

Politecnico di Torino

MASTER OF SCIENCE IN
Computer Engineering



Automation system's features under different PLC
based and low-cost controllers

Supervisor:
prof. LUIGI MAZZA

Student:
Lowai Kamaleldin Abdalla Ali

MARCH 2022

Acknowledgments

*All the acknowledgment goes to my **GOD** who continuously guides me all the life.*

*Three special "**THANK YOU**" go to:*

*Firstly, Yo my **beloved family**, for always being caring, understanding, sacrificing, and supporting me throughout my life.*

*Secondly, to my supervisor **Prof. LUIGI MAZZA**; for his help and guidance throughout this work. As well as **EDISU Piemonte**, for financially supporting me during my study.*

*Thirdly, to the long list of all **old and new friends** whom I consider my family for the ultimate support, the words of comfort and experiences, and all the etched shared memories.*

THANK YOU ALL!

GRAZIE A TUTTI!

Lowai Kamal

List of Figures

2.1	MecLab Stacking station	5
2.2	Station 1 pneumatical and electrical setup	5
2.3	Station 1 Node	6
2.4	MecLab Conveyor Station	7
2.5	Station 2 pneumatical and electrical setup	8
2.6	Station 2 Node	8
2.7	Main Cable	10
2.8	Station 3 pneumatical and electrical setup	11
2.9	Station 3 Node	11
2.10	Main Cable	12
2.11	Flow diagram	14
3.1	Raspberry GPIO	16
3.2	Raspberry Pi connection	19
3.3	Raspberry extension	19
3.4	Opto-coupler	20
3.5	Opto-coupler for Input	21
3.6	Input board scheme	22
3.7	Opto-coupler for Output	24
3.8	Output board scheme	24
3.9	Output complement scheme	25
3.10	Output complement using TIP120	26
3.11	Configuration connection schema	28
3.12	Assigned IP address	29
3.13	Updating the RPI	30
3.14	New Project Creation	30
3.15	GPIO configuration	31
3.16	GPIO Mapping	32
3.17	Raspberry Pi Network	32
3.18	Visualization	33
3.19	Creating Visualization	34
3.20	Assigning IOs to Visualization	35
3.21	Texting	35
3.22	Adding images	36
3.23	Adding images to project Visualization	36
3.24	Selecting image for Visualization	37

3.25	Dynamic variable	37
3.26	Entering optional number	38
3.27	RPI Network 1	39
3.28	RPI Network 2	39
3.29	RPI Network 3	39
3.30	RPI Network 4	40
3.31	RPI Network 5	40
3.32	RPI Network 6	40
3.33	RPI Network 7	41
3.34	RPI Network 8	41
3.35	RPI Network 9	41
3.36	RPI Network 10	41
3.37	RPI Network 11	42
3.38	RPI Network 12	43
3.39	RPI Network 13	43
3.40	Modified opto-coupler for Output	44
3.41	Modified Output board scheme	44
4.1	Siemens LOGO! PLC	46
4.2	PLC components	47
4.3	CPU Module	49
4.4	IO Expansion	49
4.5	Communication Module	50
4.6	Antenna	50
4.7	Communication Module	51
4.8	IOs connections for Communication Module	51
4.9	Creating new Project	52
4.10	Assigning IOs and Comments	53
4.11	Downloading from PC to LOGO!	54
4.12	Simulation	54
4.13	Start page	55
4.14	LAN	56
4.15	Users	56
4.16	Recipient groups	57
4.17	Overview	57
4.18	Message text	58
4.19	Signals	58
4.20	Events	59
4.21	Actions	59

4.22	Assignments	60
4.23	Mobile Wireless	60
4.24	Receipt of SMS	61
4.25	SMS alias	61
4.26	Diagnostics	62
4.27	LOGO! Logic Network 1	62
4.28	LOGO! Logic Network 2	63
4.29	LOGO! Logic Network 3	63
4.30	LOGO! Logic Network 4	64
4.31	LOGO! Logic Network 5	64
4.32	LOGO! Logic Network 6	65
4.33	LOGO! Logic Network 7	65
4.34	LOGO! Logic Network 8	65
4.35	LOGO! Logic Network 9	66
4.36	LOGO! Logic Network 10	67
4.37	LOGO! Logic Network 11	68
5.1	Easyport	69
5.2	Easyport connection	70
5.3	Easyport hardware	71
5.4	Easyport Demo	72
5.5	FluidSim OPC	73
5.6	OPC Overview	74
5.7	OPC IOs Status	74
5.8	Virtual Start	75
5.9	Pneumatic items	75
5.10	Electrical items	76
5.11	FluidSim Network 1	76
5.12	FluidSim Network 2	77
5.13	FluidSim Network 3	77
5.14	FluidSim Network 4	78
5.15	FluidSim Network 5	78
5.16	FluidSim Network 6	79
5.17	FluidSim Network 7	79
5.18	FluidSim Network 8	79
5.19	FluidSim Network 9	80
5.20	FluidSim Network 10	81
5.21	FluidSim Network 11	82
5.22	FluidSim Network 12	83

6.1	Power Supply switches	84
6.2	Start pushbutton wiring	84
6.3	Alarm led wiring	85
6.4	Panel layout	85
7.1	Final results	86

List of Tables

2.1	Station 1 IOs	5
2.2	Station 1 Pins assignment	6
2.3	Station 2 IOs	7
2.4	Station 2 Pins assignment	9
2.5	Station 3 IOs	10
2.6	Station 3 Pins assignment	12
2.7	Main cable IOs assignment	13
2.8	Required IOs	15
3.1	Complement solenoid Coils	17
3.2	IOs Assignment for RPI	18
3.3	Input board Pins assignment	22
3.4	Output board Pins assignment	25
4.1	IOs Assignment for LOGO!	48
5.1	IOs Assignment for Easyport	70

Contents

Acronyms	X
1 State of art	1
1.1 Automation in Assembly lines	1
1.2 Electropneumatic automation	2
2 MecLab	4
2.1 Stations	4
2.1.1 Stack Magazine Station	4
2.1.2 Conveyor Station	6
2.1.3 Handling Station	9
2.2 The electrical connection to controller	12
2.3 Flow chart of the process	13
2.4 Total Required IOs	15
3 Raspberry Pi	16
3.1 IOs assignment (GPIO)	16
3.2 Hardware implementations	18
3.2.1 Opto-coupler	20
3.2.2 Outputs Reduction by the complement	25
3.3 Software Implementation	26
3.3.1 Configuring the Raspberry Pi	27
3.3.2 Writing the Code	30
3.3.3 Remote controlling	33
3.4 Networks of the code	39
3.5 Testing and monitoring	43
3.5.1 Current shortage	43
3.5.2 Notes	45
4 LOGO!	46
4.1 Overview	46
4.2 IOs assignment	47
4.3 Hardware implementations	48
4.3.1 CPU Module	48
4.3.2 IO Expansion Module	49
4.3.3 Communication Module	50

4.4	Software Implementation	52
4.4.1	LOGO! Soft! Configurations	52
4.4.2	Simulation	54
4.4.3	Remote controlling	55
4.4.4	Networks of Logic	62
4.5	Testing and monitoring	68
5	EasyPort	69
5.1	Overview	69
5.2	IOs assignment	70
5.3	Hardware implementations	71
5.4	Software Implementation	71
5.4.1	Software Initialization	71
5.4.2	Coding	72
5.4.3	Remote controlling	75
5.4.4	Connections of the networks	75
5.5	Testing and monitoring	83
6	The Panel layout	84
7	Conclusions and Future Work	86
7.1	Conclusions	86
7.2	Future Work	87
8	Appendix	88
8.1	Appendix A	89
8.2	Appendix B	90
8.3	Appendix C	91
	Bibliography	92

Abstract

In current time for automation field in general and specially factories, increasing production is highly requested, while decreasing both cost and processing time. All should be compatible with high production quality and following safety standards. It worth mentioning also a well demand for analysis in order to give better decision, it is the accessibility to info, locally and globally.

In this thesis, the main aim is to provide clear practical study in the electropneumatic automation field by distinguishing advantages and drawback for different low cost controllers in an educational prototype of assembly line which consists of three stations: Pushing station, conveyor station and Gripping station. These controllers will be Raspberry Pi (with details in chapter 3), Siemens LOGO! PLC (with details in chapter 4) and Festo EasyPort (with details in chapter 5).

Wiring and mapping for these stations were reported in chapter 2. All required Hardware and software will be shown separately for each controller under its chapter in step-by-step demonstration. A panel was created to contain all the needed components for all controllers as will be illustrated in chapter 6.

11 Predefined KPIs (Key Performance Indicators) will be the the main factors for comparing these three controllers. Data will be gathered and examined against these indicators to obtain the such an analysis in chapter 7.

Acronyms

CPU

Central Processing Unit

DHCP

Dynamic Host Configuration Protocol

GPIO

General Purpose Input/Output

GPU

Graphics Processing Unit

IOs

Inputs/Outputs

OPC

Open Platform Communications

PC

Personal Computer

PLC

Programmable Logic Controller

RAM

Random Access Memory

RPI

Raspberry Pi

UART

Universal Asynchronous Receiver Transmitter

V

Voltage

Chapter 1

State of art

1.1 Automation in Assembly lines

Always production is needed to be more in both quantity and quality in such an efficient way. Hence, there are continues development to achieve it with the minimum losses in human efforts, money, and energy. Automating production lines has the ability to tolerate all successfully. Tools are used to allow ease processes during manual production. However, these tools may have dangerous impact to labor's safety. In addition to this, high standard quality cannot be assured since human error factor exists always, Despite the claims of high quality from experienced workmanship by humans, automated systems typically execute the manufacturing process with less variability, resulting in greater control and consistency of product quality. Moreover, it increases process control makes more efficient use of materials, resulting in less leftover. Also, automation ease the need to keep monitoring the production remotely to handle it when maintenance is required or during emergency.

Since air is available anytime and anyplace on earth naturally and it has the compressibility feature, it is the greatest choice to generate a reasonable force to achieve rotational movement, pick and place processes. Pneumatic process is the use of pressurized air to enable these mechanical motions.

Pneumatic power is used in industries where machines are most likely to be plumbed for compressed air to follow preplanned plan. Pneumatic automation can be done through the use of various components. It worth to mention some advantages of pneumatic automation:

1. Easy in **generation**, **transportation** through pipes and **releasing** to atmosphere without the need of processing since it is environmentally friendly.
2. **Economical** method to generate great limited forces.
3. High **durability** and **reliability** with a long operating life.
4. **Cost-effectiveness** due to the lower maintenance cost.
5. **Storage** capability, which enables the machines to be used even in the event of a power outage by using reservoir.

6. **Safety** with extremely minimum hazards on both human and machines.
7. Easy **controllability**.

However, to develop faster response time, electric is being used along the pneumatical items. It does not assure all the advantages of pneumatic only, but it adds many when only 24 volts is used. Because It guarantees faster and reliable controlling.

1.2 Electropneumatic automation

Due to its benefits, Electropneumatic automation is highly preferred in production in general and in Pick and Place especially. For the educational purposes for this study, below items will be used as the main components for the assembly line under experiment:

- **Solenoid valves** to control double acting cylinders as well as single acting cylinders.
- Proximity sensors attached to cylinders which give the feedback to elaborate the position of the piston of the specific cylinder.
- Proximity sensor to acknowledge the **availability** of the product.
- Color sensor to differentiate the **color** of the product.
- **Motor** to move the product forward or backward controlled by contacts.
- **Solenoid coil** to reject the unnecessary product.
- **Gripper** to handle the product.
- **Distribution nodes** to gather all the inputs and outputs in a single cable with common ground.

In order to control the sequence of the IOs, many kinds of controllers can be used. However, three low cost methods of controlling will be discussed in detail:

1. **Raspberry Pi** using **CODESYS** software.
2. **LOGO!** using Siemens **LOGO!Soft Comfort** software.
3. **EasyPort** using **FESTO FluidSim**.

In every controlling method, the hardware design and connections to the assembly line beside its coding will be elaborated in much details further considering the comparison between all depending on below factors:

- **Safety** of the controller itself to decrease out of service time.
- **Compactness** of the method in total to provide the full controlling.
- **Response Time** which is the time from sending the signal till achieving it. In this thesis, it will be the time from pressing the start push button till the product reaches its required target.
- **Capacity of the IOs**. Which means total allowable number of inputs and outputs the controller can control.
- **Cost** of the full controller solution.
- Required human **effort** to build the hardware and to code through the software.
- How is Human machine **interfacing**? .Whether this can be done locally near the assembly line or remotely.
- **Accessibility** to the required IOs for maintenance or during the fault investigation.

