Nikita Lillak

Designing an Automated Sorting System in Simulation Environment

Bachelor's Thesis

Bachelor of Engineering

Electrical and Automation Engineering

2024

Kaakkois-Suomen ammattikorkeakoulu

Degree title	Bachelor of Engineering
Author (authors)	Nikita Lillak
Thesis title	Designing an Automated Sorting System in Simulation Environment
Commissioned by	South-Eastern Finland University of Applied Sciences, XAMK
Time	2024
Pages	82 pages, 28 pages of appendices
Supervisor	Teemu Manninen

ABSTRACT

The main objective of this thesis work was to design and implement an Automated sorting system in virtual simulation environment Simumatik. The thesis work addresses establishing communication between Simumatik's virtual controllers and third-party software, plc programming of a stacker crane, and UR 5 robots programming with the help of Ursim robot's programming environment.

The result of this thesis work is an assembled and operating automated sorting system in Simumatik's virtual environment. Two programs were developed: a PLC program and an Ursim program. An electrical drawing of a system was created. Simulation works successfully. The UR5 robot sorts cans on a pallet, the stacker crane then stores the pallet into a storage slot.

Keywords: PLC, Robot Arm, Digital Twin, Stacker Crane, Simulation

Tutkintonimike	Insinööri (AMK)
Tekijä/Tekijät	Nikita Lillak
Työn nimi	Automatisoidun lajittelujärjestelmän suunnittelu virtuaaliseen simulaatioympäristöön
Toimeksiantaja	Kaakkois-Suomen ammattikorkeakoulu
Vuosi	2024
Sivut	82 sivua, liitteitä 28 sivua
Työn ohjaaja(t)	Teemu Manninen

TIIVISTELMÄ

Tämän opinnäytetyön päätavoitteena oli suunnitella ja toteuttaa automatisoitu lajittelujärjestelmä virtuaalisessa simulaatioympäristössä Simumatik. Opinnäytetyö käsittelee Simumatikin virtuaalisten ohjainten ja kolmannen osapuolen ohjelmiston välisen kommunikaation luomista, pinonosturin PLC-ohjelmointia sekä UR5-robotin ohjelmointia Ursim-ohjelmointiympäristön avulla.

Tämän opinnäytetyön tuloksena on koottu ja toimiva automatisoitu lajittelujärjestelmä Simumatikin virtuaaliympäristössä. Kaksi ohjelmaa on kehitetty: PLC-ohjelma ja Ursim-ohjelma. Järjestelmän sähkökuvia on luotu. Simulaatio toimii onnistuneesti. UR5-robotti lajittelee tölkkejä lavalle, ja pinonosturi varastoi lavan varastopaikan.

Asiasanat: PLC, Robot Arm, Digital Twin, Stacker Crane, Simulation

CONTENTS

1 INTRODUCTION

Industrial automation field is quickly catching up with software development field in terms of accessibility. In software development only a computer is required to practice programming, while to practice PLC programming, access to a physical machine is a requirement. With development of Digital twins and virtual commissioning environments, the industrial automation field became more accessible. Digital twins can be programmed, and the result can be simulated within a virtual environment.

I have intentionally designed this project to be executed solely on a personal computer. Tools used in this project are freely available for noncommercial and education purposes. This is great for students who want to delve into industrial automation field and decide if it suits them.

The main objective of this thesis work is to design and implement an Automated sorting system in virtual simulation environment Simumatik. The System will be divided into two parts: an automated storage system designed to store pallets in the storage, and a Sorting system responsible for positioning the items on top of the pallet. The Automated storage system utilizes a stacker crane controlled by Simumatik’s virtual PLC controller, and the sorting system’s main component UR5 robot is controlled by Simumatik’s implementation of the UR’s virtual controller. A PLC code was written in Codesys integrated development environment, while UR’s virtual controller is programmed with Ursim.

2 SIMULATION IN VIRTUAL ENVIROMENT

2.1 Digital twin and virtual commissioning

Building and changing an existing physical automation system is expensive and time consuming. Digital twin represents a computer-generated replica of a real physical system. The visuals that determine how the component looks is created by 3d software, and the components' functionality is determined by a computer code.

It is much cheaper to build a system in virtual environment first. Digital twin allows to test and commission the system before building it. Commissioning is an important step in ensuring that automated systems are installed, tested, and operating according to the system design requirements.

Virtual commissioning significantly reduces the time required in commissioning. PLC program can be designed and tested before the actual system is physically built. [1.]

2.2 Advantages of Digital Twin and Simulation Environment in Education

The application of digital twin and simulation environment is not only used in professional industry, but in education as well. Education institutions can now enhance students' practical training by incorporating Digital Twin and Simulation Technology into their curriculum, making education more accessible. Previously time consuming and financially demanding technical training, can now be easily arranged through always available simulation software. Students can build, test, and experiment with automation systems as much as they want, without the significant financial burden and risk of component damage. Additionally, simulation software eliminates the risk of damaging components, reduces the costs of physical systems, and minimizes the maintenance of those systems. [2.]

2.3 Simumatik Gateway and Communication Drivers

Simumatik Gateway is used to connect third-party software from different vendors to Simumatik, by adding communication driver to a virtual component. Communication driver is essentially a software configuration with a variety of parameters, and it is responsible for the information flow between virtual component and the gateway. Simumatik Gateway combined with a communication driver, acts as a bridge between Simumatik platform and 3d party software.

In Figure 1. Communication scheme between virtual controllers and third- party software is depicted. The input signals received by virtual components are sent to the third-party software. After receiving these signals, the application developed within the third-party software processes them and sends the output signals back to the virtual component. [3.]

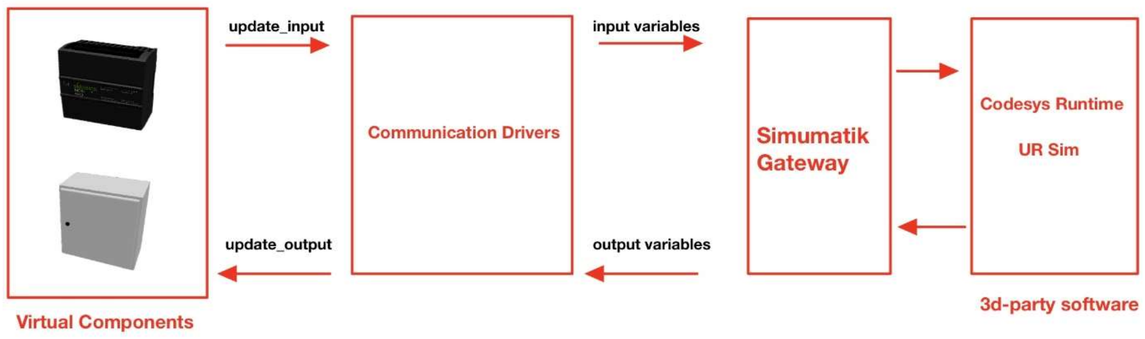


Figure 1. Communication scheme between virtual controllers and third- party software

2.4 PLC (Programmable Logic Controller)

Programmable logic controller (PLC) is essential in industrial and automation processes. It is a modern alternative to the older, physically wired relays-based control systems. PLC significantly simplified the maintaining process by replacing relay-based systems. Instead of manual rewire of the switchboard, switchboard logic can be implemented within a single program. PLCs are utilized to control components of automation systems such as switches, relays, buttons, actuators, and motors. These components typically fall into one of two categories: input or output. For example, input devices include photoelectric sensors or buttons, while output devices could range from actuators to relays. Such components are then connected to a PLC by utilizing I/O modules. Because Industrial and automation systems frequently operate in harsh environment, PLCs are designed to withstand harsh conditions, such as cold, humid, or dust. [4.]

3 OVERVIEW OF THE SOFTWARE USED IN THE PROJECT

3.1 Simumatik

The Simumatik platform is a crucial part of this project. It is simulation software, where various digital twins can be built by combining physical, electrical, pneumatic, or mechatronic components. The projects are stored in the cloud and can be accessed from anywhere. Simulation can also run on a cloud-based server or locally on the user's computer. Additionally, Simumatik is also platform independent. Simumatik's Gateway supports many third-party software and hardware, acting as a bridge between the model and third-party hardware or software. There are four key elements of the components. The visual is responsible for the component's appearance. Physics defines physical features of a component, such as its material, collision shape, and kinematic properties. Interface determines the component's connection points. Connection points are used for linking components together. Lastly, the behavior element is responsible for the component's functionality. These elements can be edited with a component editor, while the component's functionality can be edited with Python programming language. [5.]

Simumatik also has a great resource Simumatik Academy. It provides excellent tutorials and courses about Simumatik's features, components, programming as well as 3d party software integration courses. It has been great help with navigating the platform features and integrating 3d party software into the Simumatik. [6.]

3.2 Codesys and Ursim

This project involves Codesys integrated development environment for creating an application for a virtual PLC controller which controls the stacker crane, and Codesys Control WIN 3 runtime, which enables a windows-based machine to interpret and execute the application developed with CODESYS IDE. Essentially the Codesys runtime turns any device that meets the requirements into an IEC 61131-3 compatible controller. Codesys IDE uses IEC 61131-3 standard program languages. IEC 61131-3 Standard encompasses several programming languages such as Lader Logic, Structured Text, Instruction List, Function Block Diagram, and Sequential Function Chart. [7.] In this project PLC will be programmed with ladder logic, while structured text will be used to link I/O Variables.

Ursim will be used for developing an application for a virtual UR controller that controls the UR 5 robot. Ursim is Linux software and will not run on windows operating system. To run

Ursim on Windows operating system, Universal Robots created a virtual machine that can be run with the help of virtual machine software VMWARE. [8.]

4 SYSTEM DESIGN

To test if two different 3d- party software could work simultaneously in the Simumatik platform,it was necessary to select such systems, each controlled by unique software. The core of the project was formed by merging concepts from two separated Simumatik projects. The concept for the Automated Storage and Retrieval System was adopted from the implementation showcased in Simumatik's LIU Station 4 project. The concept for the Can Sorting System was inspired by Simumatik's "Showcase of Universal Robot" project.

Automated Storage and Retrieval System (ASRS) would be responsible for storing pallets into the storage slots, utilizing a stacker crane. The stacker crane is the main material handling component of the ASRS system. The stacker crane will be controlled by a PLC controller (Appendix 1, PLC 16 DIO 4AIO) connected to its actuators. The limit switches (Appendix 1, Limit Switch) will enable the PLC controller to achieve precise positioning of the stacker crane to the storage slots. Showcasing the programming of a complicated system such as stacker crane would provide valuable insight into designing an automated sorting system. In Appendix 1, Stacker Crane provides additional information about a stacker crane's functionality.

The can sorting system will sort cans on top of the pallet, which later would be stored in ASRS system's storage. As the material handling component, the can sorting system utilizes a UR 5 robot arm (Appendix 1/ 4, UR5 Robot). The control of the UR 5 robot arm is achieved via a virtual UR controller (Appendix 1/ 4, Universal Robots Controller), connected to its input axis. The systems controller's PLC and UR Controller will also control conveyors (Appendix 1, Conveyor Belts) via motor contactors and relays (Appendix 1, Motor contactor and DC Relay), while also processing inputs from photoelectric sensors (Appendix 1, Photoelectric Sensor).

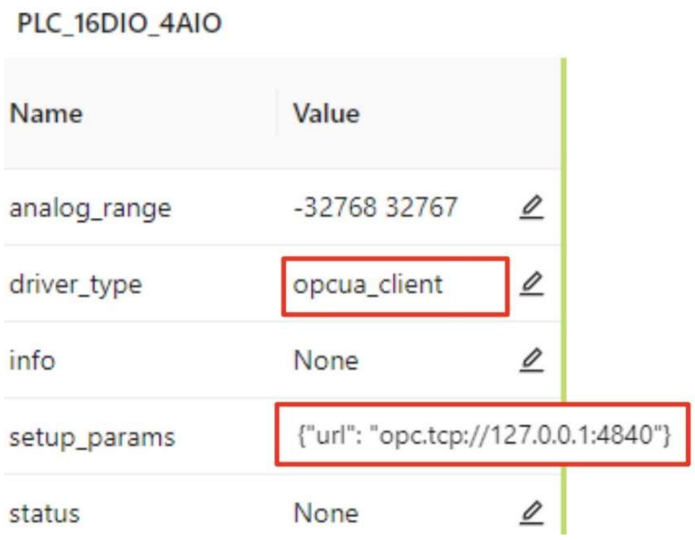
The initial operation sequence of the can sorting system was very simplistic. When a pallet with a box on it arrives at a robot arm, the robot arm would place one can inside a box. However, the programming of the robot turned out to be easier than initially expected. The cans rack with 4 slots replaced the box, as it was a more appropriate item holder, enabling this project to develop a more complex code and further differentiate from Simumatik's Showcase of Universal Robot" project.

5 IMPLEMENTATION

5.1 Establishing Gateway Connections in Simumatik

5.1.1 Gateway connection between Codesys and Simumatik

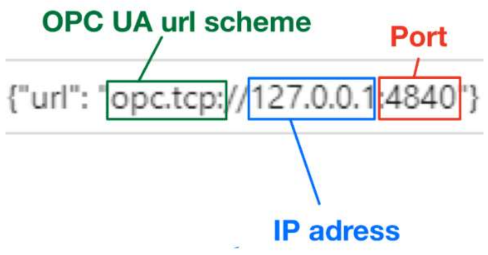
In the virtual plc setting, opcua_client is selected as a driver type (Figure 3.) and the setup_params will include URL address of opcua server. OPC UA is a protocol integrated in some plc. It allows plc devices from different manufacturers to communicate with each other. Figure 2 illustrates the parameters of OPC UA URL address.



PLC_16DIO_4AIO

Name	Value
analog_range	-32768 32767
driver_type	opcua_client
info	None
setup_params	{"url": "opc.tcp://127.0.0.1:4840"}
status	None

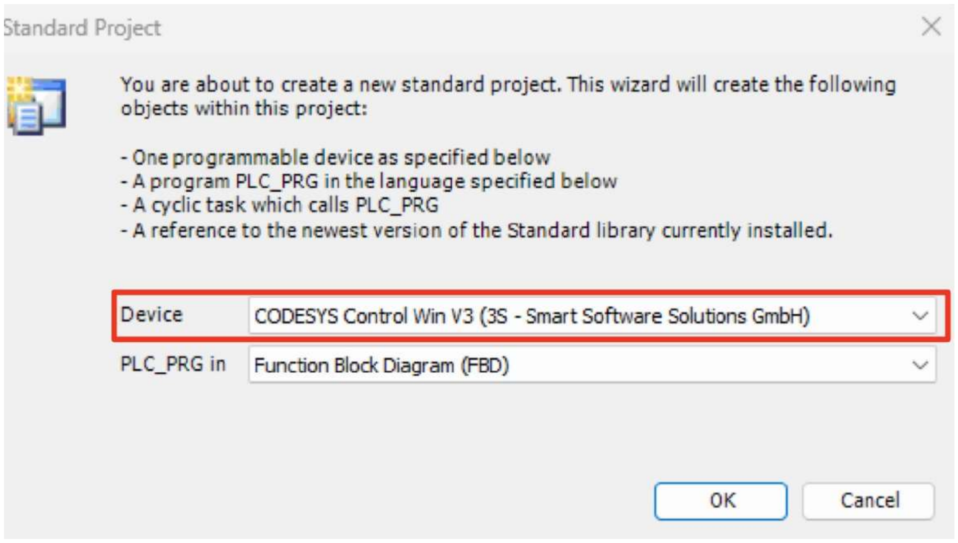
Figure 3. Simumatik's PLC controller's configuration panel



{"url": "opc.tcp://127.0.0.1:4840"}

Figure 2. OPC UA IP adress

Next in the Codesys a software project must be created, and CODESYS Control Win V3(3S- Smart Software Solutions GmbH) is selected as a device (Figure 4). CODESYS Control Win V3 is a software-based PLC with built-in OPC UA Server. It runs on a regular computer and serves as a PLC controller that can be programmed with IEC 61131-3 programming languages. [9.]



Standard Project

You are about to create a new standard project. This wizard will create the following objects within this project:

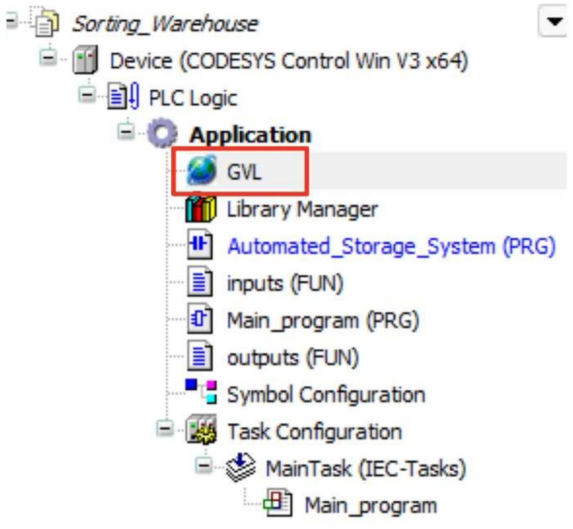
- One programmable device as specified below
- A program PLC_PRG in the language specified below
- A cyclic task which calls PLC_PRG
- A reference to the newest version of the Standard library currently installed.

Device: CODESYS Control Win V3 (3S - Smart Software Solutions GmbH)

PLC_PRG in: Function Block Diagram (FBD)

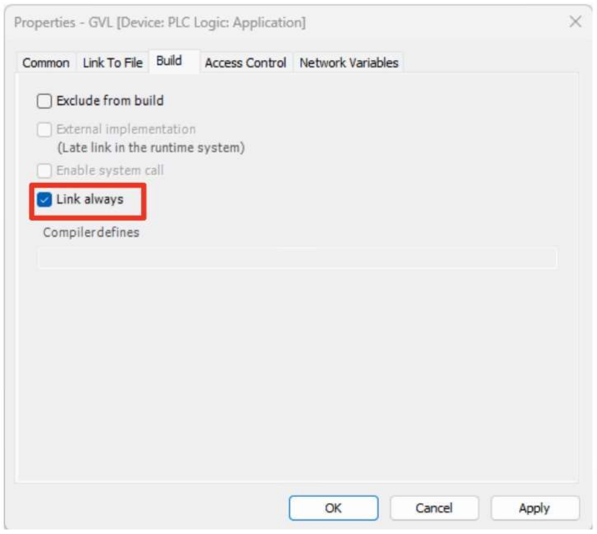
Figure 4. Codesys Project creation window

After Global Variable List (GVL) must be added (Figure 6.), and in GVL options window, under build tab, link always option must be selected (Figure 5.), to appear in the symbol configuration panel later. Global variables can be accessed from anywhere in our program. [10.]



```
Sorting_Warehouse
  Device (CODESYS Control Win V3 x64)
    PLC Logic
      Application
        GVL
        Library Manager
        Automated_Storage_System (PRG)
        inputs (FUN)
        Main_program (PRG)
        outputs (FUN)
        Symbol Configuration
        Task Configuration
          MainTask (IEC-Tasks)
            Main_program
```

Figure 6. Codesys project explorer



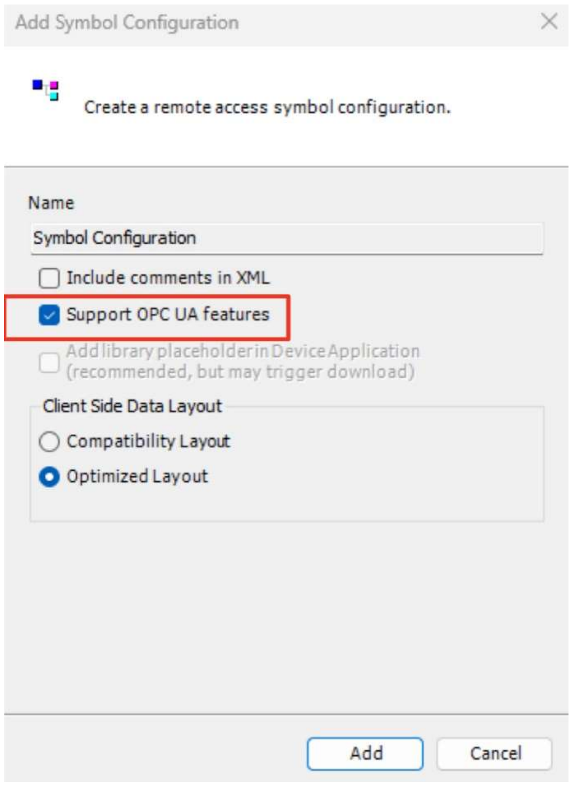
Properties - GVL [Device: PLC Logic: Application]

- [ ] Exclude from build
- [ ] External implementation (Late link in the runtime system)
- [ ] Enable system call
- [x] Link always

Compilerdefines

Figure 5. GVL's properties tab

Afterwards symbol configuration must be created, and OPC UA features (Figure 7.). This allows the OPC server to access the variables. [10.]


Add Symbol Configuration

Create a remote access symbol configuration.

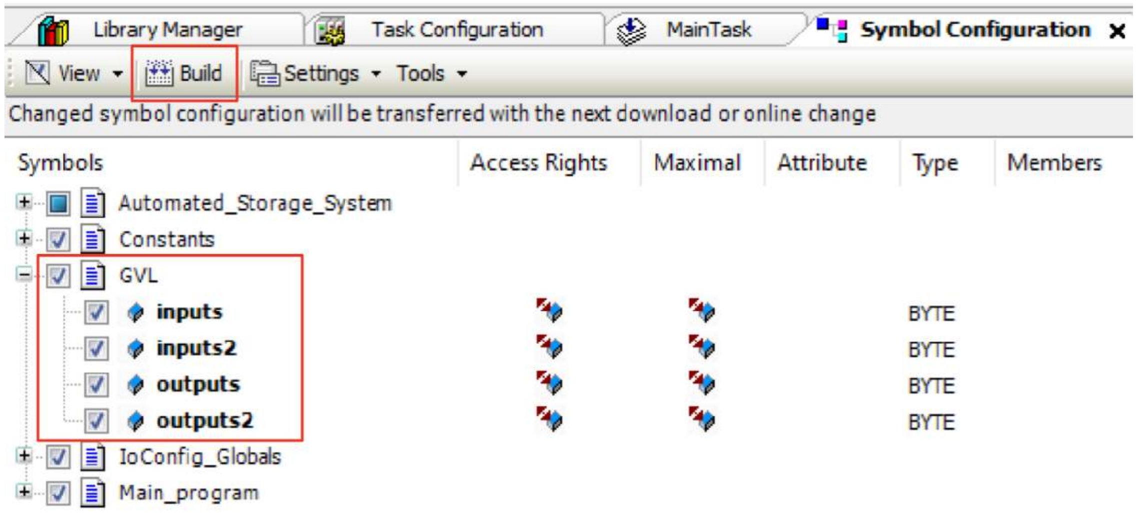
Name: Symbol Configuration

- [ ] Include comments in XML
- [x] Support OPC UA features
- [ ] Add library placeholder in Device Application (recommended, but may trigger download)

Client Side Data Layout
- ○ Compatibility Layout
- ● Optimized Layout

Figure 7. Codesys add symbol configuration window

In symbol configuration object GVL variables must be selected and build process initiated (Figure 8). [10.]



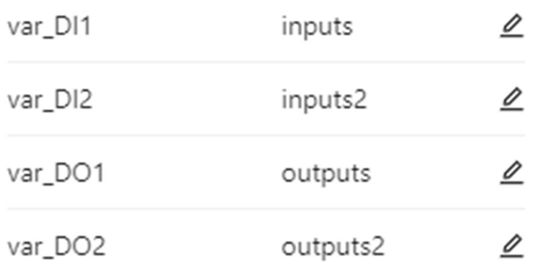
Changed symbol configuration will be transferred with the next download or online change

Symbols	Access Rights	Maximal	Attribute	Type	Members
Automated_Storage_System					
Constants					
GVL					
inputs				BYTE	
inputs2				BYTE	
outputs				BYTE	
outputs2				BYTE	
IoConfig_Globals					
Main_program					

Figure 8. Codesys symbol configuration tab

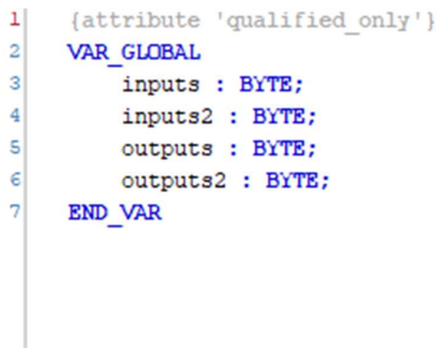
Next In the virtual plc setting, Input, and output variables var_DI1, var_DI2, var_DO1 and var_DO2 (Figure 9) must be defined in Codesys as specified in Figure 10. Since

PLC_16DIO_4AIO supports 16 inputs and 16 outputs, in Codesys Globa Variable List (GVL) the same variables must be defined as BYTE. Each input and output variable equals 8 bits, and BYTE consists of 8 bits. Future Boolean variables will be linked to a global variable.



var_DI1	inputs
var_DI2	inputs2
var_DO1	outputs
var_DO2	outputs2

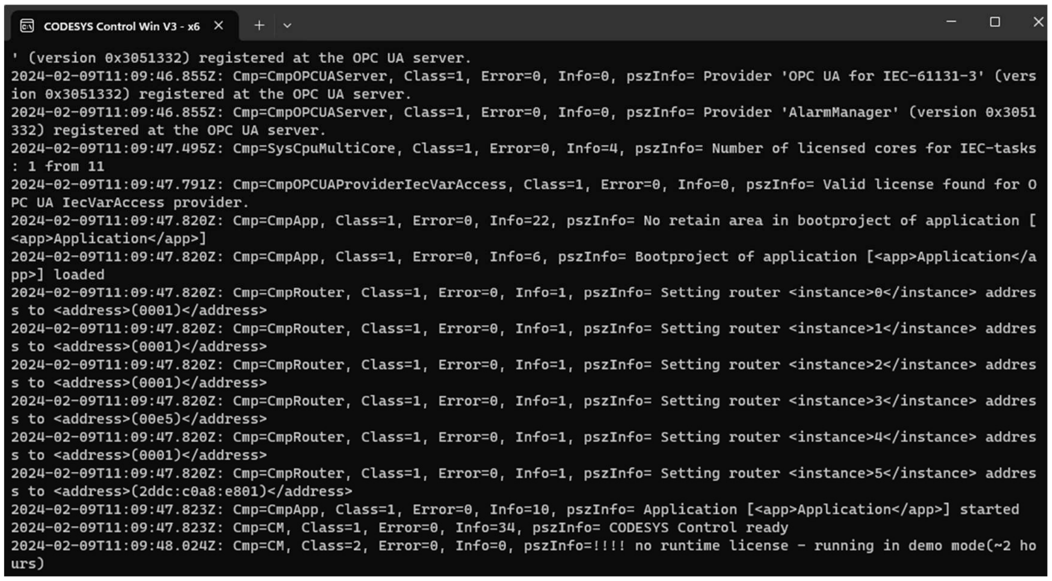
Figure 9. Simumatik's PLC 16DIO4AIO configuration window



```
{attribute 'qualified_only'}
VAR_GLOBAL
    inputs : BYTE;
    inputs2 : BYTE;
    outputs : BYTE;
    outputs2 : BYTE;
END_VAR
```

Figure 10. Global variable declaration in codesys

Next Codesys Control Win is launched (Figure 11). The free version has a runtime limit and must be reset every 2 hours.



```
' (version 0x3051332) registered at the OPC UA server.
2024-02-09T11:09:46.855Z: Cmp=CmpOPCUAServer, Class=1, Error=0, Info=0, pszInfo= Provider 'OPC UA for IEC-61131-3' (vers
ion 0x3051332) registered at the OPC UA server.
2024-02-09T11:09:46.855Z: Cmp=CmpOPCUAServer, Class=1, Error=0, Info=0, pszInfo= Provider 'AlarmManager' (version 0x3051
332) registered at the OPC UA server.
2024-02-09T11:09:47.495Z: Cmp=SysCpuMultiCore, Class=1, Error=0, Info=4, pszInfo= Number of licensed cores for IEC-tasks
: 1 from 11
2024-02-09T11:09:47.791Z: Cmp=CmpOPCUAProviderIecVarAccess, Class=1, Error=0, Info=0, pszInfo= Valid license found for O
PC UA IecVarAccess provider.
2024-02-09T11:09:47.820Z: Cmp=CmpApp, Class=1, Error=0, Info=22, pszInfo= No retain area in bootproject of application [
<app>Application</app>]
2024-02-09T11:09:47.820Z: Cmp=CmpApp, Class=1, Error=0, Info=6, pszInfo= Bootproject of application [<app>Application</a
pp>] loaded
2024-02-09T11:09:47.820Z: Cmp=CmpRouter, Class=1, Error=0, Info=1, pszInfo= Setting router <instance>0</instance> addres
s to <address>(0001)</address>
2024-02-09T11:09:47.820Z: Cmp=CmpRouter, Class=1, Error=0, Info=1, pszInfo= Setting router <instance>1</instance> addres
s to <address>(0001)</address>
2024-02-09T11:09:47.820Z: Cmp=CmpRouter, Class=1, Error=0, Info=1, pszInfo= Setting router <instance>2</instance> addres
s to <address>(0001)</address>
2024-02-09T11:09:47.820Z: Cmp=CmpRouter, Class=1, Error=0, Info=1, pszInfo= Setting router <instance>3</instance> addres
s to <address>(00e5)</address>
2024-02-09T11:09:47.820Z: Cmp=CmpRouter, Class=1, Error=0, Info=1, pszInfo= Setting router <instance>4</instance> addres
s to <address>(0001)</address>
2024-02-09T11:09:47.820Z: Cmp=CmpRouter, Class=1, Error=0, Info=1, pszInfo= Setting router <instance>5</instance> addres
s to <address>(2ddc:c0a8:e801)</address>
2024-02-09T11:09:47.823Z: Cmp=CmpApp, Class=1, Error=0, Info=10, pszInfo= Application [<app>Application</app>] started
2024-02-09T11:09:47.823Z: Cmp=CM, Class=1, Error=0, Info=34, pszInfo= CODESYS Control ready
2024-02-09T11:09:48.024Z: Cmp=CM, Class=2, Error=0, Info=0, pszInfo=!!!! no runtime license - running in demo mode(~2 ho
urs)
```

Figure 11. Codesys control WIN V3 console

Next in Codesys device configuration menu network scanning is performed (Figure 13), and soft plc selected (Figure 12).

