Alternatively, you may open the Drivers window by *Left-clicking* on the current driver displayed on the status bar. And choose S7-PLCSIM or S7-1200 (depends if you have only PLCSIM or you have an S7-1200). [D]



%I0.0	%Q0.0	Conveyor entry
%I0.1	%Q0.1	Load
%I0.2	%Q0.2	Unload
%I0.3	%Q0.3	Transf. left
%I0.4	%Q0.4	Transf. right
%I0.5	%Q0.5	Conveyor left
%I0.6	%Q0.6	Conveyor right
%I0.7	%Q0.7	Loading Conveyor 2
%I1.0	%Q1.0	
%I1.1	%Q1.1	
%I1.2	%Q1.2	Danger Indicator
%I1.3	%Q1.3	Stacker Crane 1 Lift
%I1.4	%Q1.4	Stacker Crane 1 (Left)
%I1.5	%Q1.5	Stacker Crane 1 (Right)
%I1.6	%Q1.6	Loading Conveyor 1
%I1.7	%Q1.7	P&P Convy
%I2.0	%Q2.0	Picker1 Mov X

Figure II. 27: Connecting a PLC.

- Configure you device then click connect

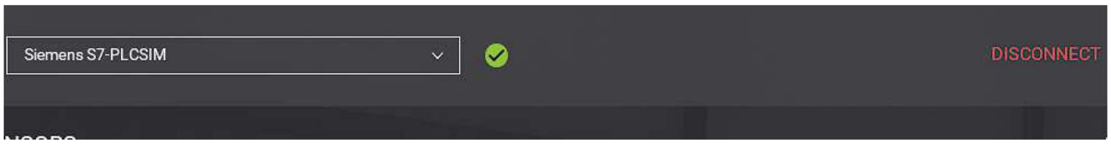


Figure II. 28: *Connecting to PLCSIM.*

III.3. Conclusion:

We conclude that TIA portal is a useful easy program to use for programming S7 PLCs As we are done from the quick guide through these programs, we introduce our hardware and software (virtual) material with brief description of each item

Chapter III: Description of the material

IV.1. Hardware:

IV.1.1. PLC:

IV.1.1.1. S7-1200:

S7-1200 CPUs are available in three versions, the standard, failsafe and SIPLUS CPUs .And every S7-1200 have at least one PROFINET port in some other versions they have two ports.S7-1200 has an internal memory, to store you program and various other functions. However, you can still use a memory card to extend your internal memory or copy your program.

Every S7-1200 have a number of inputs and outputs. However, you can add many other available extensions if needed.[B]

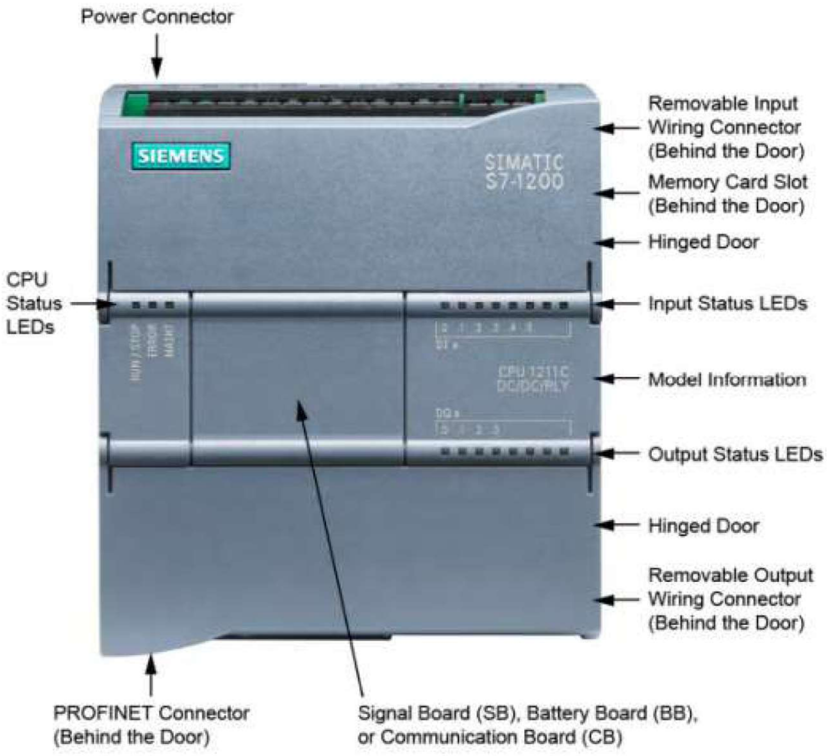

Power Connector
Removable Input Wiring Connector (Behind the Door)
Memory Card Slot (Behind the Door)
Hinged Door
CPU Status LEDs
Input Status LEDs
Model Information
Output Status LEDs
Hinged Door
Removable Output Wiring Connector (Behind the Door)
PROFINET Connector (Behind the Door)
Signal Board (SB), Battery Board (BB), or Communication Board (CB)
SIEMENS
SIMATIC S7-1200
CPU 1211C DC/DC/RLY

Figure III. 1: *S7-1200.*

IV.1.1.2. LED Status:

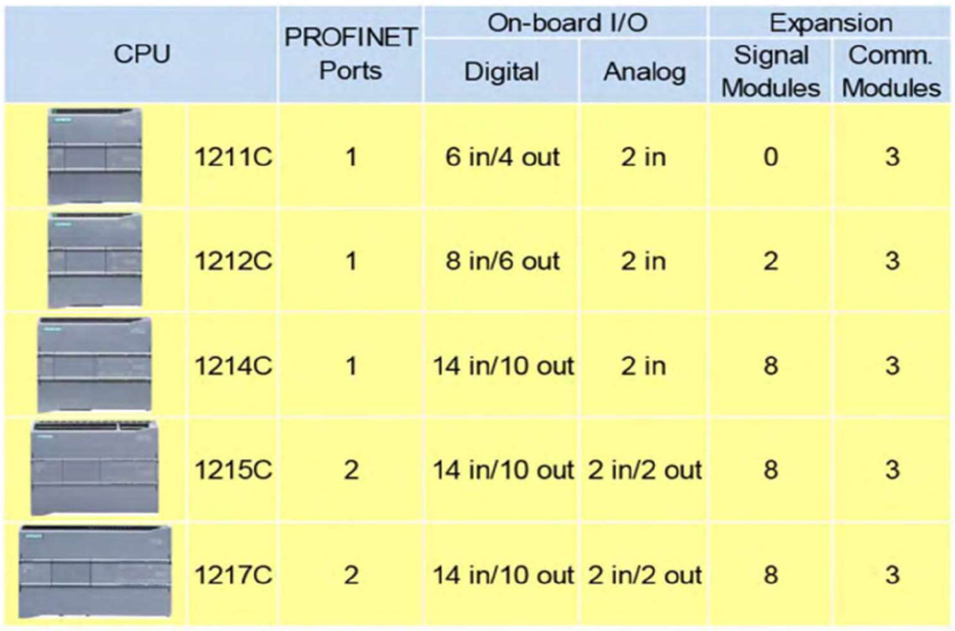
- CPU LEDS:

LED	Condition	Indication
RUN/STOP	STOP Mode	Constant Yellow
	STARTUP Mode	Alternatively Flashing Green/Yellow
	RUN Mode	Constant Green
ERROR	CPU, Memory Card, or Configuration Error	Flashing Red
	Defective Hardware	Constant Red
MAINT	When a Memory Card is Inserted	Flashing Yellow
	Maintenance Requested	Constant Yellow

Table 3: *LED Status.*

- I/O LEDS: Every I/O has a green LED, which goes on when the I/O is ON.

IV.1.1.3. CPUs:



CPU		PROFINET Ports	On-board I/O		Expansion	
			Digital	Analog	Signal Modules	Comm. Modules
	1211C	1	6 in/4 out	2 in	0	3
	1212C	1	8 in/6 out	2 in	2	3
	1214C	1	14 in/10 out	2 in	8	3
	1215C	2	14 in/10 out	2 in/2 out	8	3
	1217C	2	14 in/10 out	2 in/2 out	8	3

Figure III. 2: *S7-1200 CPUs.*

We use the CPU1214C DC/DC/DC 6ES7 214-1AG40-0XB0.

IV.1.2. SITOP PSU100S:

Is a primary-clocked power supply for connection to a 1-phase AC line supply. An electronically regulated DC voltage that can be set via a potentiometer is available at the output of the device. The LED display indicates the operating status. The state of the device can be processed via the signaling contact. [E]

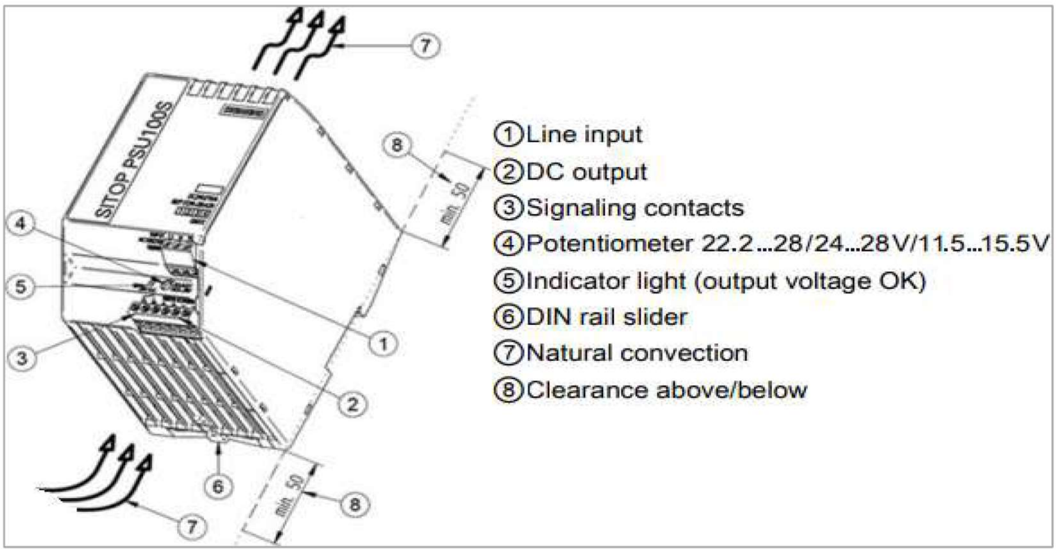


Figure III. 3: *SITOP PSU100S.*

IV.1.3Profinet:

PLCs use a variety of communication technologies. The most basic type of communication used is serial communication, where bits are sent and received one at a time. Serial communication is still used with some devices; however, more often, PLCs use network communication.

IV.1.3.1. Definition:

PROFINET is an open Industrial Ethernet standard and the leading Industrial Ethernet standard worldwide. PROFINET IO, the most widely used form of PROFINET, handles both non-time-critical IT communications and the full range of real-time control communications. [B]

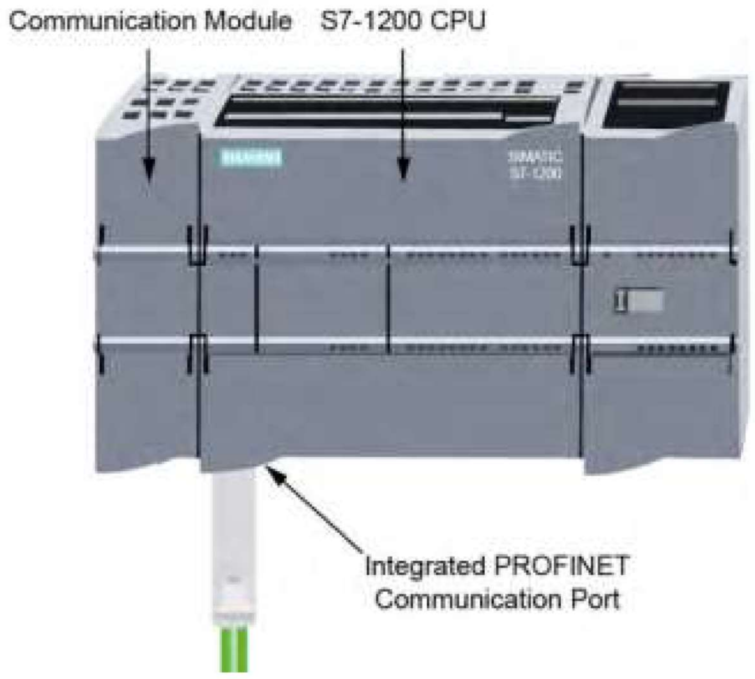


Figure III. 4: *Profinet port.*

IV.1.3.2. Device types:

A Profinet system consists of the following devices:

- The *IO-Controller*, which controls the automation task.
- The *IO-Device*, which is a field device, monitored and controlled by an IO-Controller. An IO-Device may consist of several modules and sub-modules.
- The *IO-Supervisor* is software typically based on a PC for setting parameters and diagnosing individual IO-Devices. [F]

IV.1.4. -HMI:

SIMATIC HMI TP900 COMFORT, COMFORT PANEL, TOUCH OPERATION, 9" WIDESCREEN-TFT-DISPLAY, 16 MIL. COLORS, PROFINET INTERFACE, MPI/PROFIBUS DP INTERFACE, 12 MB USER MEMORY, WINDOWS CE 6.0, CONFIGURABLE FROM WINCC COMFORT V11. [G]

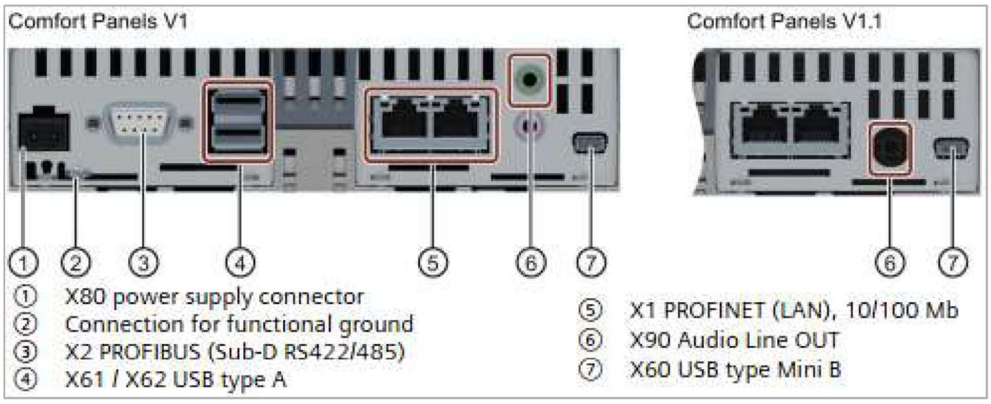

Comfort Panels V1
Comfort Panels V1.1
① X80 power supply connector
② Connection for functional ground
③ X2 PROFIBUS (Sub-D RS422/485)
④ X61 / X62 USB type A
⑤ X1 PROFINET (LAN), 10/100 Mb
⑥ X90 Audio Line OUT
⑦ X60 USB type Mini B

Figure III. 5: *TP900 Comfort.*

IV.1.5. Industrial Ethernet Switch

Industrial Ethernet Switch, which is an Ethernet switch device used in industrial control. It uses a transparent and unified TCP/IP protocol.