Chapter 2

MecLab

It is a three Electropneumatic stations manufactured by **FESTO** company. The purpose of MecLab® is to provide an educational opportunity to study the practical and theoretical aspects of Electropneumatic automation technology. It consists of three separate systems that can be used all in combination or as stand-alone separately stations.

These three stations are handling station, system conveyor station, and stacking magazine station. They are educational models of typical methods to be noticed in various industrial automated production plant. [3]

2.1 Stations

2.1.1 Stack Magazine Station

The stacking station keeps products that can be stamped with the stamping unit. In an automated production line, work pieces are stored and fed into the process in a specific timed sequence. Thus, the main aims of this Station are storing, transferring, and passing each work piece according to the controller code.

This is the first station in experimental system. It has the following components:

- 1 Single acting cylinder.
- 1 double acting cylinder equipped with 2 proximity sensors. One sensor for fully retracted position, while the second for fully extended rod.

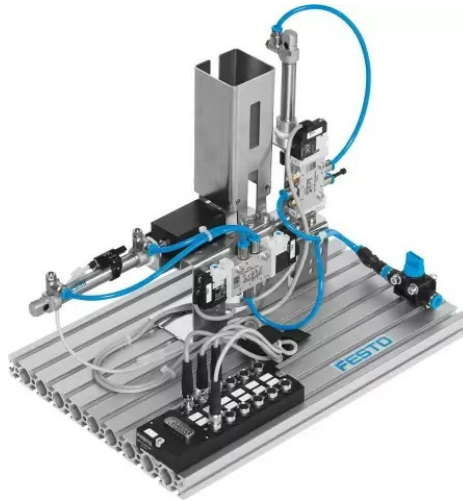


Figure 2.1: MecLab Stacking station

Below table illustrates all the IOs for this station.

IOs	Name	Comment
S1O1	Station 1, Output 1	Forward the double acting cylinder
S1O2	Station 1, Output 2	Retract the double acting cylinder
S1O3	Station 1, Output 3	Forward the single acting cylinder
S1I1	Station 1, Input 1	Double acting cylinder fully extended

Table 2.1: Station 1 IOs

The pneumatical and electrical setup is configured as below:

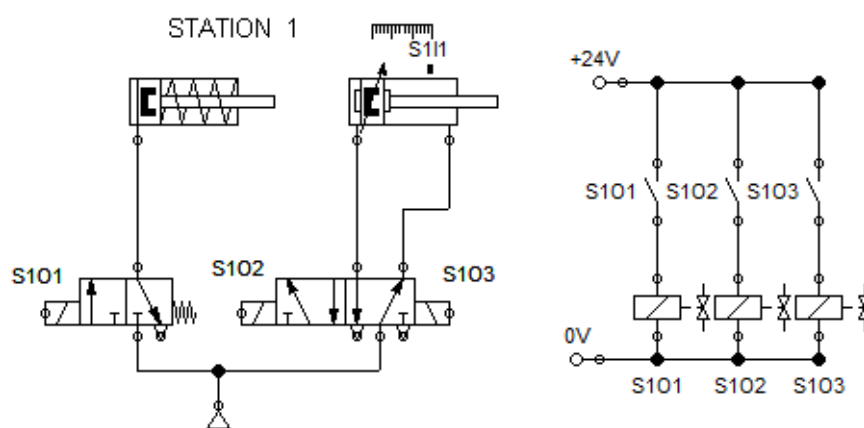


Figure 2.2: Station 1 pneumatical and electrical setup

All these IOs are being accumulated in a single node to provide a single Multi-pin plug distributor, plug Sub-D, 15-pin, with socket M8 for supplying IOs, 3-pin common

grounded for controlling as below.



Figure 2.3: Station 1 Node

These 15 pins hold the IOs according to the assignment:

Pin	IOs	Wire Color
1	S1I1	White
2	S1O1	Brown
3	Not assigned	Green
4	S1O2	Yellow
5	Not assigned	Grey
6	S1O3	Pink
7	Not assigned	Blue
8	Not assigned	Red
9	Not assigned	Black
10	Not assigned	Pink Brown
11	Not assigned	Purple
12	Not assigned	White Brown
13	+24 V	White Green
14	0 V	White Yellow
15	0 V	White Grey

Table 2.2: Station 1 Pins assignment

2.1.2 Conveyor Station

In many production assemblies' lines, work pieces are transported between “process stations” via conveyor belts. The conveyor station in MecLab provides realistic simulation of an industrial workpiece transport system.

These conveyors are actuated by motors to rotate in one direction. To rotate reversely; mechanical, or electrical components are being used. In this station, two electrical change

coils are being used to rotate forward and in reverse, depending on the flow of the current through them.

In this station, Work pieces are detected using proximity sensor, then classified thanks to the color sensor which distinguishes if the piece is black or silver color, and directed using a solenoid coil.

This is the second station in experimental system. It has the following components:

- 1 DC Motor.
- 1 set of 2 switch coils.
- 1 solenoid Coil.
- 1 color sensitive proximity sensor.
- 1 proximity sensor.

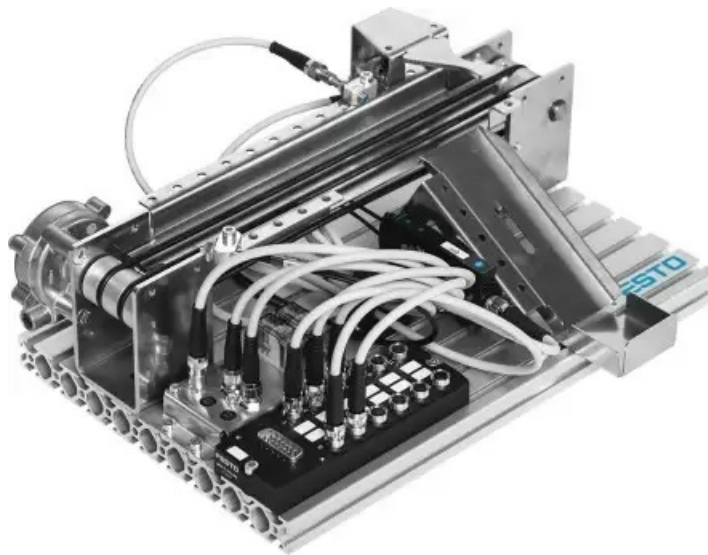


Figure 2.4: MecLab Conveyor Station

Below table illustrates all the IOs for this station.

IOs	Name	Comment
S2O1	Station 2, Output 1	Motor in backward
S2O2	Station 2, Output 2	Switch coils to move Motor forward
S2O3	Station 2, Output 3	
S2I1	Station 2, Input 1	Color sensitive proximity sensor
S2I2	Station 2, Input 2	Proximity sensor

Table 2.3: Station 2 IOs

The pneumatical and electrical setup is configured as below:

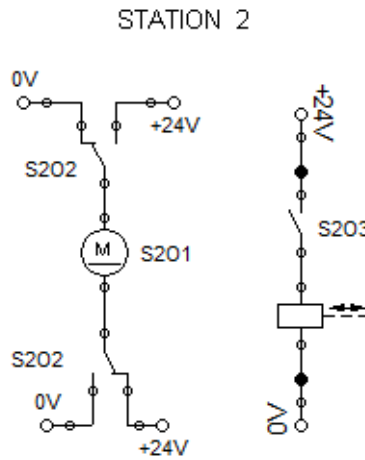


Figure 2.5: Station 2 pneumatical and electrical setup

All these IOs are being accumulated in a single node to provide a single Multi-pin plug distributor, plug Sub-D, 15-pin, with socket M8 for supplying IOs, 3-pin common grounded for controlling as below.



Figure 2.6: Station 2 Node

These 15 pins hold the IOs according to the assignment:

Pin	IOs	Wire Color
1	S2I1	White
2	S2O1	Brown
3	S2I2	Green
4	S2O2	Yellow
5	Not assigned	Grey
6	S2O3	Pink
7	Not assigned	Blue
8	Not assigned	Red
9	Not assigned	Black
10	Not assigned	Pink Brown
11	Not assigned	Purple
12	Not assigned	White Brown
13	+24 V	White Green
14	0 V	White Yellow
15	0 V	White Grey

Table 2.4: Station 2 Pins assignment

2.1.3 Handling Station

Whenever it is a simple pick place operation or highly complex assembly work, handling systems are always involved. The MecLab® Handling station consists of pneumatic cylinders with simple bearing guides and two axes. The work piece is being held with a standard gripper with two fingers. The system can be used to transport the work piece between 2 predefined points.

This is the third station in experimental system. It has the following components:

- 1 standard gripper with two fingers.
- 2 double acting cylinder equipped with 2 proximity sensors for each cylinder.

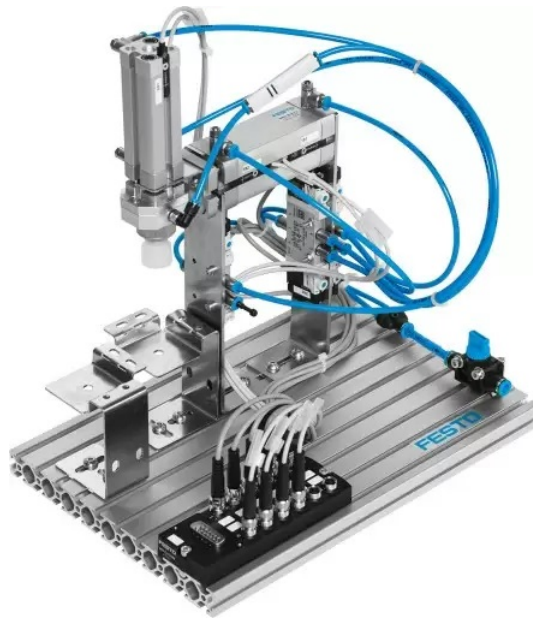


Figure 2.7: Main Cable

Below table illustrates all the IOs for this station.

IOs	Name	Comment
S3O1	Station 3, Output 1	Forward Horizontal double acting cylinder
S3O2	Station 3, Output 2	Retract Horizontal double acting cylinder
S3O3	Station 3, Output 3	Forward Vertical double acting cylinder
S3O4	Station 3, Output 4	Retract Vertical double acting cylinder
S3O5	Station 3, Output 5	Activate the Gripper
S3I1	Station 3, Input 1	Horizontal double acting cylinder fully extended
S3I2	Station 3, Input 2	Horizontal double acting cylinder fully retracted
S3I3	Station 3, Input 3	Vertical double acting cylinder fully extended
S3I4	Station 3, Input 4	Vertical double acting cylinder fully retracted

Table 2.5: Station 3 IOs

The pneumatical and electrical setup is configured as below:

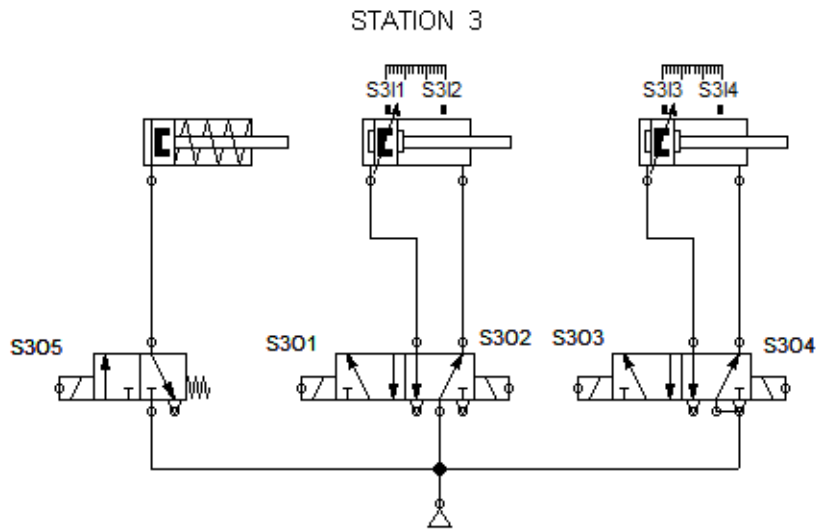


Figure 2.8: Station 3 pneumatical and electrical setup

All these IOs are being accumulated in a single node to provide a single Multi-pin plug distributor, plug Sub-D, 15-pin, with socket M8 for supplying IOs, 3-pin common grounded for controlling as below.



Figure 2.9: Station 3 Node

These 15 pins hold the IOs according to the assignment:

Pin	IOs	Wire Color
1	S3I1	White
2	S3O1	Brown
3	S3I2	Green
4	S3O2	Yellow
5	S3I3	Grey
6	S3O3	Pink
7	S3I4	Blue
8	S3O4	Red
9	Not assigned	Black
10	S3O5	Pink Brown
11	Not assigned	Purple
12	Not assigned	White Brown
13	+24 V	White Green
14	0 V	White Yellow
15	0 V	White Grey

Table 2.6: Station 3 Pins assignment

2.2 The electrical connection to controller

All the 7 input's and 11 output's signals from all three stations are accumulated in a single cable to provide a single 25 wire cable to the controller. Pin assignment was done according to below setup as in following table.