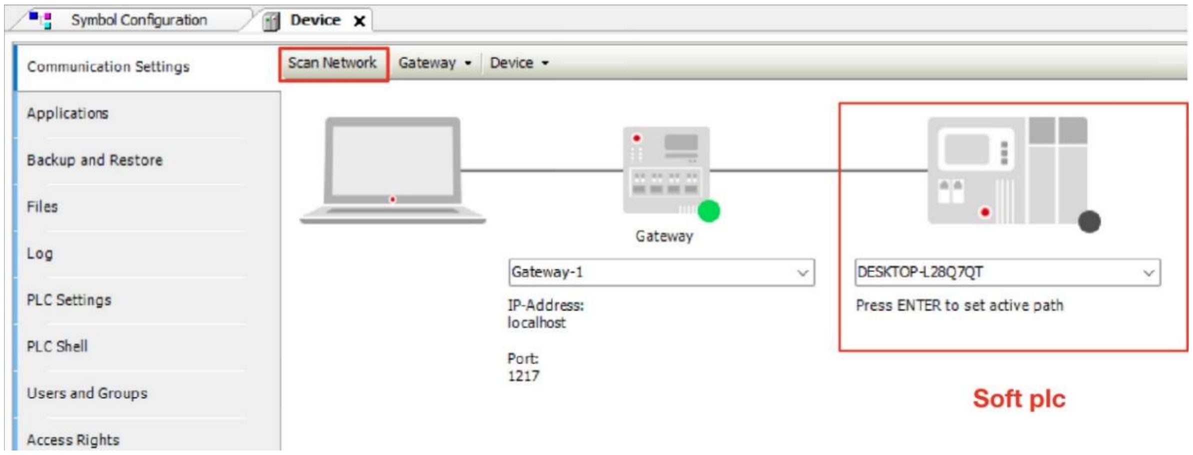


Figure 13. Codesys device configuration menu

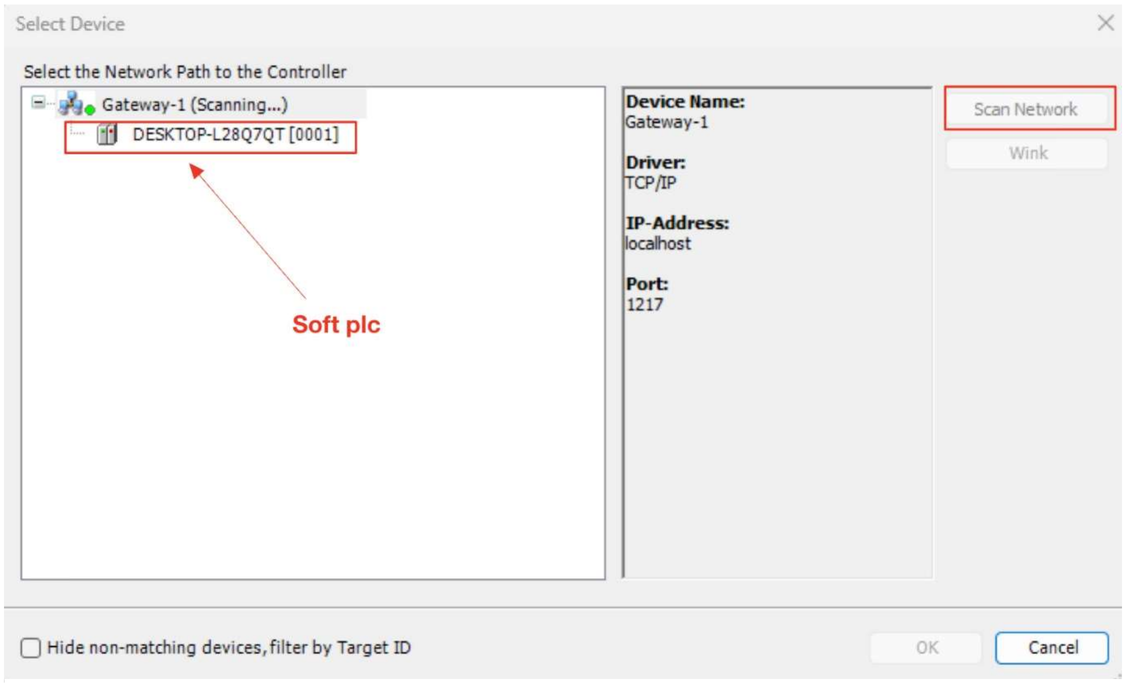


Figure 12. Codesys device configuration, select device window

Then in the device configuration setting in tab Runtime Security Policy (Figure 14), “Allow anonymous login” option must be selected to establish a connection without providing credentials (Figure 15).

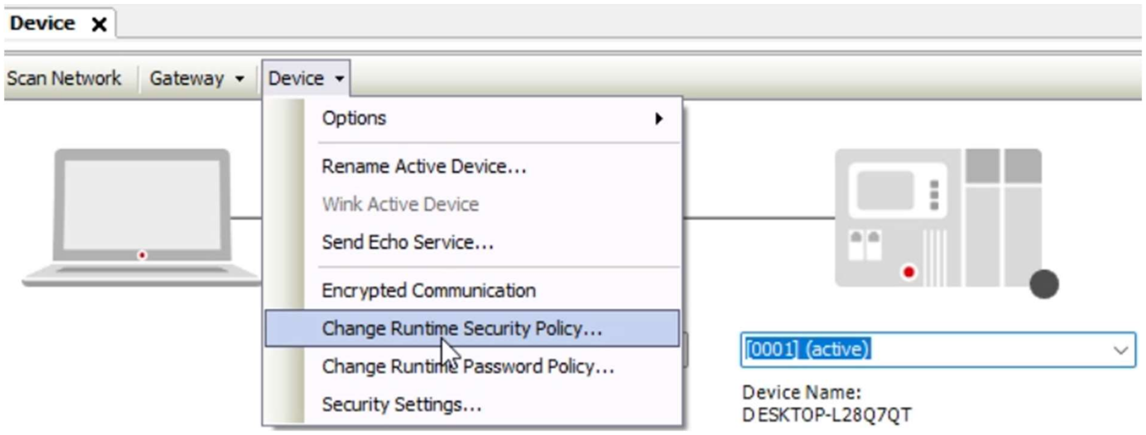
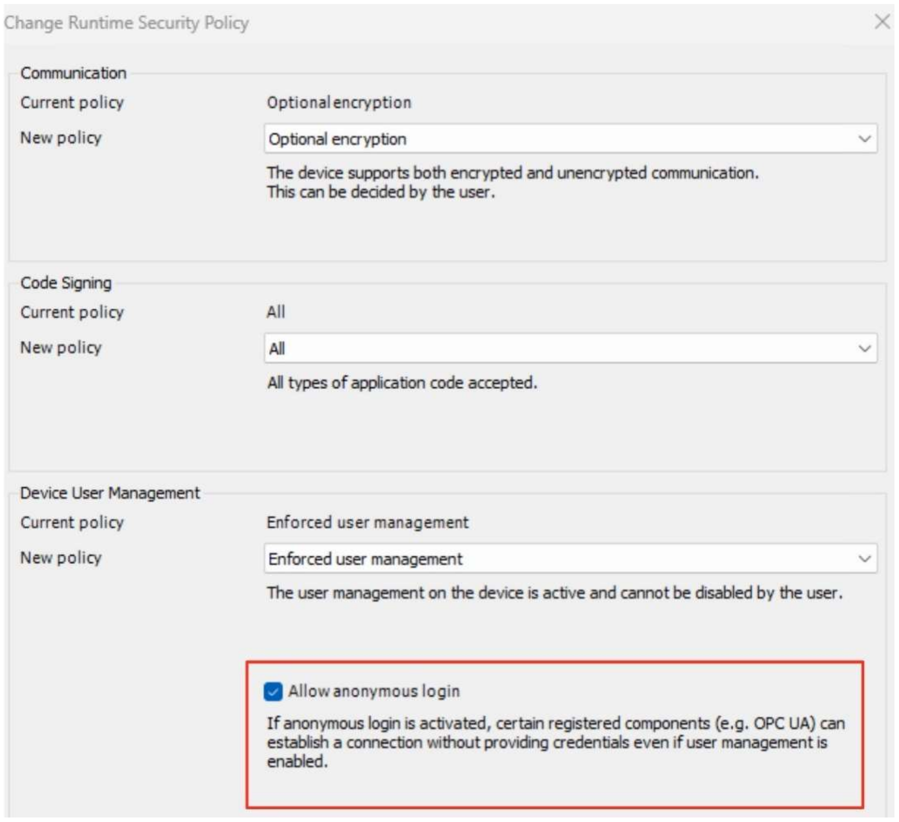


Figure 14. Location of the Codesys runtime security policy tab


Change Runtime Security Policy

Communication
Current policy Optional encryption
New policy Optional encryption
The device supports both encrypted and unencrypted communication.
This can be decided by the user.

Code Signing
Current policy All
New policy All
All types of application code accepted.

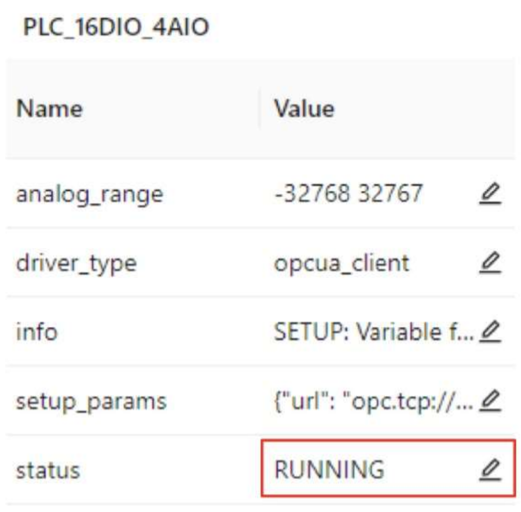
Device User Management
Current policy Enforced user management
New policy Enforced user management
The user management on the device is active and cannot be disabled by the user.

[✓] Allow anonymous login
If anonymous login is activated, certain registered components (e.g. OPC UA) can establish a connection without providing credentials even if user management is enabled.

Figure 15. Codesys runtime security policy tab

To test the connection and download code to the Codesys runtime, the user is required to log into a device in the Codesys development environment,and then establishing a gateway connection in Simumatic by selecting the gateway icon and starting the emulation.

When emulation is running, if status message RUNNING (Figure 16) appears in the PLC view window, then connection is successful. [10.]

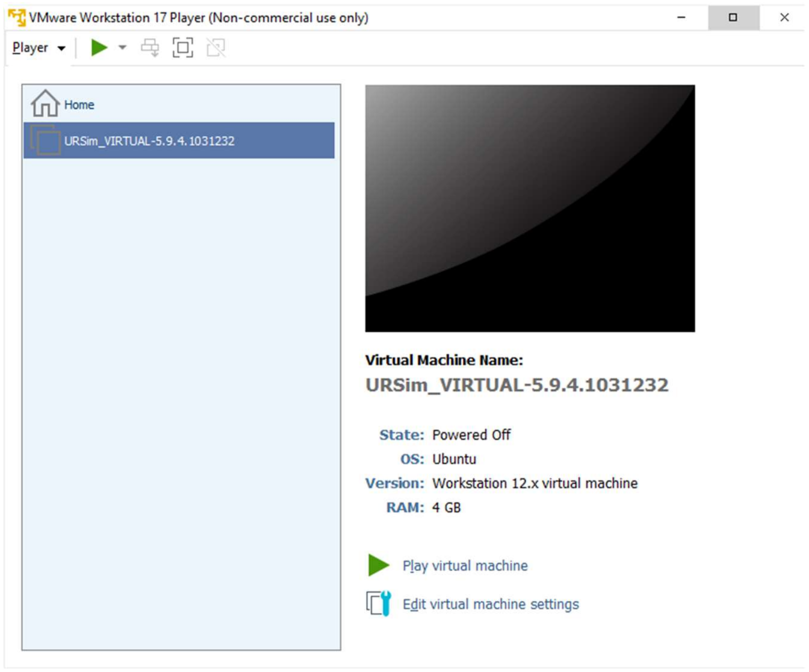

PLC_16DIO_4AIO

Name	Value
analog_range	-32768 32767
driver_type	opcua_client
info	SETUP: Variable f...
setup_params	{"url": "opc.tcp://...
status	RUNNING

Figure 16. Simumatik's PLC16DIO4AIO configuration panel

5.1.2 Gateway connection between URsim and Simumatik

URsim is unable to run directly on windows operating system due to its reliance on the Linux operating system. Therefore, Ursim will be opened on a virtual machine (Figure 17). The manufacturer of Universal Robots has developed a virtual machine, that already contains all the necessary tools. VMware Player was selected as the virtual machine software. [8; 11.]


VMware Workstation 17 Player (Non-commercial use only)
Player
Home
URSim_VIRTUAL-5.9.4.1031232
Play virtual machine
Edit virtual machine settings

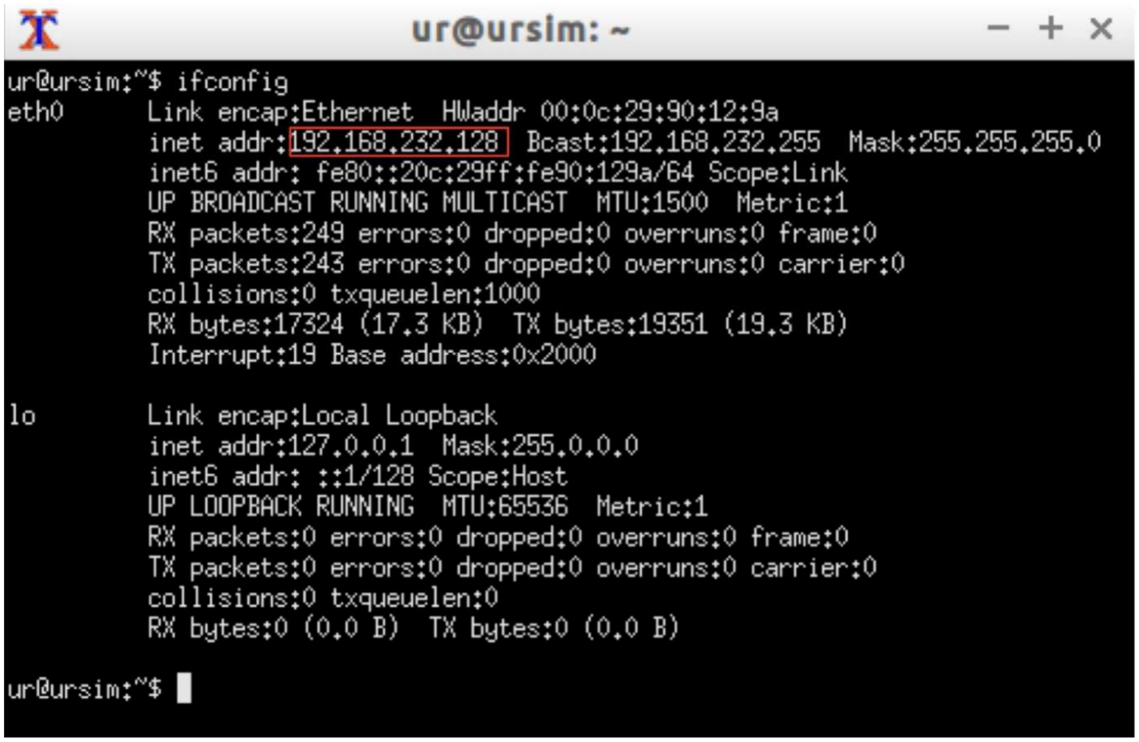
Virtual Machine Name:
URSim_VIRTUAL-5.9.4.1031232

State: Powered Off
OS: Ubuntu
Version: Workstation 12.x virtual machine
RAM: 4 GB

Figure 17. VMware Workstation Player with URsim virtual machine

It is important to have selected NAT as a network connection in Virtual machine settings. This will share the host’s Ip address. [11.]

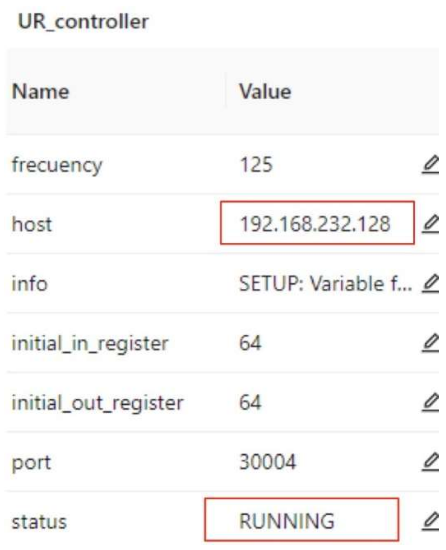
To determine the IP address of Ursim virtual machine’s, Linux terminal tool XTerm is launched with ifconfig command (Figure 18).



```
ur@ursim:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:90:12:9a
          inet addr:192.168.232.128  Bcast:192.168.232.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe90:129a/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:249 errors:0 dropped:0 overruns:0 frame:0
          TX packets:243 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:17324 (17.3 KB)  TX bytes:19351 (19.3 KB)
          Interrupt:19 Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

ur@ursim:~$
```

Figure 18. Linux terminal window in URsim virtual machine

This IP address is used to connect to Simumatik implementation of the UR Controller. Connection is successful if in the status field appears message RUNNING (Figure 19).



UR_controller

Name	Value
frecuency	125
host	192.168.232.128
info	SETUP: Variable f...
initial_in_register	64
initial_out_register	64
port	30004
status	RUNNING

Figure 19. Simumatik's UR controller configuration panel

5.2 System Configuration and Functionality Overview

5.2.1 Automated Storage and Retrieval System

ASRS system (Figure 20) consists of Storage part, stacker crane, 10 limit switches, 2 conveyor belts, photoelectric sensor, and AC motor. The storage has 9 slots (Figure 21). The sacker crane lifts items coming from the conveyor and places them in the appropriate slots with the help of limit switches. The limit switches Y (Figure 20) are responsible for vertical positioning, while the limit switches X (Figure 20) are responsible for horizontal positioning. When the crane reaches the limit switch, the stacker crane's lever (Figure 21) trigger limit switch's actuator. Afterwards, the limit switch sends a signal to a PLC, thus stopping the motor. To set the stacker crane into its initial position, an initial position limit switch has been added. Limit switch wiring to a PLC can be seen in Appendix 2, page 2 of Wiring diagrams of the Automated Sorting System.

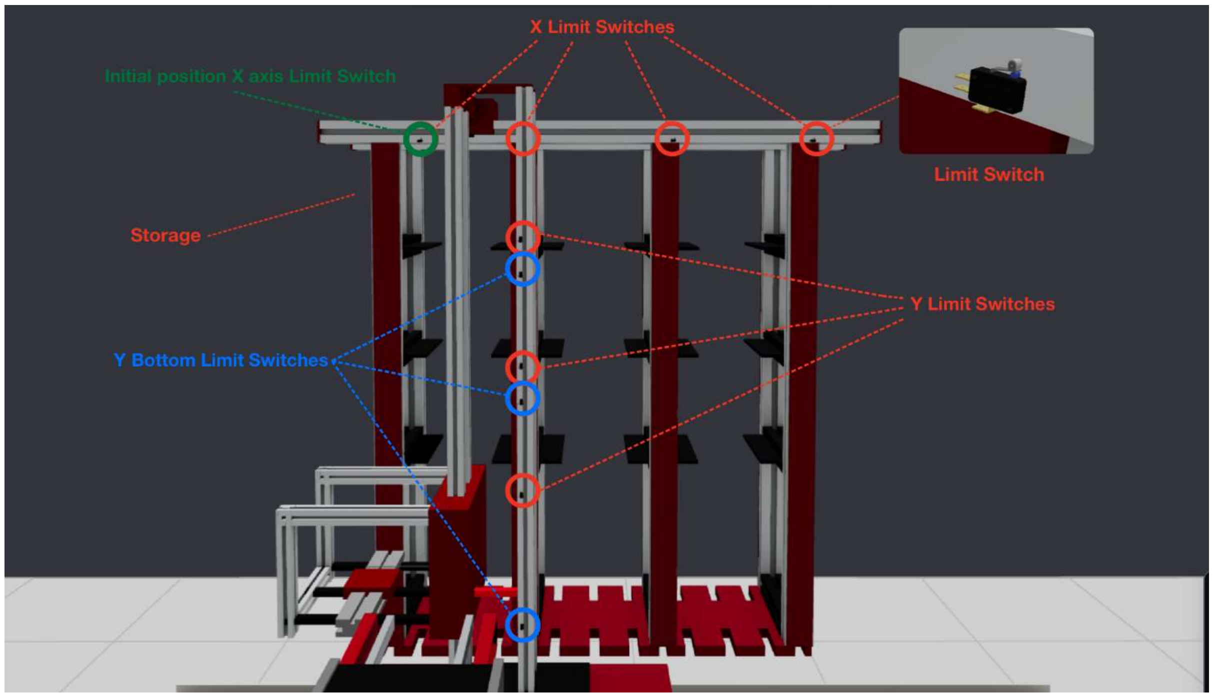


Figure 20. Location of the limit switches in the automated storage and retrieval system in Simumatik

During unloading operation, to ensure that the object stays in the desired slot, the stacker crane must descend lower that the Y limit switch. To achieve this, bottom limit switches have been added (Figure 20). The bottom limit switches are responsible for preventing the crane from descending further after placing an object into the slot.

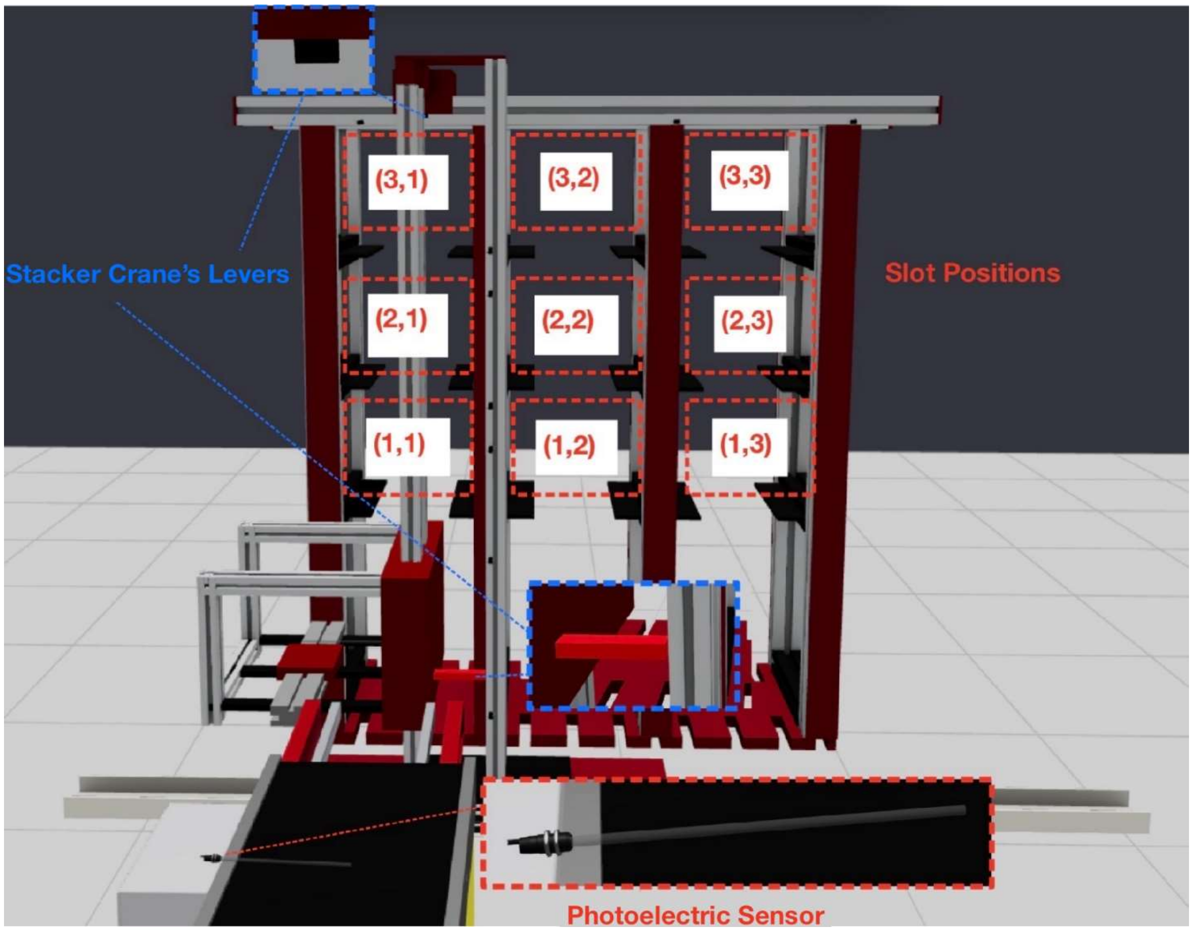


Figure 21. Location of the levers and slots in the automated storage and retrieval system in Simumatik

ASRS system is controlled by Simumatik virtual PLC controller. Additionally, is has essential components like a three-phase socket to provide power for an AC motor and DC power supply. The DC power supply supplies power to DC components such as limit switches, PLC, and photoelectric sensor. An AC motor axis is connected to both conveyors' motor ports, initiating conveyor operation. The limit switches, motor contactor, stacker crane's actuators and photoelectric sensor are all connected to a PLC. The PLC responsible for controlling the whole system. The motor contactor provides the PLC with capability to control an AC motor. The circuit breaker protects the AC motor from overcurrent.

The photoelectric sensor sends a signal to a PLC upon the arrival of a box. The photoelectric sensor located on a stacker crane's platform (Figure 23), sends a signal to a PLC when the box reaches the center of the stacker's crane platform. Each of the components previously described is illustrated in Figure 22. and the connections between components can be viewed in the wiring diagram (Appendix 2, page 1 and 2).

The stacker crane is an autonomous component and does not require a power source in this simulation and only has actuator inputs.

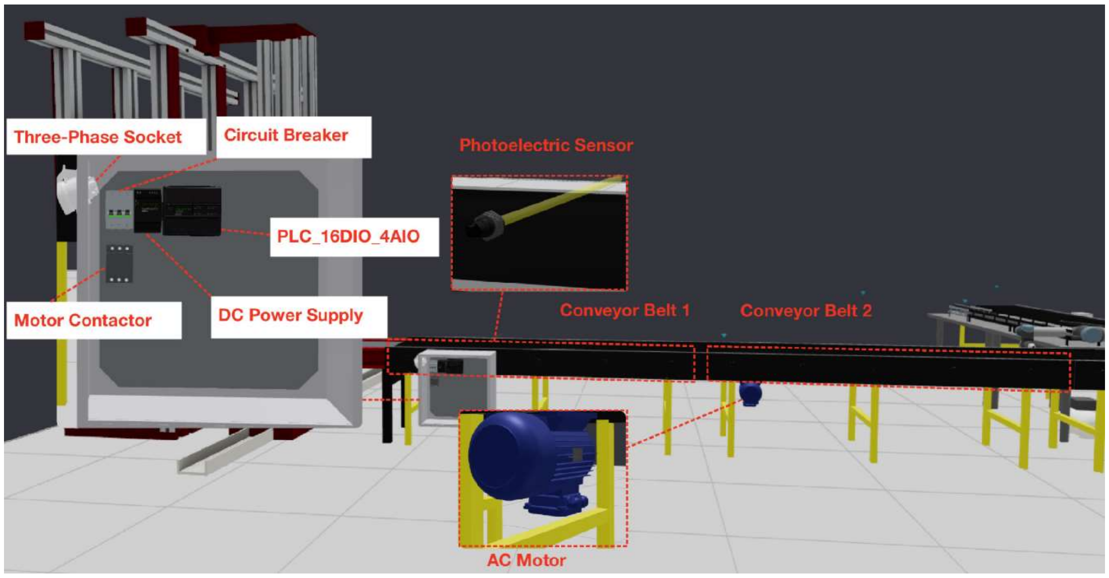


Figure 22. Locations of the electric components in the automated storage and retrieval system in Simumatik

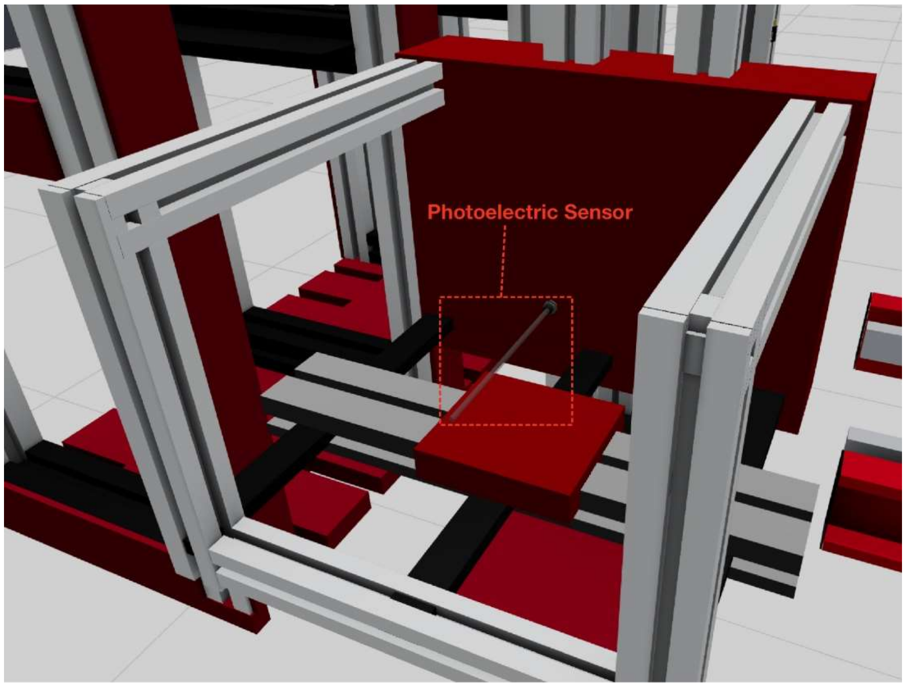


Figure 23. The photoelectric sensor located on a stacker crane’s platform in Simumatik

5.2.2 Can Sorting System

The can sorting system consists of a UR 5 robot, 3 photoelectric sensors, vacuum gripper, 4 large conveyor belts and 2 AC motors. Additionally, it includes components such as mini conveyor belts powered by 2 DC motors, a DC Relay, a DC power supply, 2 circuit breakers, 2 motor contacts for AC motor functionality, three-phase socket, pneumatic compressor, and UR Controller.