SCALANCE X208 6GK5208-0BA10-2AA3, managed IE switch, 8x 10/100 Mbit/s RJ45 ports, LED diagnostics, error-signaling contact with set button, redundant power supply, PROFINET IO device, network management.

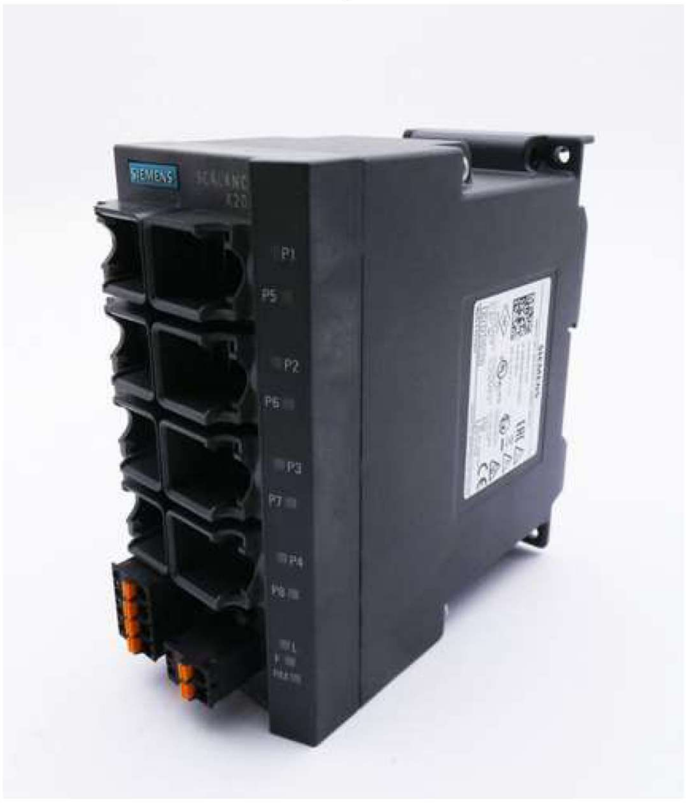


Figure III. 6: *Industrial Ethernet switch.*

IV.2. Computer:

- Operating System: Windows 10 Pro 64-bit.
- CPU: Intel Core i5 8600K @ 3.60GHzCoffee Lake 14nm Technology.
- RAM 16.0GB Dual-Channel Unknown @ 3000MHz.
- Motherboard: Micro-Star International Z370 KRAIT GAMING.
- Graphics: 6 GB GDDR5 NVIDIA GeForce GTX 1660 (MSI) .
- Storage: 476GB ADATA SX6000PNP (SSD M.2).

IV.3. Factory IO:

IV.3.1. Electric switchboard:

A switchboard is a component of an electrical distribution system which divides an electrical power feed into branch circuits while providing a protective circuit breaker or fuse for each circuit in a common enclosure. [D]

The switchboard used has: Emergency Stop, Push Buttons (Start, Stop, Reset), Light Indicators (blinking emergency light and button activation lights), Selector (Manual/Auto) and Three integer Displays for counting.

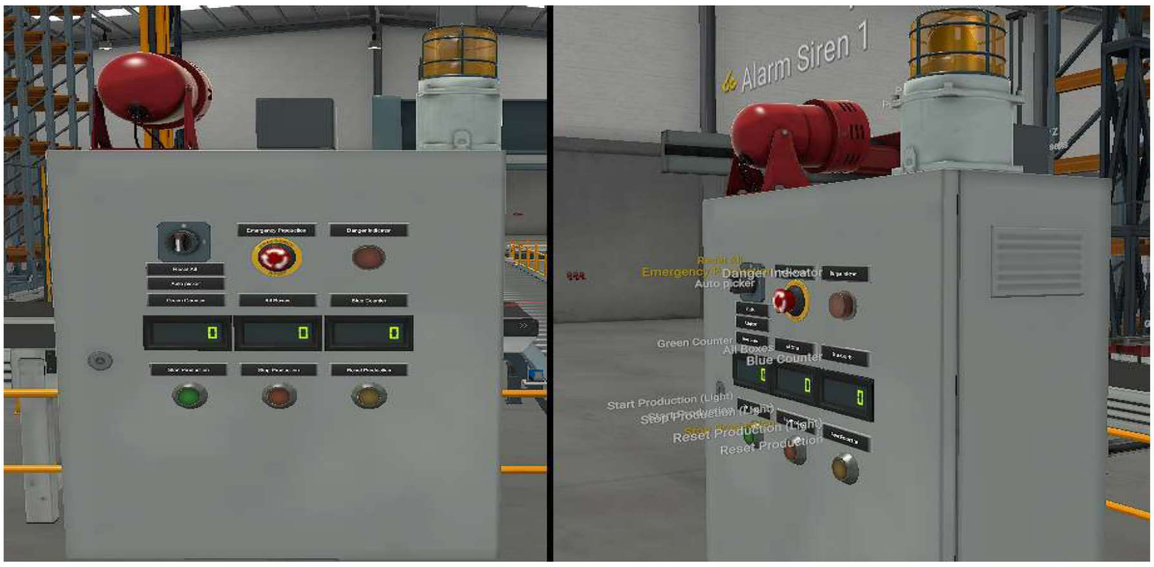


Figure III. 7: Electrical Switchboard.

IV.3.2. Sensors:

IV.3.2.1. Diffuse sensor:

Diffuse photoelectric sensor, which can detect any solid object.

- LED: red (detecting).
- Detectable materials: solids.
- Sensing range: [0 - 1.6 m].[D]

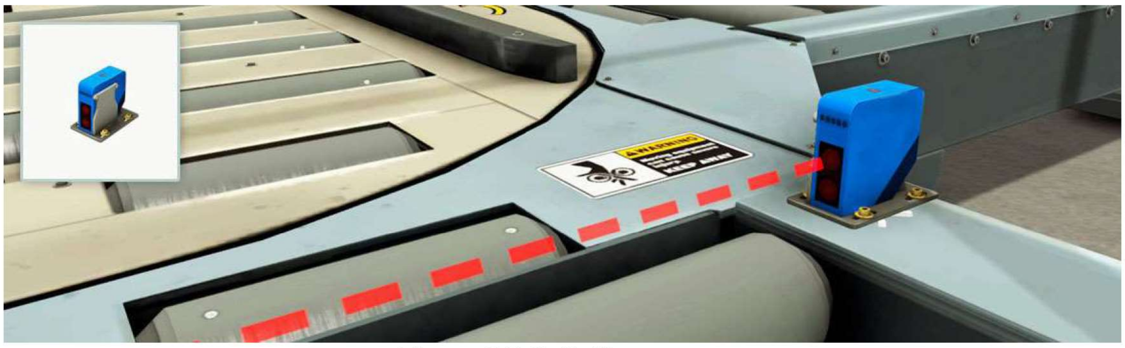


Figure III. 8: *Diffuse sensor.*

IV.3.2.2. Capacitive sensor :

Proximity sensor used for close detection of any material. It is equipped with an LED, which indicates the presence of an object within its range. The output value can be digital or analog according to the selected configuration.

- LED: green (detecting).
- Detectable materials: solids and liquids.
- Sensing range: 0 - 0.2 m. [D]

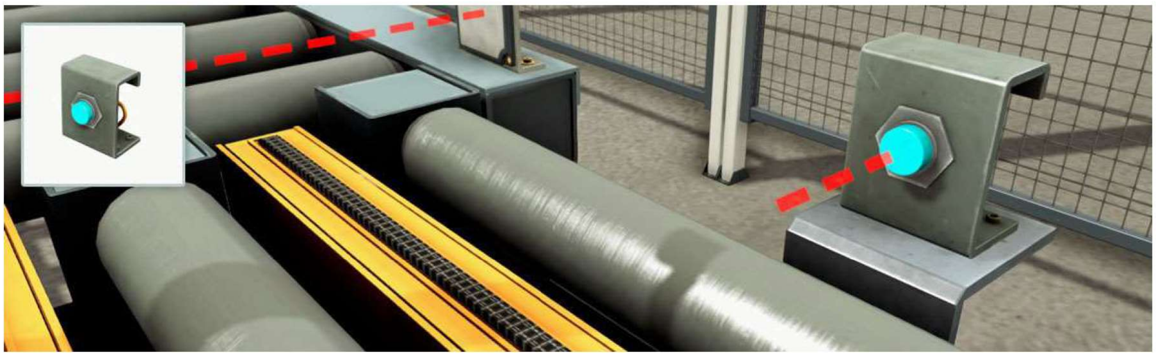


Figure III. 9: *Capacitive sensor.*

IV.3.2.3. Retro reflective Sensor and Reflector:

Unlike the other sensors, the retro reflective sensor requires a reflector. To work properly, it must be aligned with the reflector. It is equipped with two LED, which indicate the correct alignment (green), and detection status (yellow).

- Green LED: aligned with reflector.
- Yellow LED: light beam not interrupted.
- Detectable materials: solids.
- Sensing range: 0 - 6 m.[D]

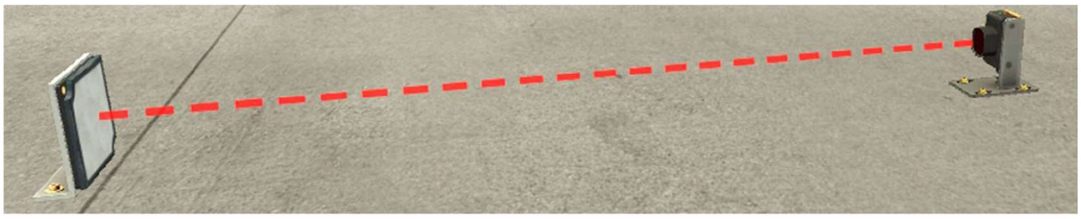


Figure III. 10: *Retro reflective sensor.*

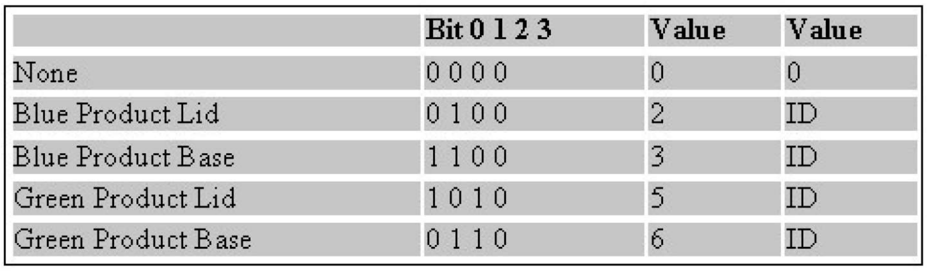
<u>IV.3.2.4. Vision Sensor**:**</u>

The Vision Sensor recognizes Raw Materials, Product Lids and Product Bases,

and their respective colors.

- LED: red (detecting)
- Detectable materials: Raw Materials, Product Bases, Product Lids
- Sensing range: 0.3 - 2 m.[D]

In this simulation, we use all digital type.



	Bit 0 1 2 3	Value	Value
None	0 0 0 0	0	0
Blue Product Lid	0 1 0 0	2	ID
Blue Product Base	1 1 0 0	3	ID
Green Product Lid	1 0 1 0	5	ID
Green Product Base	0 1 1 0	6	ID

Table 4: *Vision sensor values.*

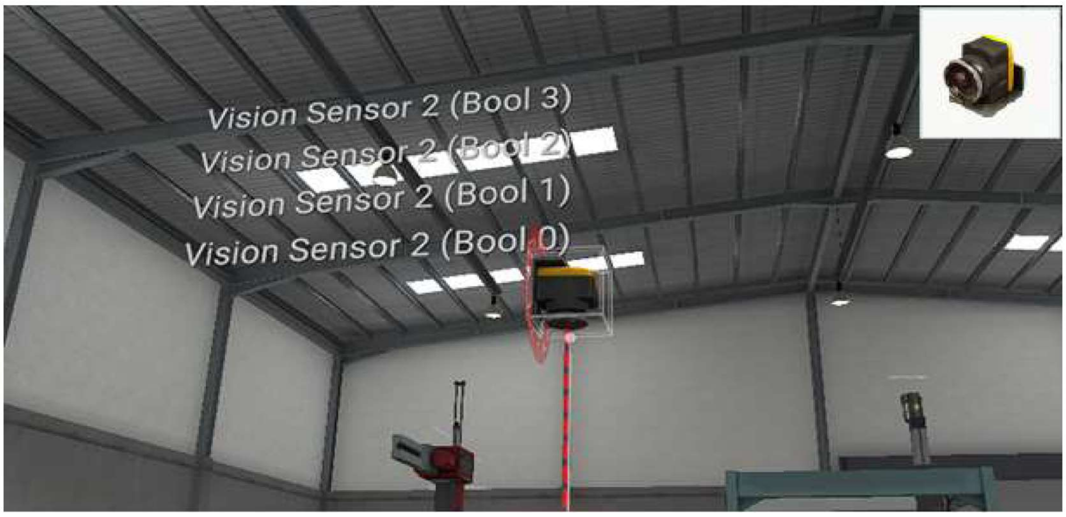


Figure III. 11: *Vision sensor.*

IV.3.3. Actuators

IV.3.3.1.Light Load Parts:

<u>IV.3.3.1.Belt Conveyors:</u>

Belt Conveyors are used to transport light load cargo.