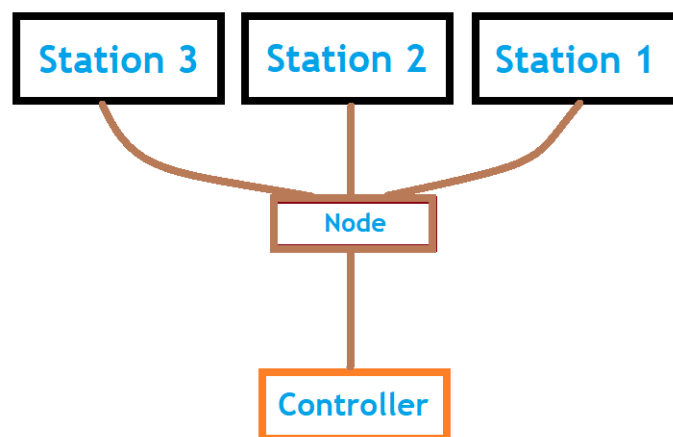


Figure 2.10: Main Cable

These 25 pins hold the IOs according to the assignment:

Pin	IOs	Wire Color
1	S3O1	White
2	S3O3	Brown
3	S2I2	Green
4	S3O4	Yellow
5	S3O2	Grey
6	S3I3	Pink
7	S3I4	Blue
8	S2I1	Red
9	S3I1	Black
10	S3O5	Pink Brown
11	S1I1	Purple
12	S2O1	White Brown
13	S2O3	White Green
14	0 V	White Yellow
15	S2O2	White Grey
16	S1O3	White Blue
17	S1O2	White Red
18	S1O1	White Black
19	S3I2	Yellow Brown
20	0 V	Green Brown
21	0 V	Blue Brown
22	0 V	Red Brown
23	0 V	Red Blue
24	0 V	Grey Brown
25	0 V	Grey Pink

Table 2.7: Main cable IOs assignment

2.3 Flow chart of the process

Whenever there is a press on the pushbutton “START”, a stored piece should be pushed from the first station forward toward the second station by the double acting cylinder “S1A” through releasing the single acting cylinder “S1B”. Both go to initial when the sensor “S1I1” is On.

The motor “S2M” goes forward by the relays control set “S2C” passes the piece to the color sensitive proximity sensor “S2I1”. If it Silver, the motor goes backward by making “S2C” off to reject it by the solenoid coil “S2R”. If it back, it goes from the second station proximity sensor “S2I2” to the specific position in the third station following the sequence for the horizontal cylinder “S3H”, the vertical cylinder “S3V”, and the Gripper “S3G”.

“ALARM” lamp indicator goes on when predefined number “N” of rejections happens. This alarm stays on till the next press on “Start”, as acknowledgment.

A virtual “Start” is needed to start the sequence remotely if possible.

A visual implementation of current status of the system is needed as well.

The displacement-step diagram can be shown as following:

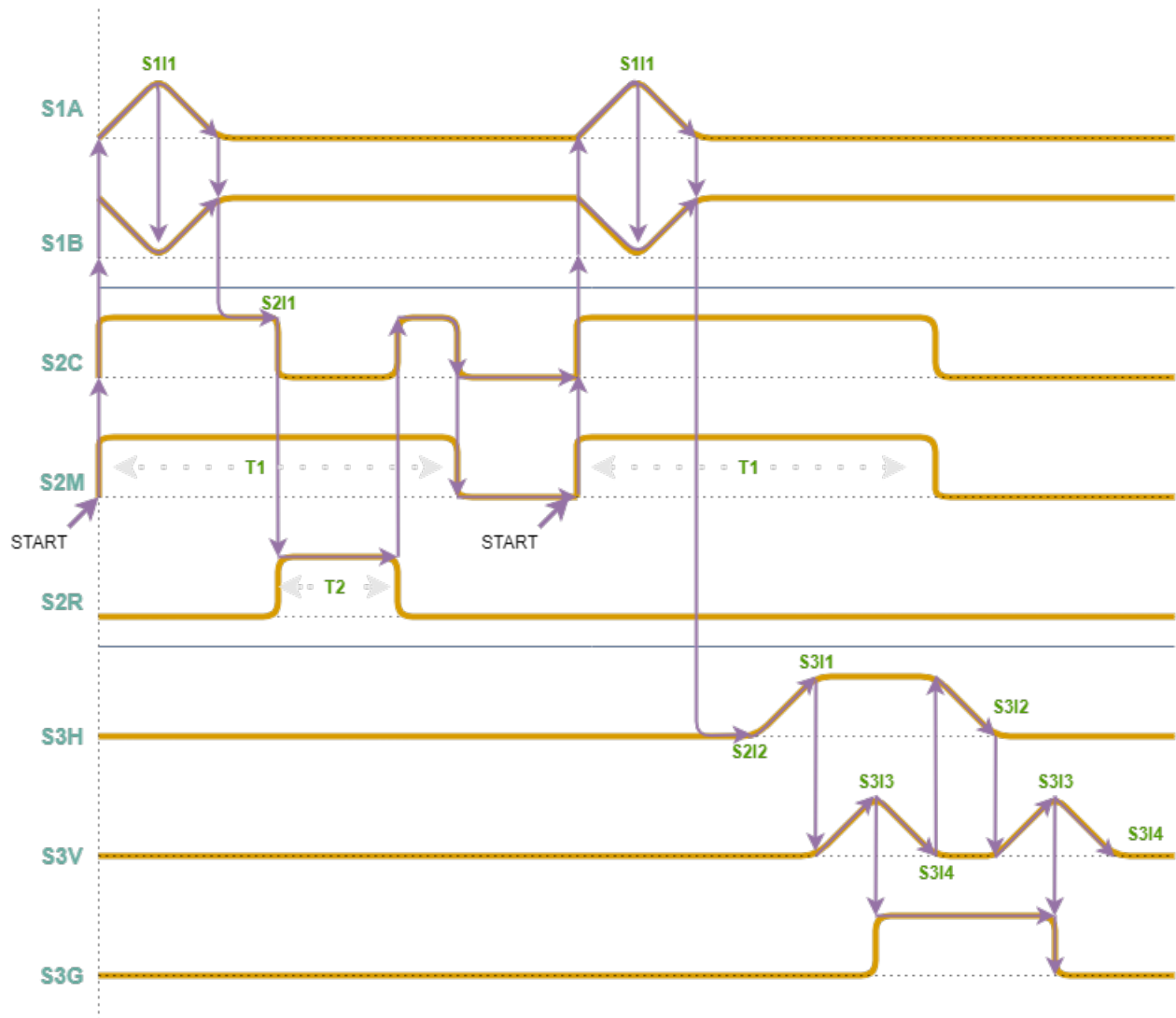


Figure 2.11: Flow diagram

2.4 Total Required IOs

Below table summarizes the number of needed IOs for this system.

Signals	Inputs	Outputs
Station 1	1	3
Station 2	2	3
Station 3	4	5
External IOs	1	1
Total IOs	8	12

Table 2.8: Required IOs

Chapter 3

Raspberry Pi

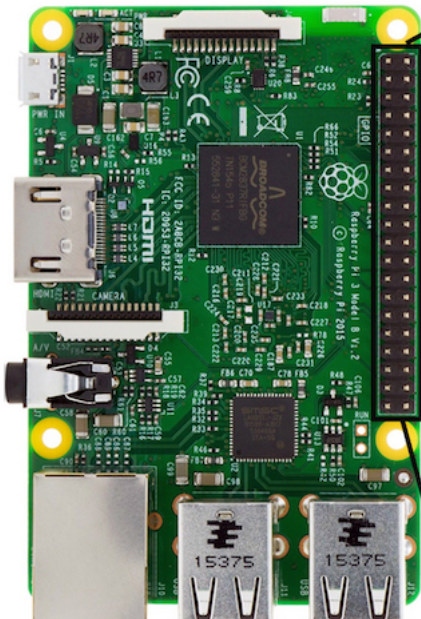
Nowadays the bulky Electronic brains are being replaced with a credit card sized computer- Raspberry pi. The raspberry pi board is nothing but a small powerful computer which contains a program memory (RAM), processor, UART, CPU, GPU, graphics chip, Ethernet port, GPIO pins, Xbee socket, both 5V and 3.3 V power source connector and various interfaces for other external devices. [6]

The Raspberry pi is used today by “**do it yourself**” users who can design, develop, and experiment with many new ideas, projects and applications.

3.1 IOs assignment (GPIO)

Raspberry pi has the ability to handle up to **17 IOs** thanks to its **GPIOs**. Each GPIO can be programmed separately as digital Inputs to receive 0-3.3 V signals, or as digital output to send 0-3.3 V signals.

Note: Although there are more than 17 GPIO, some of them are permanently fixed for other functions. Thus, these are not accessible to be programmed for assignment.



	Pin No.		
3.3V	1	2	5V
GPIO2	3	4	5V
GPIO3	5	6	GND
GPIO4	7	8	GPIO14
GND	9	10	GPIO15
GPIO17	11	12	GPIO18
GPIO27	13	14	GND
GPIO22	15	16	GPIO23
3.3V	17	18	GPIO24
GPIO10	19	20	GND
GPIO9	21	22	GPIO25
GPIO11	23	24	GPIO8
GND	25	26	GPIO7
DNC	27	28	DNC
GPIO5	29	30	GND
GPIO6	31	32	GPIO12
GPIO13	33	34	GND
GPIO19	35	36	GPIO16
GPIO26	37	38	GPIO20
GND	39	40	GPIO21

Figure 3.1: Raspberry GPIO

Since the system needs 20 IOs to control each IO fully and separately, there are only 17 GPIO available for using. In order to compensate this issue, 3 IOs will be dependent on another independent IOs.

From the flow chart of the process in figure 7.1, it can be found out these points:

- Solenoid coil of the single acting cylinder in station 1 “S1O1” follows the retracing solenoid coil of the double acting cylinder in station 1 “S1O3”. Thus, only one output signal is needed to control both.
- In station 3, Each double acting cylinder receives 2 signals to extent or retract, while these signals are always the opposite. Thus, only 1 independent output can be used one solenoid coil and the second solenoid coil dependently uses the negative of the signal. Below table illustrates it.

Movement	First solenoid Coil	Second solenoid Coil
Extension	On	Off
Retracting	Off	On

Table 3.1: Complement solenoid Coils

By implementing these reductions, only 17 GPIOs are needed. Optionally they can be assigned as in the following table.

GPIO	IOs	COMMENT
GPIO18	S1O1	Pushing cylinder advance
GPIO23	S1O2	Pushing cylinder retract
-	S1O3	Same as S1O1
GPIO12	S2O1	Motor in backward
GPIO25	S2O2	Reverse Motor
GPIO24	S2O3	Rejection Coil
GPIO21	S3O1	Vertical cylinder
NOT GPIO21	S3O2	Complement of GPIO21
NOT GPIO20	S3O3	Complement of GPIO20
GPIO20	S3O4	Horizontal cylinder
GPIO16	S3O5	Gripper
GPIO4	ALARM	Alarming Lamp
GPIO19	S1I1	Pushing cylinder advanced
GPIO13	S2I1	Color sensor
GPIO6	S2I2	Availability sensor
GPIO17	S3I1	Horizontal cylinder retracted
GPIO27	S3I2	Horizontal cylinder advanced
GPIO22	S3I3	Vertical cylinder retracted
GPIO5	S3I4	Vertical cylinder advanced
GPIO26	Start	Start Pushbutton

Table 3.2: IOs Assignment for RPI

3.2 Hardware implementations

All IOs from stations are accumulated in a single cable to the Raspberry pi's boards. There will be a board for inputs and another one for outputs as shown below. Third board was used to extend the IOs and the power supply lines "3.3 - 0V" from the Raspberry pi.