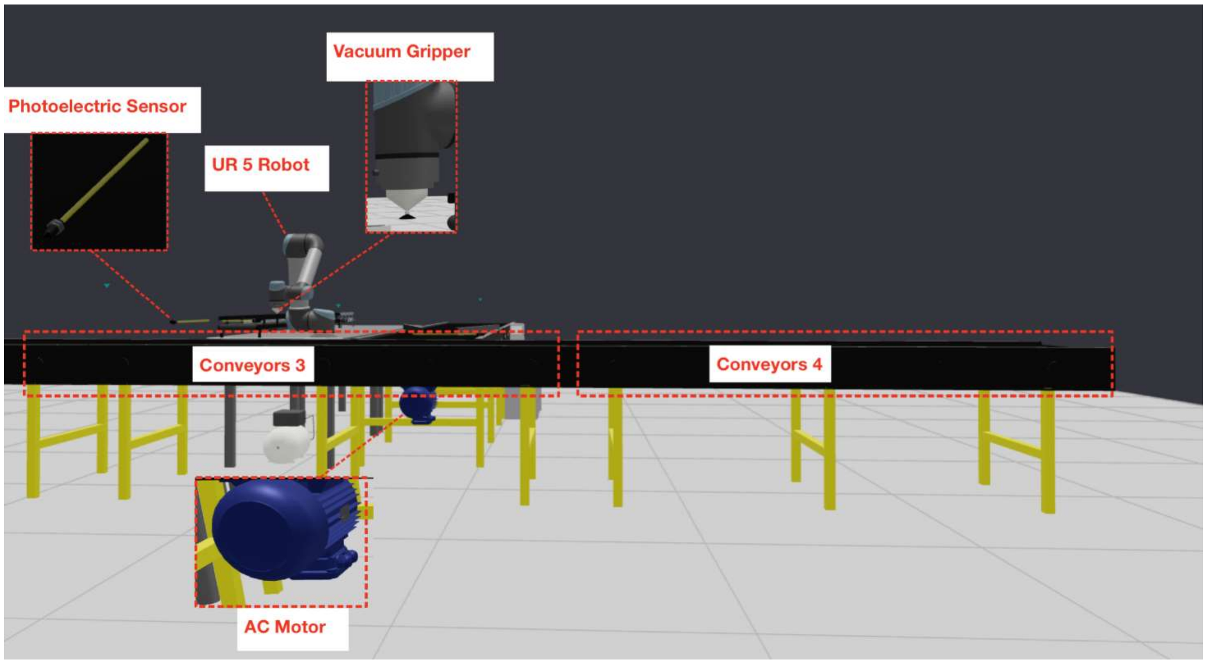


Figure 24. Locations of the can sorting system components in Simumatik, part1

UR5 robot is responsible for the sorting operation. It handles the lifting and placing of the can rack on top of the wooden pallet and inserting cans into the can rack’s slot. Photoelectric sensors send signals to the UR controller upon detecting an item, causing the conveyor to stop. Powered by a pneumatic compressor, a vacuum gripper is mounted on the UR5 Robot, and it is responsible for gripping objects.

Powered by an AC motor, large size conveyors 1 and 2 transport the can’s rack, while Conveyors 3 and 4 transport the wooden pallet. Mini conveyors are powered by DC motors, and they are used to transport cans. UR 5 controller controls DC motors with the help of DC Relay and AC motors with the help of motor contactors. A three-phase socket powers up the AC motors, the UR5 Robot and the DC power supply. Each of these component’s locations are illustrated in Figure 24, Figure 25 and Figure 26 and the connections between components can be viewed in the wiring diagram (Appendix 2, page 3 and 4) .

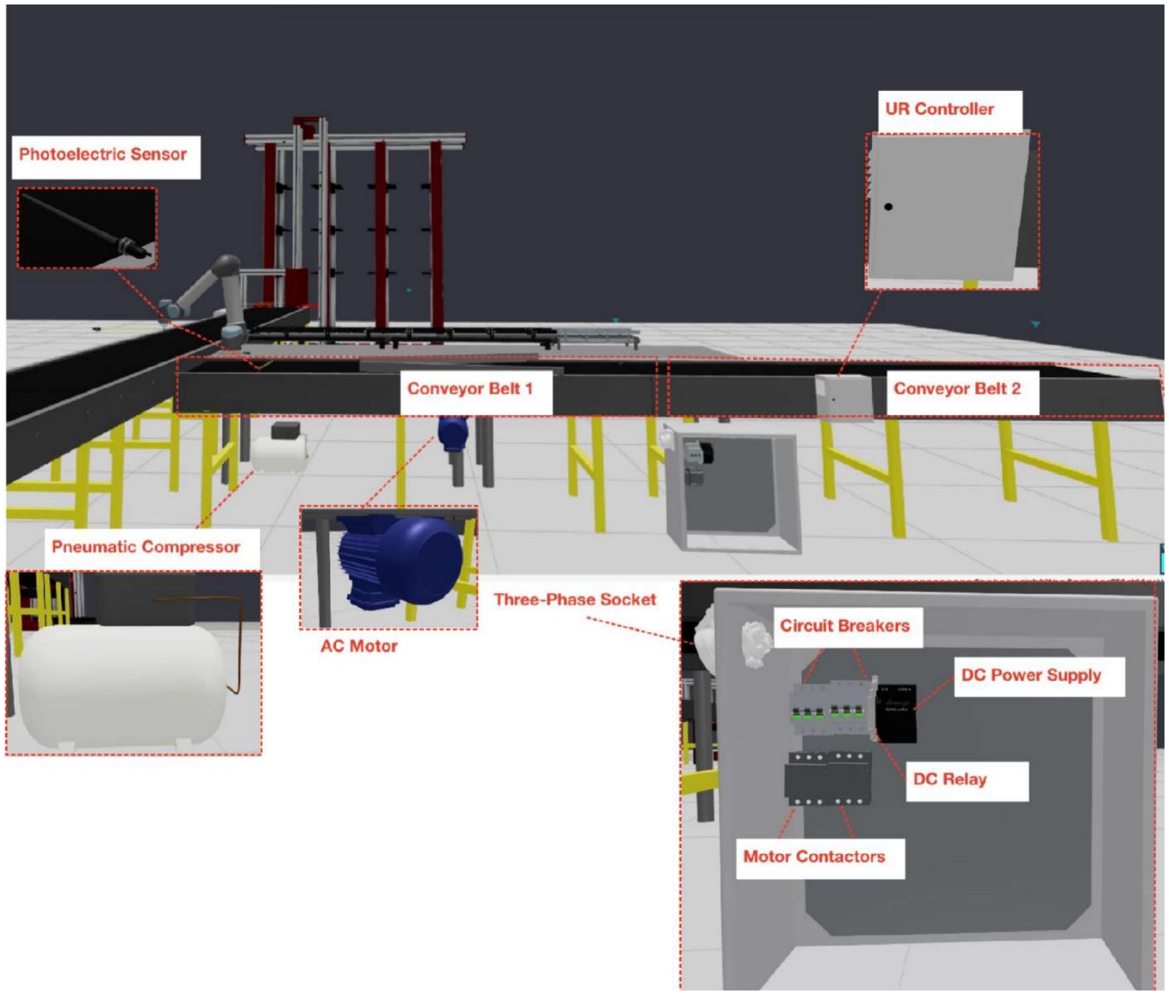


Figure 26. Locations of the can sorting system components in Simumatik, part2

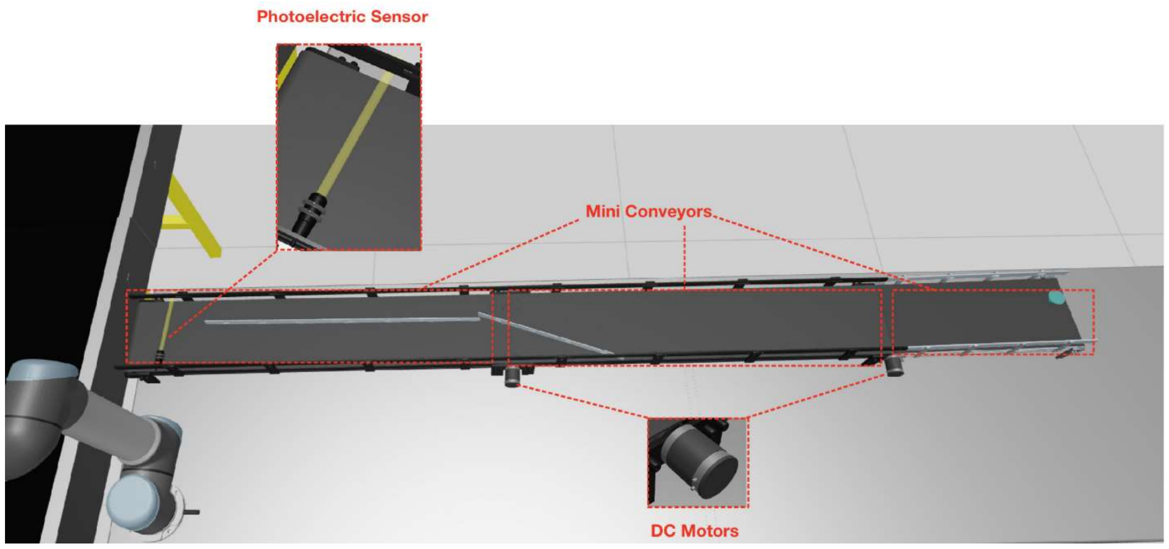


Figure 25. Locations of the can sorting system components in Simumatik, part3

5.3 Programming portion using PLC

5.3.1 I/O Mapping

I/O Mapping is a process of linking application variables with actual inputs and outputs of the PLC. First Global Variable List (GVL) must be created, and global variables defined as a BYTE. Simumatik virtual PLC has 16 inputs and 16 outputs. Four global variables are created: “inputs”, “inputs2”, “outputs” and “outputs2”. Each global variable is defined as a BYTE, as a BYTE consists of 8 bits (Figure 27). Each bit will be linked to each I/O application variable. For this purpose, inputs and outputs functions are created. [12;13.]
Function inputs mapping input variables to application variables, as illustrated in Figure 29, and function outputs mapping output variables to application variables as shown in Figure 28.

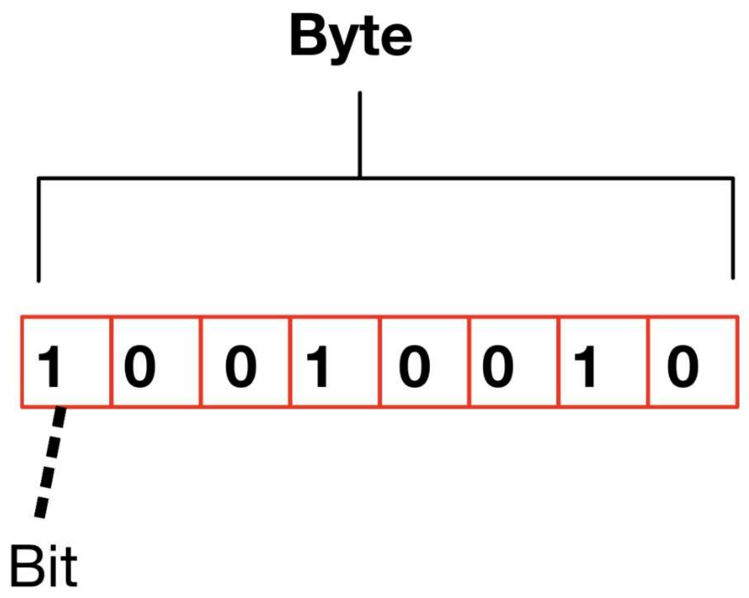
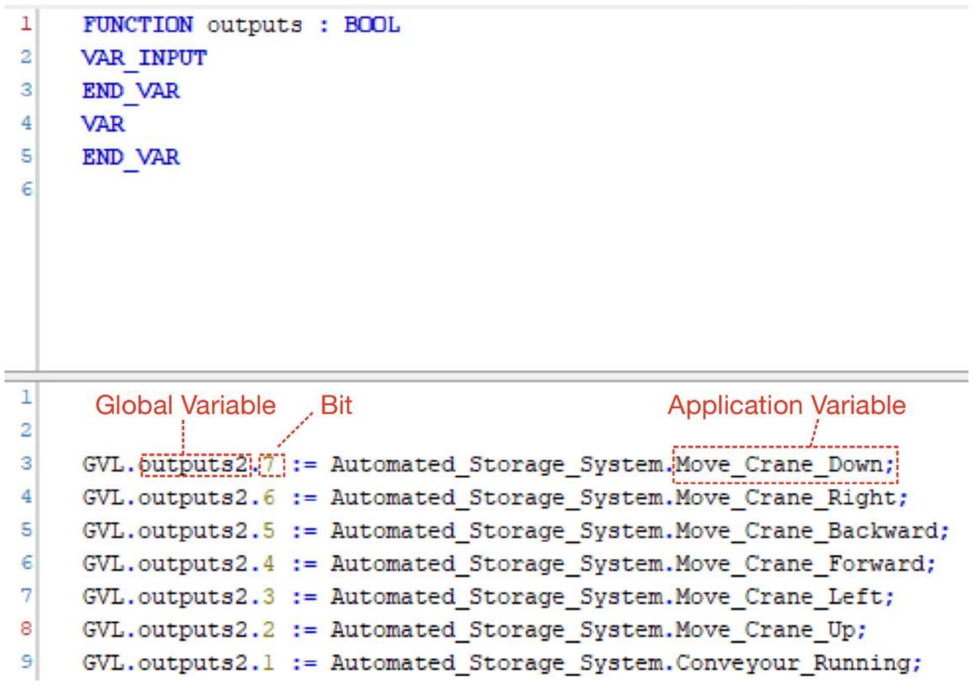


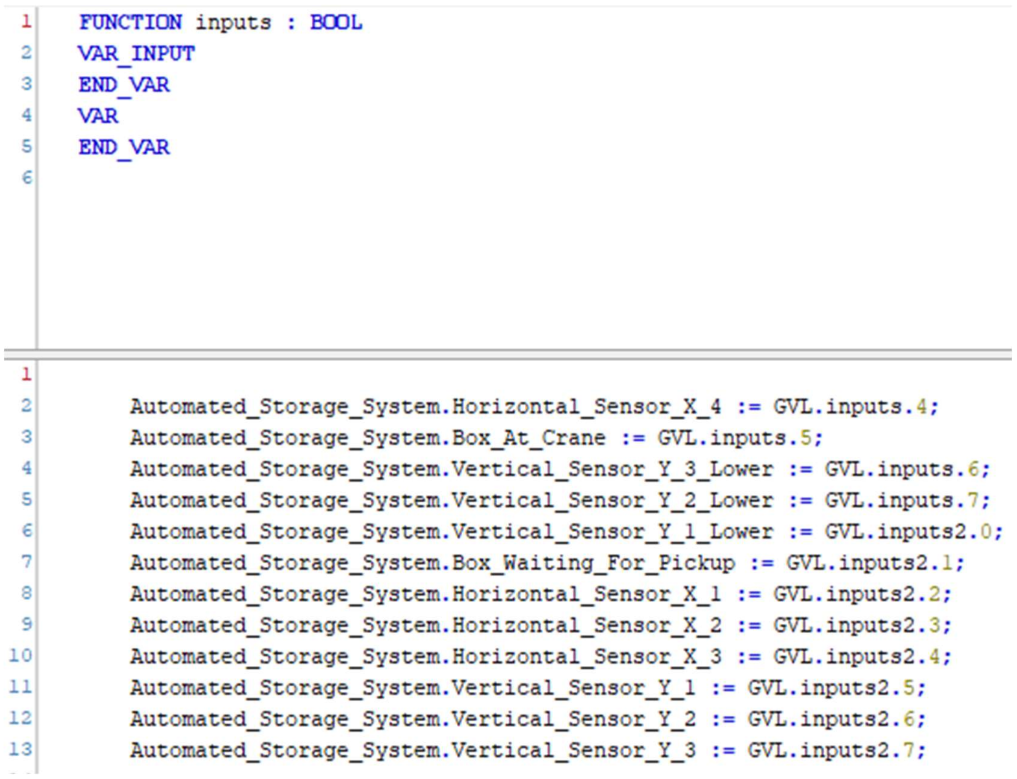
Figure 27. Structure of a byte



```
FUNCTION outputs : BOOL
VAR_INPUT
END_VAR
VAR
END_VAR
```

```


GVL.outputs2.7 := Automated_Storage_System.Move_Crane_Down;
GVL.outputs2.6 := Automated_Storage_System.Move_Crane_Right;
GVL.outputs2.5 := Automated_Storage_System.Move_Crane_Backward;
GVL.outputs2.4 := Automated_Storage_System.Move_Crane_Forward;
GVL.outputs2.3 := Automated_Storage_System.Move_Crane_Left;
GVL.outputs2.2 := Automated_Storage_System.Move_Crane_Up;
GVL.outputs2.1 := Automated_Storage_System.Conveyour_Running;
```

Figure 28. Function outputs, mapping global variables outputs2 to application variables in Codesys



```
FUNCTION inputs : BOOL
VAR_INPUT
END_VAR
VAR
END_VAR
```

```

    Automated_Storage_System.Horizontal_Sensor_X_4 := GVL.inputs.4;
    Automated_Storage_System.Box_At_Crane := GVL.inputs.5;
    Automated_Storage_System.Vertical_Sensor_Y_3_Lower := GVL.inputs.6;
    Automated_Storage_System.Vertical_Sensor_Y_2_Lower := GVL.inputs.7;
    Automated_Storage_System.Vertical_Sensor_Y_1_Lower := GVL.inputs2.0;
    Automated_Storage_System.Box_Waiting_For_Pickup := GVL.inputs2.1;
    Automated_Storage_System.Horizontal_Sensor_X_1 := GVL.inputs2.2;
    Automated_Storage_System.Horizontal_Sensor_X_2 := GVL.inputs2.3;
    Automated_Storage_System.Horizontal_Sensor_X_3 := GVL.inputs2.4;
    Automated_Storage_System.Vertical_Sensor_Y_1 := GVL.inputs2.5;
    Automated_Storage_System.Vertical_Sensor_Y_2 := GVL.inputs2.6;
    Automated_Storage_System.Vertical_Sensor_Y_3 := GVL.inputs2.7;
```

Figure 29.Function inputs, mapping global variables inputs and inputs2 to application variables in Codesys

To simplify above I/O mapping, Figure 30 illustrates I/O mapping process to a Simumatik’s virtual PLC16DIO4AIO controller’s input and output ports.

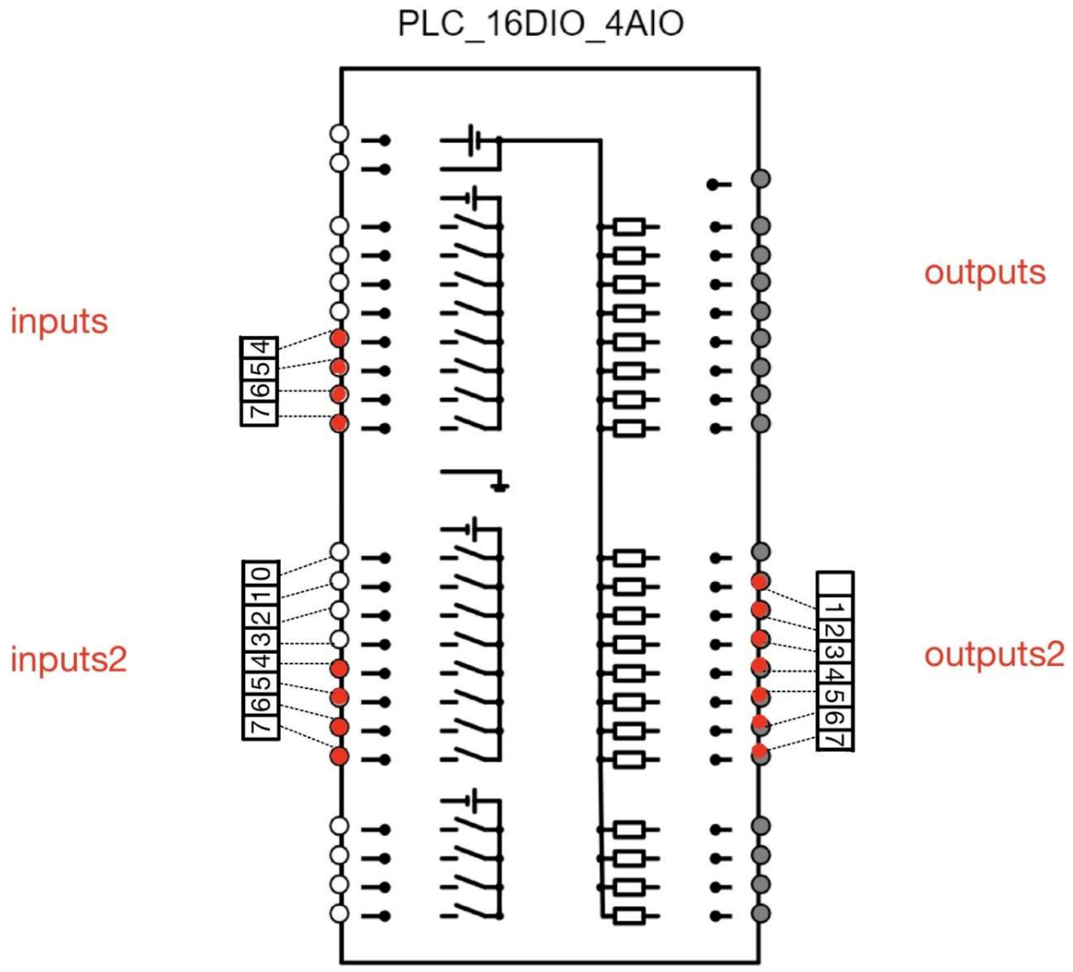


Figure 30. Simumatik's PLC16DIO4AIO controller, mapping process to a GVL variables

5.3.2 Plc code development

The development of plc application for a stacker crane was particularly difficult. There is very limited amount of information available on the internet about programming stacker cranes. As a result, only a basic sorting application was developed. This application sorts pallets into the stacker crane's storage, arranging them from left to right, switching to the next row once the current row is filled.

The complete plc program contains Main_program, inputs, outputs and the Automated_Storage_System (Figure 31) where all logic is executed. The main program functionality involves executing the inputs' function, where it reads the inputs. Afterwards Automated_Storage_System program is executed, later, the outputs' function writes the outputs (Figure 32).

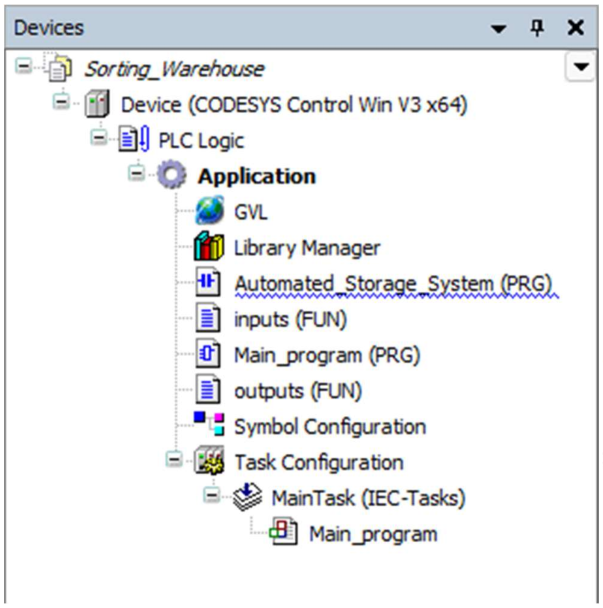
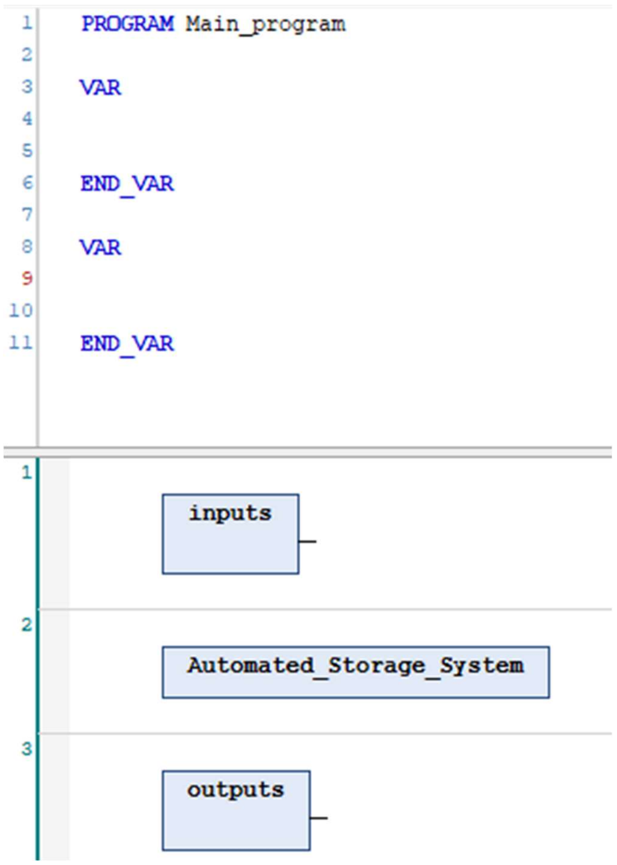


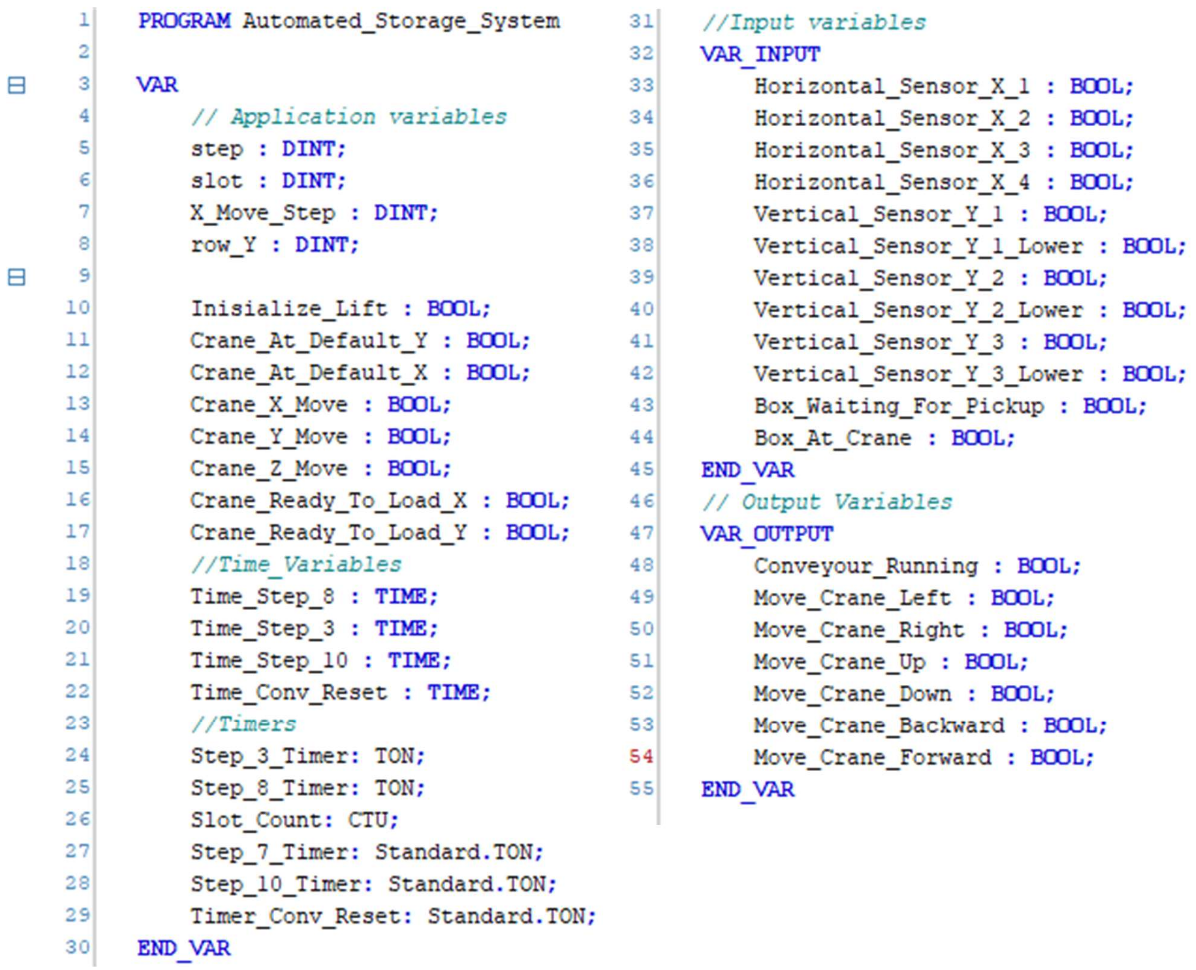
Figure 31. Structure of the Codesys project



```
PROGRAM Main_program

VAR


END_VAR

VAR


END_VAR
```

Figure 32. Contents of the main program in Codesys

Automated_Storage_System is the main logic program responsible for the behavior of the storage system part of the project. Presented below in Figure 33, are the variables of the Automated_Storage_System program.



```
PROGRAM Automated_Storage_System

VAR
    // Application variables
    step : DINT;
    slot : DINT;
    X_Move_Step : DINT;
    row_Y : DINT;

    Inisialize_Lift : BOOL;
    Crane_At_Default_Y : BOOL;
    Crane_At_Default_X : BOOL;
    Crane_X_Move : BOOL;
    Crane_Y_Move : BOOL;
    Crane_Z_Move : BOOL;
    Crane_Ready_To_Load_X : BOOL;
    Crane_Ready_To_Load_Y : BOOL;
    //Time_Variables
    Time_Step_8 : TIME;
    Time_Step_3 : TIME;
    Time_Step_10 : TIME;
    Time_Conv_Reset : TIME;
    //Timers
    Step_3_Timer: TON;
    Step_8_Timer: TON;
    Slot_Count: CTU;
    Step_7_Timer: Standard.TON;
    Step_10_Timer: Standard.TON;
    Timer_Conv_Reset: Standard.TON;
END_VAR
//Input variables
VAR_INPUT
    Horizontal_Sensor_X_1 : BOOL;
    Horizontal_Sensor_X_2 : BOOL;
    Horizontal_Sensor_X_3 : BOOL;
    Horizontal_Sensor_X_4 : BOOL;
    Vertical_Sensor_Y_1 : BOOL;
    Vertical_Sensor_Y_1_Lower : BOOL;
    Vertical_Sensor_Y_2 : BOOL;
    Vertical_Sensor_Y_2_Lower : BOOL;
    Vertical_Sensor_Y_3 : BOOL;
    Vertical_Sensor_Y_3_Lower : BOOL;
    Box_Waiting_For_Pickup : BOOL;
    Box_At_Crane : BOOL;
END_VAR
// Output Variables
VAR_OUTPUT
    Conveyour_Running : BOOL;
    Move_Crane_Left : BOOL;
    Move_Crane_Right : BOOL;
    Move_Crane_Up : BOOL;
    Move_Crane_Down : BOOL;
    Move_Crane_Backward : BOOL;
    Move_Crane_Forward : BOOL;
END_VAR
```

Figure 33. Code snipped of the Automated_Storage_System's variables

Variables step, slot, X_Move_Step and row_Y are defined as DINT. DINT has 32 bits.[9] Each bit can be set on and off. The main idea is to use just one variable instead of many. A variable step is used to count steps. A variable slot is used to count items and arrange them into correct slot. X_Move_Step is supplementary variable, utilized within the Crane_X_Move contact as a counter for tracking at which horizontal sensor the crane should stop, while moving along the X axis.