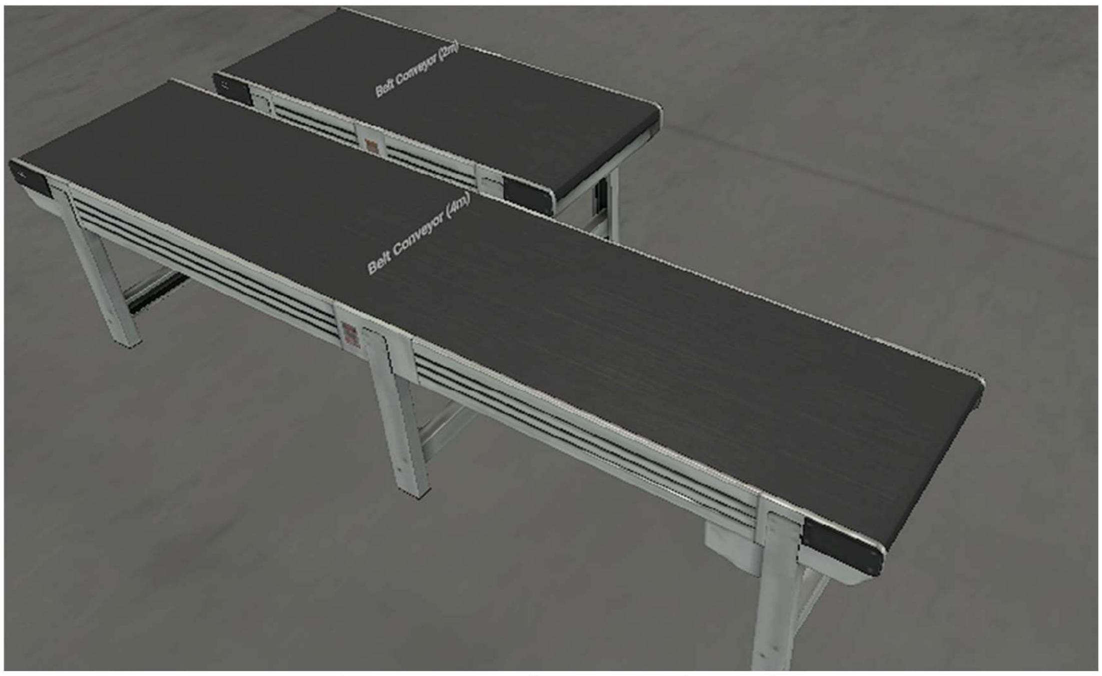


Figure III. 12: *Belt conveyors (2m, 4m).*

IV.3.3.2. Positioning Bars:

A machine that clamps both Lid or Base into place to allow the picker ark to move the Lid on top of the Base.

Vertical stroke: 0.373 m, Clamper stroke: 0.48 m.[D]

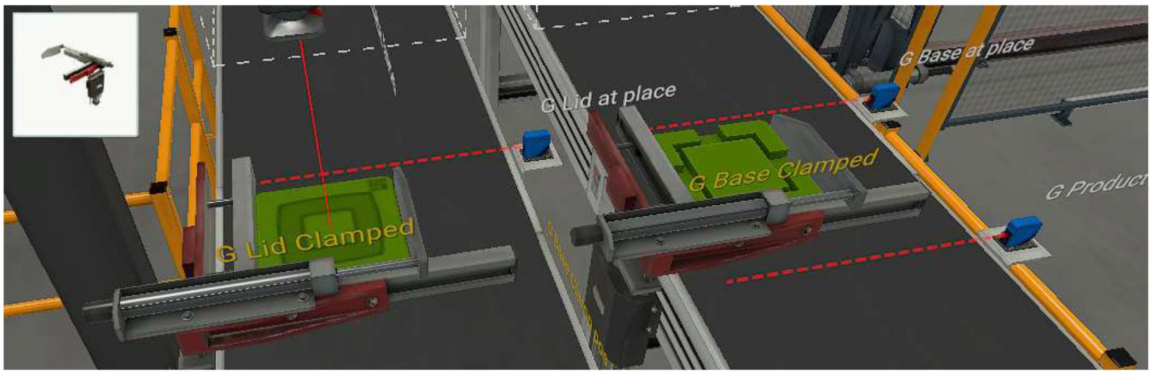
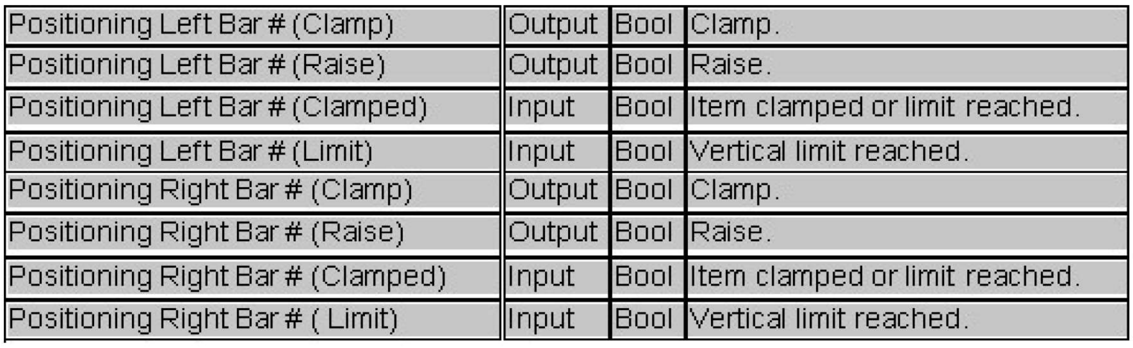


Figure III. 13: *Positioning bars*



Positioning Left Bar # (Clamp)	Output	Bool	Clamp.
Positioning Left Bar # (Raise)	Output	Bool	Raise.
Positioning Left Bar # (Clamped)	Input	Bool	Item clamped or limit reached.
Positioning Left Bar # (Limit)	Input	Bool	Vertical limit reached.
Positioning Right Bar # (Clamp)	Output	Bool	Clamp.
Positioning Right Bar # (Raise)	Output	Bool	Raise.
Positioning Right Bar # (Clamped)	Input	Bool	Item clamped or limit reached.
Positioning Right Bar # (Limit)	Input	Bool	Vertical limit reached.

Table 5: *Positioning Bar I/O*

IV.3.3.4. Heavy Load Parts:

IV.3.3.4. Roller Conveyor:

Heavy-duty roller conveyor, can be controlled by digital and analog values according to the selected configuration.

- Roll Radius: 0.046 m.
- Maximum conveying speed: 0.45 m/s (digital); 0.8 m/s (analog).[D]

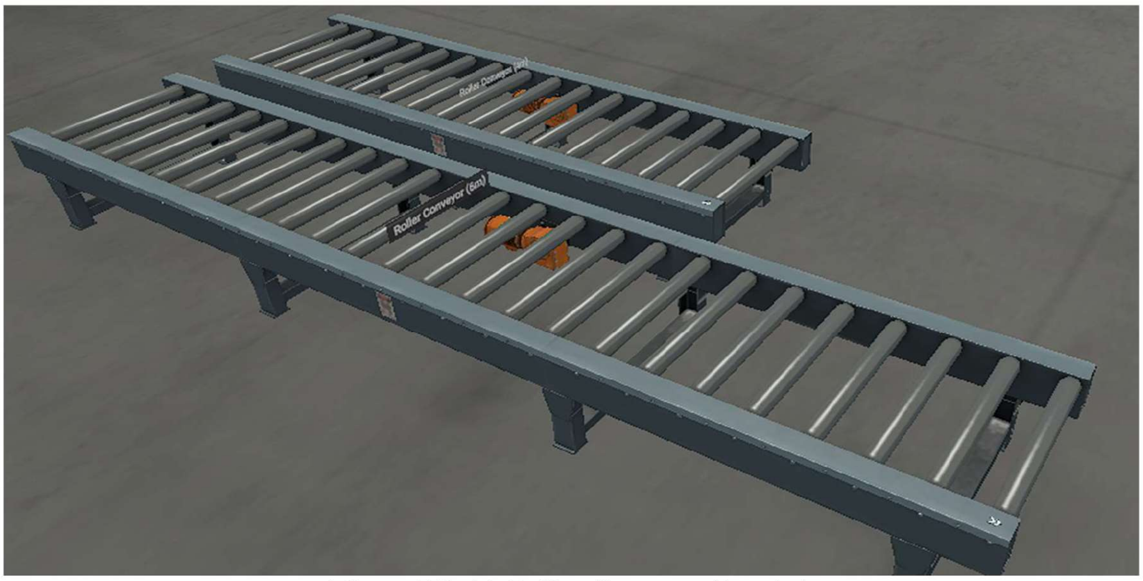


Figure III. 14: *Roller Conveyor (4m, 6m).*

IV.3.3.3. Loading conveyor:

Heavy duty conveyor, mostly used to load/unload cargo onto a Stacker Crane. Can be controlled by digital or analog values.

- Roll radius: 0.046 m.
- Maximum conveying speed: 0.45 m/s (digital); 0.8 m/s (analog).[D]

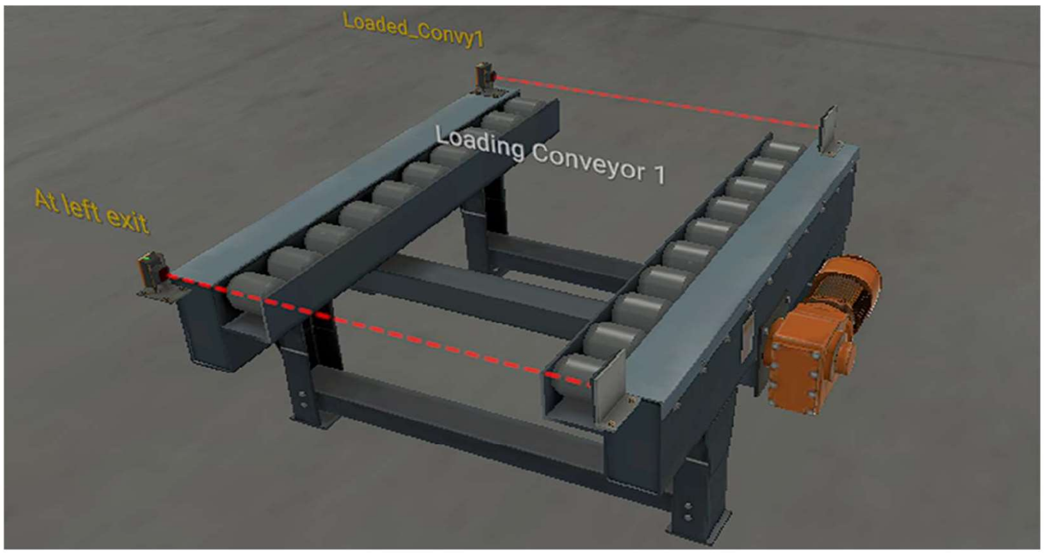


Figure III. 15: *Loading Conveyor.*

IV.3.3.4. Chain Transfer:

Used to transfer cargo onto adjacent conveyors, works best with square pellets. It is made of loading rolls and three chain tracks.

- Chain track stroke: 0.04 m.
- Maximum conveying speed: 0.45 m/s.
- Maximum chain speed: 0.45 m/s.[D]

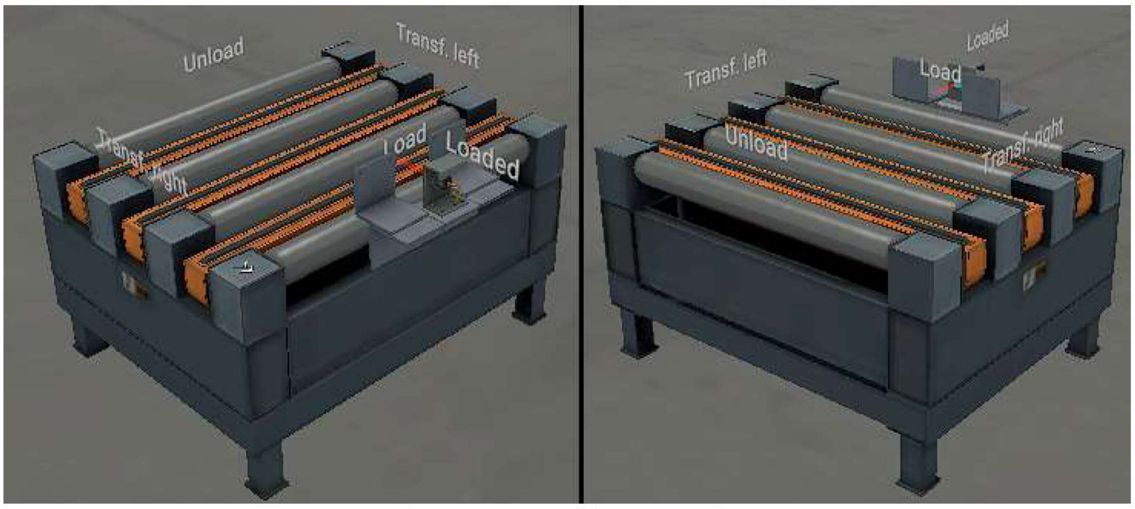


Figure III. 16: *Chain Transfer.*

Chain Transfer # (+)	Output	Bool	Roll (arrow direction).
Chain Transfer # (-)	Output	Bool	Roll.
Chain Transfer # (Left)	Output	Bool	Raise platform and rotate chain left.
Chain Transfer # (Right)	Output	Bool	Raise platform and rotate chain right.

Table 6: *Chain Transfer I/O.*

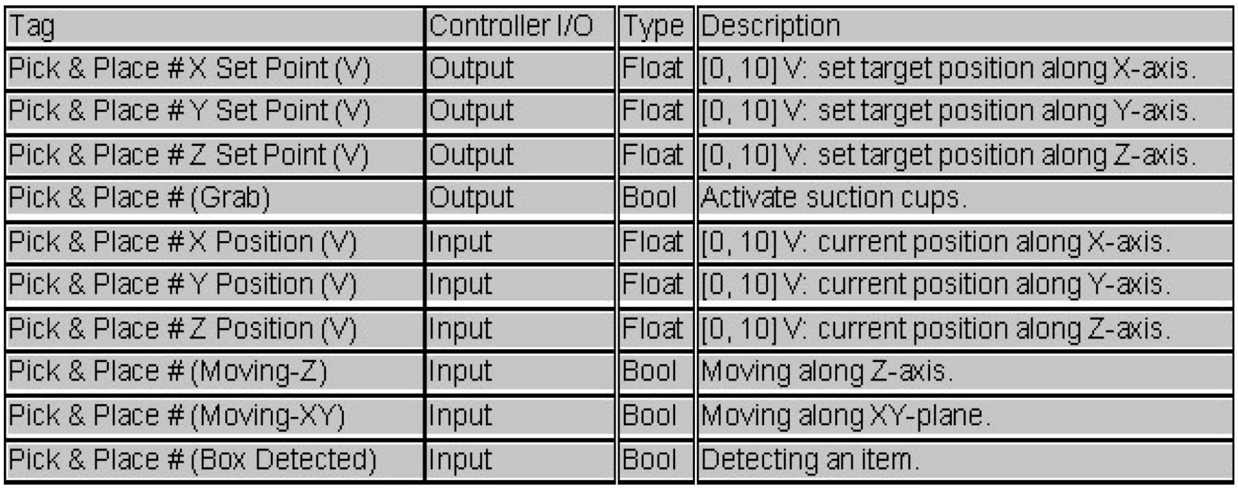
IV.3.4. Stations:

IV.3.4.1. Pick & Place:

Gantry Pick and Place station with three axes controlled by servomotors. Often used to move light load cargo (e.g. cardboard boxes) into other conveyors or pallets.