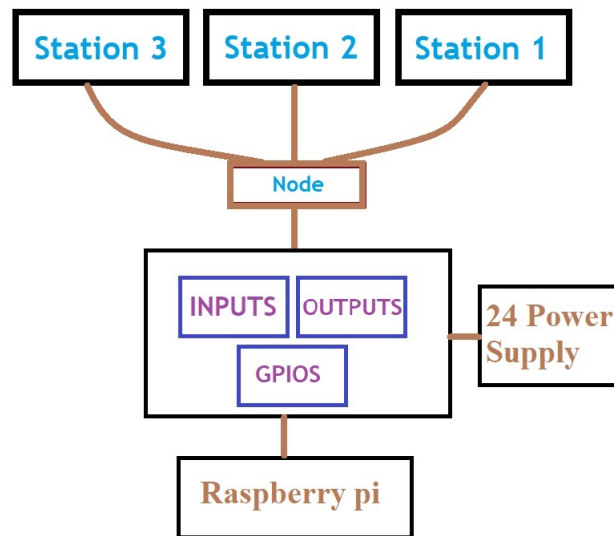


Figure 3.2: Raspberry Pi connection

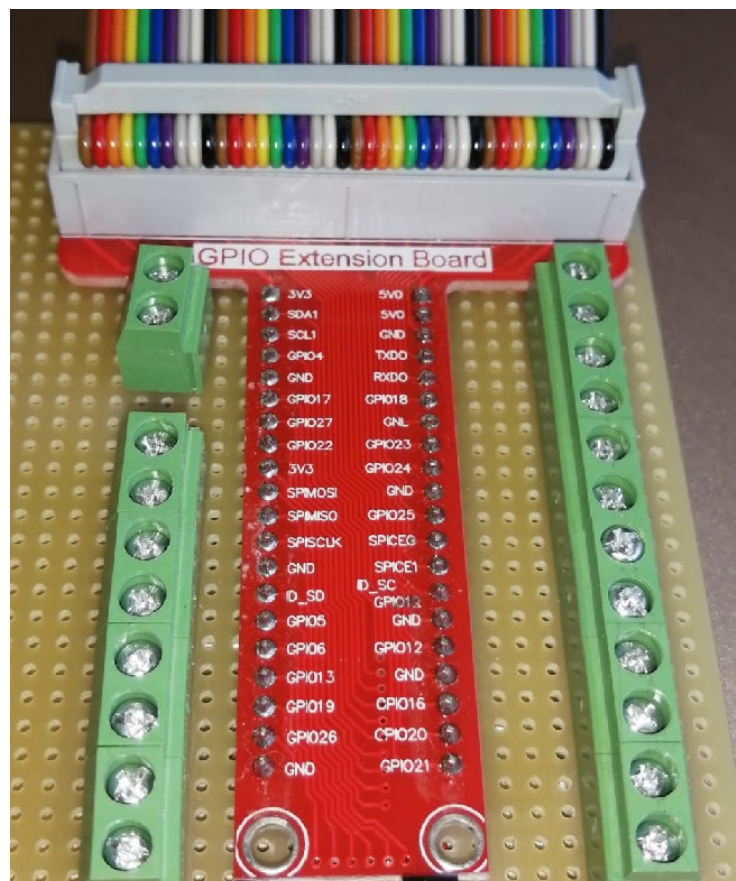


Figure 3.3: Raspberry extension

3.2.1 Opto-coupler

Voltage **conversion** is need; Because all the IOs for the system use 24 V, while the Raspberry pi uses 3.3 V. There are many methods to do it, however to completely separate the 24 VDC from the 3.3 VDC without sharing their grounds, using Opto-coupler is great choice. By considering using Opto-coupler, the Raspberry pi will be totally isolated from the 24V; which is preferable.

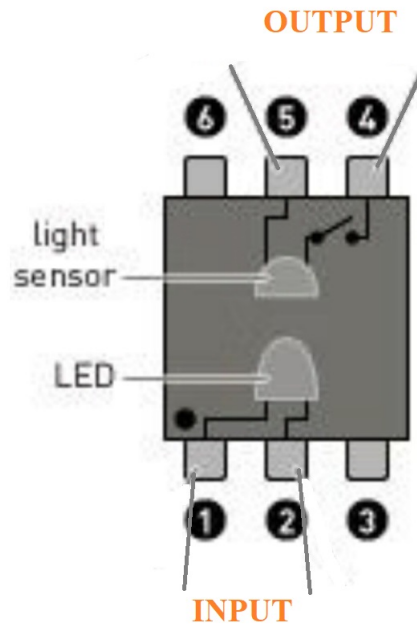


Figure 3.4: Opto-coupler

Inputs Board:

Inputs are supplied by 24 V. When they are ON, they send 24 V. To let the Raspberry pi receive this ON status, It is required to have 3.3 V to the GPIO pin.

From the opto-coupler's input side, voltage should be with the range of 1.3 V to bridge the output "short circuit". Voltage divider law is used to reduce the voltage from 24 to be 1.3 V considering there is voltage drop for the building transistor inside the optocoupler itself of 0.7 V.

from **Ohm's law**:

$$\text{Minimum resistance} = \text{Minimum voltage} / \text{Maximum current}$$

$$\text{Required voltage} = 24 - 1.3 - 0.7 = \mathbf{22\ V}$$

$$\text{Required current} = \mathbf{10\ mA\ [1]}$$

$$\text{Then resistance} = \mathbf{2.2K\ Ohm}$$

Thus, **2.4K Ohm** will be used to tolerate.

From the opto-coupler's output side, depending on Current divider law, 10K Ohm resistance will be used to let the GPIO receive 0 V as default. In case of ON input, the current will flow to GPIO through the 1K Ohm forcing it to sense ON.

The input board scheme will is following:

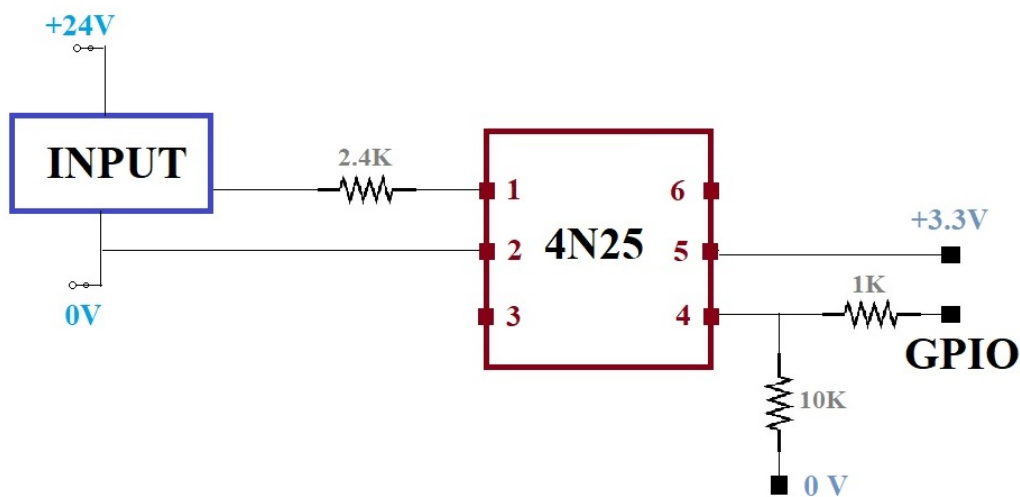


Figure 3.5: Opto-coupler for Input

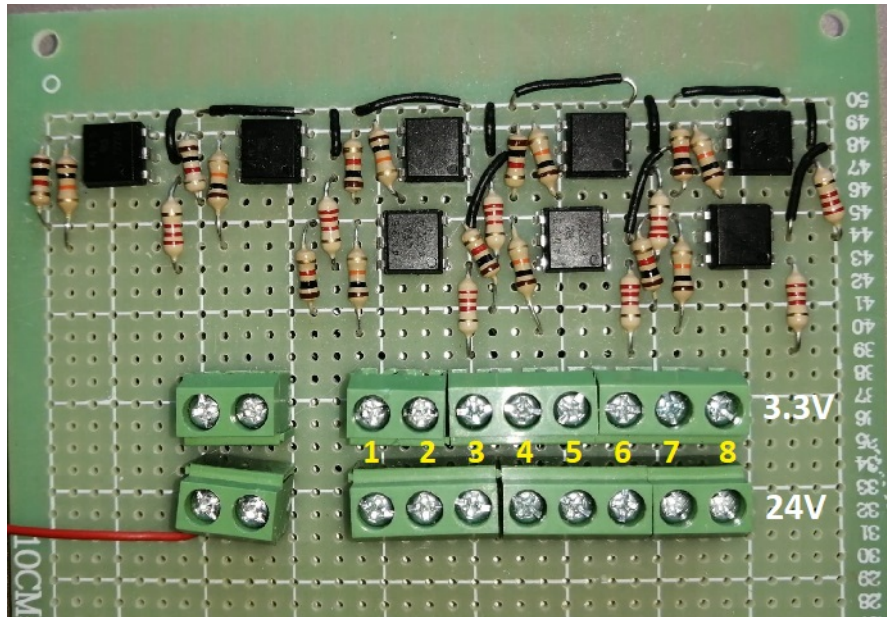


Figure 3.6: Input board scheme

Pins assignment are done according to the following table:

GPIO	IOs	Pin
GPIO6	S2I2	1
GPIO22	S3I3	2
GPIO5	S3I4	3
GPIO13	S2I1	4
GPIO17	S3I1	5
GPIO19	S1I1	6
GPIO27	S3I2	7
GPIO29	Start	8

Table 3.3: Input board Pins assignment

Outputs Board:

Outputs are supplied by 0 V while they need only +24V to operate. When the GPIO from the Raspberry pi is ON, it sends 3.3 V. To let the output receive this ON status, It is required to connect the +24V; which is done through the optocoupler.

From the opto-coupler's input side, voltage should be with the range of 1.3 V to bridge the output "short circuit". Voltage divider law is used to reduce the voltage from 3.3 to be 1.3 V considering there is voltage drop for the building transistor inside the optocoupler itself of 0.7 V.

from **Ohm's law**:

Minimum resistance = Minimum voltage / Maximum current

Required voltage = $3.3 - 1.3 - 0.7 = \mathbf{1.3\ V}$

Required current = **50 mA** [1]

Then resistance = **26 Ohm**

Thus, **39 Ohm** will be used to tolerate.

The output board scheme is as following:

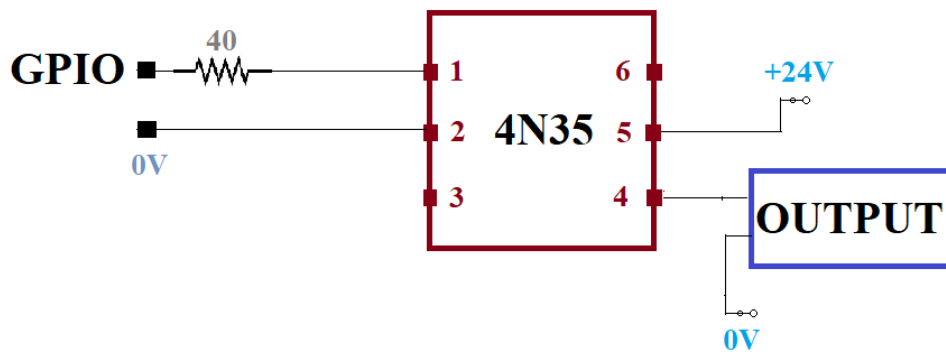


Figure 3.7: Opto-coupler for Output

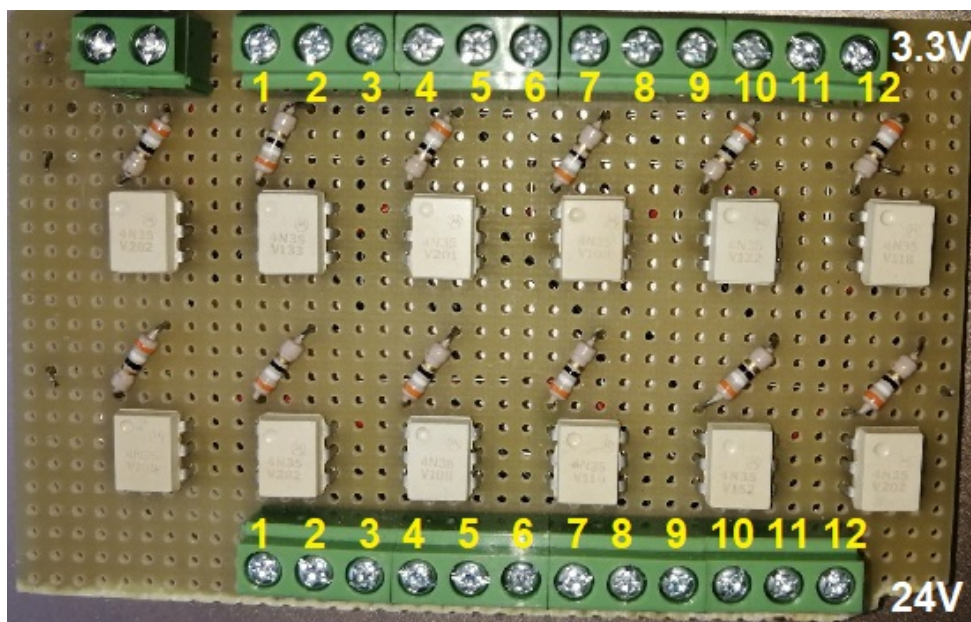


Figure 3.8: Output board scheme

Pins assignment are done according to the following table:

GPIO	IOs	Pin
GPIO21	S3O1	4
GPIO22	S3O3	12
GPIO20	S3O4	5
GPIO13	S3O2	11
GPIO16	S3O5	1
GPIO12	S2O1	2
GPIO24	S2O3	6
GPIO25	S2O2	3
GPIO4	ALARM	9
GPIO23	S1O2	8
GPIO18	S1O1	10

Table 3.4: Output board Pins assignment

3.2.2 Outputs Reduction by the complement

As stated before, there will be a compensation for some outputs depending on other outputs. A transistor is the cheapest and easiest solution to generate the complement of the input.