The program utilizes various timers to determine when to cease operations of the component that does not rely on any sensors. Input variables represent variables that are linked to the PLC's input components, such as sensors. Output variables are linked to PLC actuators, such as motor contacts and crane actuators.

Below is the first network of the Automated_Storage_System program (Figure 34). Initialize_Lift is responsible for initiating the stacker crane's lifting sequence. In this network when Initialize_Lift sequence is activated, Conveyor_Running coil is reset. Timer_Conv_Reset provides a 5 second delay before executing the reset operation.

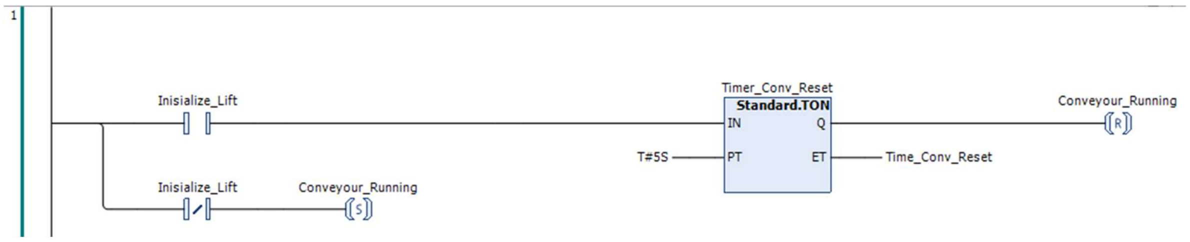


Figure 34. Automated_Storage_System ladder logic first network

The second network of the program (Figure 35) is responsible for counting incoming items. Box_Waiting_For_Pickup variable is connected to a photoelectric sensor (Sensor_Last_Belt) which is located at the last conveyor. The contact provides a rising edge pulse rather than just activating the contact.

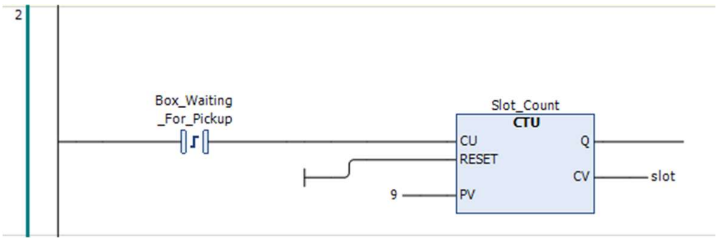


Figure 35. Automated_Storage_System Lader logic second network

The third network of the program (Figure 36) is responsible for determining the appropriate Y-axis row assignment. The storage has 3 slots horizontally. Slot 4 indicates that the crane should fill slots in the second row, while slot 7 indicates that the crane should fill slots in the third row. When row_Y.3 activated, row_Y.2 deactivated.

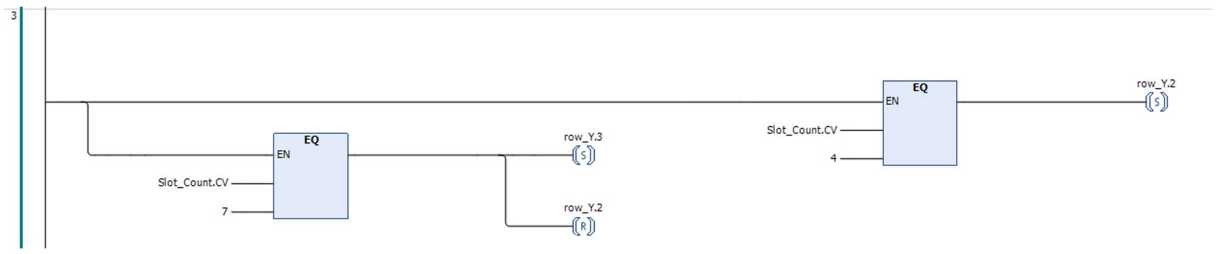


Figure 36. The third network of the Automated_Storage_System's ladder logic

The fourth network (Figure 37) upon detecting item sends a rising edge pulse to set Initialize_Lift sequence.

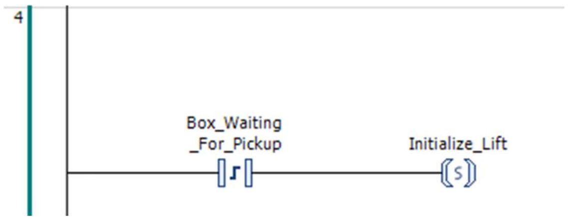


Figure 37. The third network of the Automated_Storage_System's ladder logic

The fifth network (Appendix 3, network 5) is a complete 12 step lifting and storing item sequence. I decided that using graphs to describe the program functionality would be the best approach. The graph is illustrated below in Figure 38 and Figure 39.

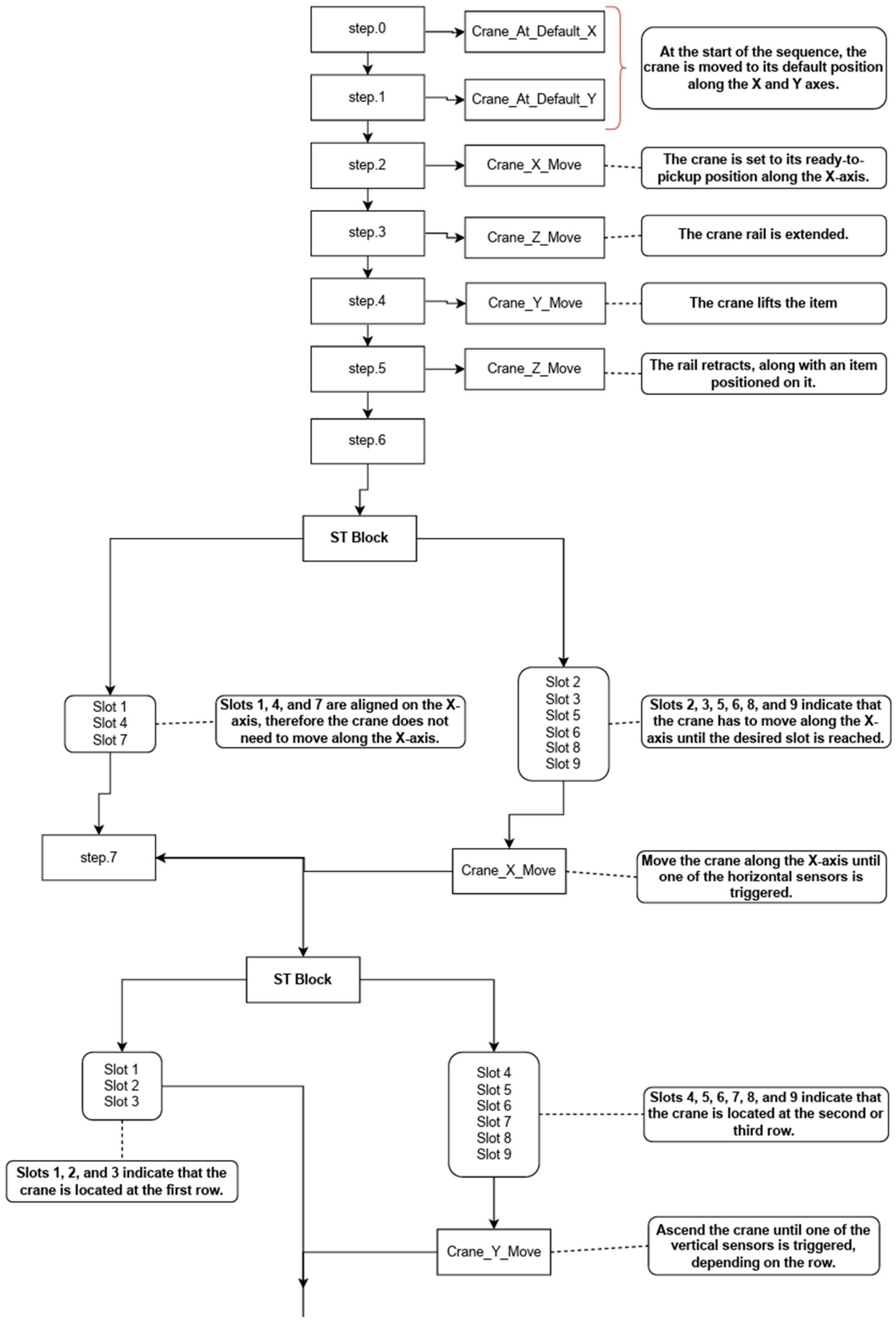

step.0
Crane_At_Default_X
step.1
Crane_At_Default_Y
At the start of the sequence, the crane is moved to its default position along the X and Y axes.
step.2
Crane_X_Move
The crane is set to its ready-to-pickup position along the X-axis.
step.3
Crane_Z_Move
The crane rail is extended.
step.4
Crane_Y_Move
The crane lifts the item
step.5
Crane_Z_Move
The rail retracts, along with an item positioned on it.
step.6
ST Block
Slot 1 Slot 4 Slot 7
Slots 1, 4, and 7 are aligned on the X-axis, therefore the crane does not need to move along the X-axis.
Slot 2 Slot 3 Slot 5 Slot 6 Slot 8 Slot 9
Slots 2, 3, 5, 6, 8, and 9 indicate that the crane has to move along the X-axis until the desired slot is reached.
step.7
Crane_X_Move
Move the crane along the X-axis until one of the horizontal sensors is triggered.
ST Block
Slot 1 Slot 2 Slot 3
Slots 1, 2, and 3 indicate that the crane is located at the first row.
Slot 4 Slot 5 Slot 6 Slot 7 Slot 8 Slot 9
Slots 4, 5, 6, 7, 8, and 9 indicate that the crane is located at the second or third row.
Crane_Y_Move
Ascend the crane until one of the vertical sensors is triggered, depending on the row.

Figure 38. Graph of the 12-step lifting and storing item sequence part 1

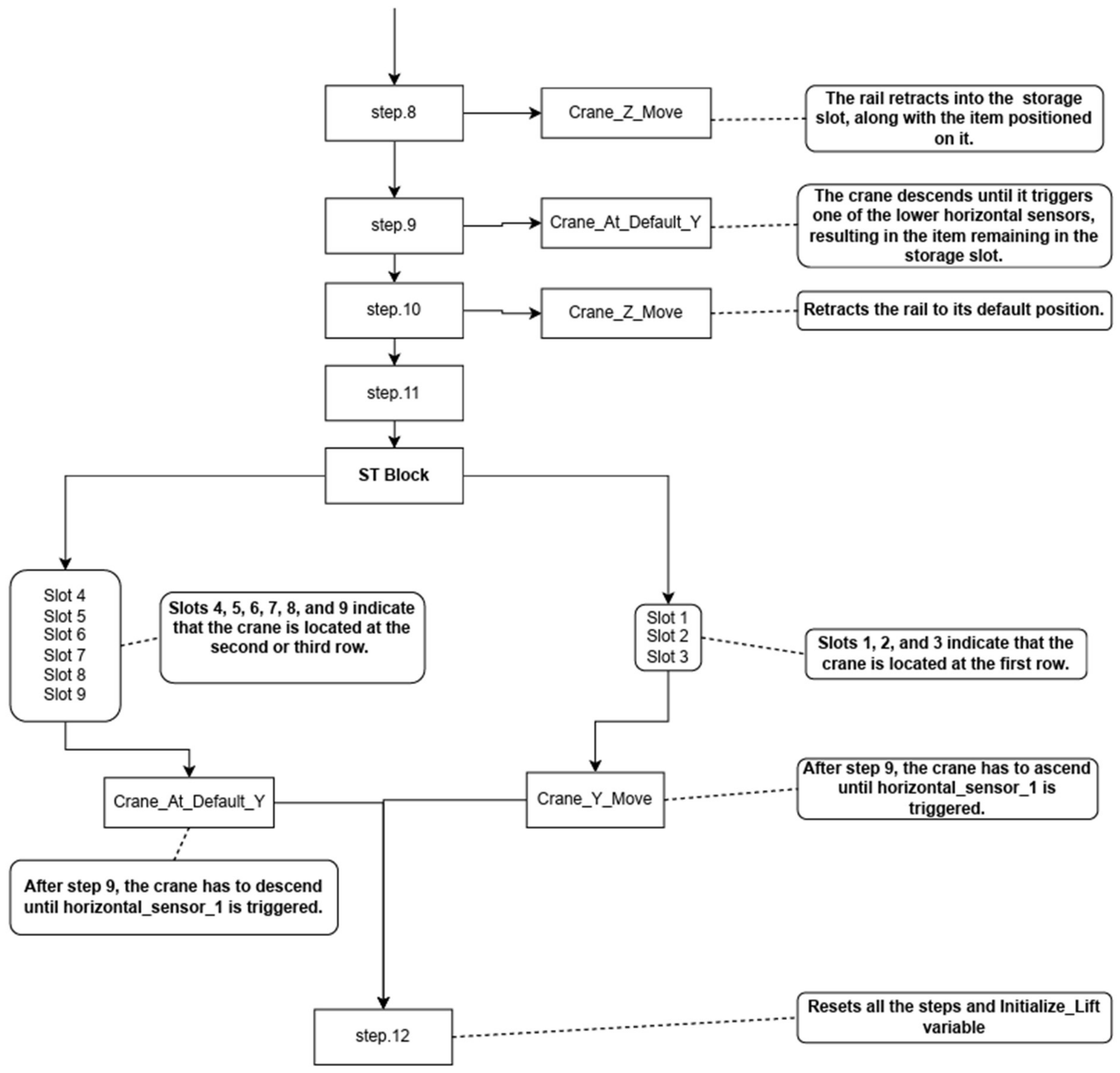

step.8
Crane_Z_Move
The rail retracts into the storage slot, along with the item positioned on it.
step.9
Crane_At_Default_Y
The crane descends until it triggers one of the lower horizontal sensors, resulting in the item remaining in the storage slot.
step.10
Crane_Z_Move
Retracts the rail to its default position.
step.11
ST Block
Slot 4 Slot 5 Slot 6 Slot 7 Slot 8 Slot 9
Slots 4, 5, 6, 7, 8, and 9 indicate that the crane is located at the second or third row.
Slot 1 Slot 2 Slot 3
Slots 1, 2, and 3 indicate that the crane is located at the first row.
Crane_At_Default_Y
After step 9, the crane has to descend until horizontal_sensor_1 is triggered.
Crane_Y_Move
After step 9, the crane has to ascend until horizontal_sensor_1 is triggered.
step.12
Resets all the steps and Initialize_Lift variable

Figure 39. Graph of the 12-step lifting and storing item sequence part 2

Contacts Crane_At_Default_X, Crane_At_Default_Y, Crane_Z_Move, Crane_X_Move and Crane_Y_Move have functionality to determine the crane's movement direction, based on the current step in the fifth network.

The sixth network (Figure 40) is responsible for moving the crane to the left, its default position along the X axis. It also sets contact Crane_Ready_To_Load_X, so the program knows that the crane is at its default position along the X axis.

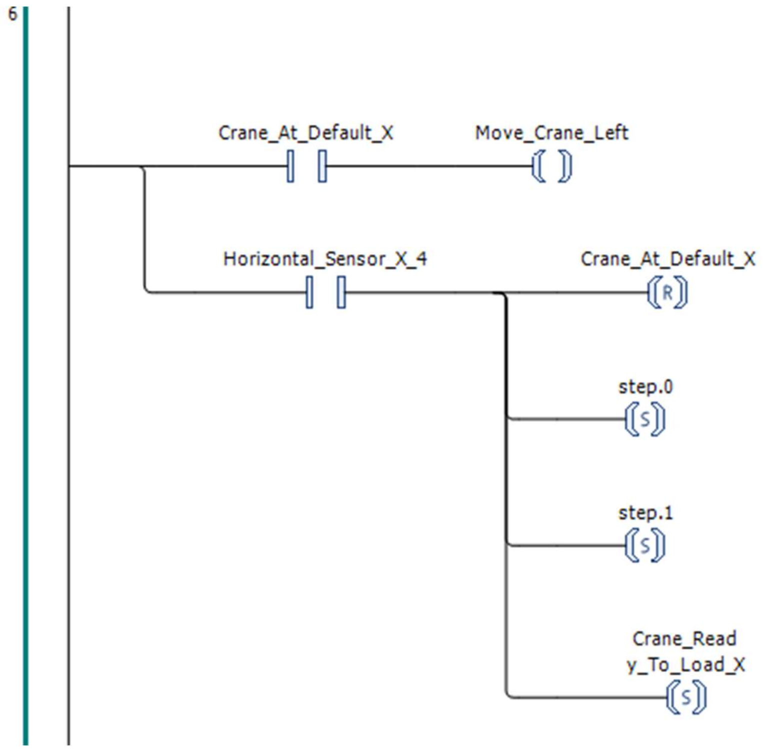


Figure 40. The sixth network of the Automated_Storage_System's ladder logic

The seventh network (Figure 41) is responsible for moving the crane down, its default position along the Y axis. When the action is complete contact Crane_Ready_To_Load_Y is set. When both Crane_Ready_To_Load_X and Crane_Ready_To_Load_Y are set, The crane is ready to initiate pickup operation.

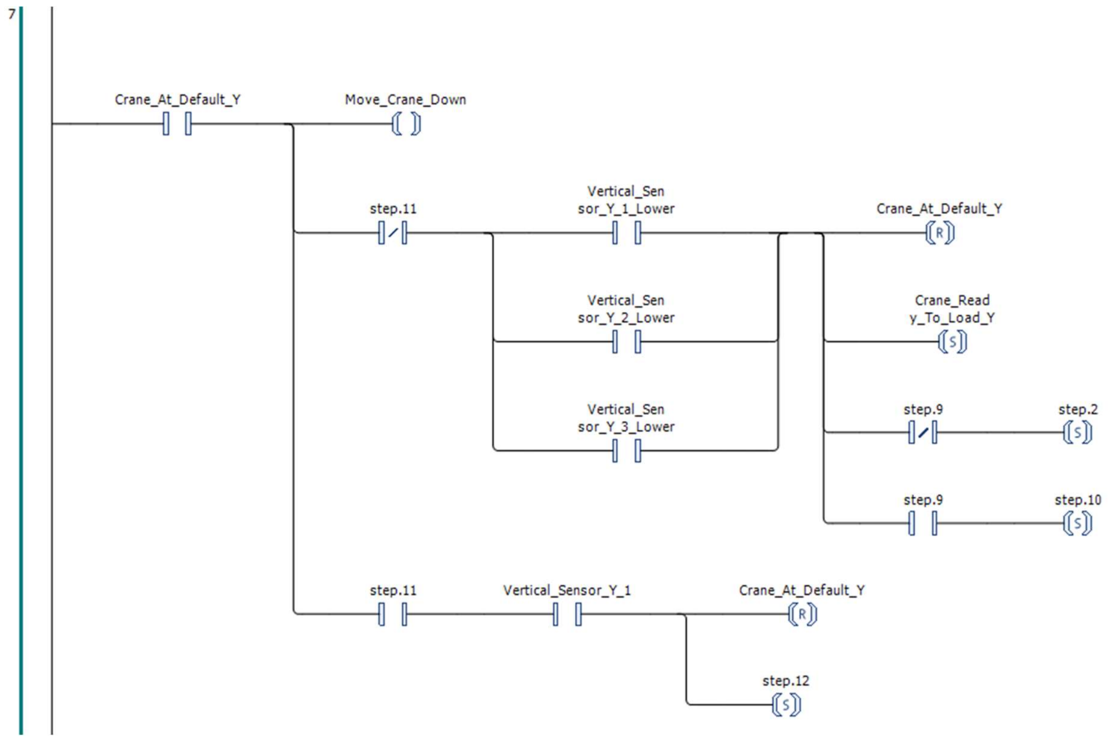


Figure 41. The seventh network of the Automated_Storage_System's ladder logic

The eighth network (Figure 42) is responsible for the crane’s rail, extension, and retraction. The rail does not have any sensors, therefore timers are used to determine the appropriate length of extension or retraction.

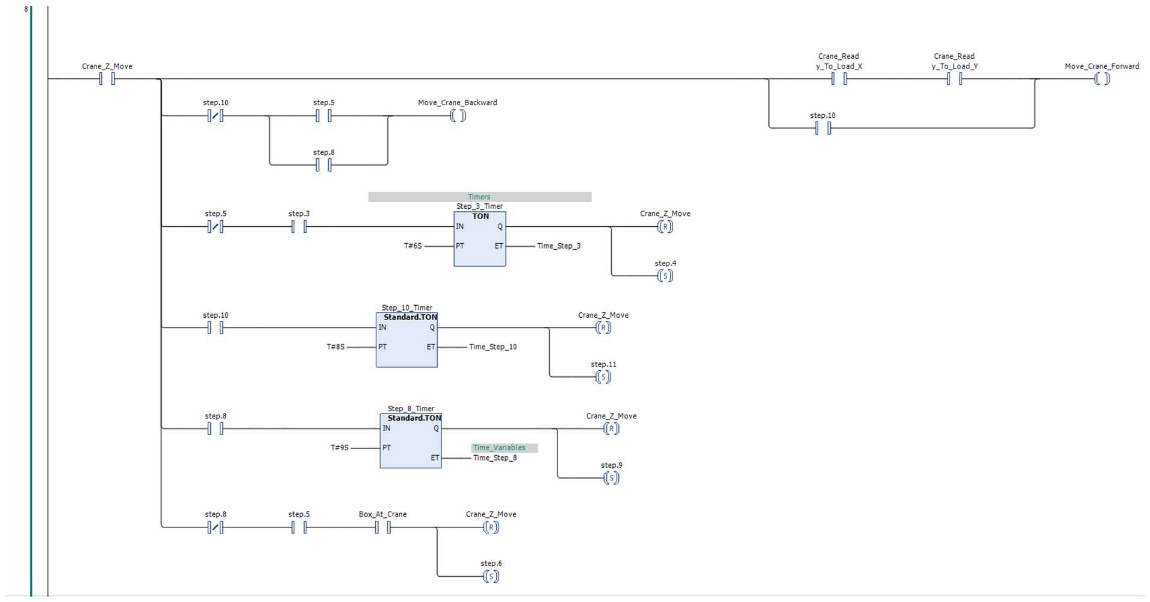


Figure 42. The eighth network of the Automated_Storage_System's ladder logic

The ninth network (Figure 43) is responsible for crane movement to the right. Supplementary variable X_Move_Step helps to determine at which horizontal sensor the crane stops.

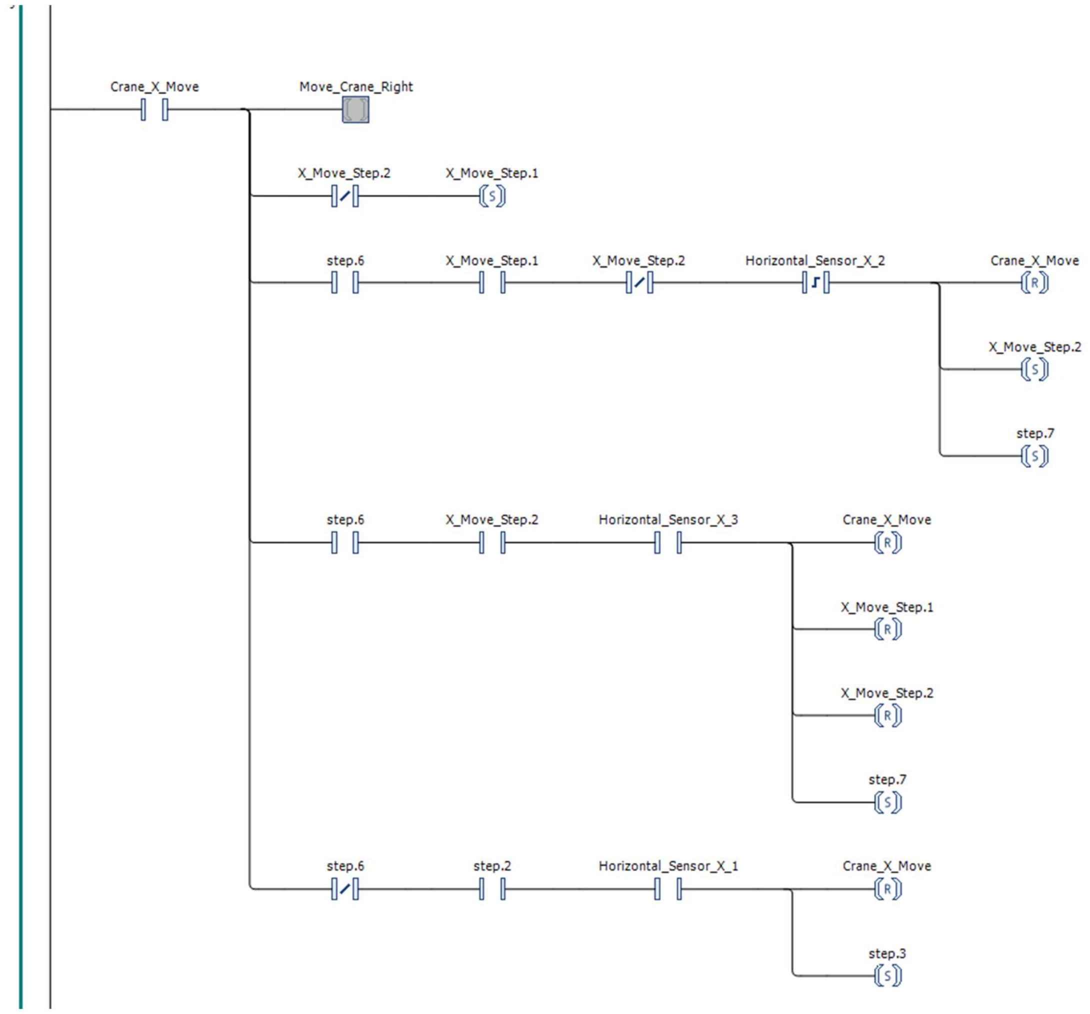


Figure 43. The ninth network of the Automated_Storage_System's ladder logic

The final tenth network (Figure 44) is responsible for the crane's upward movement. Variable row_Y helps to determine the current row within the sequence.