The Pick and Place has four degrees of freedom, three correspond to the axes linear movement and another to gripper rotation. The gripper is enabled by suction cups and includes a proximity sensor. Can be controlled by digital and analog values, according to the selected configuration. When controlled with digital I/O, axis movement is performed incrementally (step by step) on each rising edge of the controlling tag value.

- Y-axis stroke: 1.25 m.
- X-axis stroke: 2.125 m.
- Z-axis stroke: 0.5m.
- Step: 0.125 m.
- Beam speed: 1.5 m/s.
- Gripper angular speed: 4.6 rad/s. [D]



Tag	Controller I/O	Type	Description
Pick & Place # X Set Point (V)	Output	Float	[0, 10] V: set target position along X-axis.
Pick & Place # Y Set Point (V)	Output	Float	[0, 10] V: set target position along Y-axis.
Pick & Place # Z Set Point (V)	Output	Float	[0, 10] V: set target position along Z-axis.
Pick & Place # (Grab)	Output	Bool	Activate suction cups.
Pick & Place # X Position (V)	Input	Float	[0, 10] V: current position along X-axis.
Pick & Place # Y Position (V)	Input	Float	[0, 10] V: current position along Y-axis.
Pick & Place # Z Position (V)	Input	Float	[0, 10] V: current position along Z-axis.
Pick & Place # (Moving-Z)	Input	Bool	Moving along Z-axis.
Pick & Place # (Moving-XY)	Input	Bool	Moving along XY-plane.
Pick & Place # (Box Detected)	Input	Bool	Detecting an item.

***Table** 7: Pick & Place I/O*

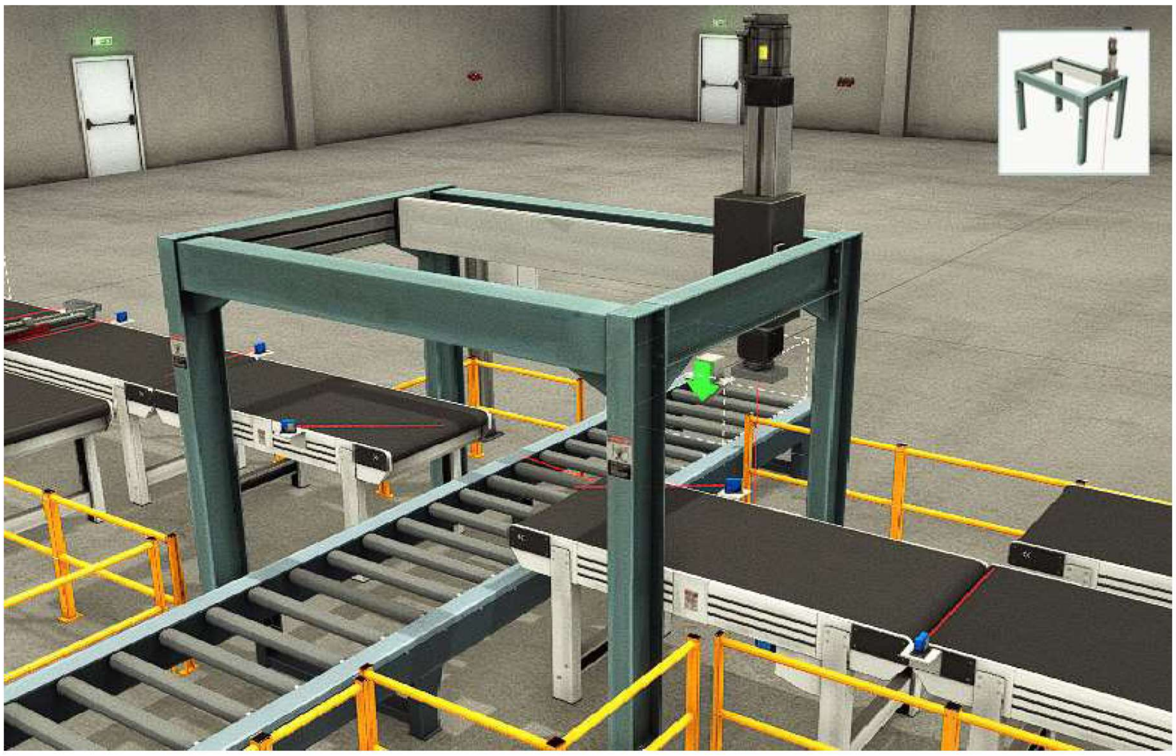


Figure III. 17: *Pick & Place.*

IV.3.4.2. Stacker Crane and Rack:

Rail mounted stacker crane used to stock heavy cargo. Includes a cart, a vertical platform, and two forks that may slide to both sides.

Two laser rangefinders, placed on the cart and the platform, measure the horizontal and vertical position of the platform. Racks are upright steel frames connected by horizontal steel beams with the purpose to store loads. The available rack is single-deep rack type, also known as selective rack, which only allows loads to be stored one pallet deep. Loads can be stored from both sides of the rack.

Each rack must be aligned with one of the rail ends, making the stacker crane stop at the correct position. The stacker crane can be controlled by Digital, Numerical and Analog values, according to the selected configuration.

- Forks stroke: 1.2 m.
- Cart stroke: 10.5 m.
- Platform stroke: 6.625 m.
- Cart speed: 1.4 m/s.
- Forks speed: 0.5 m/s.
- Platform speed: 1.7 m/s.
- Number of cells: 18. [D]

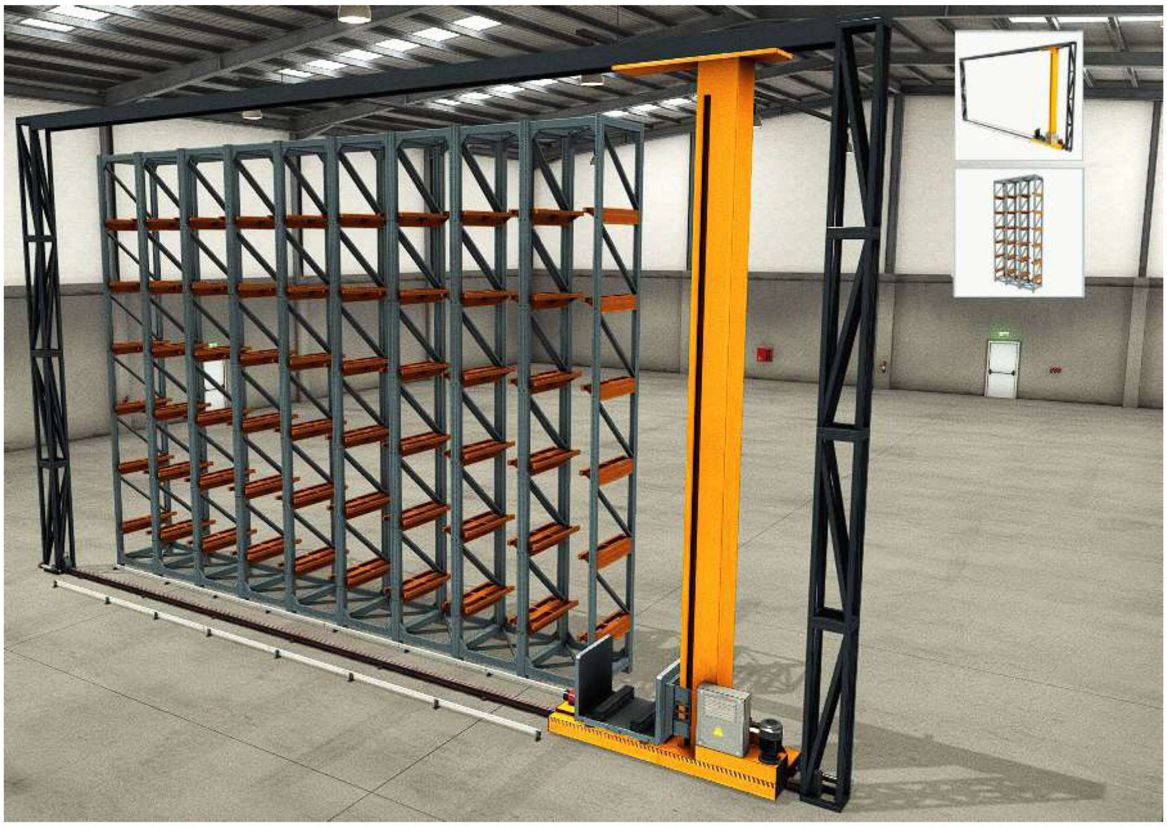


Figure III. 18: *Stacker crane.*

The stacker type we use is numerical with a sensor in every rack; the target cell can be defined by an integer value between 1 and 54. If this value is set to zero, the stacker crane stops at the current position. However, if it is higher than 54, it will move to the rest position (55).[D]

Tag	Controller I/O	Type	Description
Stacker Crane #Target Position	Output	Integer	Move to the desired cell.
Stacker Crane #(Left)	Output	Bool	Move forks left.
Stacker Crane #(Right)	Output	Bool	Move forks right.
Stacker Crane #Lift	Output	Bool	Slightly lift the platform.
Stacker Crane #Moving-X	Input	Bool	Moving along X-axis.
Stacker Crane #Moving-Z	Input	Bool	Moving along Z-axis.
Stacker Crane #Left Limit	Input	Bool	Forks at left limit.
Stacker Crane #Middle Limit	Input	Bool	Forks at middle.
Stacker Crane #Right Limit	Input	Bool	Forks at right limit.

Table 8: *Stacker crane I/O.*

IV.3.4.3. Two-Axis Pick & Place:

This part can be used to assemble Lids on Bases or pick and place items from one place to another. To guarantee a correct fit, Positioning Bars should properly align bases and lids.

- X-axis stroke: 1.125 m.
- Z-axis stroke: 0.625 m.
- Arm and picker speed: 2 m/s.[D]

In this example, we use the Digital configuration for Two-Axis Pick & Place.

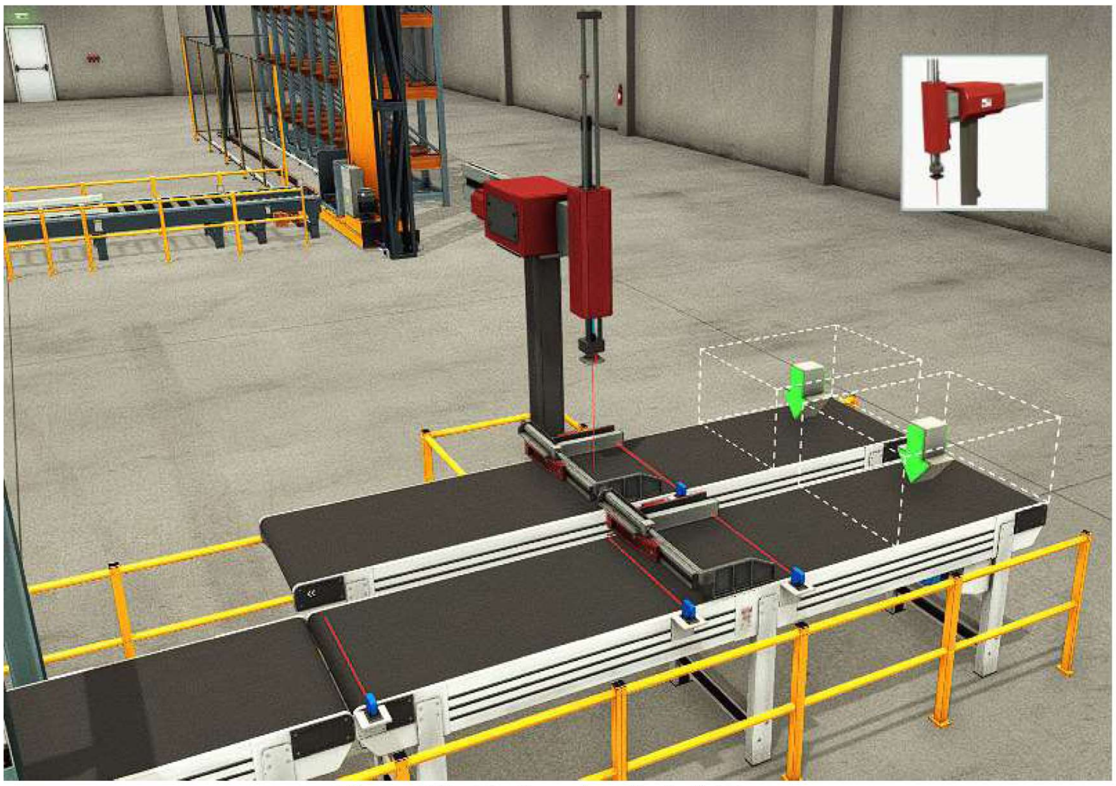


Figure III. 19: *Two-Axis Pick & Place.*

Tag	Controller I/O	Type	Description
Two-Axis Pick & Place # Z	Output	Bool	Move along Z-axis.
Two-Axis Pick & Place # X	Output	Bool	Move along X-axis.
Two-Axis Pick & Place # (Grab)	Output	Bool	Activate suction cup.
Two-Axis Pick & Place # (Moving X)	Input	Bool	Moving along X-axis.
Two-Axis Pick & Place # (Moving Z)	Input	Bool	Moving along Z-axis.
Two-Axis Pick & Place # (Detected)	Input	Bool	Detecting an item.

Table 9: *Two-Axis Pick & Place I/O.*

IV.3.5.Safety material:

IV.3.5.1. Handrails:

Metal tube structures used to provide safety to operators on a shop floor.