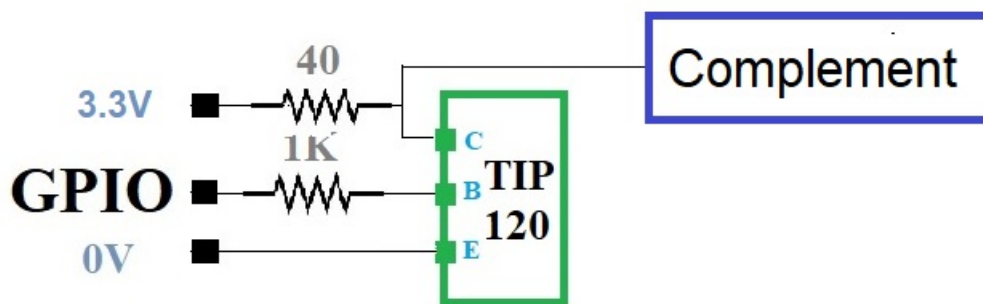


Figure 3.9: Output complement scheme

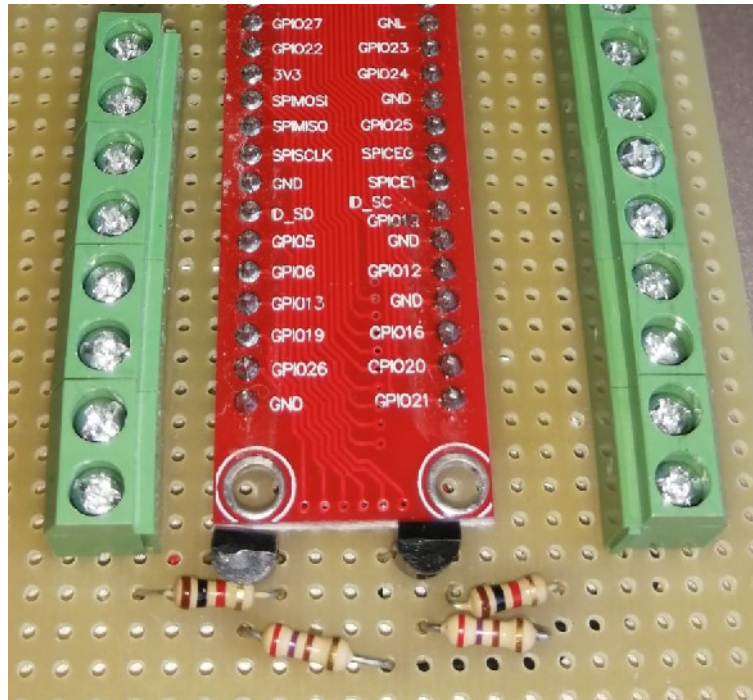


Figure 3.10: Output complement using TIP120

3.3 Software Implementation

In order to communicate to the Raspberry Pi via the use of PC, Codesys software is needed. Through Codesys, all GPIOs can be configured, assigned, and monitored. However, there should be always a link between the Codesys and the Raspberry Pi to do so.

Free version for educational purposes can be installed from the internet. It works fine except it has a time limitation of 2 hours. Re launching is needed to keep it monitoring the Raspberry pi.

To reach the final goal, It is necessarily to proceed step by step and solve several steps:

1. **Configuring** the Raspberry Pi.
2. **Recognition** of the RPI device by the PC.
3. **Designing** of the circuit on the breadboard.
4. **Configuring** the Raspberry Pi on Codesys.
5. **Creation** of the ladder network.
6. **"Visualization"** interface.
7. **Results**.

3.3.1 Configuring the Raspberry Pi

Before proceeding to the milestone of the project, operating system inside the Raspberry Pi must be installed. It is possible to choose between different "**open-source**" software available:

- Raspbian
- Kali Linux
- Pidora
- Ubuntu Core
- Arch Linux ARM

In this thesis install **Raspbian** was chosen as it is one of the more stable and robust versions than the Linux distribution offers. This operating system is particularly suitable for home automation, as it is tailor-made optimized for hardware.

The procedure

1. Downloading NOOBS which can be found from internet to the PC. Two different versions of it can be found:
 - NOOBS: an "installer" that allows to easily install and configure Raspbian.
 - NOOBS Lite: an "installer" which, during installation, allows to connect to the Internet and download the desired operating system.

For simplicity, it was preferred to use NOOBS in order to have a higher speed during the installation process. Once the zip file is been extracted, a folder with the name: NOOBS will be obtained.

2. Formatting the micro SD card:

It is necessary to have a Micro SD card, which will be the Raspberry Pi's hard disk. This little card must have precise characteristics. The choice depends on two factors:

- Memory capacity
- Class of the Memory

Class identifies the minimum transmission speed. It is evaluated in numbers from 2 to 10: plus the number is higher, the higher the quality. In this case it is important to have a high class: therefore class 10 is chosen to use a micro SD from 32GB.

Formatting the Micro SD card using a suitable data formatting program. In this case the SD Memory Card Formatter was used. It is available free of charge on internet.

3. Copying NOOBS_xx to the micro SD card

To transfer the file to the micro SD it is necessary to proceed through a writing program. In this case, Win32DiskImager was used.

Through it, writing the NOOBS file on the micro SD can be done. After choosing the previously downloaded NOOBS folder and choosing the drive to which the micro SD is associated on the PC, writing the image can be done.

4. Removing the micro SD card and inserting it into the Raspberry Pi

5. Connecting the monitor, keyboard, and mouse to the Raspberry Pi to be used during installing the operating system as in below schema. Since Wireless connection will be depended on, no Ethernet cable is needed.

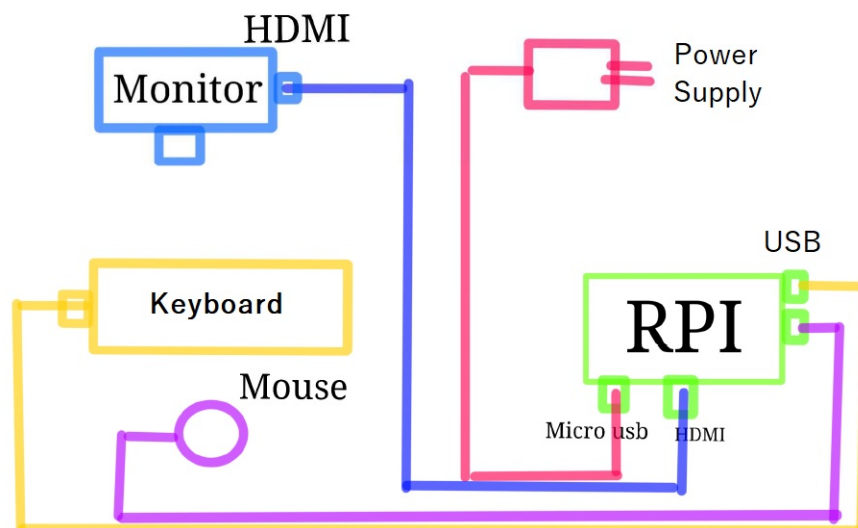
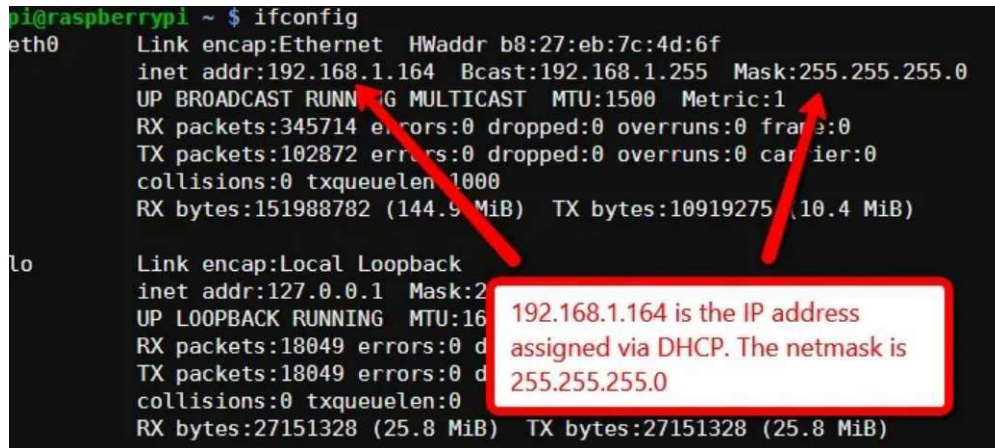


Figure 3.11: Configuration connection schema

6. After switching on and following the directions the full desktop version operating system can be installed.
7. After rebooting, the graphical interface starts to configure the time zone, administrative password and Wifi network. The wifi network should be the same to which the Codesys PC is connected.
8. The IP of the PC is assigned automatically and seen by RPI through a protocol called DHCP (Dynamic Host Configuration Protocol). The IPv4 address that in this case is: **192.168.0.1**

9. Back to the monitor of the RPI, calling the terminal to assign a STATIC IP so that the PC recognizes this IP without having to follow each possible change.

To see which IP address is assigned by default to the RPI, it is needed to run this command on the terminal: **ifconfig**



```
pi@raspberrypi ~ $ ifconfig
eth0      Link encap:Ethernet  HWaddr b8:27:eb:7c:4d:6f
          inet addr:192.168.1.164  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:345714 errors:0 dropped:0 overruns:0 frame:0
          TX packets:102872 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:151988782 (144.9 MiB)  TX bytes:10919275 (10.4 MiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.255.255.0
          UP LOOPBACK RUNNING  MTU:16384  Metric:0
          RX packets:18049 errors:0 dropped:0 overruns:0 frame:0
          TX packets:18049 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:27151328 (25.8 MiB)  TX bytes:27151328 (25.8 MiB)
```

Figure 3.12: Assigned IP address

Since the RPI is keeping connecting to the Wifi, it keeps its same IP which will be used in Codesys.

10. To confirm the PC is connected to the RPI, "Ping" command followed by the RPI's IP is used from the PC terminal.
11. RPI communicates with Codesys:

Codesys has implemented a package that allows to be able to do this new interfacing. However, for this to happen both the Codesys and the Codesys package must belong to the same version. To download this version, it can be found after registering on the Codesys website. **"CODESYS Control for Raspberry Pi MC SL"** version was selected for installation since it is more complete. It is possible to use it through a demo version, free, but for a limited time. Its validity is about two hours: this implies that; if it passes, downloading the package again is must.

12. After installing Codesys, **"update Raspberry Pi"** under **"Tools"** is needed. Also login credentials are needed: entering credentials of the RPI previously saved (Username, Password) and selecting target: entering STATIC IP address previously noted or it can be found by scanning it. Then Proceeding to Codesys Runtime Package and **"Install"**, under **"Standard"**.

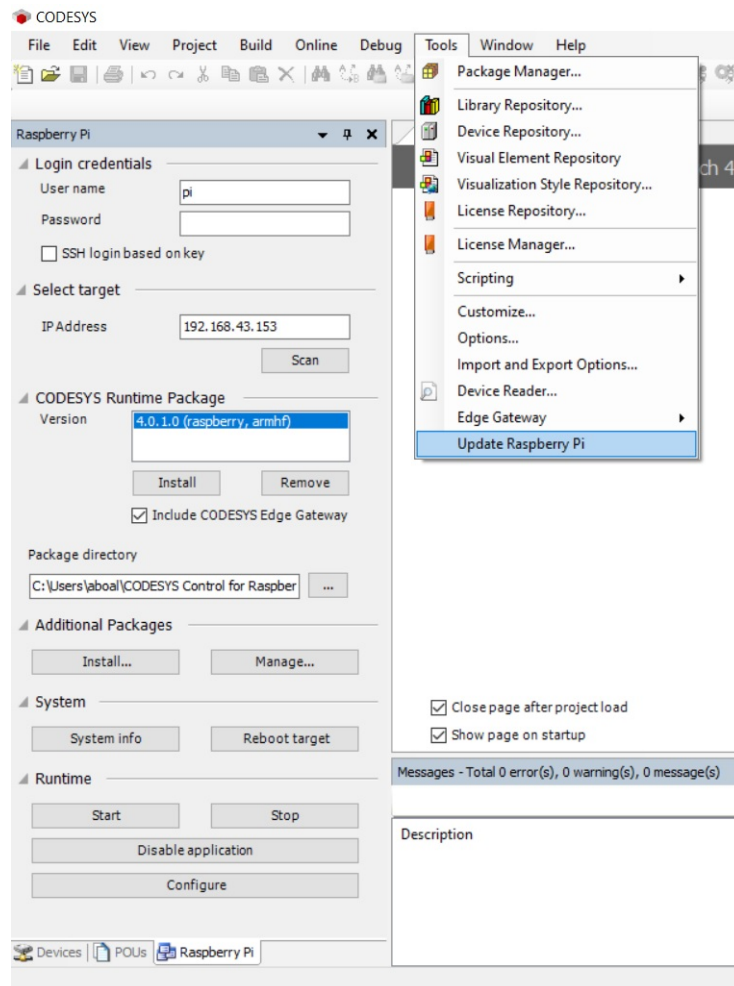


Figure 3.13: Updating the RPI

3.3.2 Writing the Code

- Under “File”, “New Project”, “Standard Project”, the name can be specified.