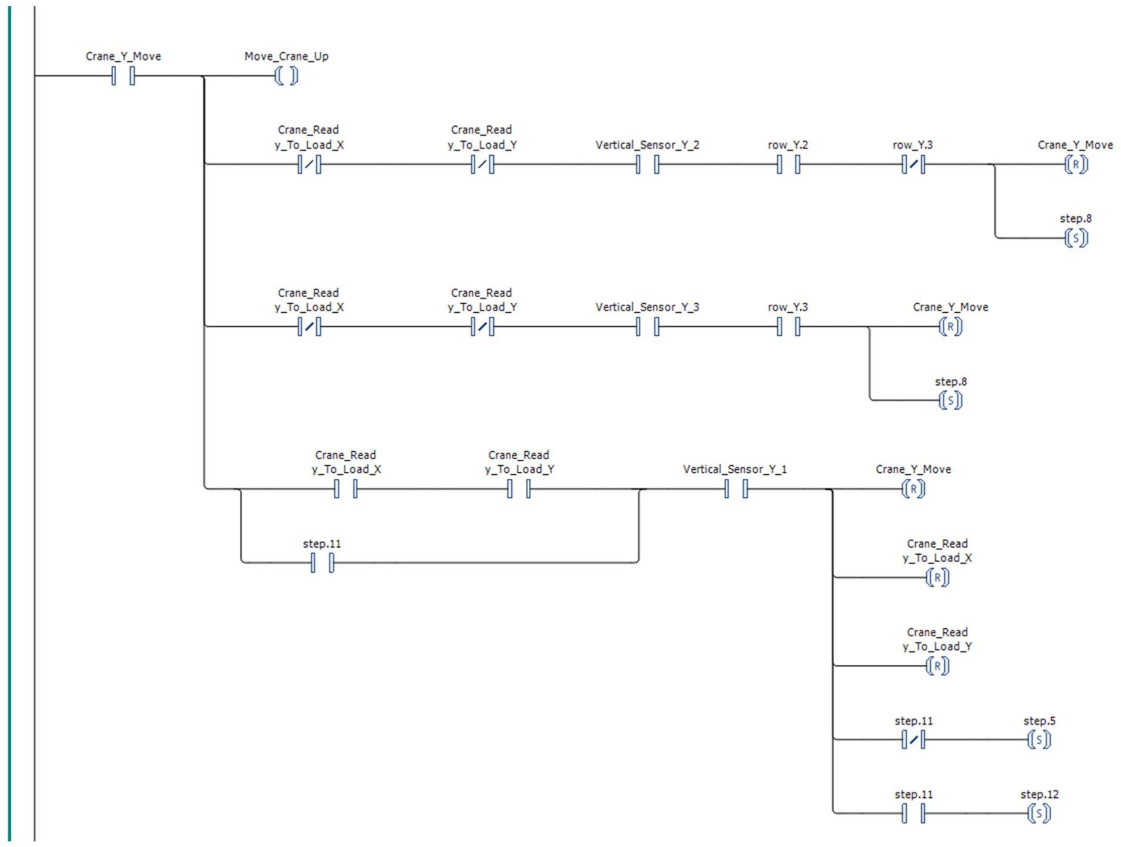
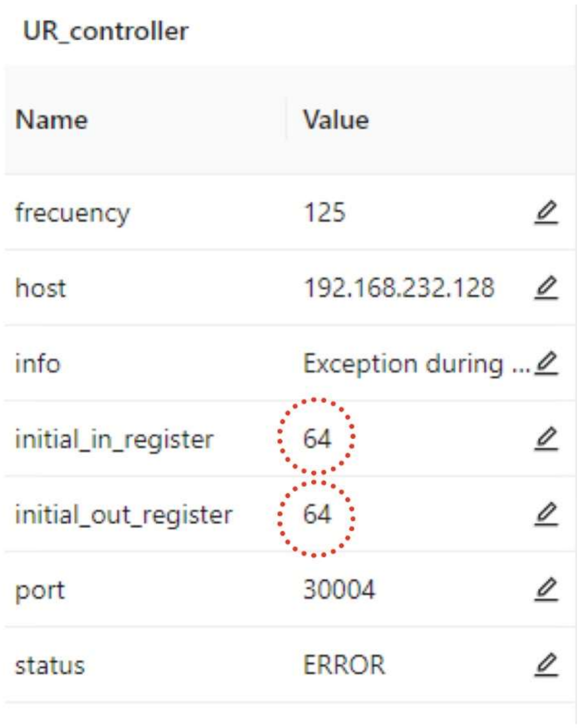


Figure 44. The tenth network of the Automated_Storage_System's lader logic

5.4 Programming portion using URsim

5.4.1 I/O Setup

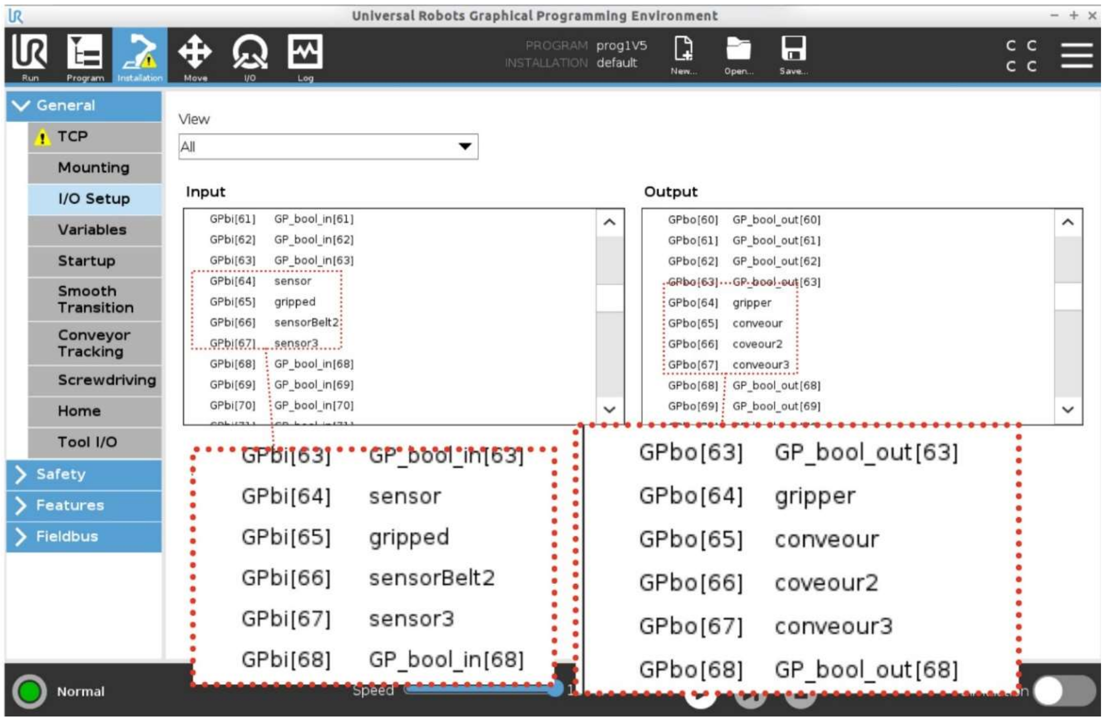
The virtual implementation of the UR_controller is able to utilize input and output registers exposed to the external devices from Ursim. Reading of registers begins with value specified in initial_in_register and initial_out_register, which is 64 (Figure 45). [13]



UR_controller

Name	Value
frecuency	125
host	192.168.232.128
info	Exception during ...
initial_in_register	64
initial_out_register	64
port	30004
status	ERROR

Figure 45. Simumatik's UR controller configuration panel



View: All

Input

GPbi[61]	GP_bool_in[61]
GPbi[62]	GP_bool_in[62]
GPbi[63]	GP_bool_in[63]
GPbi[64]	sensor
GPbi[65]	gripped
GPbi[66]	sensorBelt2
GPbi[67]	sensor3
GPbi[68]	GP_bool_in[68]
GPbi[69]	GP_bool_in[69]
GPbi[70]	GP_bool_in[70]

Output

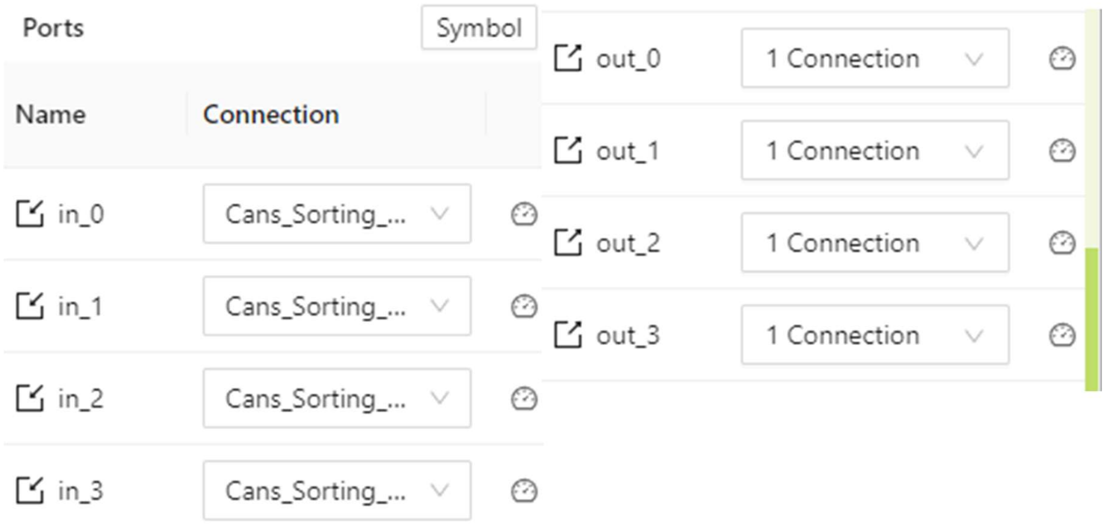
GPbo[60]	GP_bool_out[60]
GPbo[61]	GP_bool_out[61]
GPbo[62]	GP_bool_out[62]
GPbo[63]	GP_bool_out[63]
GPbo[64]	gripper
GPbo[65]	conveour
GPbo[66]	coveour2
GPbo[67]	conveour3
GPbo[68]	GP_bool_out[68]
GPbo[69]	GP_bool_out[69]

GPbi[63]	GP_bool_in[63]
GPbi[64]	sensor
GPbi[65]	gripped
GPbi[66]	sensorBelt2
GPbi[67]	sensor3
GPbi[68]	GP_bool_in[68]

GPbo[63]	GP_bool_out[63]
GPbo[64]	gripper
GPbo[65]	conveour
GPbo[66]	coveour2
GPbo[67]	conveour3
GPbo[68]	GP_bool_out[68]

Figure 46. Universal Robots graphical programming environment

The register “conveyour” activates a small conveyor that is transporting cans, while “conveyor2” is responsible for transporting the can’s rack. The register “conveyour3” is responsible for the main conveyor transporting a wooden pallet. The register “sensor” reads the output from a sensor that is located at the end of the conveyor which is transporting cans. Meanwhile, the register “sensorBelt2” reads the output from a sensor that is located at the end of the conveyor which is transporting the can’s rack. Finally, the register “sensor3” reads the output from a sensor that is located at the end of the conveyor which is transporting the wooden pallet. All the above registers are illustrated in Figure 46. The registers are assigned to UR controller’s input and output ports (Figure 47).

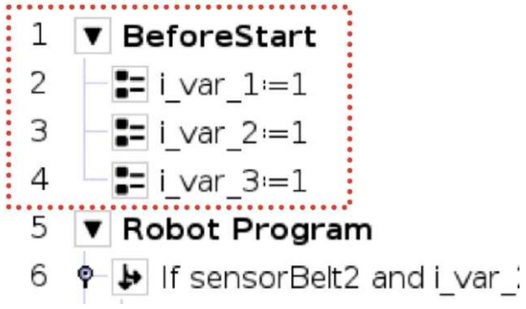


Name	Connection
in_0	Cans_Sorting_...
in_1	Cans_Sorting_...
in_2	Cans_Sorting_...
in_3	Cans_Sorting_...
out_0	1 Connection
out_1	1 Connection
out_2	1 Connection
out_3	1 Connection

Figure 47. UR controller's ports configuration panel in Simumatik

5.4.2 Code

The UR 5 robot’s program consists of five parts. Part “BeforeStart” (Figure 48) runs at the beginning of the code. This part resets supplementary variables, that used to identify which step the program must execute.

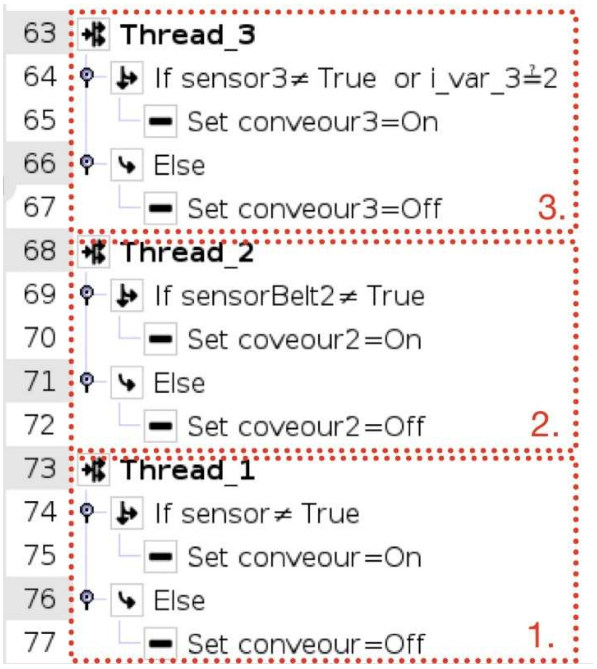


```
▼ BeforeStart
    i_var_1:=1
    i_var_2:=1
    i_var_3:=1
▼ Robot Program
    If sensorBelt2 and i_var_
```

Figure 48. Code snipped of UR 5 robot program from URsim

The program also has 3 threads (Figure 49). These threads run alongside the robot’s main program, allowing it to control conveyors while simultaneously operating the robot arm.

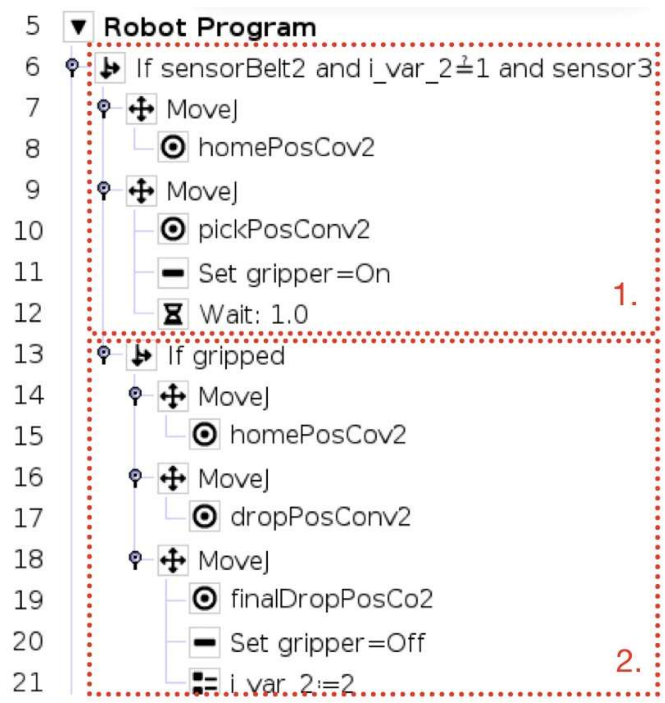
When an item triggers a sensor, Ursim deactivates the relay or contactor, therefore stopping the conveyor. The sensor also triggers the robot’s main program sorting sequence that lifts the items. Once the item has been lifted, the conveyor resumes operation until the sensor is triggered again. The Ursim program threads control three conveyors.



```
Thread_3
  If sensor3≠ True  or i_var_3≟2
    Set conveour3=On
  Else
    Set conveour3=Off
Thread_2
  If sensorBelt2≠ True
    Set coveour2=On
  Else
    Set coveour2=Off
Thread_1
  If sensor≠ True
    Set conveour=On
  Else
    Set conveour=Off
```

Figure 49. Code snipped of UR 5 robot program’s threads from URsim

The main program consists of two pickup sequences for handling materials. The first pickup sequence is responsible for lifting a can’s rack and securely placing it on a wooden pallet. To begin the first pick up sequence the following conditions must be met. A wooden pallet triggers the sensor 3 and the can’s rack triggers sensor 2, stopping both conveyors. Variable i_var_2 value must be one. After that the UR robot initiates the first pickup sequence. In this sequence the robot picks up the can’s rack and places it on the wooden pallet. In Figure 50 in stage one (1.) the robot moves to its default position, followed by a shift to the pickup position. The vacuum gripper then activates, clamping onto the rack. At stage two (2.), the robot returns to its default stance, lifting the rack. This is followed by a move to the drop position and, the final drop position where deactivating the vacuum gripper releases the item, allowing it to fall on the wooden pallet. At the end of the code block variable i_var_2 value is changed to two. This allows the second sequence to proceed.

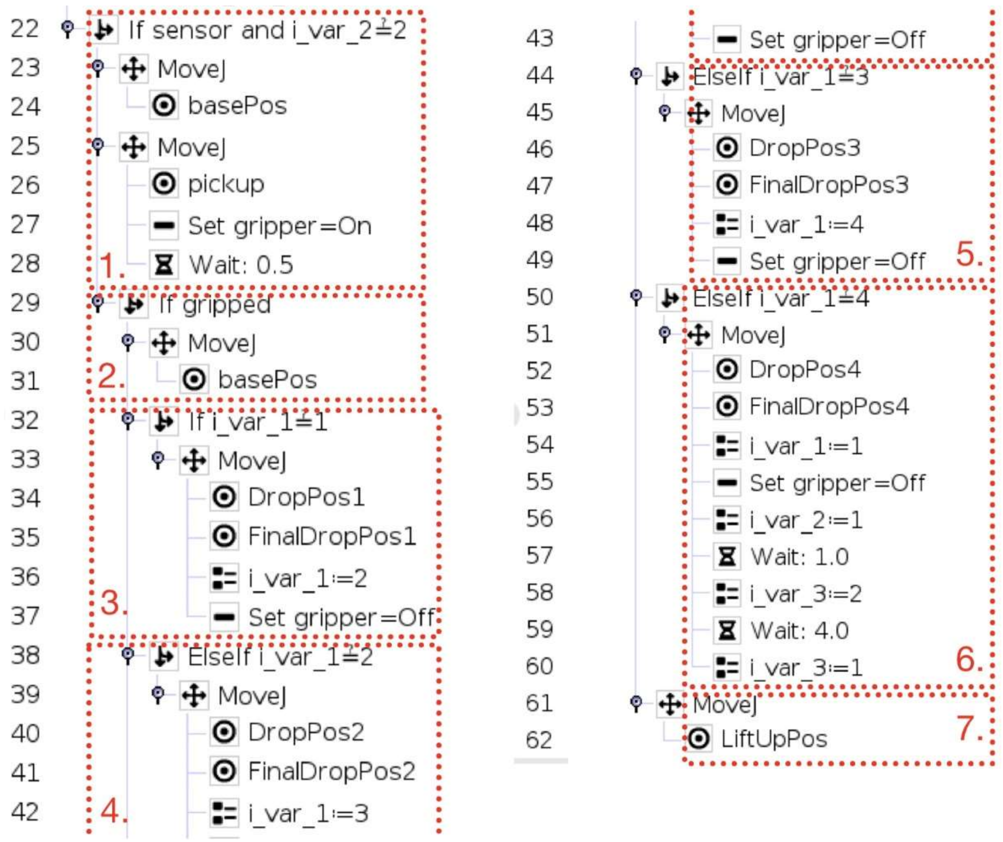


```
Robot Program
  If sensorBelt2 and i_var_2≟1 and sensor3
    MoveJ
      homePosCov2
    MoveJ
      pickPosConv2
      Set gripper=On
      Wait: 1.0                              1.
    If gripped
      MoveJ
        homePosCov2
      MoveJ
        dropPosConv2
      MoveJ
        finalDropPosCo2
        Set gripper=Off                      2.
        i_var_2:=2
```

Figure 50. Code snipped of UR 5 robot is main program from UR-sim

The following conditions initiate the second sequence. Approaching Can triggers the sensor, located at the end of the conveyor that transports the cans, and variable i_var_2 value must be equal to two after completing the first sequence. In Figure 50 in the stage one (1.) the robot moves to the default can retrieval position, designed for picking up cans, followed by a shift to the pickup position. At the stage two (2.), (Figure 50), The Vacuum gripper then activates, clamping onto the can, and moving into the default can retrieval position, lifting a can. At the stage three (3.), (Figure 51) robot moves to the drop position and places the can into the rack. Stages four (4.), five (5.), and six (6.), (Figure 51) follow the same procedure as stage three (3.), but with adjustments to both, drop and final drop positions, to place cans into appropriate slots on the can rack.

Also, at the stage six (6.), (Figure 51) all variables are reset to ensure that the program can start the next cycle from the beginning. The variable's i_var_3 value is set to 2 for 4 seconds, ensuring that the wooden pallet moves further into a storage slot.
At the stage seven (7.), (Figure 51) the robot lifts, providing the space for leaving the wooden pallet.



```
If sensor and i_var_2≟2
  MoveJ
    basePos
  MoveJ
    pickup
    Set gripper=On
    Wait: 0.5
  If gripped
    MoveJ
      basePos
    If i_var_1≟1
      MoveJ
        DropPos1
        FinalDropPos1
        i_var_1≔2
        Set gripper=Off
    ElseIf i_var_1≟2
      MoveJ
        DropPos2
        FinalDropPos2
        i_var_1≔3
        Set gripper=Off
    ElseIf i_var_1≟3
      MoveJ
        DropPos3
        FinalDropPos3
        i_var_1≔4
        Set gripper=Off
    ElseIf i_var_1≟4
      MoveJ
        DropPos4
        FinalDropPos4
        i_var_1≔1
        Set gripper=Off
        i_var_2≔1
        Wait: 1.0
        i_var_3≔2
        Wait: 4.0
        i_var_3≔1
    MoveJ
      LiftUpPos
```

Figure 51. Code snipped of UR 5 robot is main program from UR-sim

6 RESULTS

At the beginning all the necessary software must be launched. In Simumatik, emulation process is initiated by pressing Start the emulation button (Figure 52) To connect to both Ursim and Codesys Control Win V3 soft plc, Simumatik's gateway is launched (Figure 52)

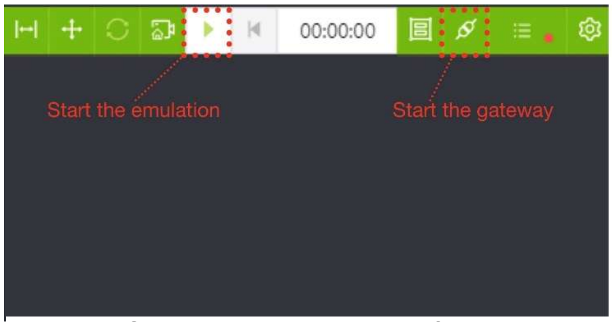


Figure 52. Simumatik's emulation interface

At the beginning all the necessary software must be launched. Ursim is launched through VMware. In Ursim's Linux operating system, UR 5 application must be launched (Figure 53) stage (1.) and the program loaded stage (2.). Then the robot must be powered on followed by the break release stage (3.). Finally, the program is initiated, and the robot moved to its default position stage (4.).

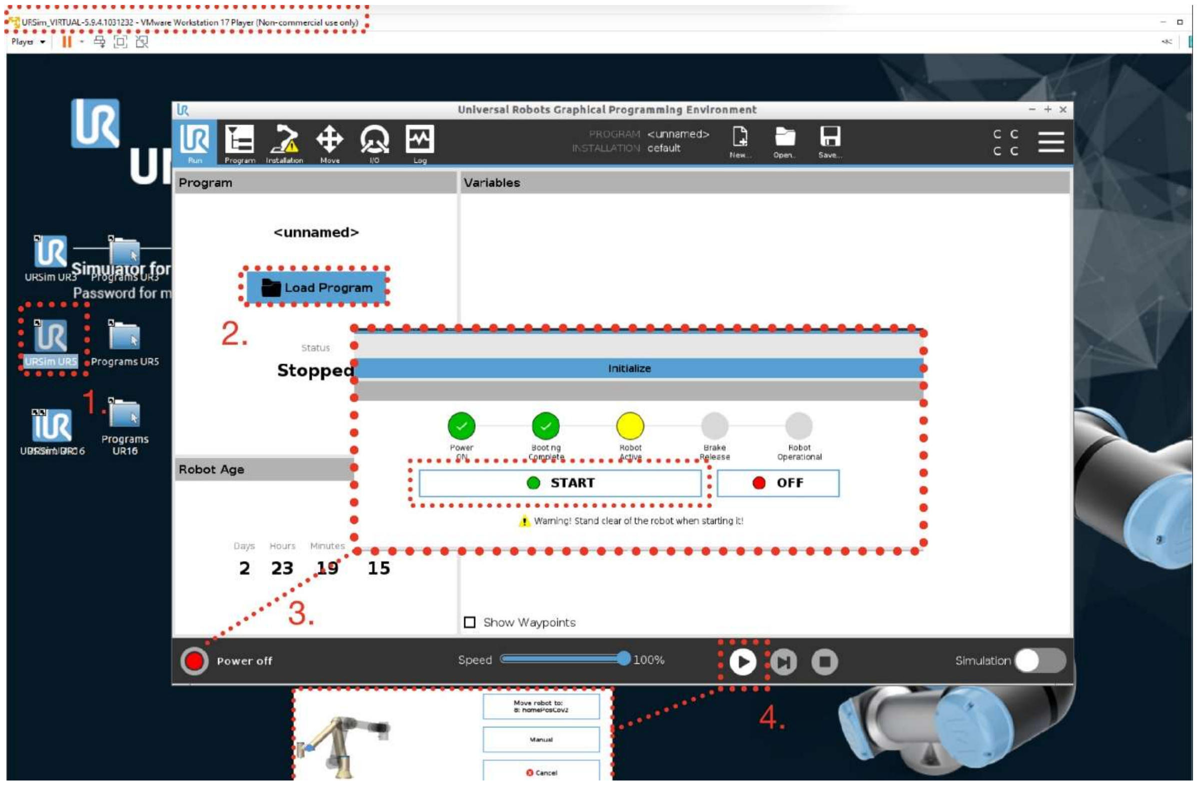
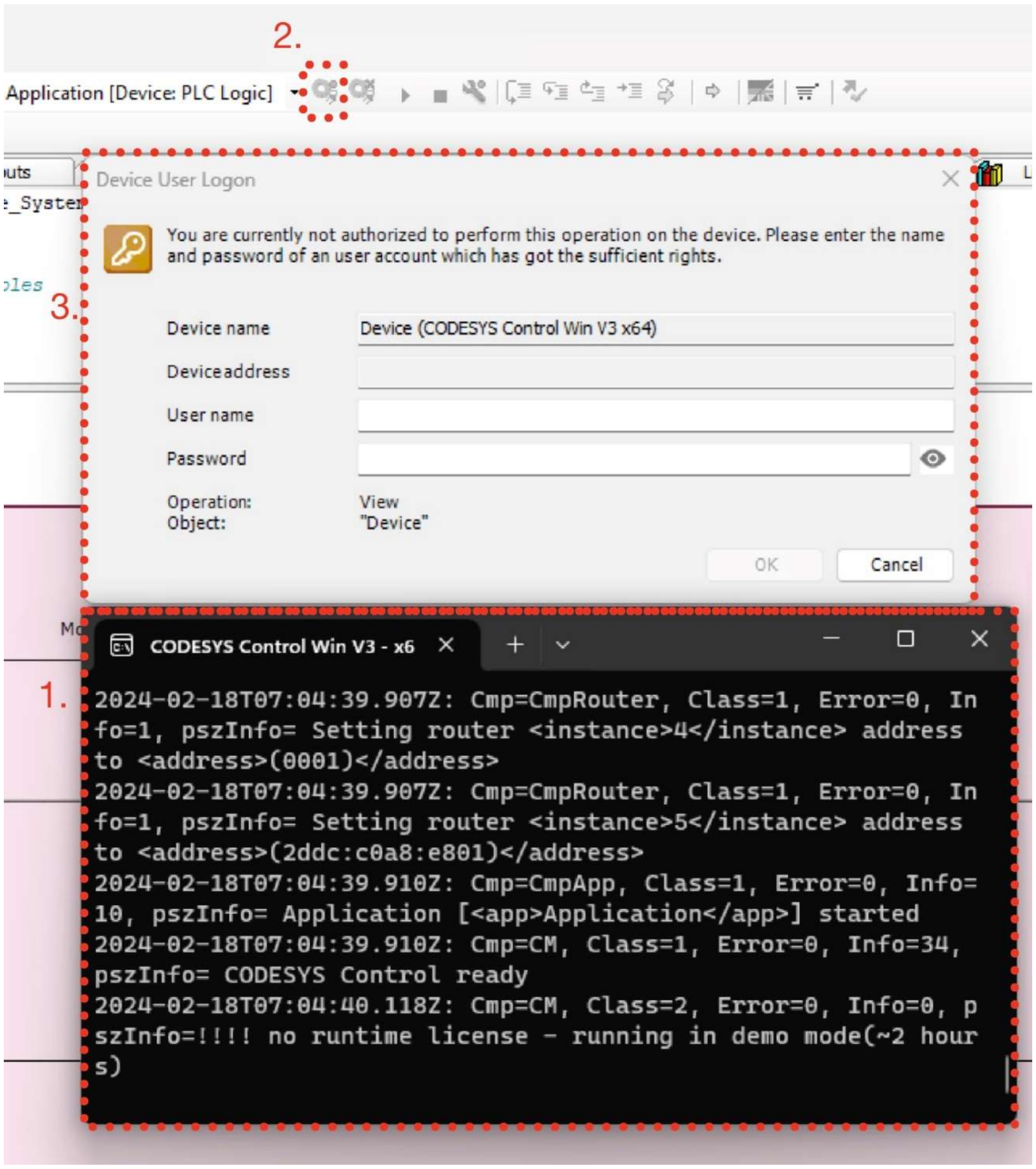


Figure 53. Universal Robot's graphical programming environment, loading a program and starting the ro-

Before launching Codesys, Soft PLC Codesys Control Win V3 is launched (Figure 54) stage 1. The Application is then downloaded and launched by logging into the Device, stage (2.) and (3.)