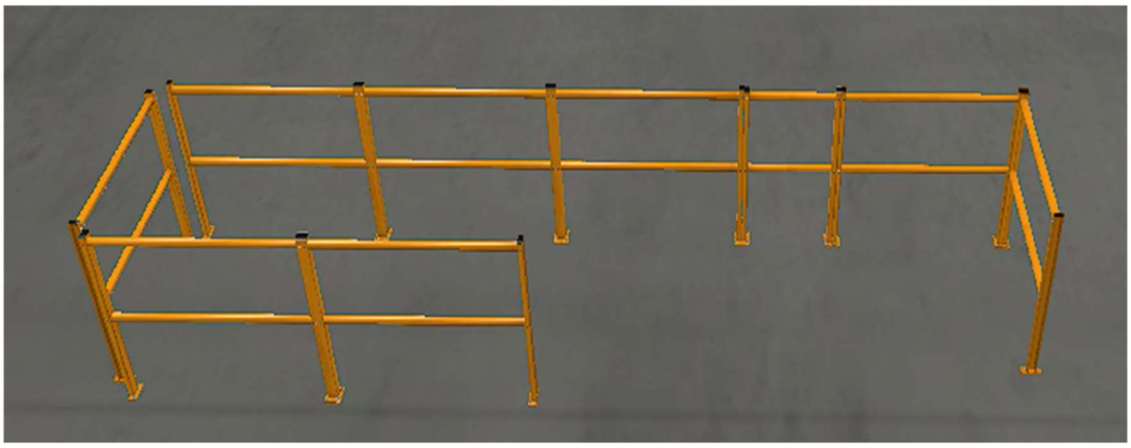


Figure III. 20: *Handrails.*

IV.3.5.2. Safeguard:

Grid metal structures commonly used to define safe perimeters around machines or working areas. [D]

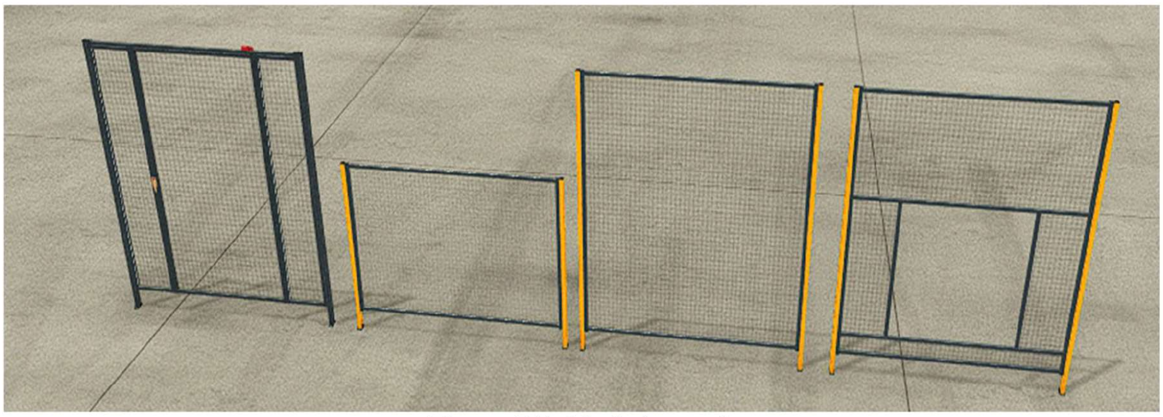


Figure III. 21: *Safeguard.*

IV.4. Conclusion:

In this chapter, we covered the basic items that we use in chapter IV, separately in brief descriptions.

In the next chapter, we can jump straight into setting up the machine to communicate together and function as one.

Chapter IV: Simulation and programming

This finale chapter, we will implement what we have learned in the last few chapter, then make the program for this simulation.

V.1. Setting up Tia portal with Factory I/O:

Starting from chapter II Tia portal steps on how to create a project in TIA Portal, and how to setup your Factory I/O scene.

We need an extra step to establish the connection between PLCSIM (S7-1200) and Factory I/O. and it's downloading a special template from Factory I/O, and then changing to you specific desired CPU.

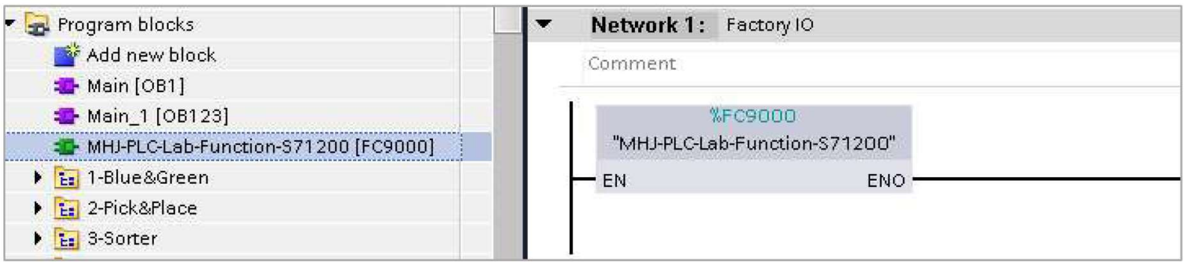


Figure IV. 1: *Factory I/O Function block.*

Also to make sure there isn't any security problems in the connection between Factory I/O and TIA Portal, we need to check the Permit Access PUT/GET.

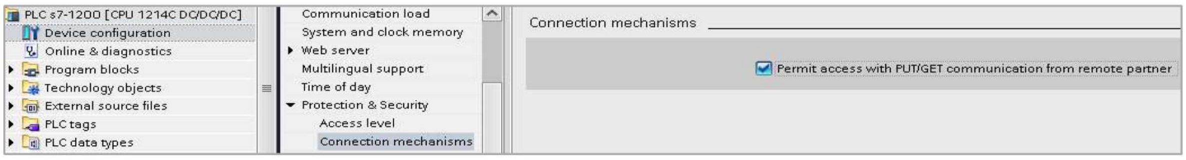


Figure IV. 2: *Permit access with PUT/GET.*

The rest of the other steps are already mentioned in Chapter II.

V.2. PLC & HMI Connection:

To add HMI in Tia portal we go to Add new Device then we Click HMI. And you we choose the desired HMI.

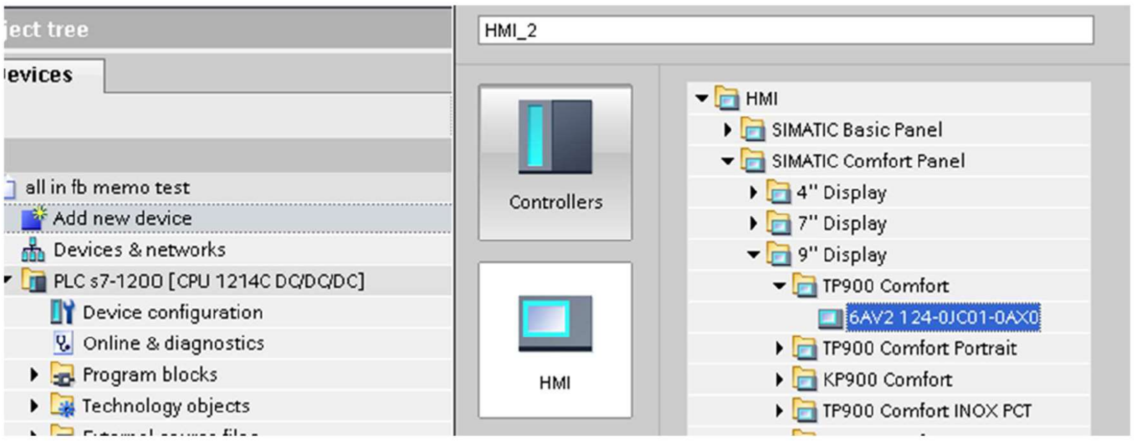


Figure IV. 3: *Add HMI.*

After that, we establish Ethernet connection in Devices & networks (network view).

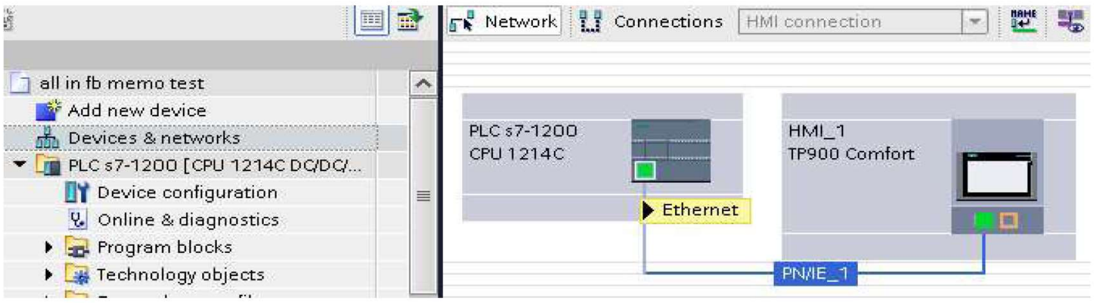
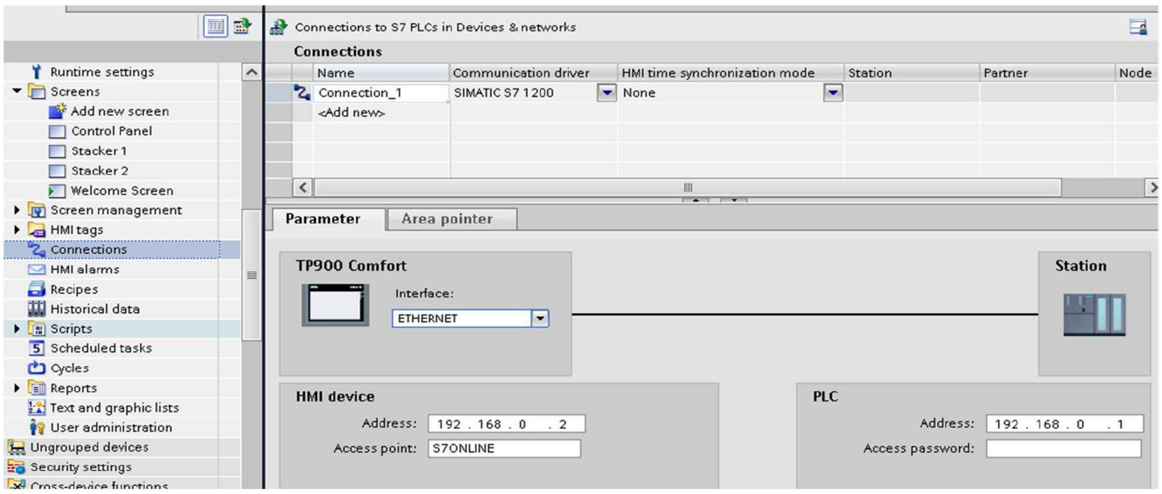


Figure IV. 4: *HMI/PLC connection.*

Finally, we go to connections and we choose add new, and we choose the communication driver (S7-1200), also make sure to choose Ethernet interface in HMI

Moreover, be careful to setup the correct IPs.



Connections

Name	Communication driver	HMI time synchronization mode	Station	Partner	Node
Connection_1	SIMATIC S7 1200	None			
<Add new>					

TP900 Comfort

Interface: ETHERNET

Station

HMI device

Address: 192 . 168 . 0 . 2

Access point: S7ONLINE

PLC

Address: 192 . 168 . 0 . 1

Access password:

Figure IV. 5: *Establishing HMI PLC connection.*

V.3. Hardware connections:

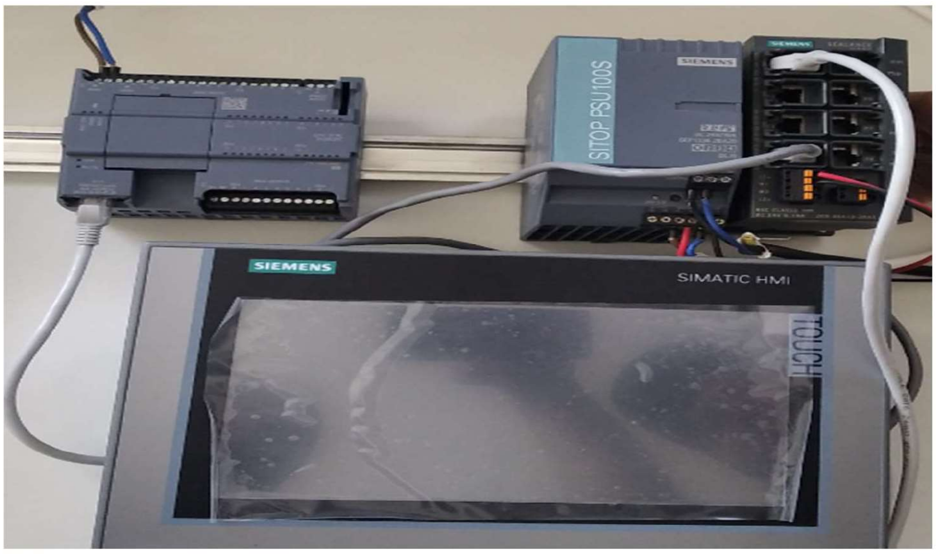


Figure III. 22: *Wiring.*

V.4. Factory I/O scene:

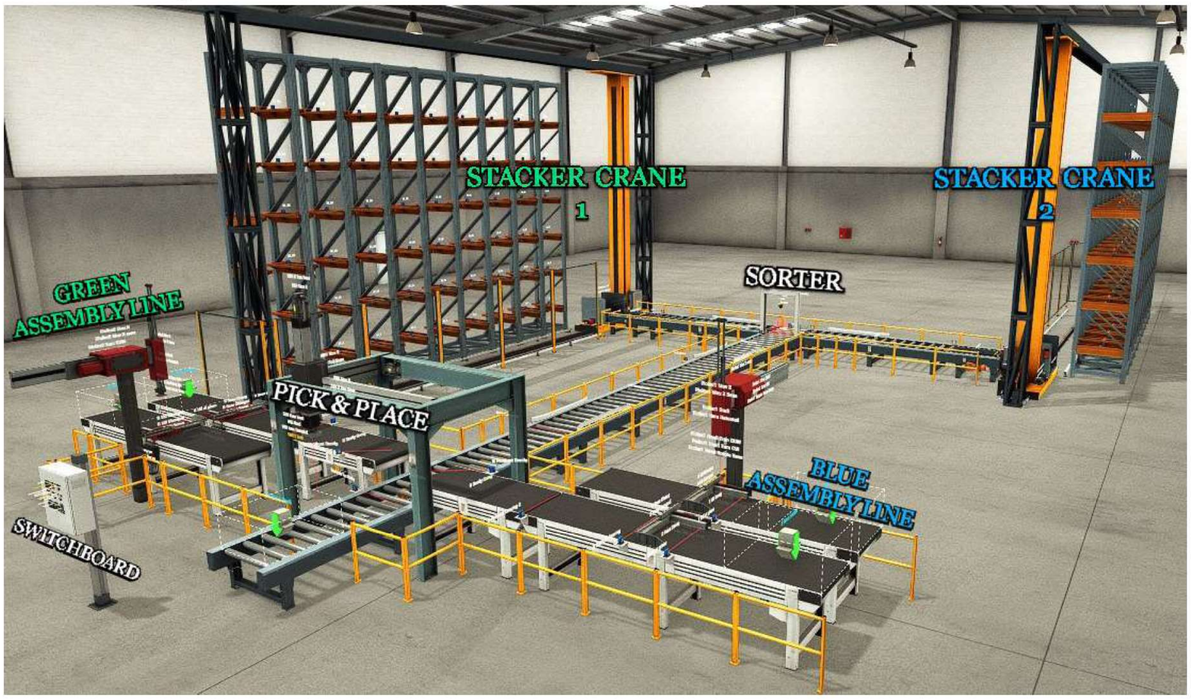


Figure IV. 6: *Factory I/O project scene.*

V.4. Project functioning explained:

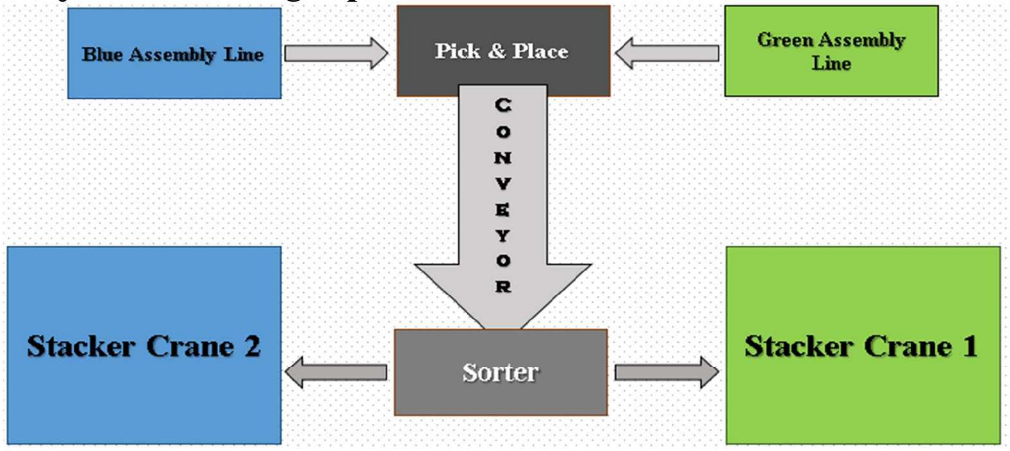


Figure IV. 7: *Global Project Aim.*

In Fig IV.8, we see our entire material that we previously described in chapter III. Both of the assembly lines start producing a Lid clamped on top of a Base (green or blue), then they move under the pick & place machine. Both production lines stop if the product under the pick & place machine is not picked. In addition, the Green product always get picked first by the machine, then the blue one after it.

Once the product is picked by the pick & place machine, it waits for a stackable box to be under it. It drops the product in the boxes so the conveyor moves forward and when another product and stackable box is ready again the process repeats it's self, alternating from Green product to Blue product.

The conveyors move the boxes until they reach the sorter, where the Lid colors are sensed, if it is a green Lid it sorted to the left, if it is blue it is sorted to the right. The sorter does not work if the stacker crane is busy.

When a box reaches to the stacker crane, the stacker detects which sensor is not on meaning which spot is empty, so the box can be lifted and stored in that empty space.

If the stacker 54 storage spots are full, then it waits for a spot to be empty. If there is no sot available then the sorter will stop sorting the colors to the stacker concerned with that color, if the sorter is stopped then the pick & place conveyor won't move, which means the stackable boxes won't be ready, which also means the production will be stopped.

While all this process is happening there is counter for both products and a counter for each one of them separately.

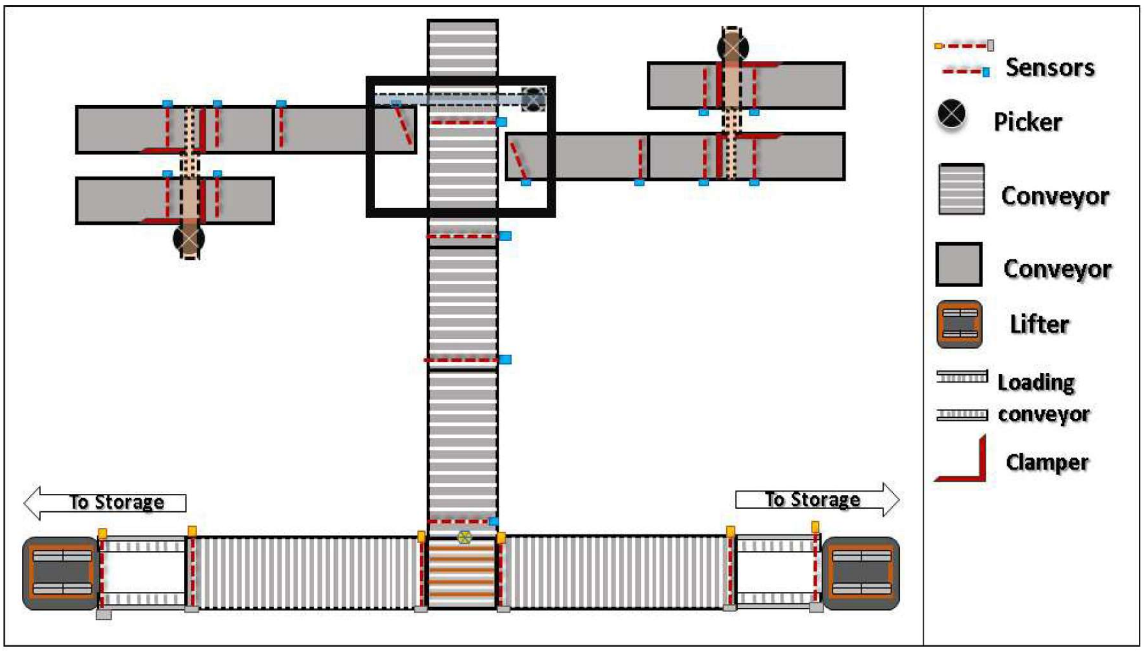


Figure IV. 9: *Project Planning.*

In addition, we have an HMI, which indicates the empty storage spots in both stackers, to keep track of our automated storage system from the control room.

If the HMI indicated RED in the spot, then the spot is full.

If it's GREEN then the spot is empty.

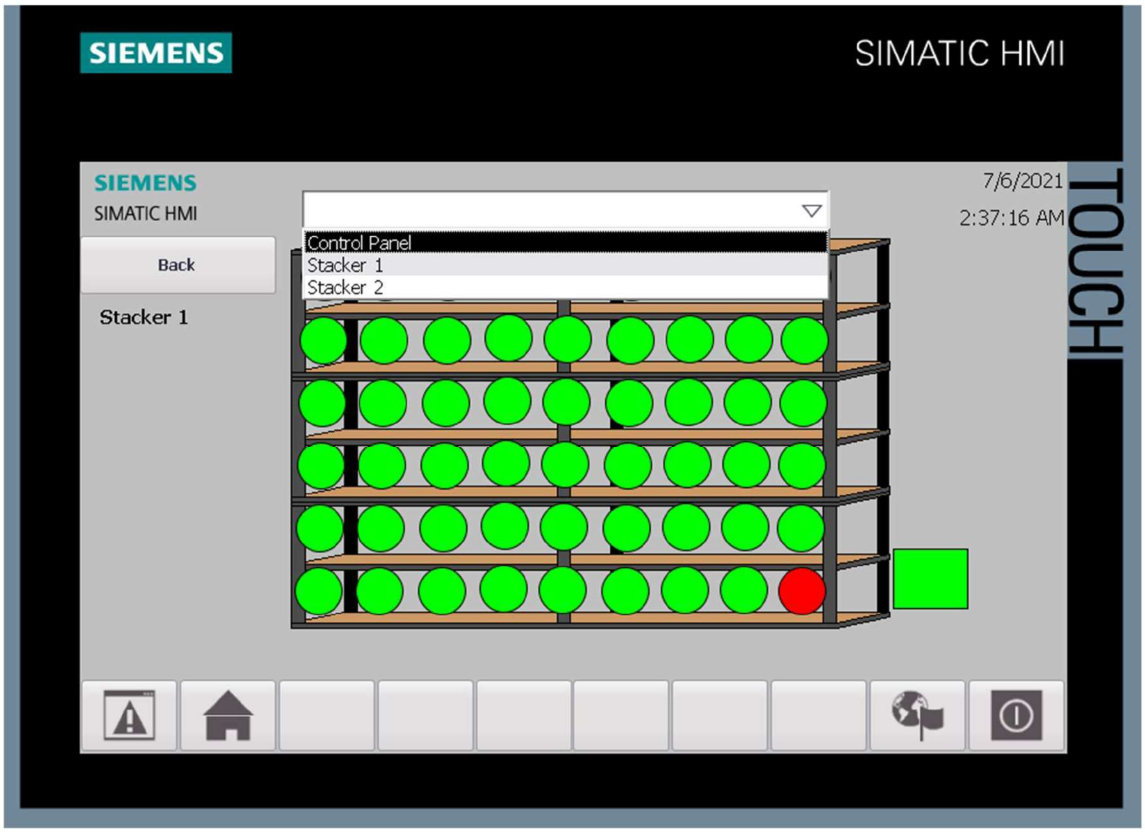


Figure IV. 10: *HMI Stacker Vue.*

V.5. Description of primary programming functions:

The program is distributed on many FC or FBs for every specific machine:

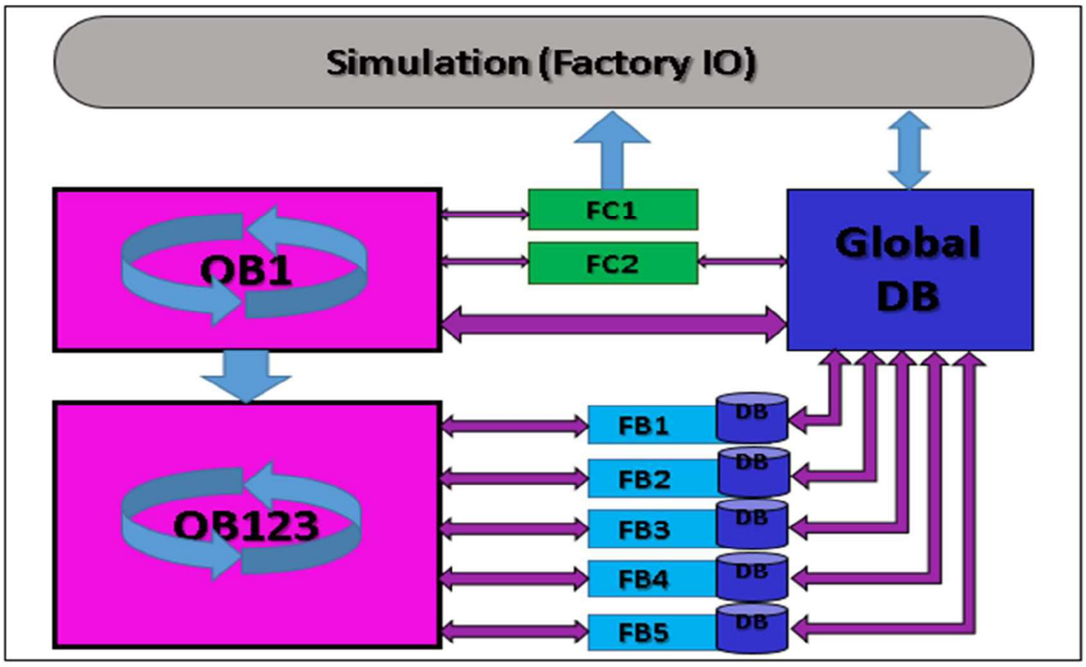


Figure IV. 11: *FC & FB program.*

V.5.1.Main OB123:

Calls both of the Green & Blue assembly lines, sorter, and Stacker crane 1 & 2.

V.5.2.Assembly Line: Both assembly line work the same way.

V.5.2.1. Start/Busy LAD Program:

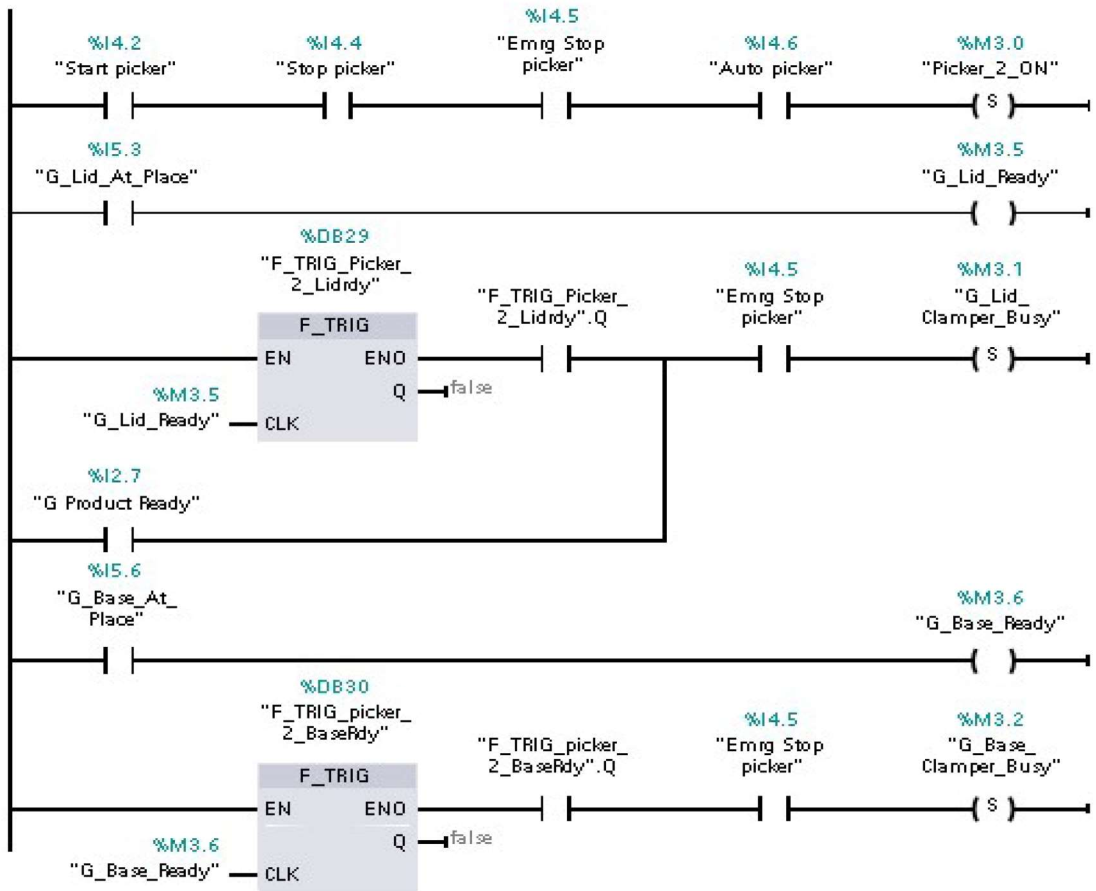


Figure IV. 12: *Assembly Line Start LAD.*

V.5.2.2. Belt Conveyors LAD Program:

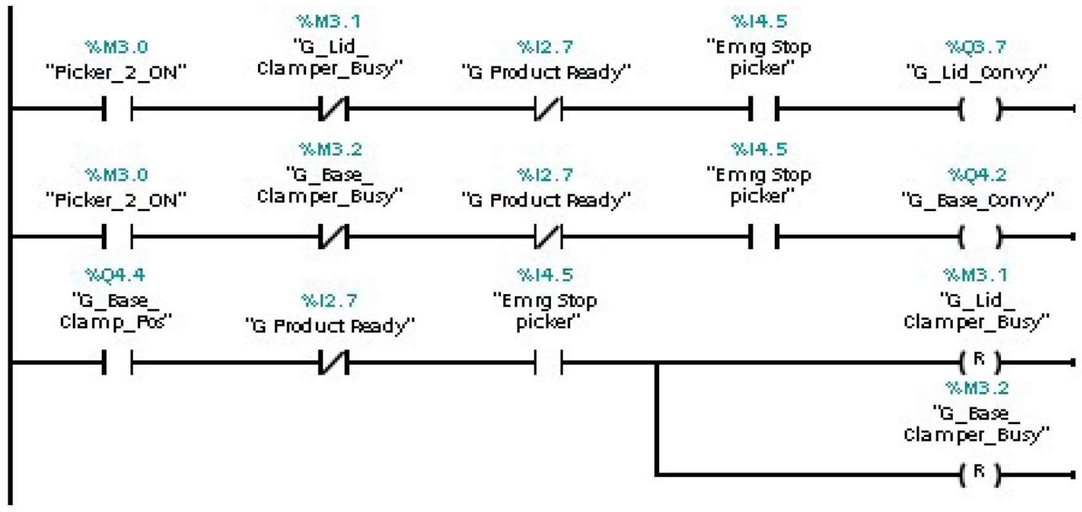


Figure IV. 13: *Assembly Line Conveyors LAD.*

V.5.2.3. Clamper LAD Program:

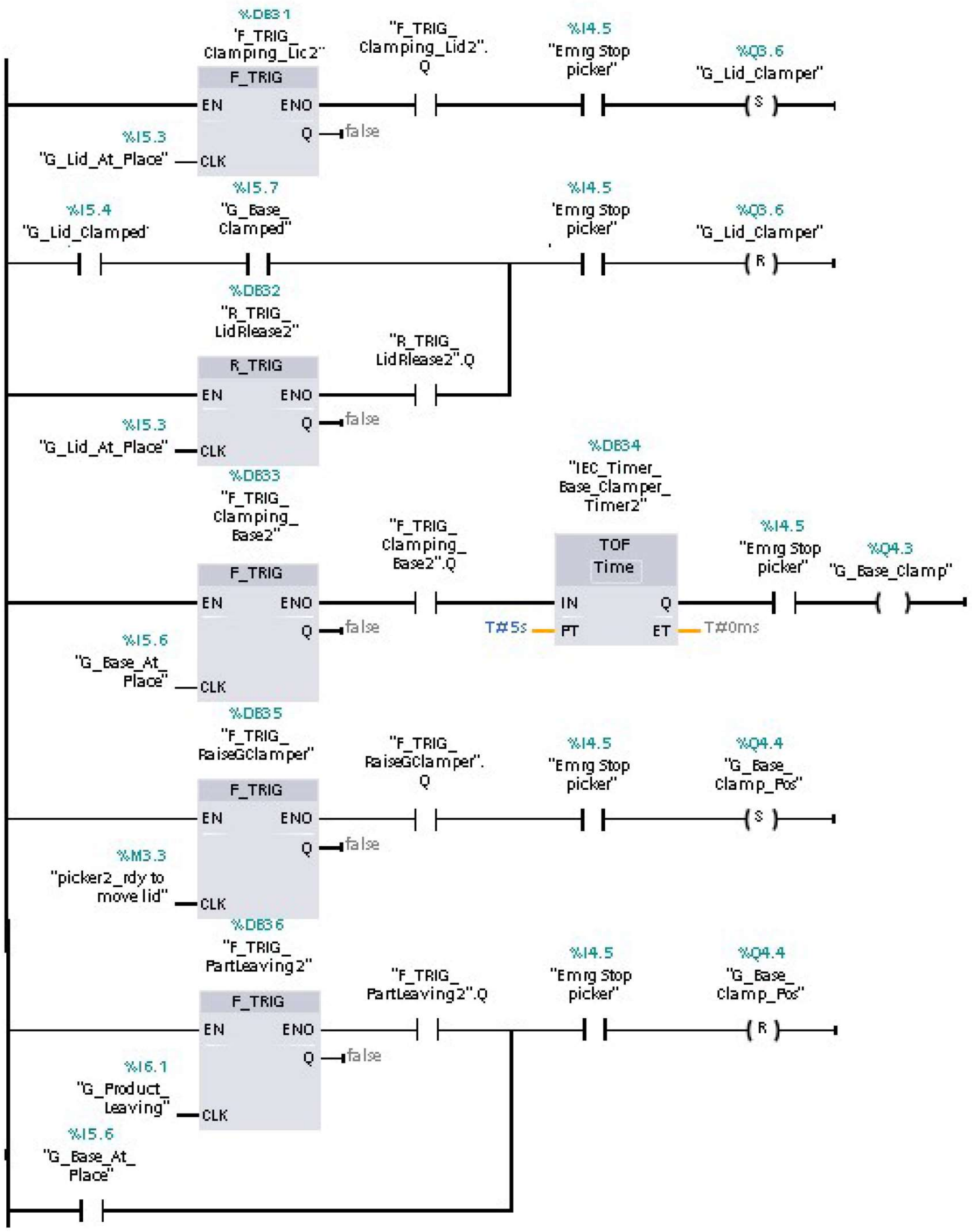


Figure IV. 14: *Assembly Line Clamper LAD.*

V.5.3. Pick & Place:

For the pick &place we use a steps each step the picker move, then we go to the next time and so on. Picking Green product is from 0 to 5 and Blue is from 5 To 8 is for the Blue product.

This first LAD program is in Pick & Place FC, and the second one is in OB1

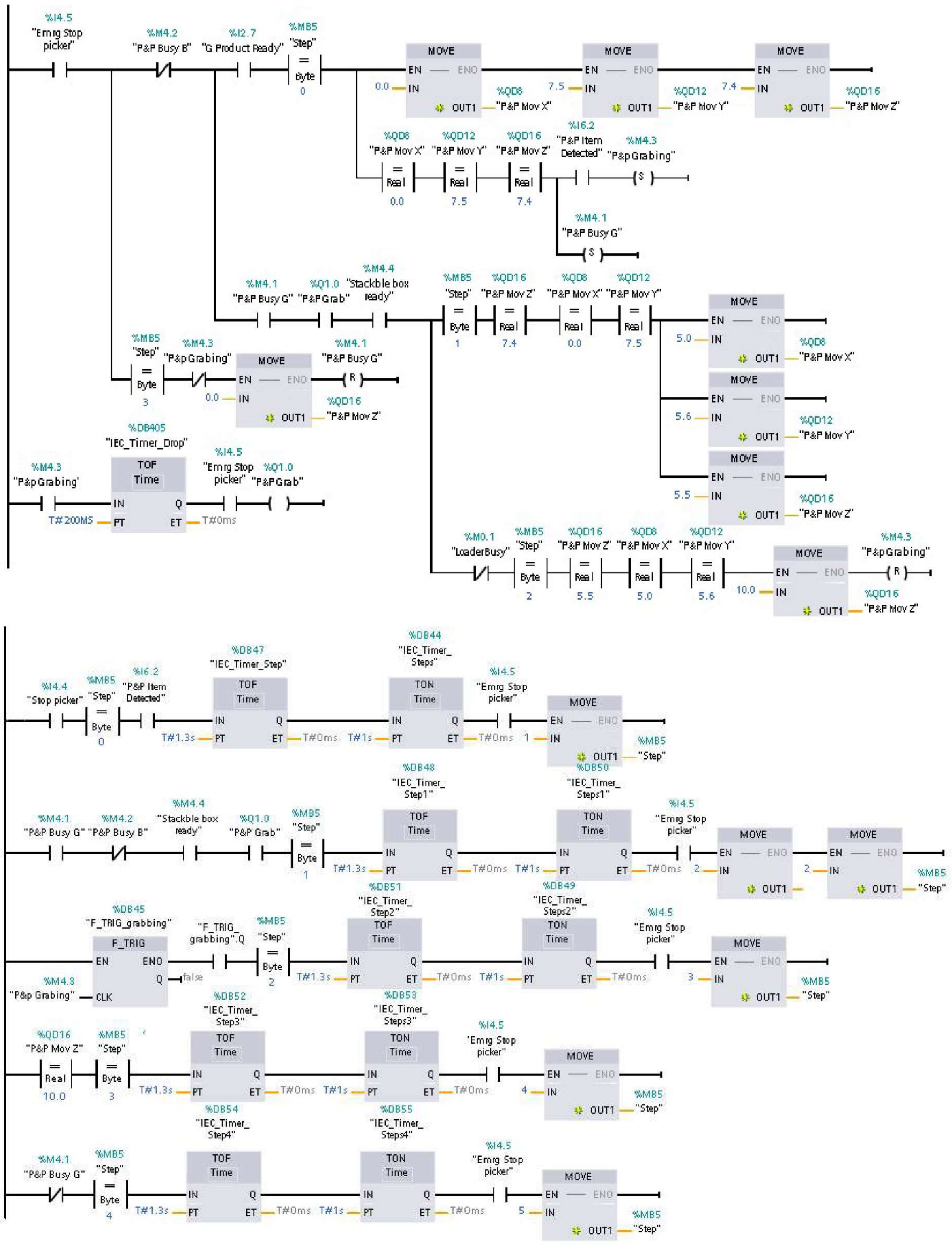


Figure IV. 15: Pick & Place Green LAD.

V.5.4. Stacker Crane:

Stacker crane start and lifts the box then wait for the available spot, if there is no available spot then its keep on waiting. This leads to the sorter to also wait which leads to the whole system waiting for available spot .Both programs are in FB Stacker Crane1, and they are called from OB123.

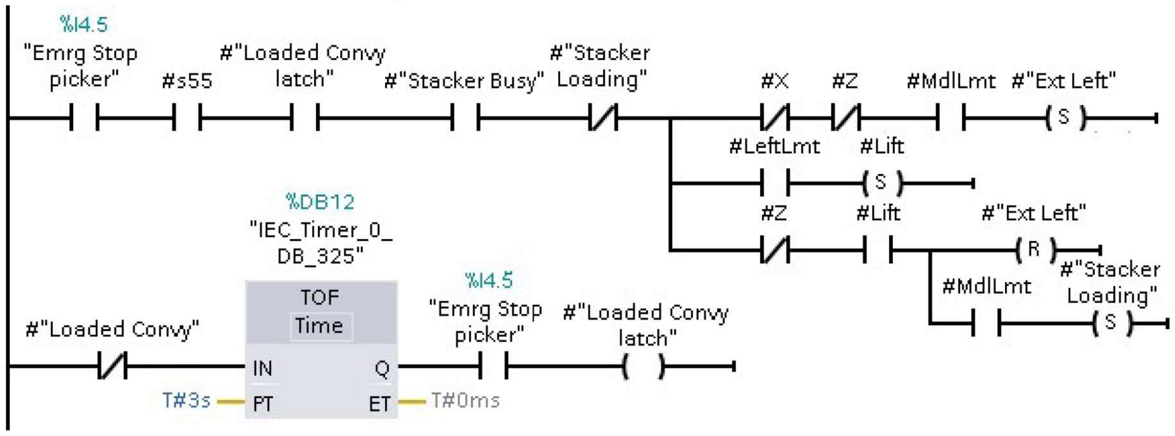


Figure IV. 16: *Stacker Crane 1 LAD.*

For the Stacker crane spot they are all the same the only thing that is different from the spots is the numbers, so it is the same as from S2 to S54 except S1 that doesn't have a spot before it.

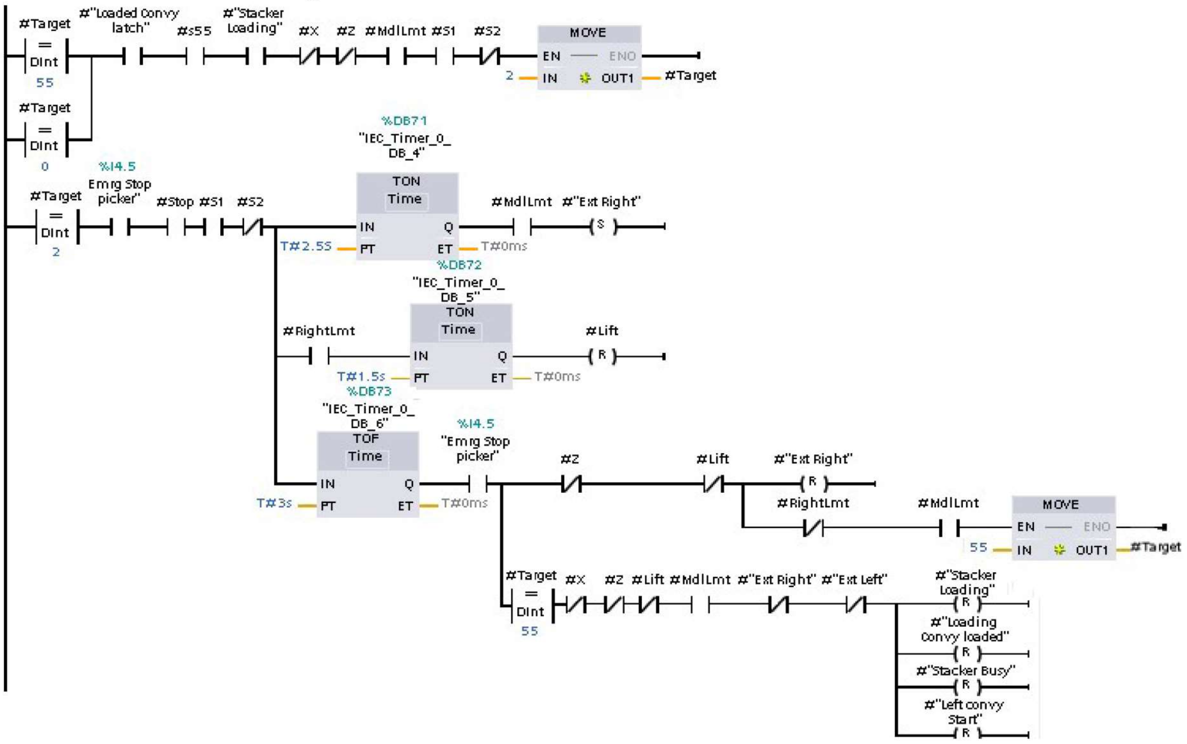


Figure IV. 17: *Stacker Crane 1 Spot2 LAD.*

Name	Path	Data Typ	Logical Addr
SCra_1_Z	Crane	Bool	%I1.7
SCra_1_L_Lim	Crane	Bool	%I2.0
SCra_1_Left	Crane	Bool	%Q1.4
SCra_1_Lift	Crane	Bool	%Q1.3
Scra_1_Loading	Crane	Bool	%M6.3
SCra_1_Mdl_Lim	Crane	Bool	%I2.2
SCra_1_R_Lim	Crane	Bool	%I2.1
SCra_1_Right	Crane	Bool	%Q1.5
SCra_1_Targ_Pos	Crane	DWord	%QD36
SCra_1_X	Crane	Bool	%I1.6