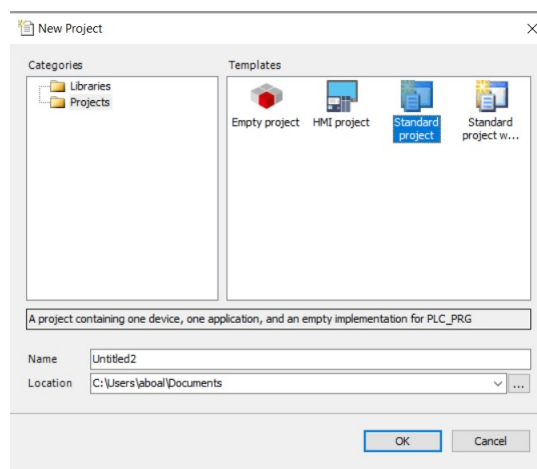


Figure 3.14: New Project Creation

- Under Device, “CODESYS Control for Raspberry Pi SL” is selected. PLC_PRG is “Ladder Logic Diagram (LD)”.
- This will open a project on the device: RPI. It is possible to note the presence of the following within the RPI hardware:
 - I2C
 - SPI
 - GPIOs_A_B
 - Camera Device
- Clicking on **PLC_PRG (PRG)** to create the LADDER NETWORKS.
- Considering the requirements from the logic flow diagram, networks were created as in the Appendix A 8.1.
- Selecting GPIOs Configuration whether input or output is done under “**GPIOs Parameters**”.

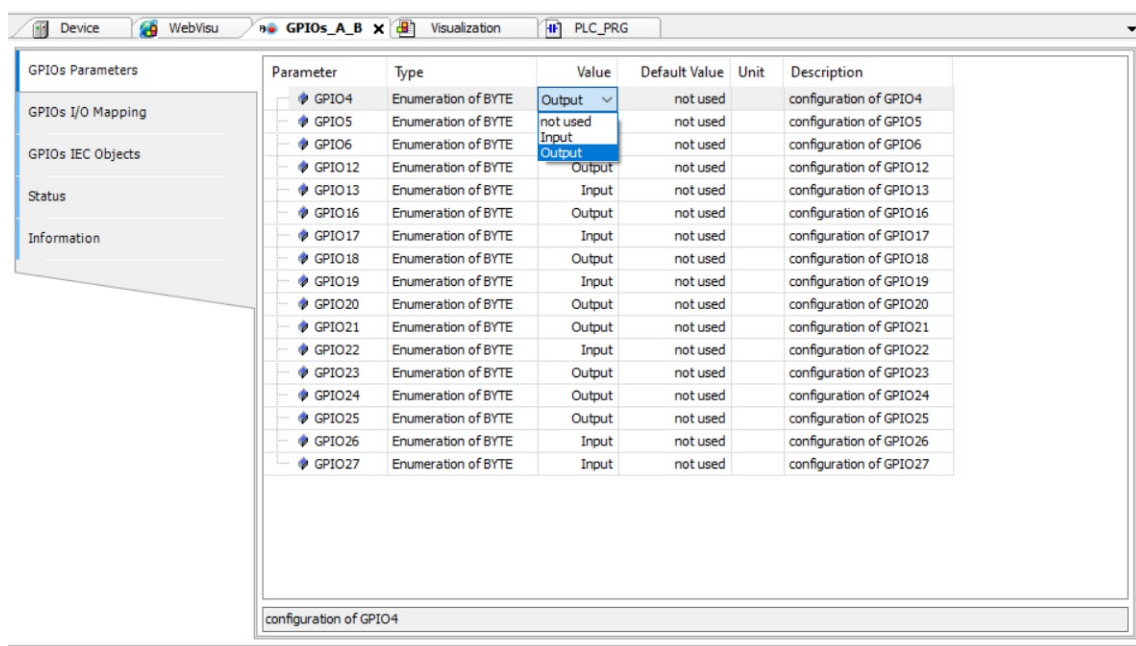


Figure 3.15: GPIO configuration

- Selecting GPIOs Mapping depending on the names from the networks is done under “GPIOs I/O Mapping”.

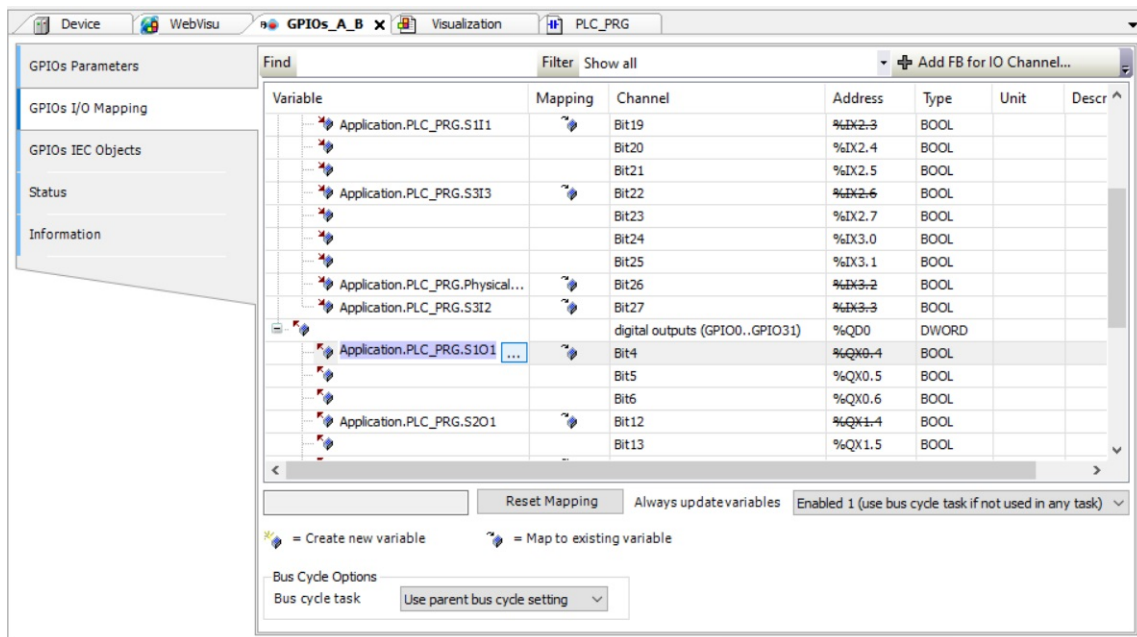


Figure 3.16: GPIO Mapping

- Then, configuring the device for the project is needed. It is done by going to Device (CODESYS Control for Raspberry PI SL), clicking on "**Browse the network**", then Codesys will recognize the device under "**raspberrypi**".

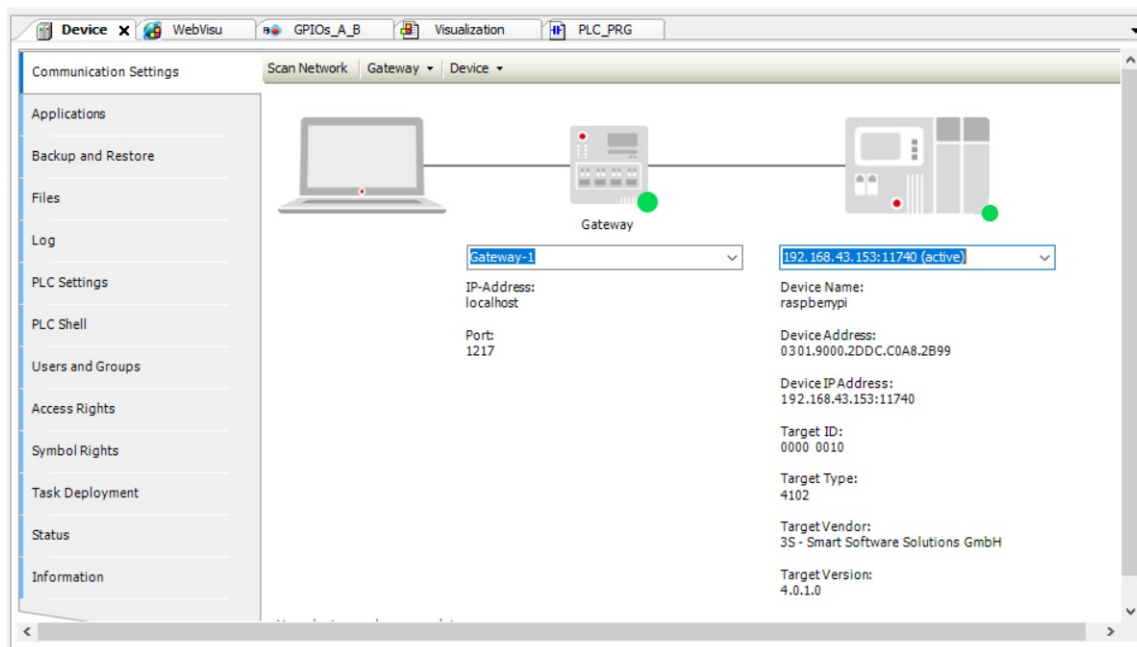


Figure 3.17: Raspberry Pi Network

- After doing all, by pressing "**Compile**", "**Login**" and "**play**" the instant monitoring for IOs can be.

3.3.3 Remote controlling

- - Thanks to **Codesys**, It can easily establish remote visualization and control. This is done by choosing from many styles and options which all are supported.
- - It can be created under “**Application**”, “**Add object**”, then “**Visualization**”.

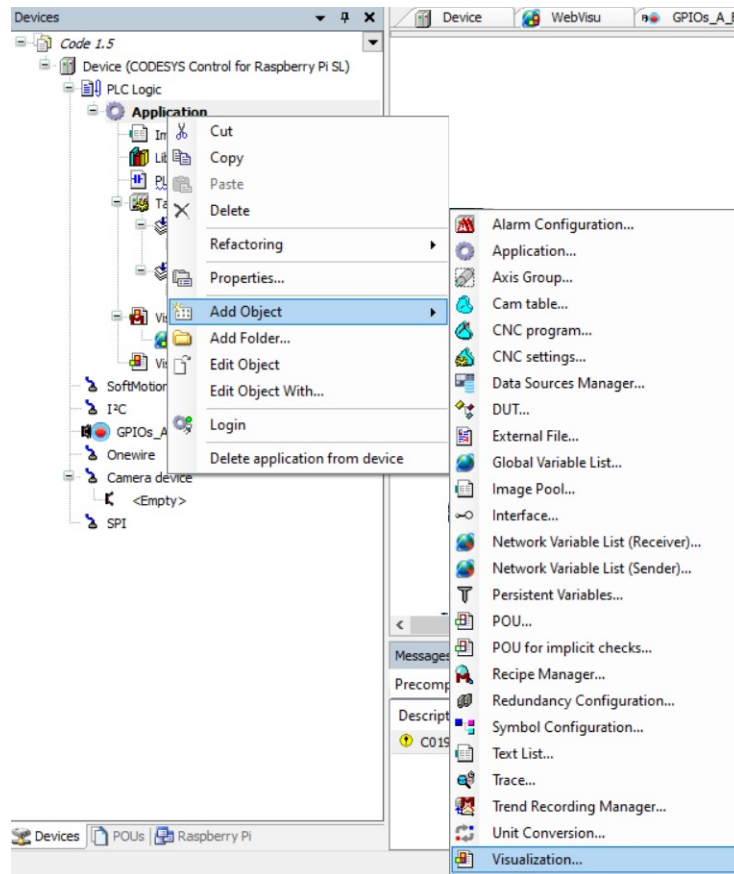


Figure 3.18: Visualization

- “Push Switch” to act as the **virtual START** and “Lamp” to indicate the status of the output can be added to the visualization following these steps:



Figure 3.19: Creating Visualization

- Assigning the variable to each IOs can be made as illustrated in below steps.