Device User Logon

You are currently not authorized to perform this operation on the device. Please enter the name and password of an user account which has got the sufficient rights.

Device name	Device (CODESYS Control Win V3 x64)
Deviceaddress	
User name	
Password	
Operation:	View
Object:	"Device"

OK Cancel

```
2024-02-18T07:04:39.907Z: Cmp=CmpRouter, Class=1, Error=0, In
fo=1, pszInfo= Setting router <instance>4</instance> address
to <address>(0001)</address>
2024-02-18T07:04:39.907Z: Cmp=CmpRouter, Class=1, Error=0, In
fo=1, pszInfo= Setting router <instance>5</instance> address
to <address>(2ddc:c0a8:e801)</address>
2024-02-18T07:04:39.910Z: Cmp=CmpApp, Class=1, Error=0, Info=
10, pszInfo= Application [<app>Application</app>] started
2024-02-18T07:04:39.910Z: Cmp=CM, Class=1, Error=0, Info=34,
pszInfo= CODESYS Control ready
2024-02-18T07:04:40.118Z: Cmp=CM, Class=2, Error=0, Info=0, p
szInfo=!!!! no runtime license - running in demo mode(~2 hour
s)
```

Figure 54. Codesys download and launch of the PLC application to a Codesys Control runtime

The UR robot can be precisely programmed to perform different actions. However, it is important that the item must be placed in a consistent position. The spawn point that generates the items lacks consistency. Conveyor guides were installed (Figure 55) to ensure that the item is in the exact same spot every time, for the robot’s pickup.

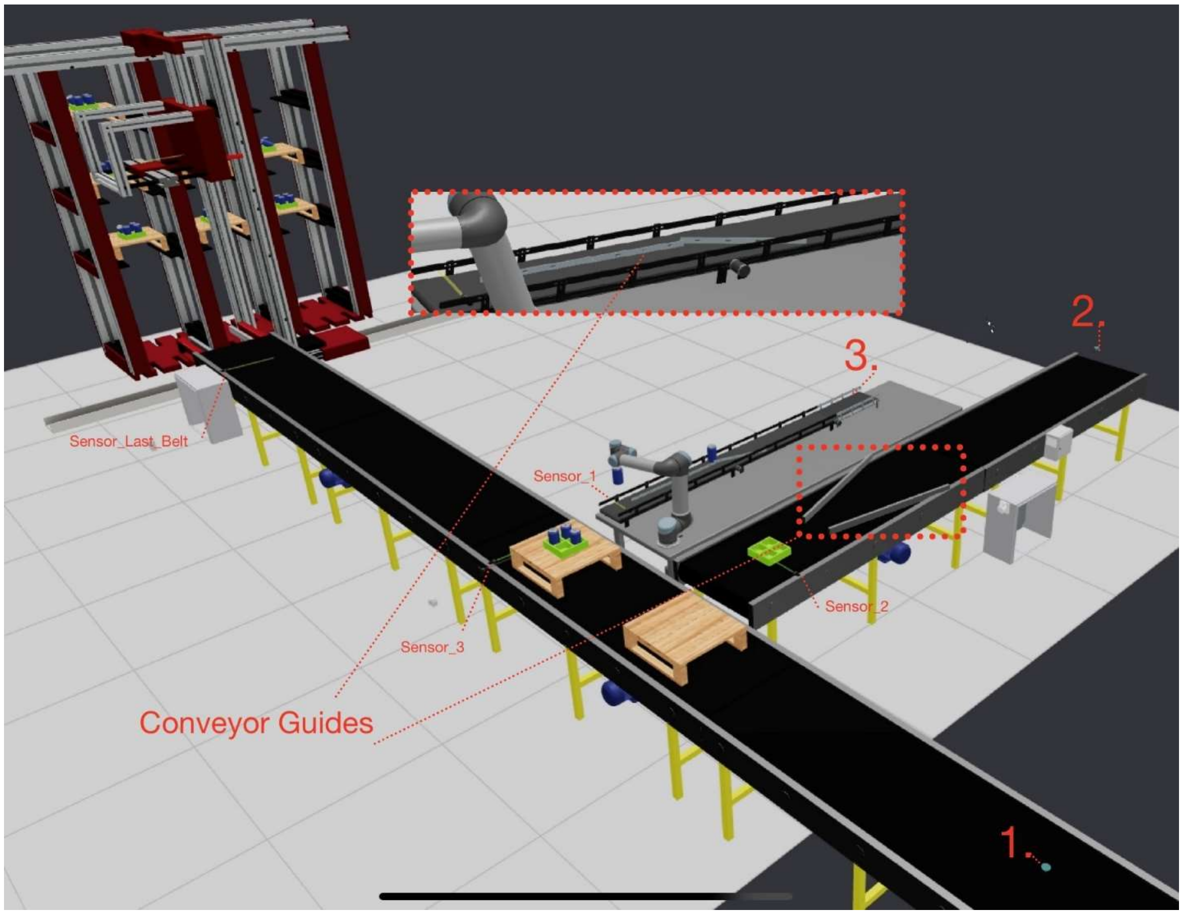


Figure 55. Automated Sorting System in full operation

When emulation begins items appear at the blue spawn points sensor (Figure 55) stages 1., 2., 3. Wooden pallet appears at spawn point 1, the can’s rack appears at the second (2.) spawn point and blue cans appear at the third (3.) spawn point. Items then move towards photoelectric sensors and upon triggering them the conveyor stops. Once both the wooden pallet and the can’s rack are in place, the UR robot lifts the can’s rack and places it on top of the wooden pallet (Figure 56).

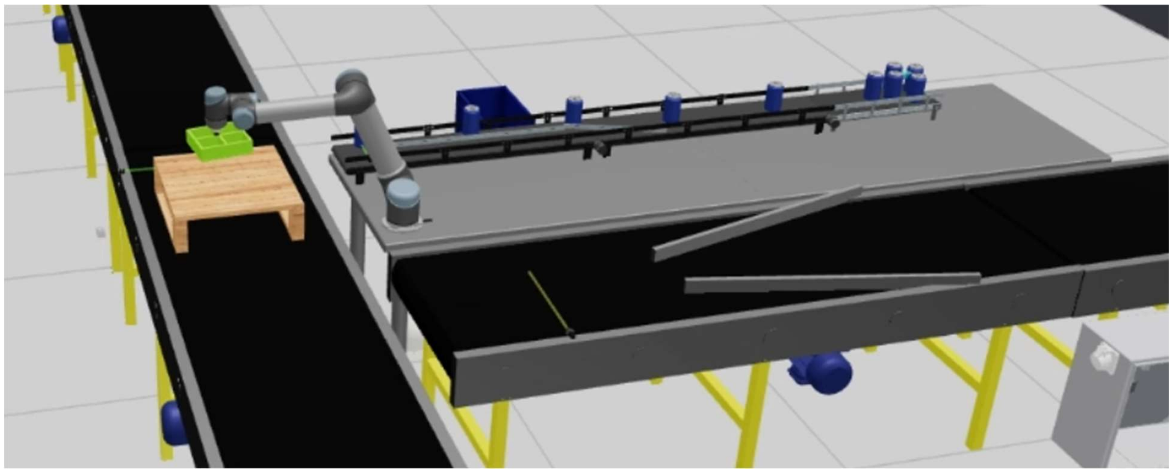


Figure 56. UR 5 robot lifting can’s rack in Simumatik

After that the UR 5 Robot initiates the can’s sorting sequence (Figure 57). In the Can sorting sequence, the robot lifts a can and inserts it in the appropriate slot of the can’s rack. Once the rack is full, the wooden pallet continues to move towards the stacker crane.

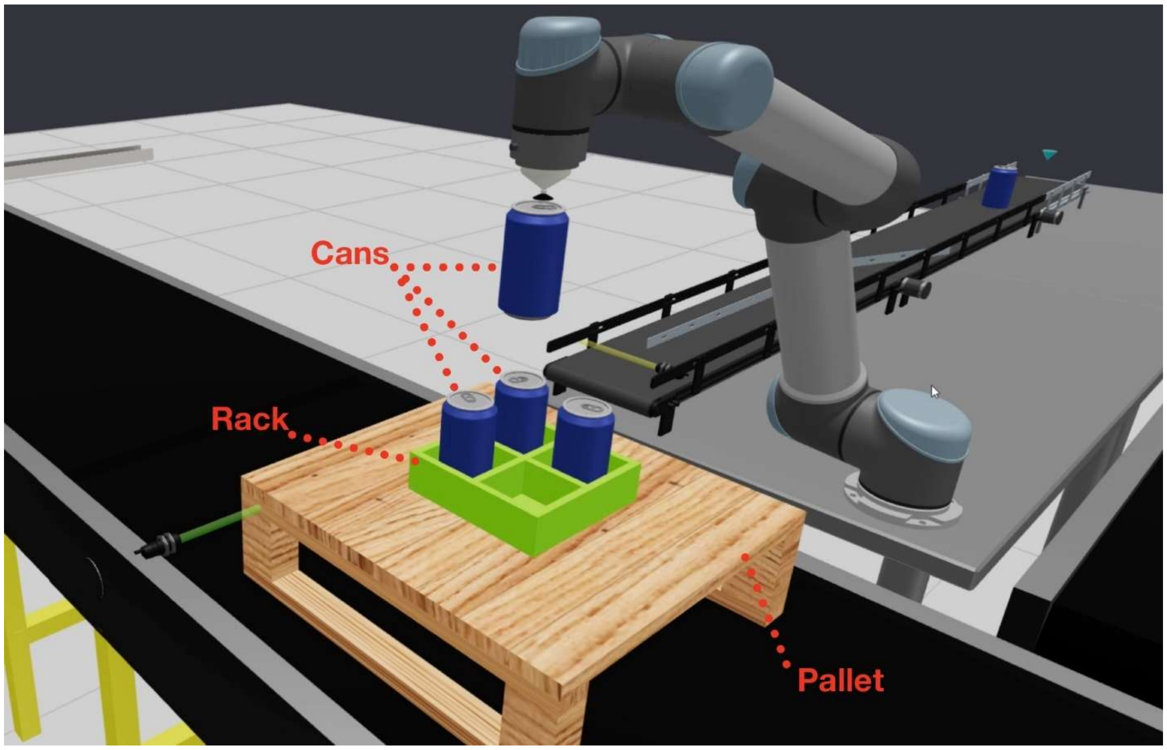


Figure 57. UR 5 inserting cans into the slots of the can’s rack

The Triggering sensor (“Sensor_Last_belt”, Figure 55) near the stacker crane initiates the Stacker crane’s pickup and loading sequence. In the pickup and loading sequence the stacker crane lifts the wooden pallet and loads it into the appropriate slot. During the simulation a total of 9 wooden pallets, 9 can’s racks and 36 cans are spawned and go through the sorting process. The stacker crane’s storage is 9 slots in total. The result is successful. Both the UR5 robot’s program and the stacker crane’s PLC program work as intended, effectively sorting the items at the conveyor, and placing them into the storage slots. The result can be seen in Figure 55.

7 CONCLUSION

The automated sorting system was successfully completed, integrating two different third-party software, Codesys Control Win V3 and URsim into a Simumatik. The system was assembled in a virtual environment. Both the UR5 robot's and the Stacker crane's plc programs were implemented. Electrical drawings of the system were created using Simumatik 2 D view feature.

During the project I delved into various automation technology topics. This project broadened my understanding of these technologies, as well as demonstrated the practical application and effectiveness of a virtual simulation environment in the development of automated systems.

This thesis highlights the important role of virtual simulation in education. Physical machines are expensive, and their assembly takes a long time. The components also wear out and have reliability issues. During education students may accidentally damage the components or devices. Virtual simulation software provides the solution to this issue. Students can experiment, make mistakes and learn, without the risk of damaging expensive components or devices. Without the simulation software, I would not have been able to implement such a complex project. The cost of the components would be too high for any individual.

7.1 Drawbacks of the project

The project complexity went beyond my capabilities, affecting the PLC program of the stacker crane. The retrieval feature of the stacker crane, which retrieves pallets back at the conveyor, was not implemented. Moreover, the stacker crane's program fails to identify which slots are occupied and which are empty. It sorts items from left to right and from the bottom row upward.

The automated sorting system also lacks important features such as safety mechanisms and control panel with start and stop buttons. The PLC code would benefit from improved readability and modularity. The primary logic is concentrated in the Automated_Storage_System program. Improving code readability and modularity could be achieved by breaking down the logic program into separate programs. Additionally, ladder logic code could be rewritten in a structured text plc programming language, which is more suited for programming a complex automation device, such as a stacker crane.

The items that the system sorts were reused from other projects and not designed well for the stacker crane. The wooden pallet occasionally slides off and falls when being unloaded into the storage slots. This issue is caused by the can's rack, where uneven weight distribution leads to the rack sliding off during the unloading process. The results were better with boxes or wooden pallets loaded individually. Another issue arises when, despite the installation of conveyor guides on some conveyors, items placed at the pickup position are slightly misaligned. This misalignment causes UR 5 robot failing to fit cans properly into the slots of the can's rack.

REFERENCES

1. Digital Twins and Virtual Commissioning in the Manufacturing Industry. Visual Components. WWW page. Available from: https://www.visualcomponents.com/resources/blog/digital-twins-and-virtual-commissioning-in-industry-4-0/ [Accessed 2023 Dec 5].

2. The Important Use of Simulation Software in Education. Bin95. WWW page. Available from: https://bin95.com/articles/electrical/engineering/discrete-event-plc-simulation.htm [Accessed 2024 Mar 14].

3. Gateway and Integration, Introduction to Gateway and Communication Driver. Simumatik. Video. Available from: https://academy.simumatik.com/path-player?courseid=gateway-and-integration&unit=63e519dcda193461b7026a2eUnit [Accessed 2024 Mar 16].

4. What is a PLC? Definition and Details. Paessler. WWW page. Available from: https://www.paessler.com/it-explained/plc [Accessed 2024 Feb 25].

5. Open Emulation Platform - User Manual. Simumatik. WWW page. Available from: https://simumatik.com/learn/open_emulation_platform/ [Accessed 2023 Dec 5].

6. Launch of the Simumatik Academy. Simumatik. WWW page. Available from: https://simumatik.com/launch-of-simumatik-academy/ [Accessed 2024 Mar 13].

7. CODESYS Runtime. Clarify. WWW page. Available from: https://www.clarify.io/integrations-browse/codesys-runtime [Accessed 2024 Mar 4].

8. Universal Robots - Offline Simulator - CB-Series - Non-Linux - URSim 3.15.8. Universal Robots. WWW page. Available from: https://www.universal-robots.com/download/software-cb-series/simulator-non-linux/offline-simulator-cb-series-non-linux-ursim-3158/ [Accessed 2024 Mar 4].

9. Control. CODESYS. WWW page. Available from: https://www.codesys.com/products/codesys-runtime/control.html [Accessed 2024 Feb 6].

10. Codesys Tutorial - Simumatik. YouTube. Video. Available from: https://www.youtube.com/watch?v=SBspYJM7tB0 [Accessed 2023 Nov 25].

11. UR SIM Integration- Simumatik. YouTube. Video. Available from: https://www.youtube.com/watch?v=GvUhSvtKh5A [Accessed 2023 Dec 5].

12. Configuring Devices and I/O Mapping. CODESYS. WWW page. Available from: https://help.codesys.com/api-content/2/codesys/3.5.14.0/en/_cds_configuring_devices_mapping_ios/#id3 [Accessed 2024 Jan 3].

13. Advanced Simulation, Digital Twin Technology - the Simumatik Platform. Simumatik. WWW page. Available from: https://academy.simumatik.com/path-

player?courseid=gateway-and-integra-
tion&unit=63e51e5cf00b138791014b8fUnit [Accessed 2024 Jan 10].

LIST OF FIGURES

Catalog of Components for the Automated Sorting System

CONTENTS

1 INTRODUCTION

This document will provide a brief description of the system's components. The components are virtual representations of a real-life counterparts. Each component has ports and variables. The ports are the equivalent of terminals in the electrical components, serving as connection points between components.

1.1 Material handling machinery

1.1.1 Stacker Crane

The stacker crane is called "LIU Crane" in Simumatik. Because the stacker crane is a new addition to the Simumatik's component library, it is currently unfinished. It has only input actuators, with the power source yet to be implemented. However, because this is a virtual component and the code defines its functionality, the actuators are still functioning, despite the unimplemented power source.

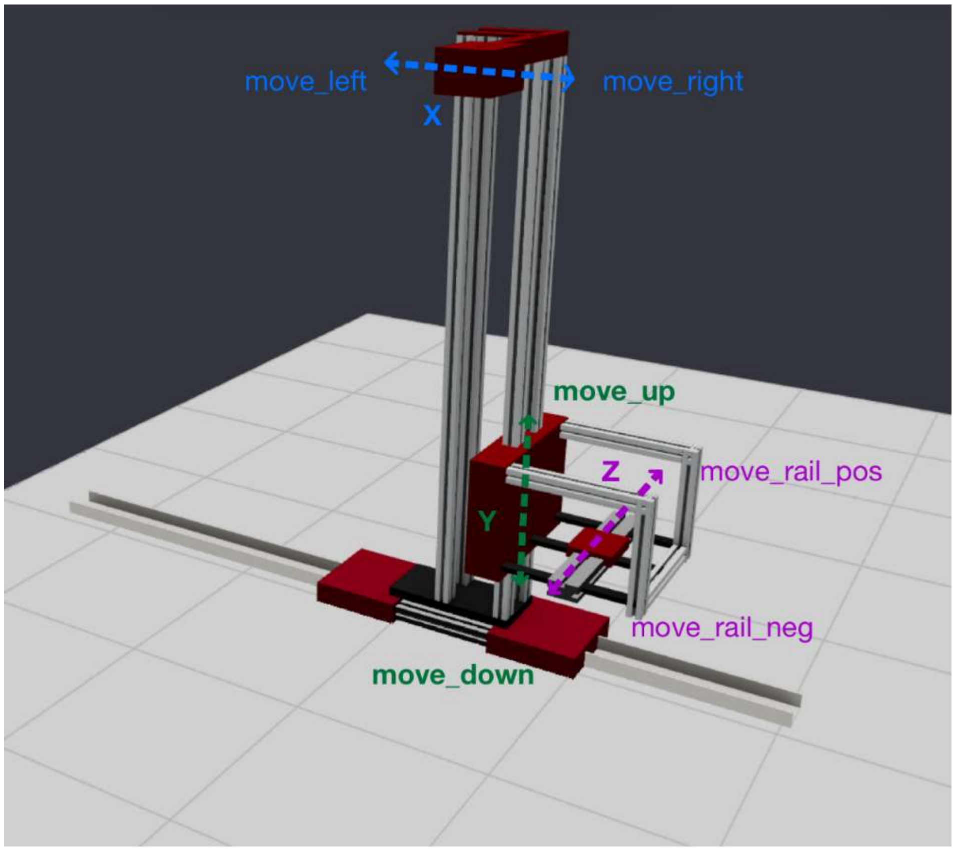
The Figure (1) depicts directional movements of the actuators in the stacker crane. The actuators move_left and move_right, responsible for horizontal movement of the crane along the X axis. While the actuators move_up and move_down determines vertical movements of the crane along the Y axis. Finally, actuators move_rail_pos and move_rail_neg controls extension and retraction of the crane's rail, therefore moving along the Z axis. Component LIU crane has the following inputs and variables that will be connected to the plc.

Ports:

- **move_left:** When energized, moves crane to the left
- **move_right**: When energized, moves crane to the right
- **move_up**: When energized, moves crane up
- **move_down:** When energized, moves crane down
- **move_rail_pos:** When energized, extends the rail forward
- **move_rail_neg:** When energized, extends the rail backward

Variables

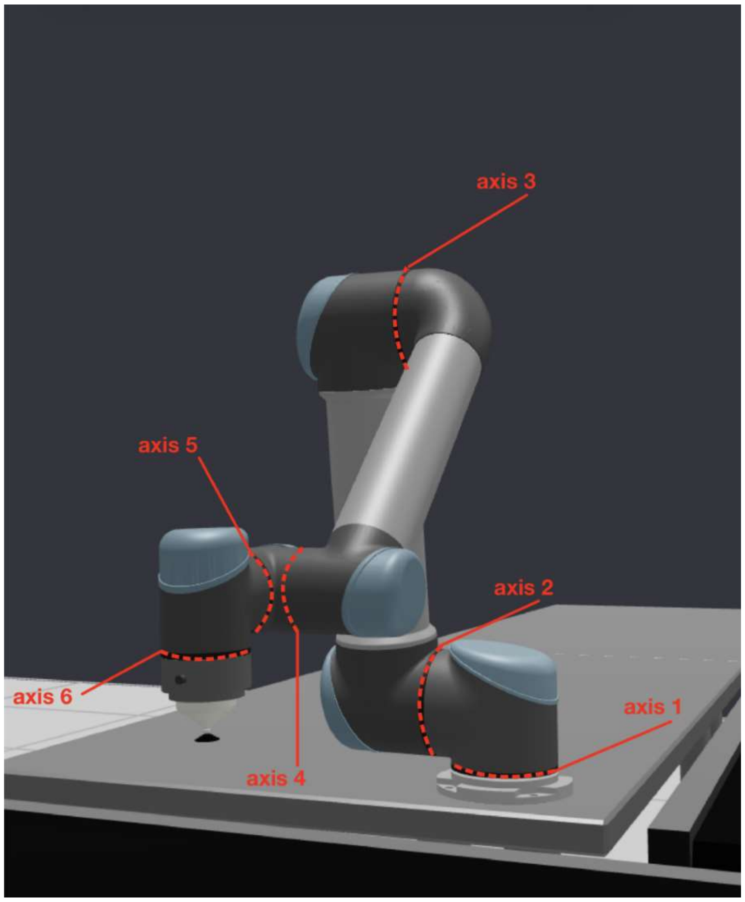
- **GEAR_RATIO_X:** Responsible for the speed adjustment of the x axis
- **GEAR_RATIO_Y:** Responsible for the speed adjustment of the y axis
- **GEAR_RATIO_Z:** Responsible for the speed adjustment of the z axis



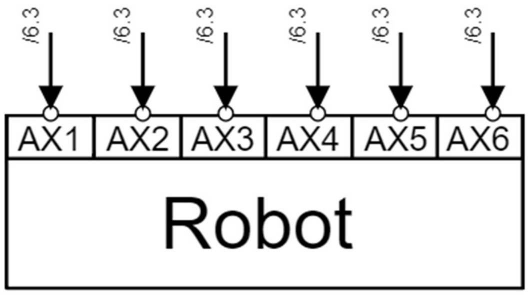
1. Simumatik's Virtual Stacker crane's directions of the actuators

1.1.2 UR5 Robot

For picking up cans' virtual component UR5 robot is selected. UR5 virtual component have 6 axis inputs (Figure 3). Each axis controls the robot's rotational joint (Figure 2). Inputs can be connected to a UR virtual controller in Simumatic.



2. UR 5 robot's rotational joints

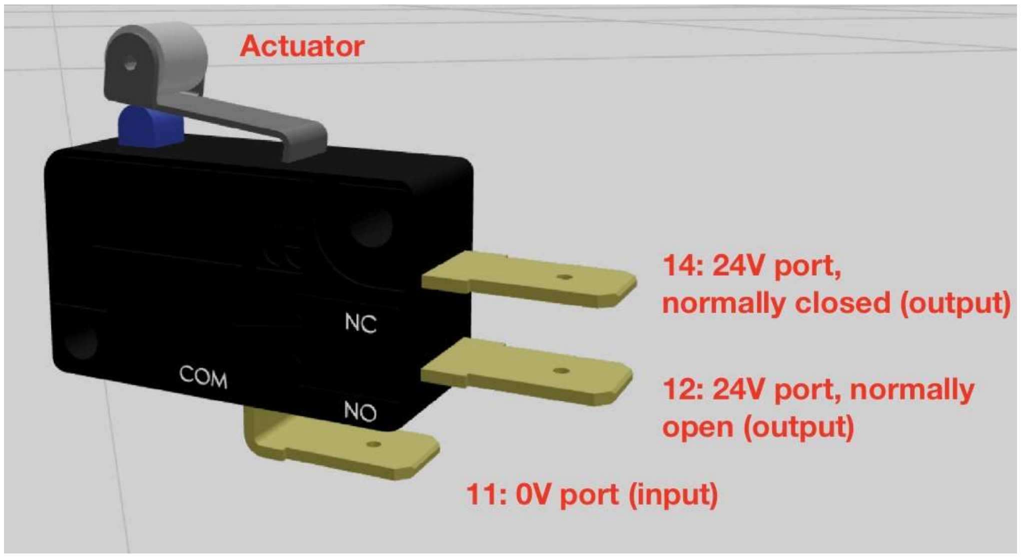


3. Simumatik's UR 5 robot's 2D diagram and its axis inputs

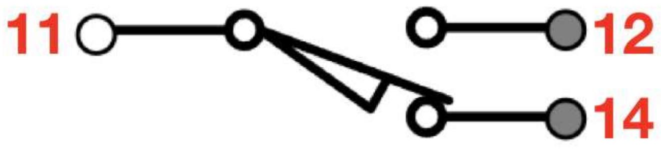
2 ELECTRICAL COMPONENTS

2.1 Limit Switch

Limit switches are used for position detection for industrial equipment.
The limit switch is used to detect the presence of a moving part and has 1 input port and 2 output ports. When a moving part triggers an actuator, it either opens or closes the circuit, depending on which output port is used.


Actuator
NC
COM
NO
14: 24V port, normally closed (output)
12: 24V port, normally open (output)
11: 0V port (input)

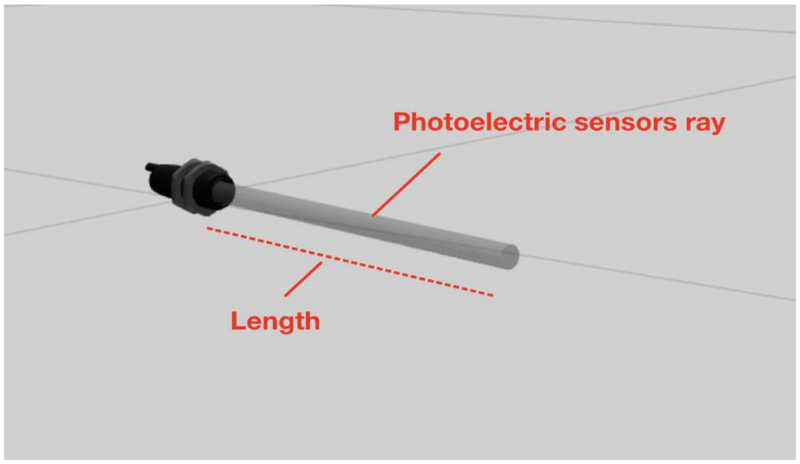
4. Simumatik's limit switch and its ports



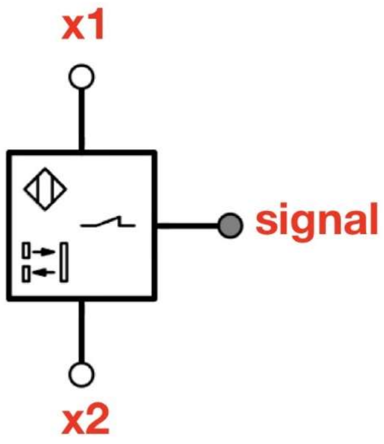
5. Simumatik’s limit switch 2D

2.2 Photoelectric Sensor

A photoelectric sensor is used to detect any physical objects like cans, boxes, and pallets. It can be configured to be either normally open or normally closed.



6. Simumatik's photoelectric sensor



7. Simumatik's photoelectric sensor 2D

Ports:

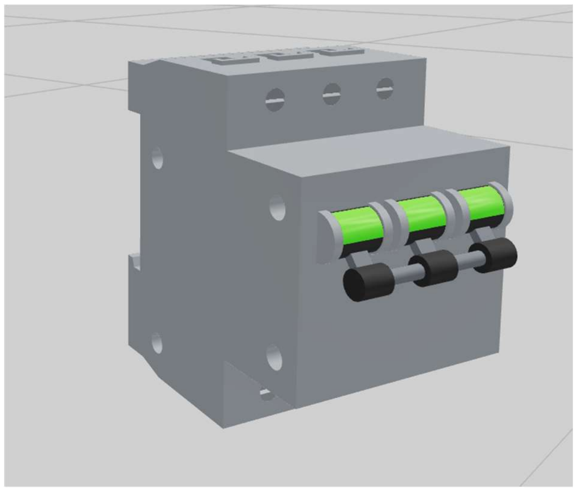
- **x1**: 24V port (input)
- **x2**: 0v port (input)
- **Signal port (output)**: Produces an electrical output upon detection of a physical object, outputting either 24V or 0V.