Table 10: *Stacker Crane 1 Inputs/Outputs.*

V.5.5. Sorter:

Most important Sorter LAD Program:

V.5.5.1. Start System:

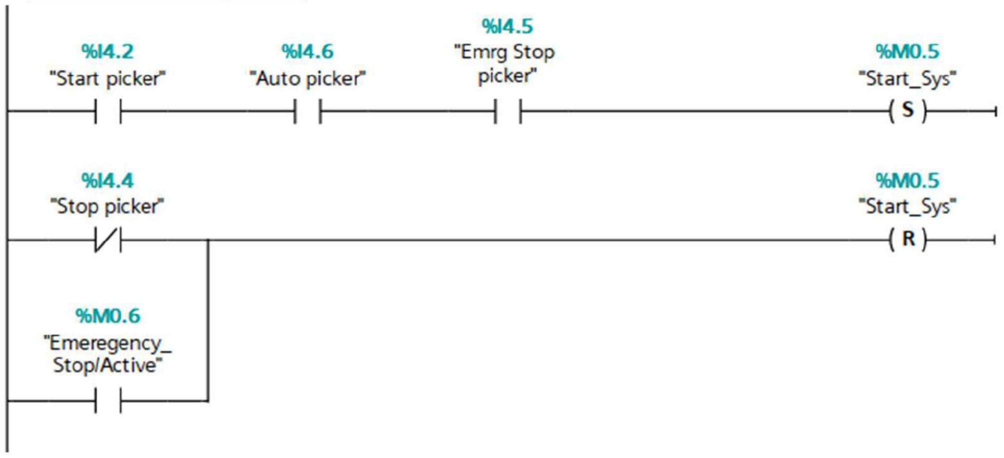


Figure IV. 18. *Sorter Start LAD.*

V.5.5.2. Color sensing:

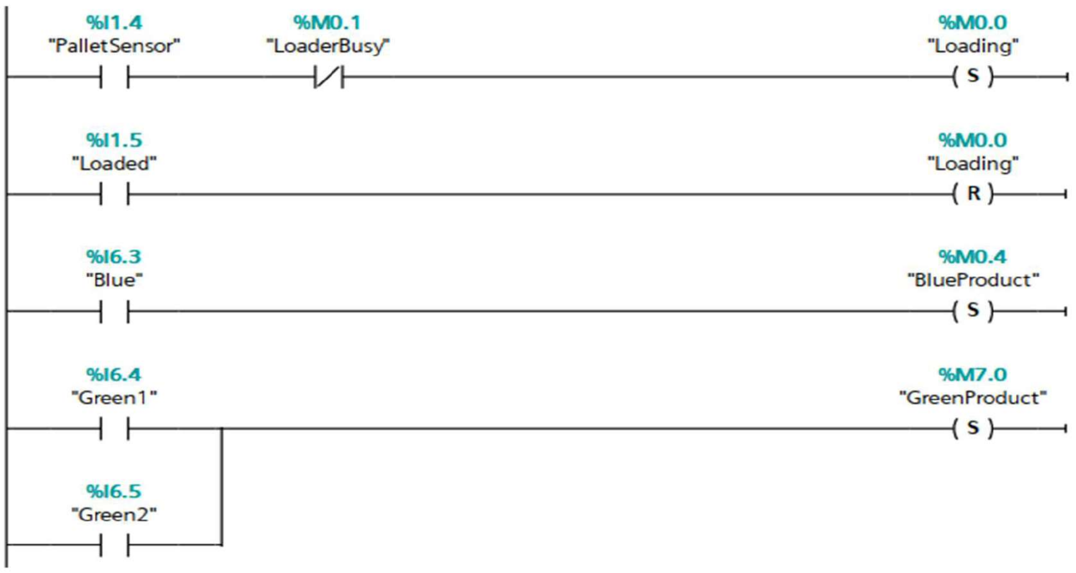


Figure IV. 19: *Color Sensing LAD.*

V.5.5.3. Sorting Left/Right:

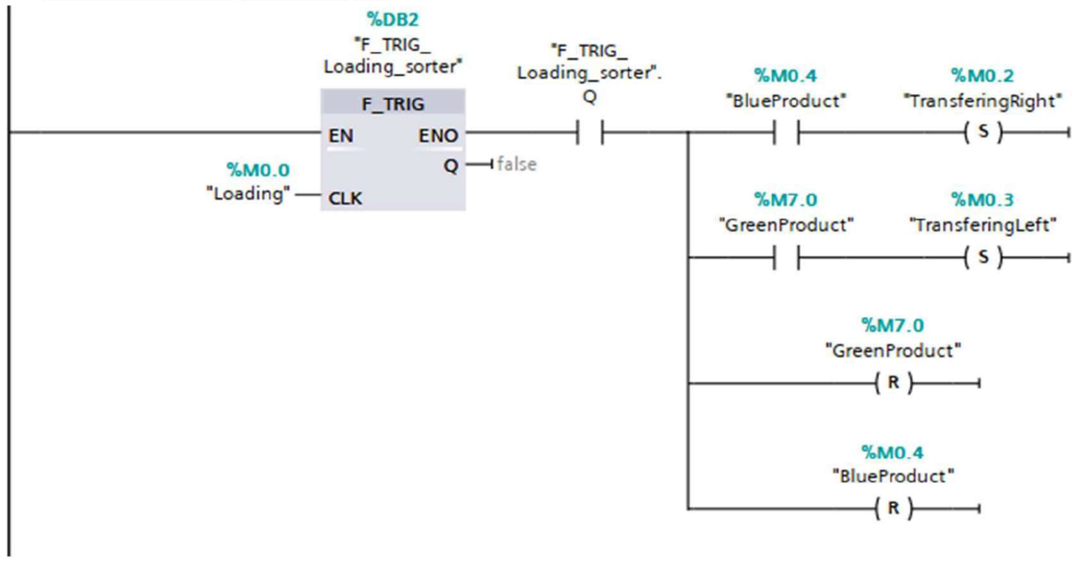


Figure IV. 20: *Sorting LAD.*

V.5.5.4. Counters:

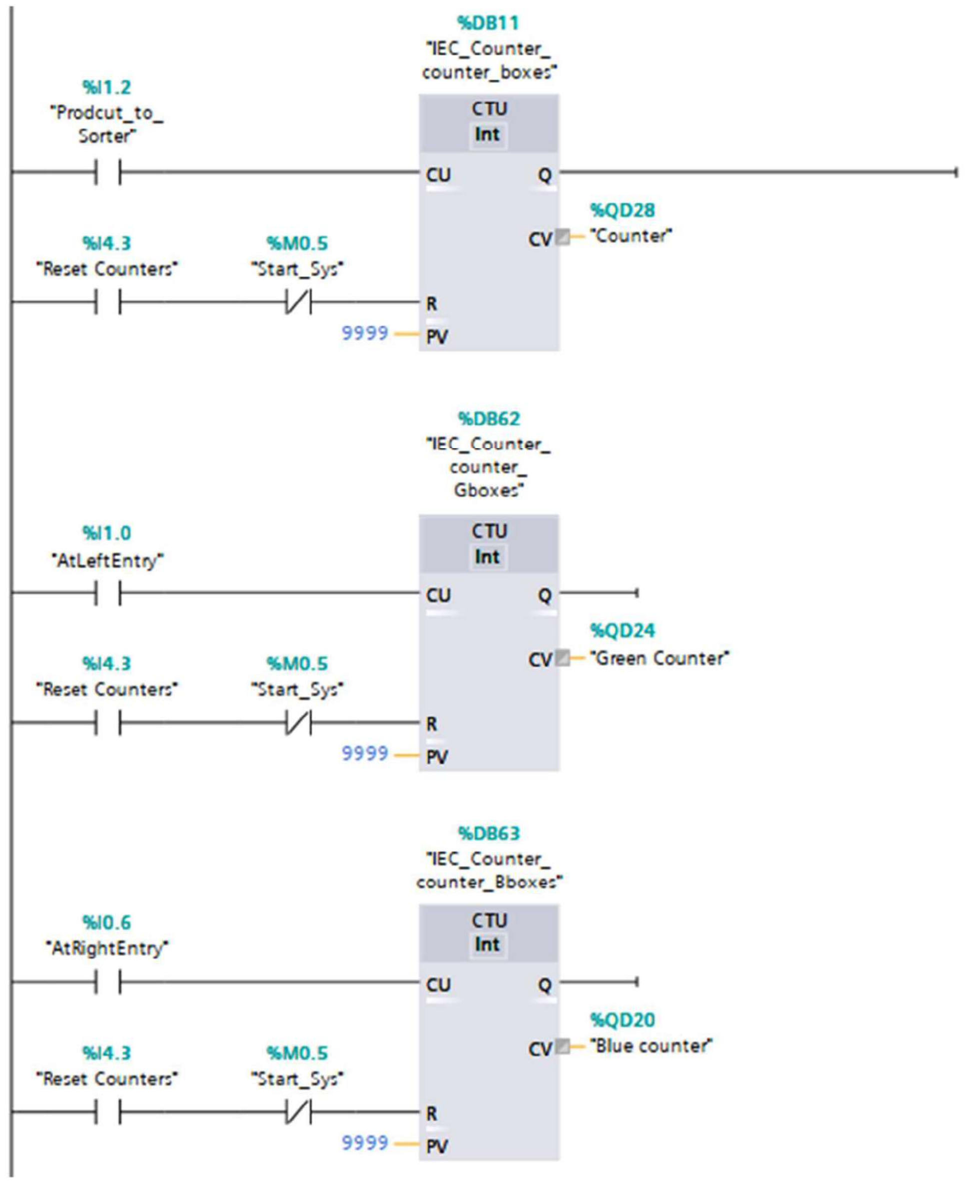


Figure IV. 21: *Counters LAD.*

IV.6.Project Statics and information:

In this project, we used 269 Different Tags:

- 162 Inputs
- 49 Outputs
- 50 memory

Inputs are used for 132 different sensors:

124 diffuse sensors 6 Retro reflective

One capacitive

One vision sensors

Three buttons
One switch
One emergency
Button and other sensors integrated in the machines).

Outputs are used for actuators:
Six conveyor belts
Five roller conveyors
Two loading conveyors
One chain transfer
Two two-axis pick & place
One pick and place
Two stackers with 3 racks (each rack with 18 storage spot)
Four positioning bars,
Three digital displays,
Two light indicators and
Three button lights indicators,
One siren and other actuators integrated in the stations and machines)

For the program, we use:

Five Function Blocks and two Organization Blocks, over 400 DB (most of them of Timers or F_trig/R_trig)

OB1: contains 6 networks and one FC call (pick & place) (Factory IO doesn't count)

OB123: contains five networks and 5 FB Calls

FB1: Stacker crane one; contains 55 networks and 163 DB

FB2: Stacker crane two; contains 55 networks and 163 DB

FB1 and 2 share the same tag list of 136 Tag mainly storage diffuse sensors.

FB3: Blue picker; contains 5 networks, 23 DB and 22 Tag (11 Inputs, 9 Outputs, 10 memory).

FB4: Green picker; contains 5 networks, 23 DB and 22 Tag (11 Inputs, 9 Outputs, 10 memory).

FB5: Color sorter; contains 9 networks, 29 DB and 37 Tag (11 Input, 10 Output, 17 memory).

FC2: Pick&place; uses 3 networks, 43DB and 12 Tag (2 Inputs, 5 Outputs, 5 memory).

Some other Tags are common so we don't count them above.

[redacted] doesn't send back the information to TIA Portal.

- We had to use OB123 because when trying to call some FB or FC from OB1 Factory I/O send a pulsed signal.
- Switching from PLCSIM to S7-1200 in the simulation was taking most of the time because we had to redo all the Input and Outputs from scratch, and we did

not have the inputs and outputs extensions, so S7-1200 wasn’t sending the full signals.

- We had some struggle with the Laptops that we own to run TIA Portal and Factory I/O and HMI simulation, so we had to swtich to a Desktop Gaming PC from friend for the finale simulation, so we always had the problem of low performance laptops.
- Also Multi-instance DB didn’t work for some reason. So I had to put every TON and TOF in the stackers program by its own Single-instance DB.

IV.7.Conclusion:

In this chapter, we have seen the simulation of the full process, and we used the LADDER programming languages to synchronize the machine work, while we control the process using HMI.

General Conclusion

Conclusion

We conclude that the use of digitalization of a factory using a PLC as S7-1200 is beneficial for the industrial revolution and as engineers we can benefit from it fruits.

To even benefit more of this simulation we add an HMI to more control the work of our automated storage, also to intervene from the control panel in case of a fault or emergency.

Bibliography

Books:

[1] Frank Lamb, Industrial automation Hands-On. 2013
[2] M P. Groover, Automation, Production Systems, and Computer-Integrated Manufacturing 4th Edition. 1980
[3] Terry R. Borden Richard A. Cox, Technician's Guide to Programmable Controllers 6th Edition. 2012
[4] W. Bolton Programmable Logic Controllers, Fifth Edition. 2009
[5] Nebojsa Matic, introduction to PLC controllers, for begginers to. 2008

Sites and others:

[A] https://www.britannica.com/topic/Siemens-AG
[B] http://www.usa.siemens.com/step
[C] TIA Portal v16 Help.
[D] https://docs.factoryio.com/manual/parts/
[E] SIEMENS SITOP power supply SITOP PSU100S Operating Instructions.
[F] https://en.wikipedia.org/wiki/Profinet
[G] Siemens SIMATIC HMI HMI devices Comfort Panels Operating Instructions.