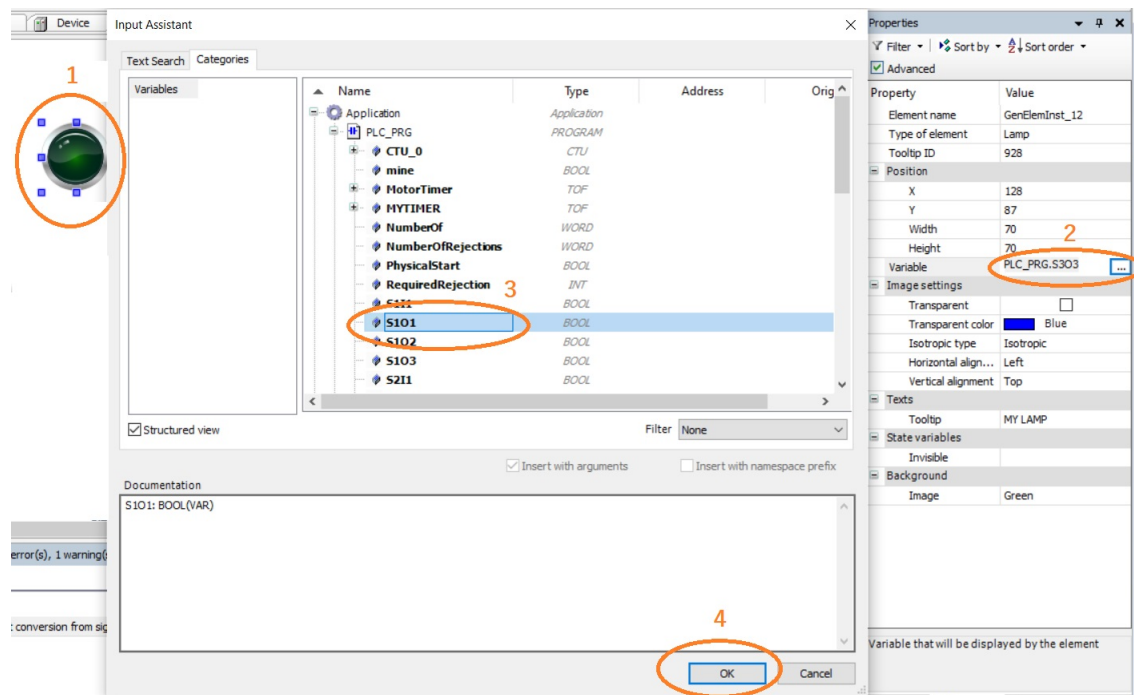


Figure 3.20: Assigning IOs to Visualization

- Texting Labels are added as below

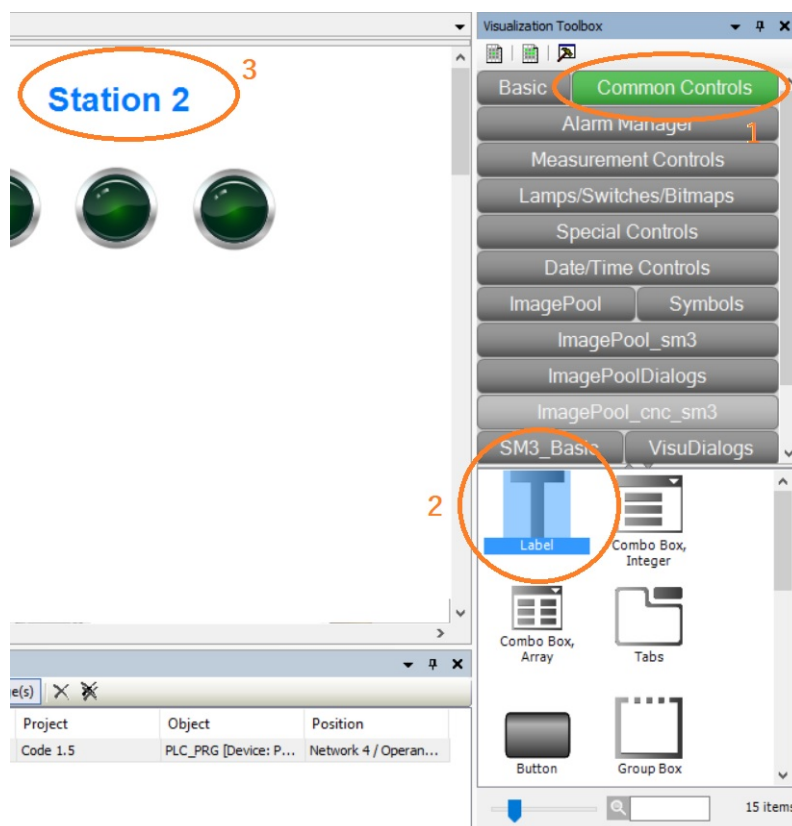


Figure 3.21: Texting

- Images can be added also to the graphic by these steps:

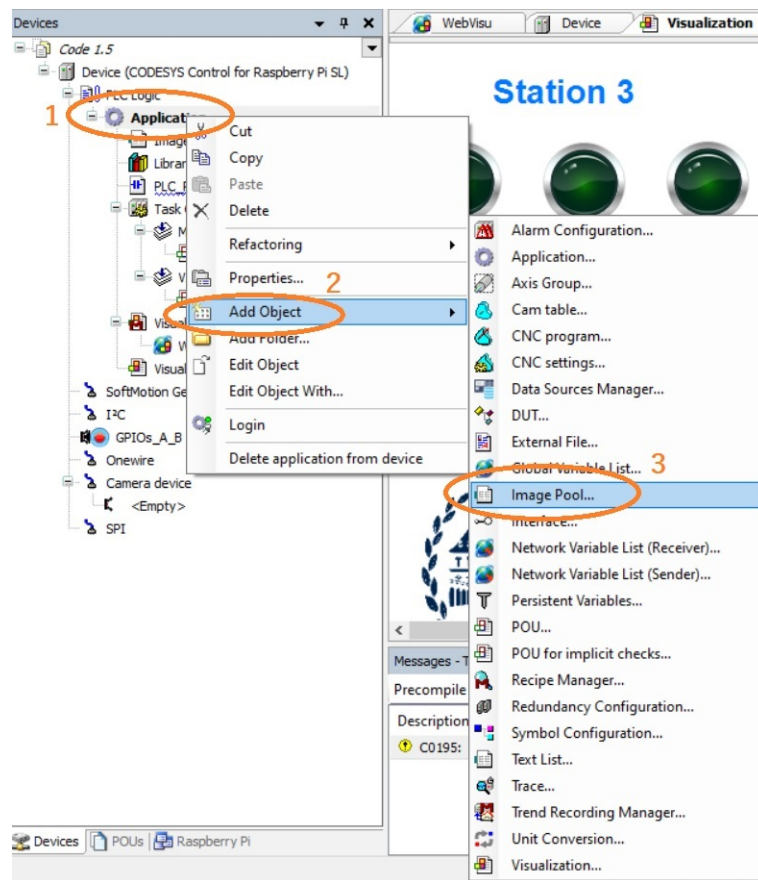


Figure 3.22: Adding images

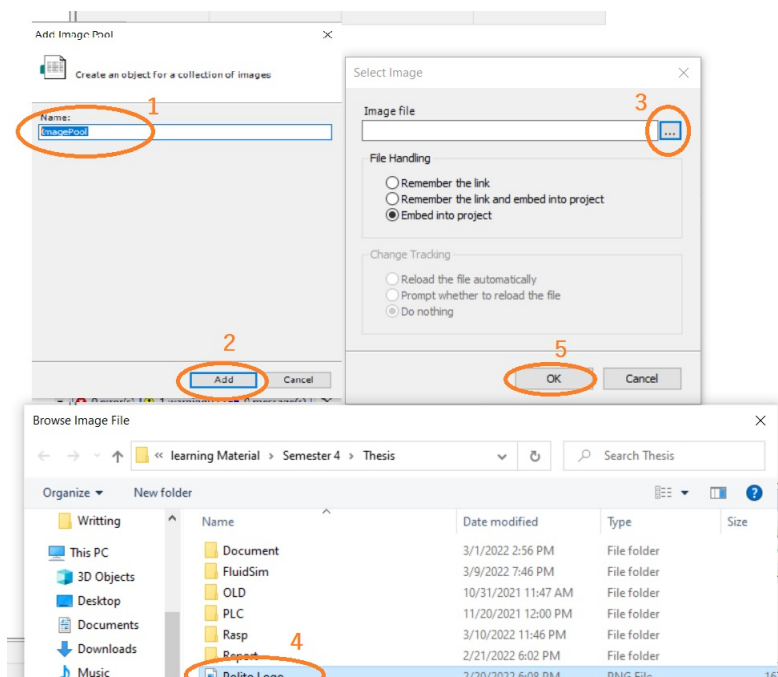


Figure 3.23: Adding images to project Visualization

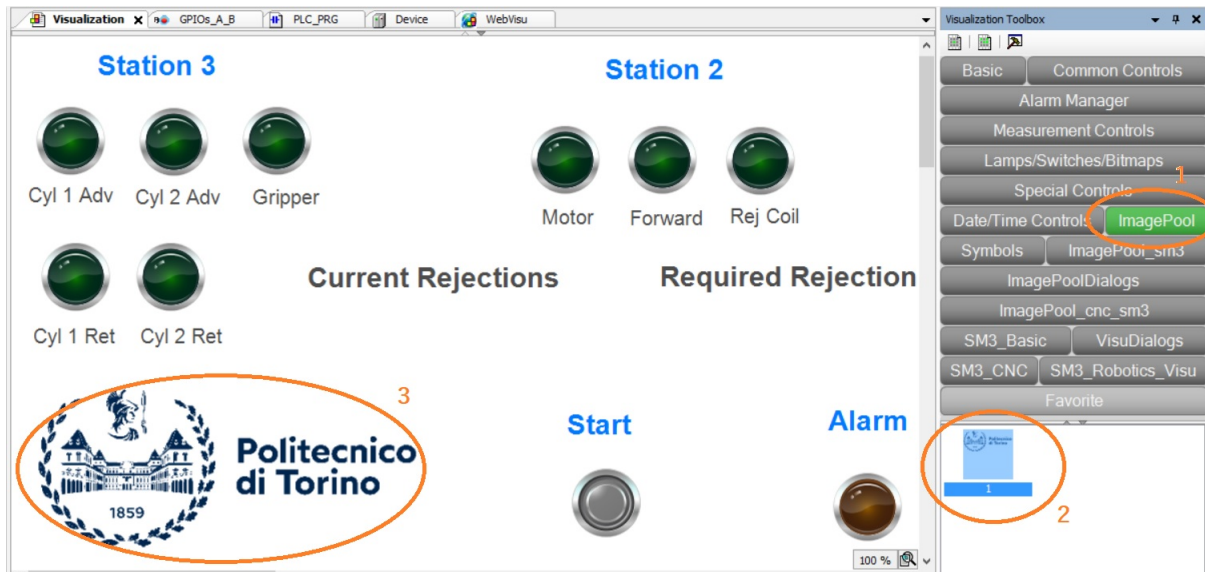


Figure 3.24: Selecting image for Visualization

- Dynamic variables can be added to the graphic by firstly, adding “Labels”. Then putting the sign “%I” to indicate the integer variable. Lastly, assigning the required variable as in below figure:

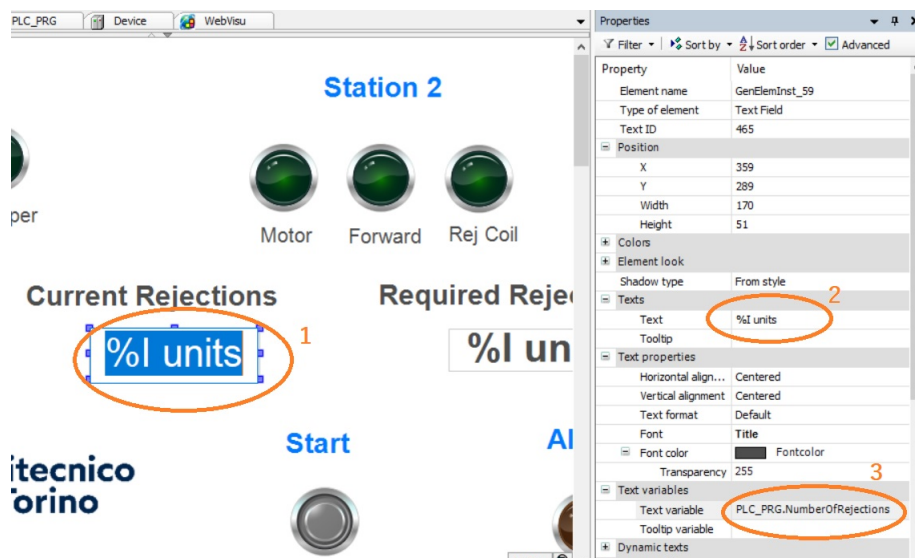


Figure 3.25: Dynamic variable

- For entering arbitrary input by the user, dynamic variable must be set as before. In addition to that, following steps must be followed.