Variables

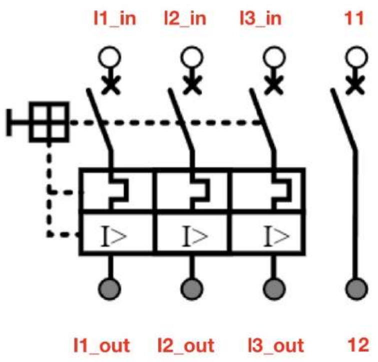
- **ray_length**: Determines the length of the ray.
- **ray_visible**: Controls the visibility of the ray in simulation.
- **normally_open**: When enabled, the sensor output will be activated upon detecting an object; if disabled, functions like normally closed

2.3 Circuit Breaker

A circuit breaker is an electrical safety device, designed to interrupt the current during fault events, protecting the devices connected to it. The circuit breaker instantly breaks the connection if the current exceeds the variable 'max_current'.



9. Simumatik's circuit breaker



8. Simumatik's circuit breaker 2D

Ports:

- **I1_ in**: The first-phase input of three-phase electrical power.
- **I2_ in**: The second-phase input of three-phase electrical power.
- **I3_ in**: The third-phase input of three-phase electrical power.

- **I1_ out**: The first-phase output of three-phase electrical power.
- **I2_ out**: The second-phase output of three-phase electrical power.
  I3_ out: The third-phase output of three-phase electrical power.

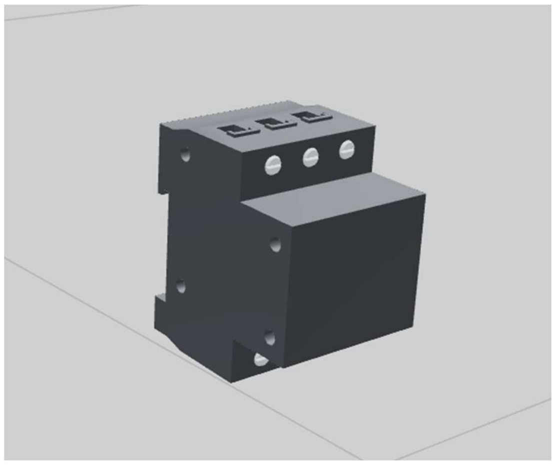
- **11**: Electrical safety port(input)
- **12**: Electrical safety port(output)

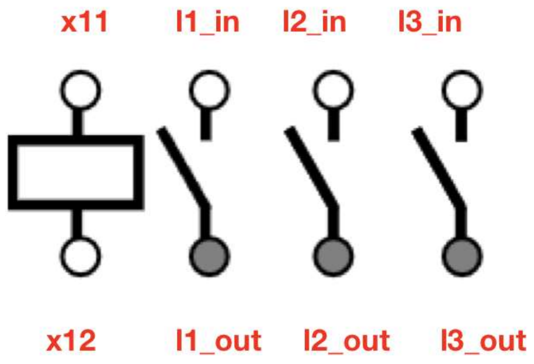
Variables

- **max_current**: maximum allowed value of current, before breaking the connection

2.4 Motor contactor

A motor contactor is used to turn a three-phase electrical motor on and off. It has three inputs and three outputs, along with the two auxiliary contacts, x11 and x12. PLCs and other VDC control devices cannot control a three-phase high-voltage motor directly. A VDC control device is connected via ports x11 and x12. When electrical current flows through ports x11 and x12, it energizes the coil. The coil generates a magnetic field, which attracts the contacts to complete the circuit, activating the device connected to the contactor.


11. Simumatik's motor contactor



10. Simumatik's motor contactor 2D

Ports:

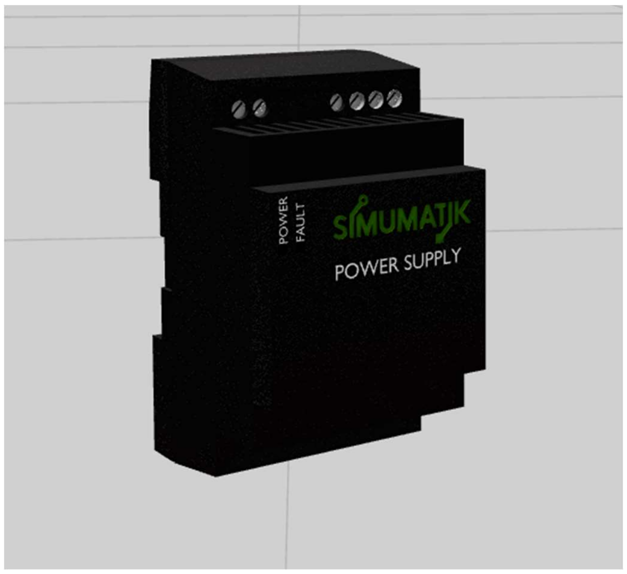
- **l1_ in**: The first-phase input of three-phase electrical power.
- **l2_ in**: The second-phase input of three-phase electrical power.
- **l3_ in**: The third-phase input of three-phase electrical power.

- **l1_ out**: The first-phase output of three-phase electrical power.
- **l2_ out**: The second-phase output of three-phase electrical power.
  l3_ out: The third-phase output of three-phase electrical power.

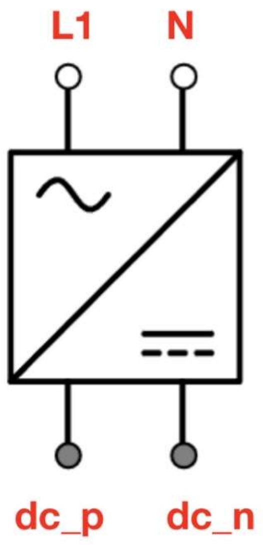
- **x11**: 24V port (input)
- **x12**: 0V port (input)

2.5 DC Power Supply

DC power supply is used to convert alternating current (AC) to direct current (DC) for powering different VDC electronic devices.



12. Simumatik's DC power supply



13. Simumatik's DC power supply 2D

Ports:

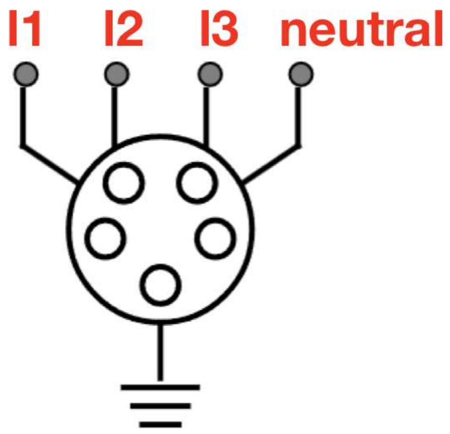
- **L1**: Phase line that is used to connect to one of the three- phase electric power.
- **N**: Neutral line that is used to connect to neutral line of the tree- phase electric power.
- **dc_p**: 24V port (output)
- **dc_n**: 0V port (output)

Variables:

- **max_current**: Maximum allowed value of current, before breaking the connection.
- **overload**: Activates red led light and sends a signal if power is off.

2.6 Three Phase Industrial Socket

14. Simumatik's three phase industrial socket



15. Simumatik's three phase industrial socket 2D

A three-phase industrial socket used to supply power to devices that require three-phase electricity.

Ports:

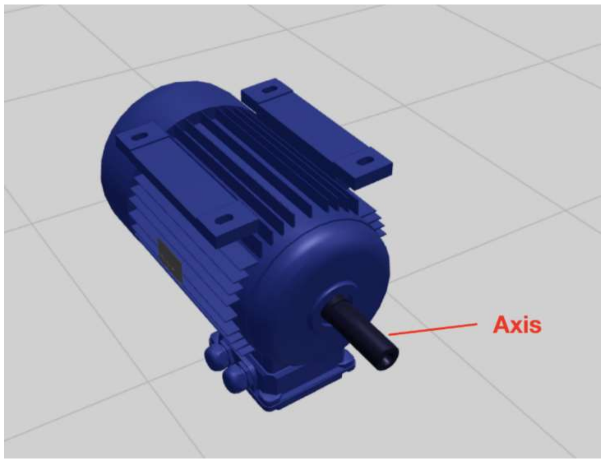
- **l1**: The first-phase output of three-phase electrical power.
- **l2**: The second-phase output of three-phase electrical power.
  l3: The third-phase output of three-phase electrical power.
- **neutral**: 0V Neutral connection.

Variables

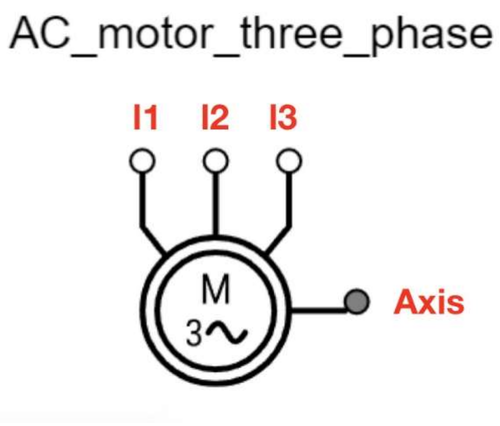
- **max_current**: maximum allowed value of current, before breaking the connection
- **overload**: Sends a signal if power is off.

2.7 AC Motor

A three-phase electrical motor is used for various tasks in industrial settings, such as powering heavy machinery like conveyors.



17. Simumatik's AC motor



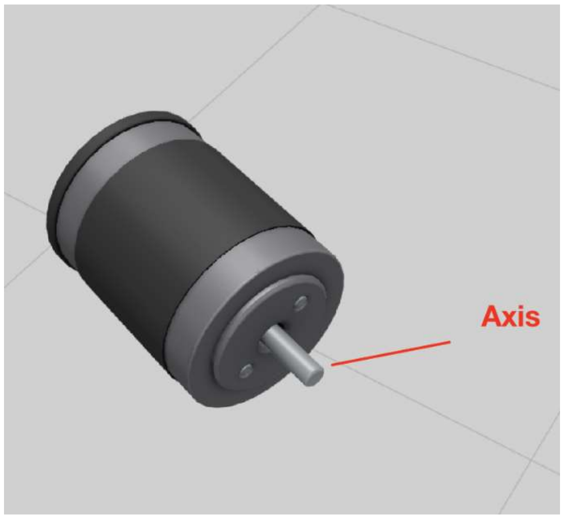
16. Simumatik's AC motor 2D

Ports:

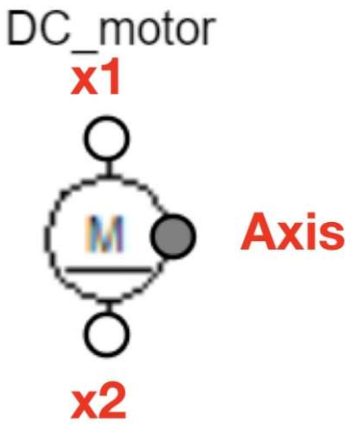
- **I1**: The first-phase input of three-phase electrical power.
- **I2**: The second-phase input of three-phase electrical power.
- **I3**: The third-phase input of three-phase electrical power.
- **axis**: Output that displays and transfers motors rpm value to the connected machinery.

2.8 DC motor

Dc motor is used to power up smaller equipment that does not require much torque. It operates on 24VDC.



19. Simumatik's DC motor

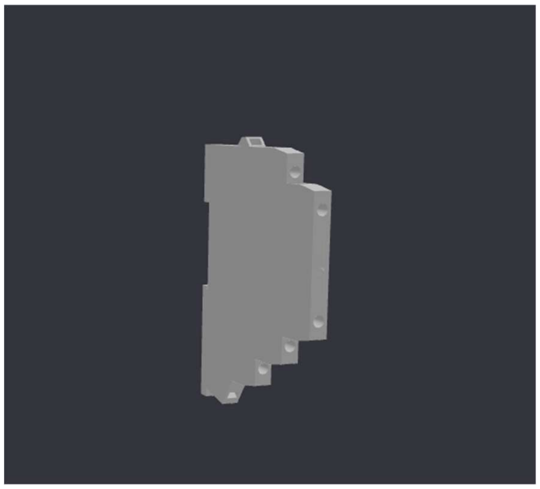


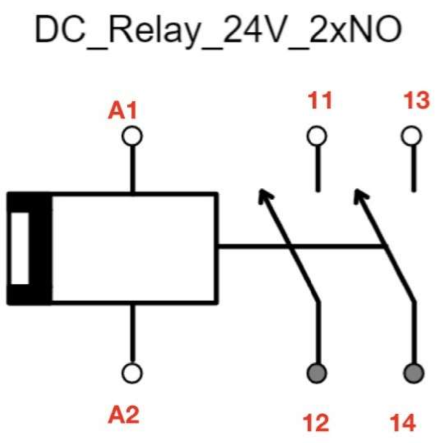
18. DC motor 2D

Ports:

- **x1**: 24V input port
- **x2**: 0V input port
- **axis**: Output that displays and transfers motors rpm value to the connected machinery.

2.9 DC Relay


21. Simumatik's DC relay



20. Simumatik's DC relay 2D

DC_Relay_24V_2xNO is used to control one or multiple VDC devices. Contacts A1 and A2 power the relay, while 2 normally open contacts (11, 12) and (13, 14) serve as auxiliary ports for connecting other devices. When ports A1 and A2 are energized, normally open contacts complete the circuit, and the devices that are connected to relays contacts are activated.

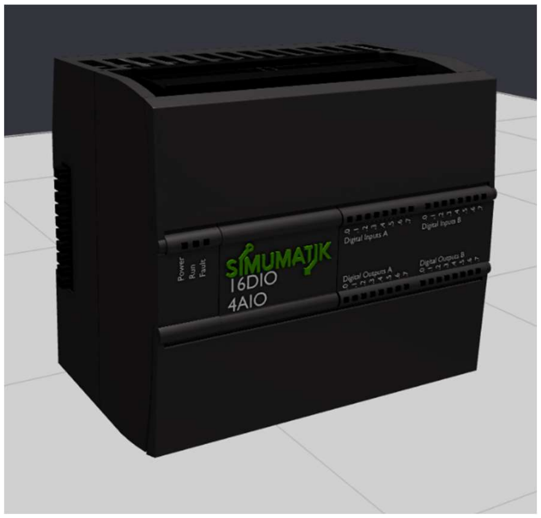
Ports:

- **A1**:24V port
- **A2**: 0V port
- **11**: input port
- **12**: output port
- **13**: input port
- **14**: output port

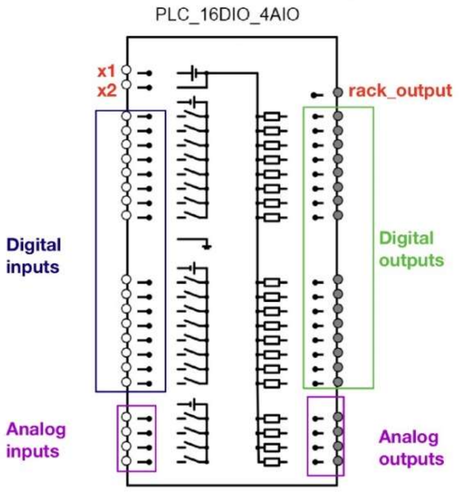
3 CONTROL DEVICES

3.1 PLC 16 DIO 4AIO

PLC 16 DIO 4AIO has 16 digital inputs and 16 digital outputs, as well as 4 analog input and output ports. The controller can be configured with the help of various communication drivers, such as opcua_client, twincat_ads, s7protocol, and many more, to establish connections with third-party software.



23. Simumatik's PLC16DIO4AIO controller



22. Simumatik's PLC16DIO4AIO controller 2D

Ports:

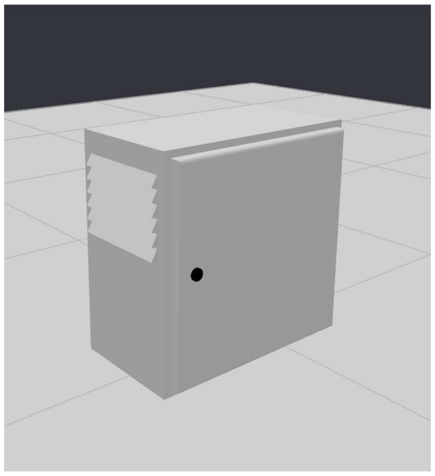
- **x1**: 24V input port
- **x2**: 0V input port
- **in_0 - in_15**: The range of digital input ports
- **out_00 - out_15**: The range digital outputs
- **analog_in_0 – analog_in_3**: The range of analog input ports
- **anallog_out_0 – analog_out_3**: The range of analog output ports
- **rack_output**: Optional IO-card

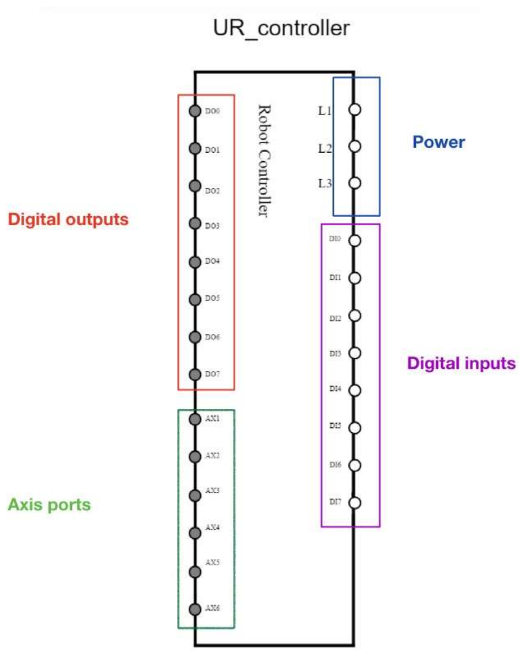
Variables:

- **driver_type**: Driver selection
- **setup_params**: The parameters forwarded to the gateway, like ip- address and port.
- **var_DI1**, **var_DI2**: [Byte] Variable names for the corresponding input signals.
- **var_DO1, var_DO2**: [Byte] Variable names for the corresponding output signals
- **var_AI1** – **var_AI4**: [INT] Variable names for the corresponding analog input signals
- **var_AO1** – **var_AO4**: [INT] Variable names for the corresponding analog output signals
- **voltage_range**: Analog range voltage setting
- **Analog_range**: Range of values for analog signals in the PLC software

3.2 Universal Robots Controller

UR controller uses the ur_driver to communicate with the URsim simulation software and controls the robot. Simumatik’s virtual representation of the UR controller differs in functionality from an actual counterpart.


25. Simumatik's UR controller



24. Simumatik's UR controller 2D

Ports:

- **l1**: The first-phase input of three-phase electrical power.
- **l2**: The second-phase input of three-phase electrical power.
- **l3**: The third-phase input of three-phase electrical power.
- **in_0 – in_7**: Single input bits
- **out_0 – out_7**: Single output bits
- **axis1 – axis6:** axis values in radians, prepared for a robot (output port)

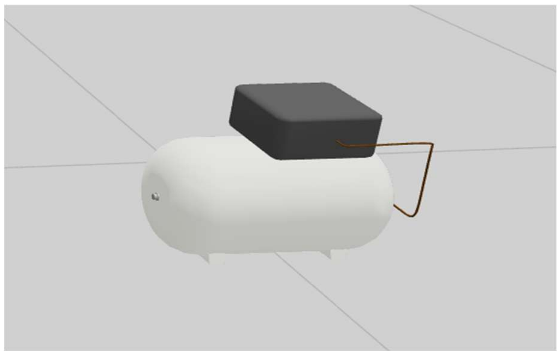
Variables:

- **controller:** Name of the robot
- **read_interval:** Refreshing time value

4 OTHER COMPONENTS

4.1 Pneumatic compressor

Pneumatic compressor used to power up various pneumatic components.



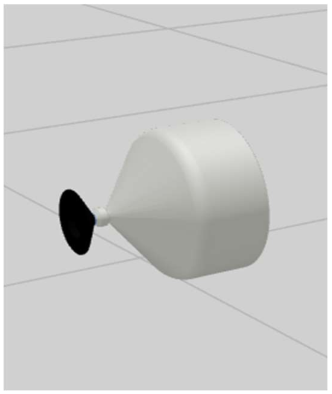
26. Simumatik's pneumatic compressor

Ports:

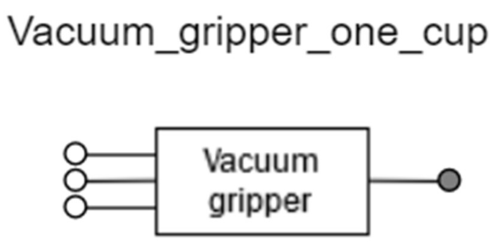
- **out:** Pneumatic power output

4.2 Vacuum Gripper

This ideal vacuum gripper is used to grab an object with a suction-cup using pneumatic power. It can be attached to various robots. The Gripper has a built-in sensor that provides an output signal to its connected device.



27. Simumatik's vacuum gripper



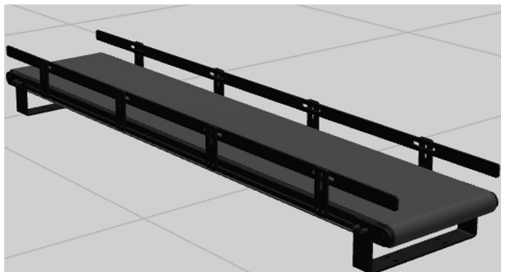
28. Simumatik's vacuum gripper 2D

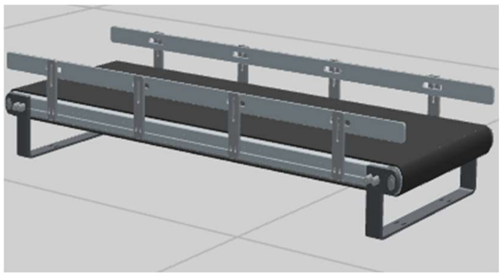
Ports:

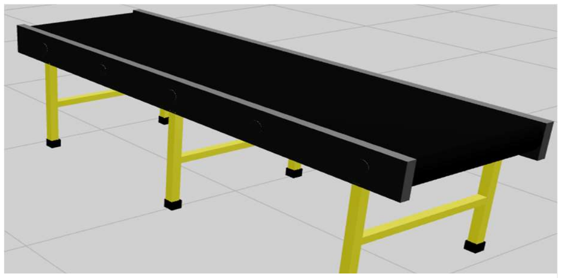
- **x1**: 24V port for powering up gripper sensor
- **x2**: 0V port for powering up gripper sensor
- **p1**: Pneumatic power input, supplied by pneumatic compressor
- **signal**: Output port, that sends detection signal to its connected device

4.3 Conveyor Belts

Every conveyor has motor axis, which can be used to connect a motor. Conveyor belts are used to transport items from one point to another.


30. Simumatik's mini conveyor 1000 x 200 mm


29. Simumatik's mini conveyor 500 x 200 mm

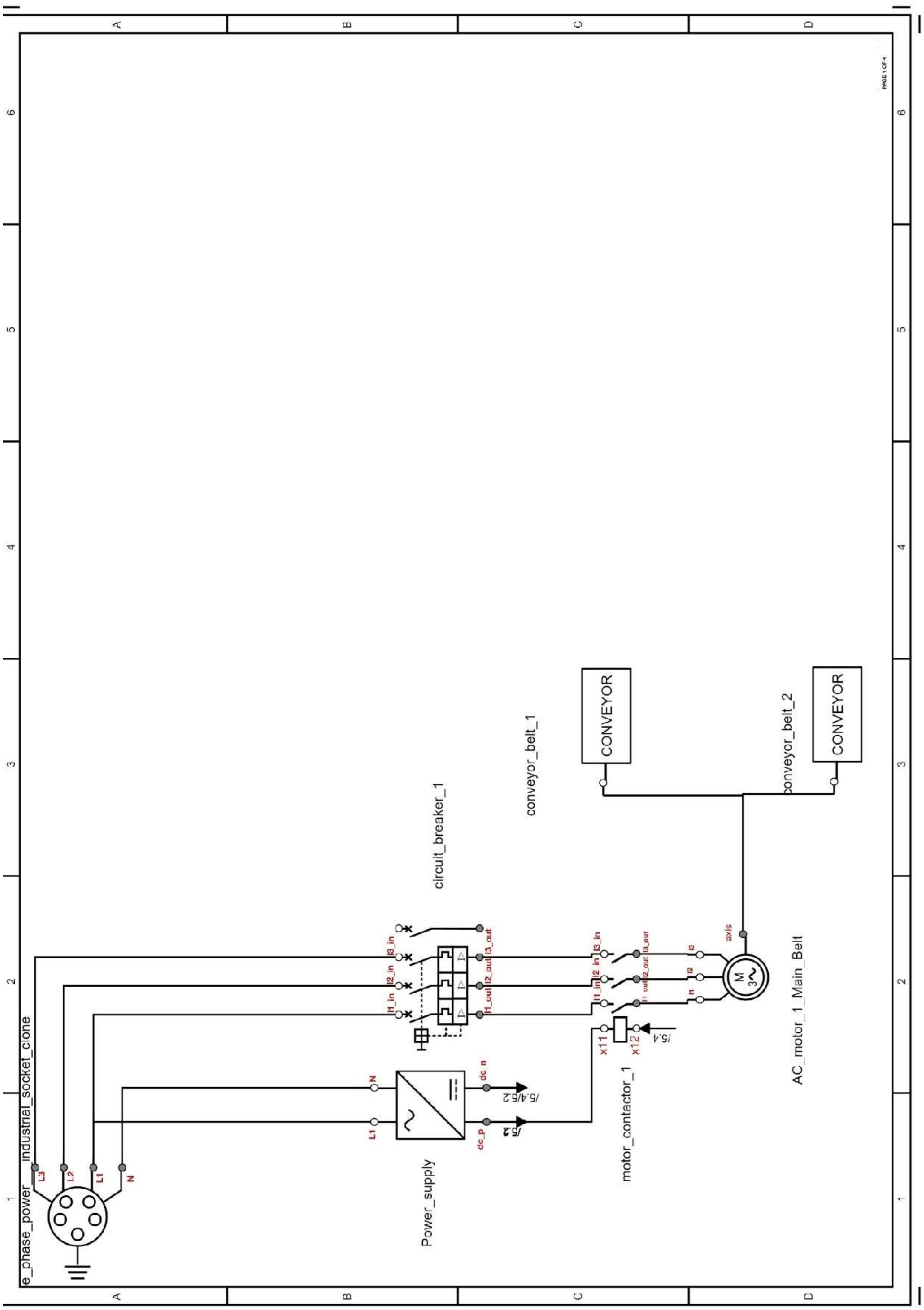

31. Simumatik's conveyor belt

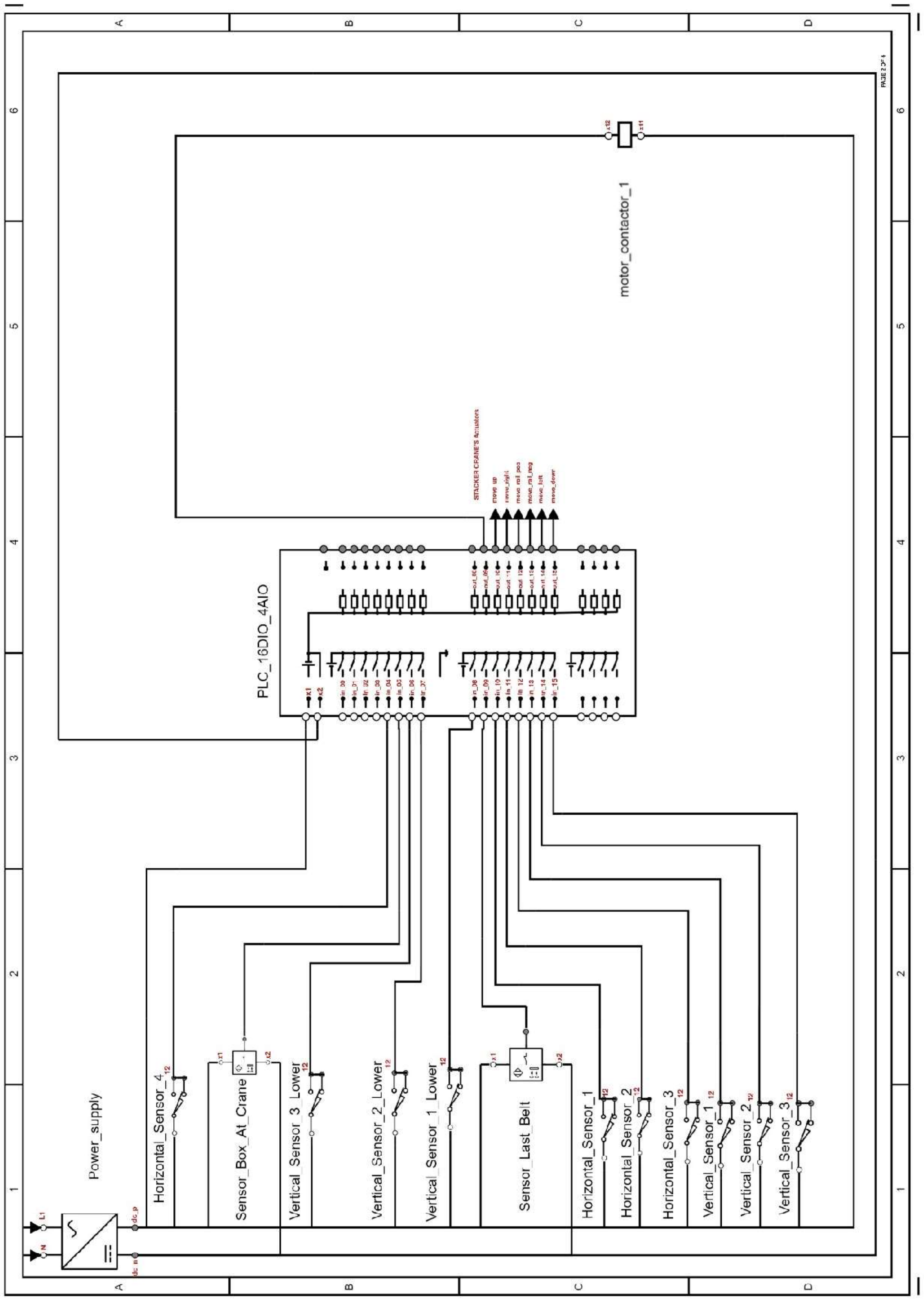
Ports:

- motor_axis

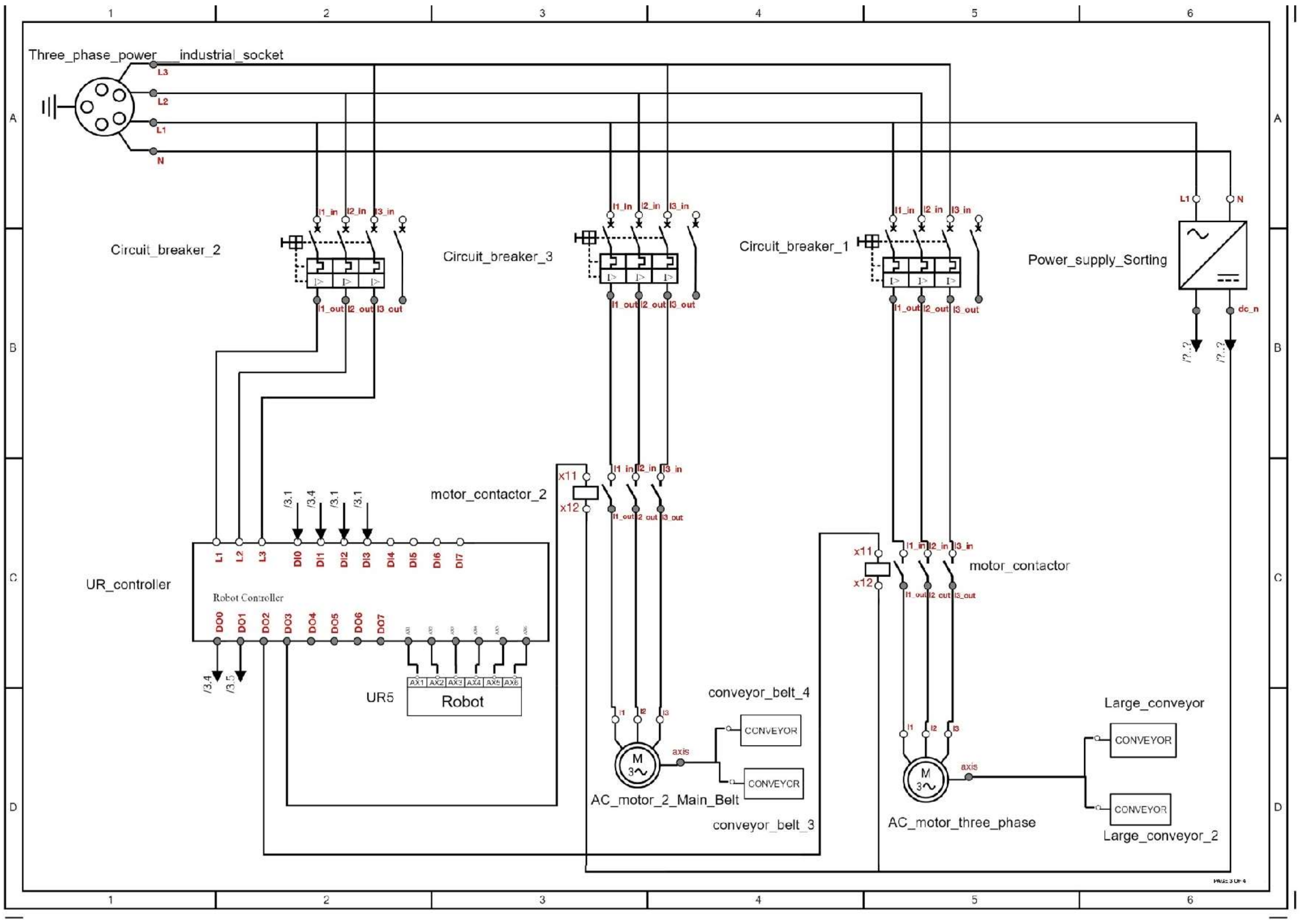
Appendix 2/4

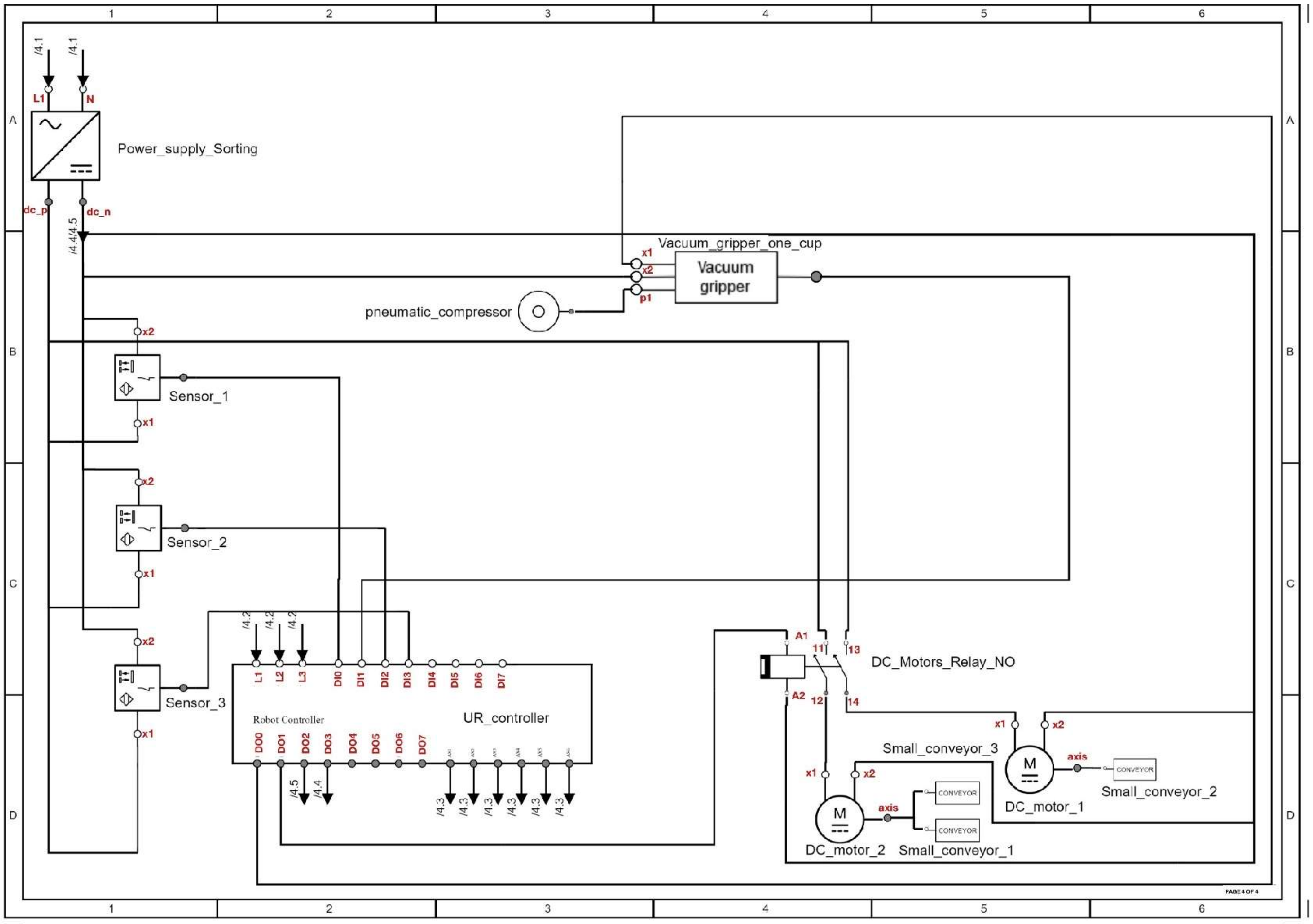
Wiring diagrams of the Automated Sorting System





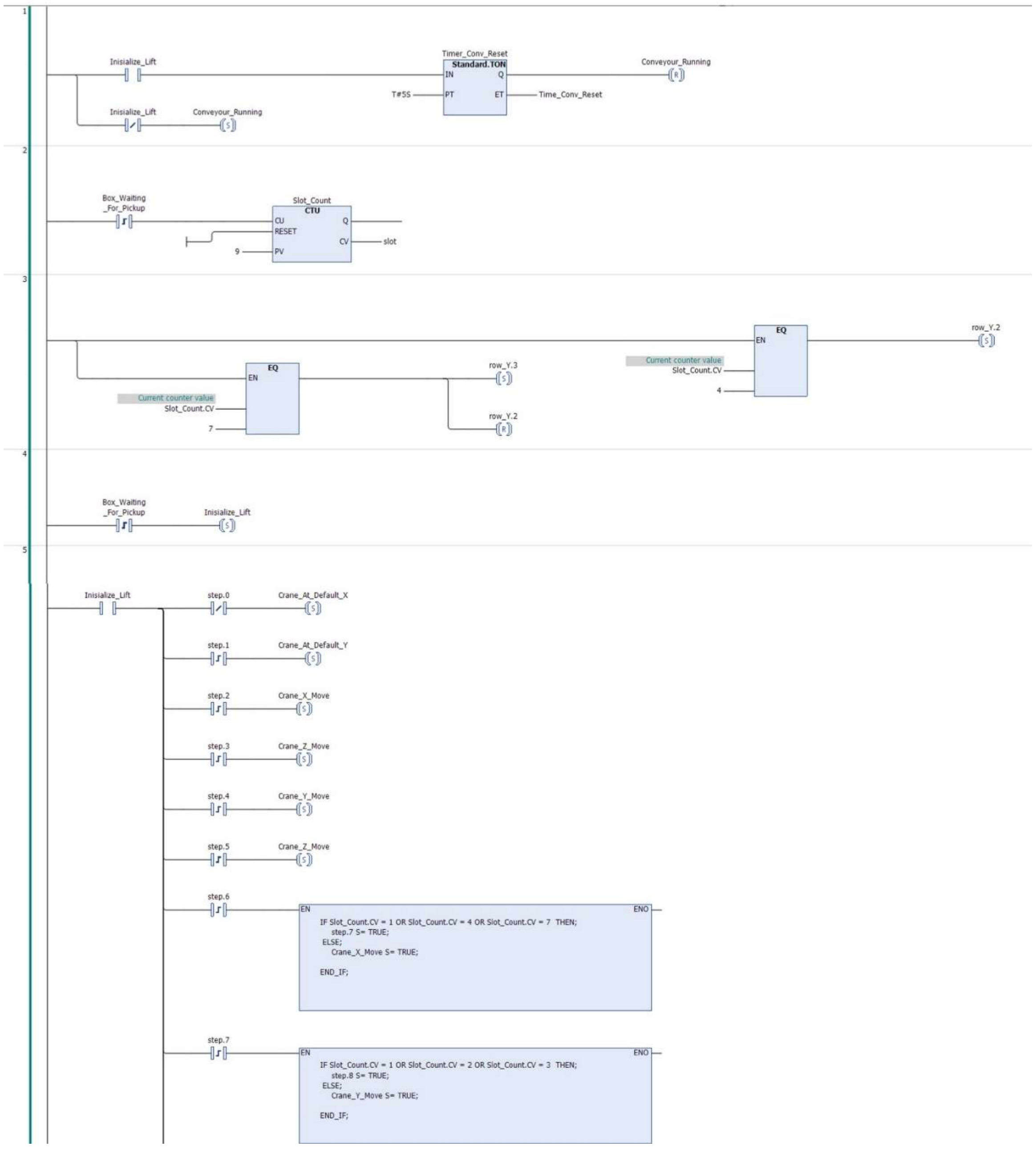
Power_supply
L1
N
dc_p
dc_n
Horizontal_Sensor_4
Sensor_Box_At_Crane
Vertical_Sensor_3_Lower
Vertical_Sensor_2_Lower
Vertical_Sensor_1_Lower
Sensor_Last_Belt
Horizontal_Sensor_1
Horizontal_Sensor_2
Horizontal_Sensor_3
Vertical_Sensor_1
Vertical_Sensor_2
Vertical_Sensor_3
PLC_16DIO_4AIO
STACKER CRANE'S Actuators
move up
move_right
move_rail_pos
move_rail_neg
move_left
move_down
motor_contactor_1
A
B
C
D
1
2
3
4
5
6


Three_phase_power___industrial_socket
L3
L2
L1
N
Circuit_breaker_2
Circuit_breaker_3
Circuit_breaker_1
Power_supply_Sorting
dc_n
I1_in
I2_in
I3_in
I1_out
I2_out
I3_out
motor_contactor_2
motor_contactor
x11
x12
UR_controller
Robot Controller
DI0
DI1
DI2
DI3
DI4
DI5
DI6
DI7
DO0
DO1
DO2
DO3
DO4
DO5
DO6
DO7
/3.1
/3.4
/3.5
UR5
Robot
AX1
AX2
AX3
AX4
AX5
AX6
axis
AC_motor_2_Main_Belt
AC_motor_three_phase
conveyor_belt_4
conveyor_belt_3
Large_conveyor
Large_conveyor_2
CONVEYOR
PAGE 3 OF 4



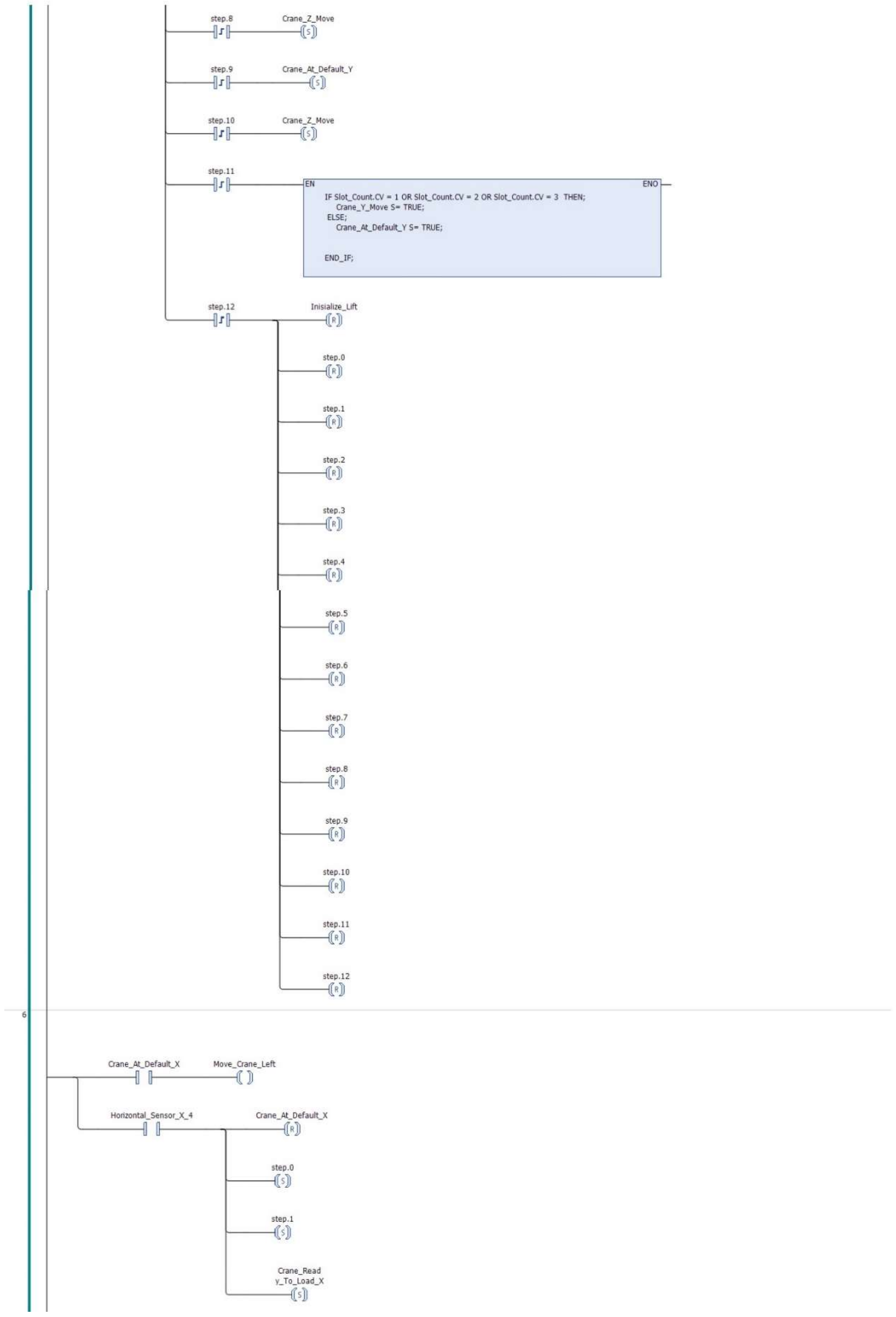
Power_supply_Sorting
L1
N
/4.1
/4.1
dc_p
dc_n
/4.4/4.5
Vacuum_gripper_one_cup
Vacuum gripper
x1
x2
p1
pneumatic_compressor
Sensor_1
Sensor_2
Sensor_3
x1
x2
Robot Controller
UR_controller
L1 L2 L3
DI0 DI1 DI2 DI3 DI4 DI5 DI6 DI7
DO0 DO1 DO2 DO3 DO4 DO5 DO6 DO7
/4.2
/4.5
/4.4
/4.3
DC_Motors_Relay_NO
A1
A2
11
12
13
14
DC_motor_1
DC_motor_2
M
axis
Small_conveyor_1
Small_conveyor_2
Small_conveyor_3
CONVEYOR
PAGE 4 OF 4

PLC Program for the Stacker Crane

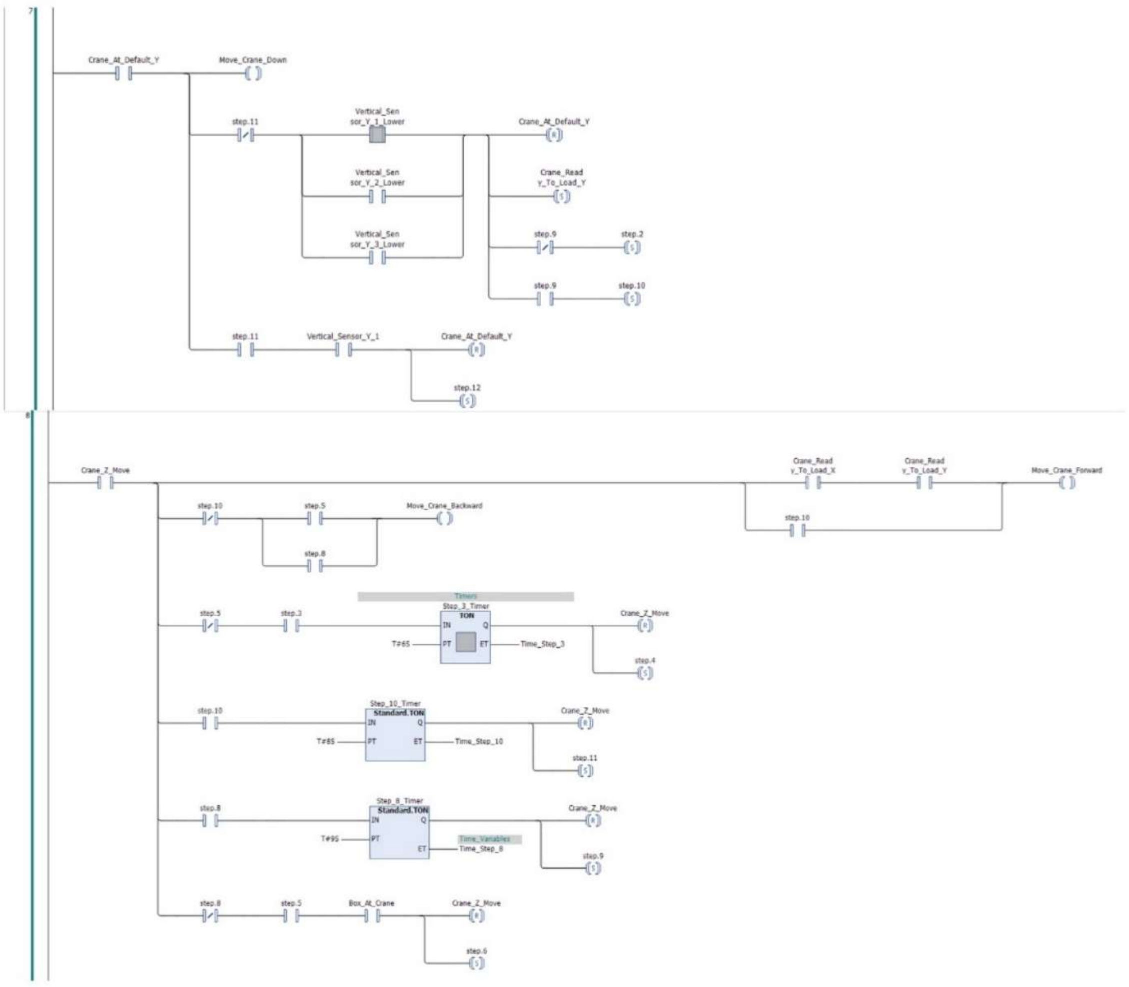


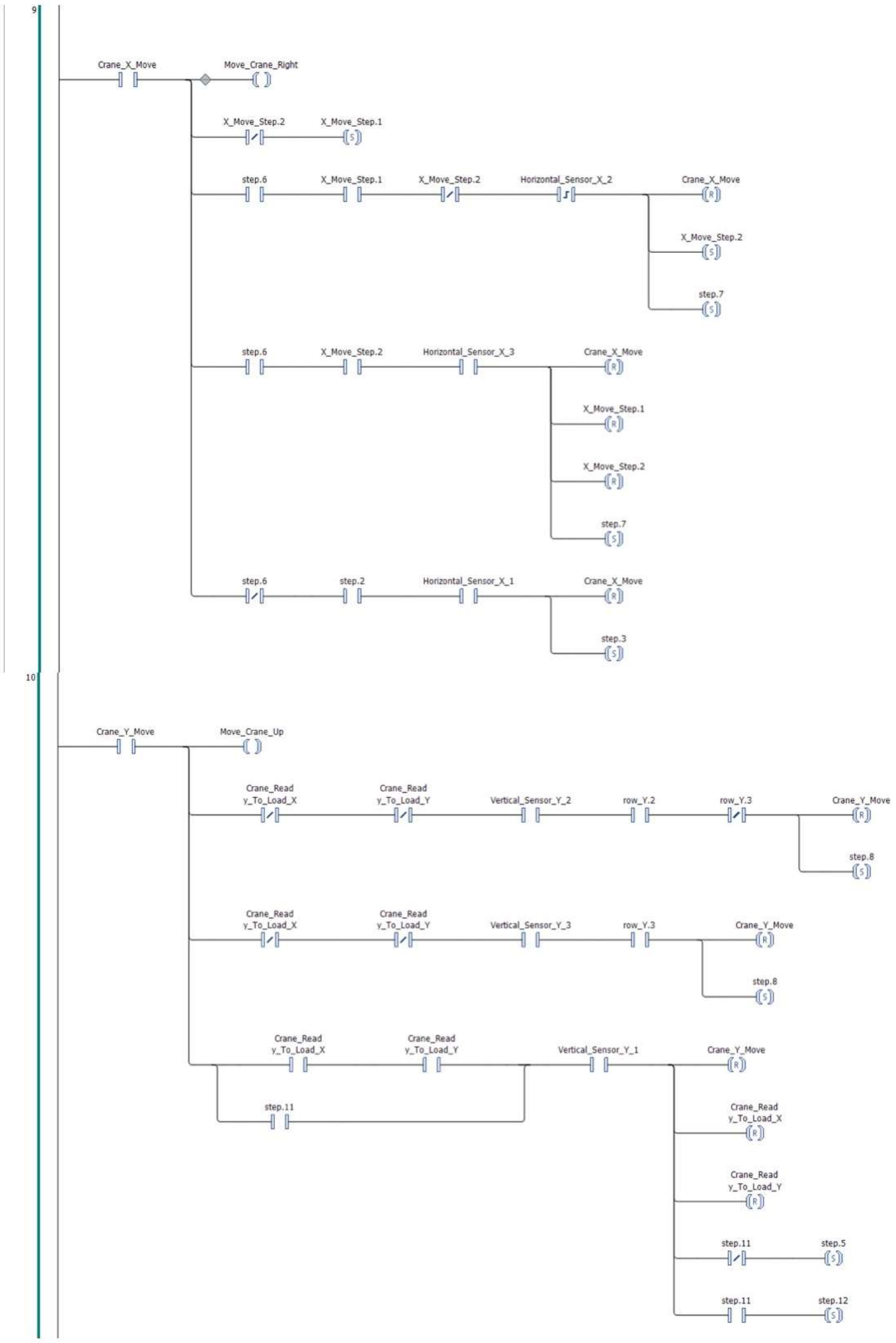
```
IF Slot_Count.CV = 1 OR Slot_Count.CV = 4 OR Slot_Count.CV = 7  THEN;
    step.7 S= TRUE;
 ELSE;
    Crane_X_Move S= TRUE;

END_IF;
```

```
IF Slot_Count.CV = 1 OR Slot_Count.CV = 2 OR Slot_Count.CV = 3  THEN;
    step.8 S= TRUE;
 ELSE;
    Crane_Y_Move S= TRUE;

END_IF;
```



```
IF Slot_Count.CV = 1 OR Slot_Count.CV = 2 OR Slot_Count.CV = 3  THEN;
    Crane_Y_Move S= TRUE;
 ELSE;
    Crane_At_Default_Y S= TRUE;


END_IF;
```


7
Crane_At_Default_Y
Move_Crane_Down
step.11
Vertical_Sensor_Y_1_Lower
Crane_At_Default_Y
Vertical_Sensor_Y_2_Lower
Crane_Ready_To_Load_Y
Vertical_Sensor_Y_3_Lower
step.9
step.2
step.9
step.10
step.11
Vertical_Sensor_Y_1
Crane_At_Default_Y
step.12
8
Crane_Z_Move
Crane_Ready_To_Load_X
Crane_Ready_To_Load_Y
Move_Crane_Forward
step.10
step.10
step.5
Move_Crane_Backward
step.8
Timers
step.5
step.3
Step_3_Timer
TON
IN
Q
T#6S
PT
ET
Time_Step_3
Crane_Z_Move
step.4
step.10
Step_10_Timer
Standard.TON
IN
Q
T#8S
PT
ET
Time_Step_10
Crane_Z_Move
step.11
step.8
Step_8_Timer
Standard.TON
IN
Q
T#9S
PT
ET
Time_Variables
Time_Step_8
Crane_Z_Move
step.9
step.8
step.5
Box_At_Crane
Crane_Z_Move
step.6


9
Crane_X_Move
Move_Crane_Right
X_Move_Step.2
X_Move_Step.1
step.6
X_Move_Step.1
X_Move_Step.2
Horizontal_Sensor_X_2
Crane_X_Move
X_Move_Step.2
step.7
step.6
X_Move_Step.2
Horizontal_Sensor_X_3
Crane_X_Move
X_Move_Step.1
X_Move_Step.2
step.7
step.6
step.2
Horizontal_Sensor_X_1
Crane_X_Move
step.3
10
Crane_Y_Move
Move_Crane_Up
Crane_Read y_To_Load_X
Crane_Read y_To_Load_Y
Vertical_Sensor_Y_2
row_Y.2
row_Y.3
Crane_Y_Move
step.8
Crane_Read y_To_Load_X
Crane_Read y_To_Load_Y
Vertical_Sensor_Y_3
row_Y.3
Crane_Y_Move
step.8
Crane_Read y_To_Load_X
Crane_Read y_To_Load_Y
Vertical_Sensor_Y_1
Crane_Y_Move
step.11
Crane_Read y_To_Load_X
Crane_Read y_To_Load_Y
step.11
step.5
step.11
step.12

Program for Universal Robot 5

```
Program
   BeforeStart
     i_var_1≔1
     i_var_2≔1
     i_var_3≔1
   Robot Program
     If sensorBelt2 and i_var_2≟1 and sensor3
       MoveJ
         homePosCov2
       MoveJ
         pickPosConv2
         Set gripper=On
         Wait: 1.0
       If gripped
         MoveJ
           homePosCov2
         MoveJ
           dropPosConv2
         MoveJ
           finalDropPosCo2
           Set gripper=Off
           i_var_2≔2
     If sensor and i_var_2≟2
       MoveJ
         basePos
       MoveJ
         pickup
         Set gripper=On
         Wait: 0.5
       If gripped
         MoveJ
           basePos
         If i_var_1≟1
           MoveJ
             DropPos1
             FinalDropPos1
             i_var_1≔2
             Set gripper=Off
         ElseIf i_var_1≟2
           MoveJ
             DropPos2
             FinalDropPos2
             i_var_1≔3
             Set gripper=Off
         ElseIf i_var_1≟3
           MoveJ
```

```
          DropPos3
          FinalDropPos3
          i_var_1≔4
          Set gripper=Off
      ElseIf i_var_1≟4
        MoveJ
          DropPos4
          FinalDropPos4
          i_var_1≔1
          Set gripper=Off
          i_var_2≔1
          Wait: 1.0
          i_var_3≔2
          Wait: 4.0
          i_var_3≔1
      MoveJ
        LiftUpPos
Thread_3
  If sensor3≠ True  or i_var_3≟2
    Set conveour3=On
  Else
    Set conveour3=Off
Thread_2
  If sensorBelt2≠ True
    Set coveour2=On
  Else
    Set coveour2=Off
Thread_1
  If sensor≠ True
    Set conveour=On
  Else
    Set conveour=Off
```