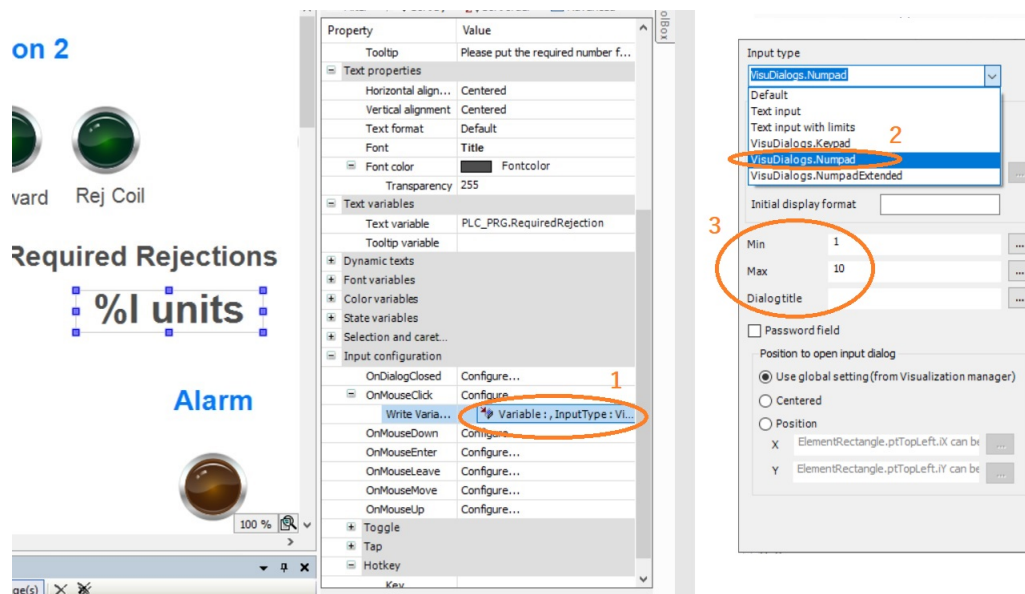


Figure 3.26: Entering optional number

- To access the designated graphic remotely from any device through an internet browser, the device must be in same network with the RPI. Then accessing to the un-safe link: **http://IP + :Port + /webvisu.htm**

Example:

http://192.168.43.153:8080/webvisu.htm

Note: Secure link “https” does not allow accessibility.

3.4 Networks of the code

Following the proposed scenario in 2.11 “flow chart of the process”, networks will be explained.

- **Network1:** The output coil “Start” will be ON, if there is any signal from “PhysicalStart” from the panel or “Virtual Start” from the remote control to start the sequence.



Figure 3.27: RPI Network 1

- **Network2:** Whenever “Start” coil is ON, the double acting cylinder in station 1 “S1O2” advances till it reaches its maximum thanks to the latching. When the full advance sensor gives signal “S1I1”, this coil goes OFF.



Figure 3.28: RPI Network 2

- **Network3:** When the double acting cylinder goes advancing, its retracting solenoid coil “S1O1” goes OFF. Also the single acting cylinder does the same. This is been implemented by a NOT gate of “S1O2”.



Figure 3.29: RPI Network 3

- **Network4:** Whenever the rejection coil “S2O3” goes ON, a counter counts up. The preset value is been supplied through the visualization as “RejuiredRejection”, while its default is 2. Current number of the counter “CV” is being shown through visualization as “NumberOfRejection”. When the CV equals to PV, “Alarm” of “S1O3” goes ON. The counter is reset when there is coil Start goes ON while the Alarm is ON.

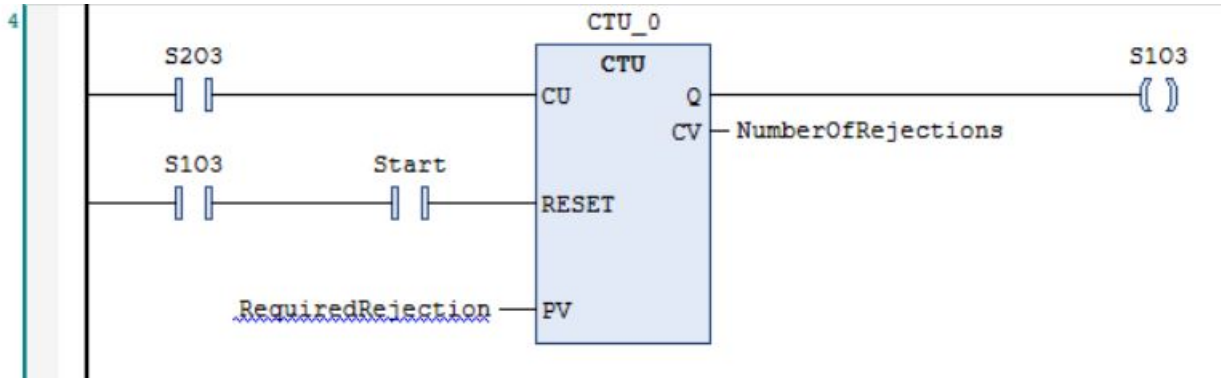


Figure 3.30: RPI Network 4

- **Network5:** When the color sensor “S2I1” detects the silver color, it goes ON to enable timer OFF “T1” which goes ON till it reaches its preset time in “PT”.

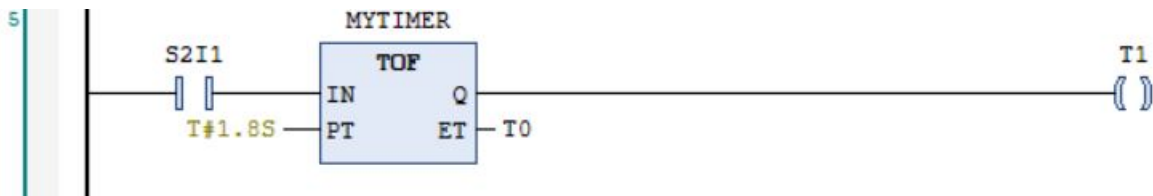


Figure 3.31: RPI Network 5

- **Network6:** When timer T1 has signal of ON, the motor directional coil “S2O2” becomes OFF forcing the motor to rotate backward and vice versa.



Figure 3.32: RPI Network 6

- **Network7:** If the Start Coil goes ON, immediately the motor “S2O1” must rotate for preset time “PT”. After this time, the motor stops to save energy.

Figure 3.33: RPI Network 7

- **Network8:** If there is a rejection timing by “T1”, the rejection coil “S2O3” becomes ON following “T1”.



Figure 3.34: RPI Network 8

- **Network9:** The retracting coil for the Horizontal double acting cylinder in station 3 “S3O2” is the complement of the advancing coil “S3O1”.



Figure 3.35: RPI Network 9

- **Network10:** The retracting coil for the Horizontal double acting cylinder in station 3 “S3O4” is the complement of the advancing coil “S3O3”.



Figure 3.36: RPI Network 10

- **Network11:** The advancing coil for the Horizontal double acting cylinder in station 3 “S3O1” is being set by the availability sensor “S2I2” and when its retracting sensor “S3I3” is ON. It retracts or resets when it reaches the full advancing sensor “S3I3” and the gripper “S3O5” is OFF. Thus:

$$S = S2I2.S3I3$$

$$R = S3O5 \cdot S3I3$$

Since the future state “X” can be controlled through the old state “x” as:

$$X = (S+x).R'$$

$$\text{Then } S3O1 = (S2I2.S3I3 + S3O1) \cdot (S3O5 \cdot S3I3)'$$

$$= (S2I2.S3I3 + S3O1) . (S3O5' + S3I3')$$

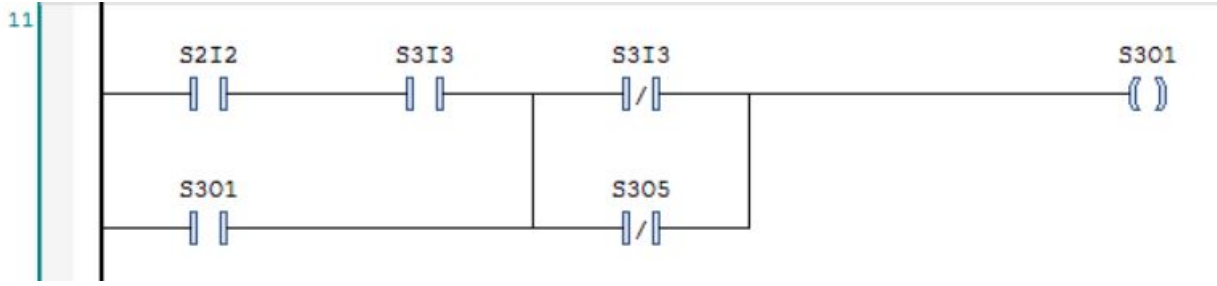


Figure 3.37: RPI Network 11

- **Network12:** The advancing coil for the Vertical double acting cylinder in station 3 “S3O3” is being set two times:

- When the gripper “S3O5” is ON and retracting sensor “S3I1” is ON.
- When the gripper “S3O5” is OFF and retracting sensor “S3I2” is ON.

It retracts or resets two times”

- When it reaches the full advancing sensor “S3I2” and the gripper “S3O5” is ON.
- When the full retracting sensor “S3I1” is ON and the gripper “S3O5” is OFF.

Thus:

$$S = S3O5.S3I1 + S3O5'.S3I2$$

$$R = (S3O5 . S3I2) + (S3O5' . S3I1)$$

Since the future state “**X**” can be controlled through the old state “x” as:

$$\mathbf{X} = (S+x).R'$$

$$\text{Then } \mathbf{S3O3} = (S3O5.S3I1 + S3O5'.S3I2 + S3O3) . ((S3O5 . S3I2) + (S3O5' . S3I1))'$$

$$= (S3O5.S3I1 + S3O5'.S3I2 + S3O3) . (S3O5' + S3I2') . (S3O5 + S3I1')$$

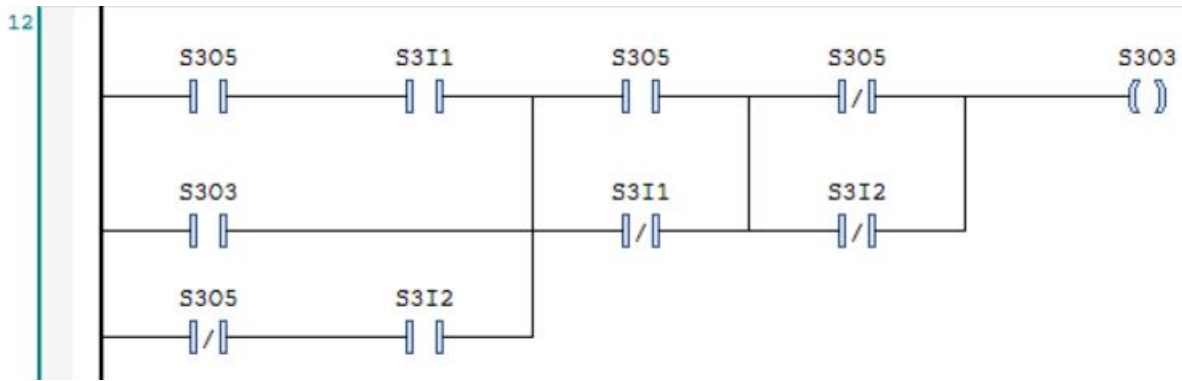


Figure 3.38: RPI Network 12

- Network 13: The gripper in station 3 “S3O5” is being set when both the full advancing sensor “S3I2” and full advancing sensor “S3I4” are ON. It retracts or resets only when both full retracting sensor “S3I1” and full advancing sensor “S3I4” are ON. Thus:

$$S = S3I2.S3I4$$

$$R = S3I1 \cdot S3I4$$

Since the future state “X” can be controlled through the old state “x” as:

$$X = (S+x).R'$$

$$\text{Then } S3O1 = (S3I2.S3I4 + S3O1) \cdot (S3I1 \cdot S3I4)'$$

$$= (S2I2.S3I3 + S3O1) \cdot (S3I1' + S3I4')$$

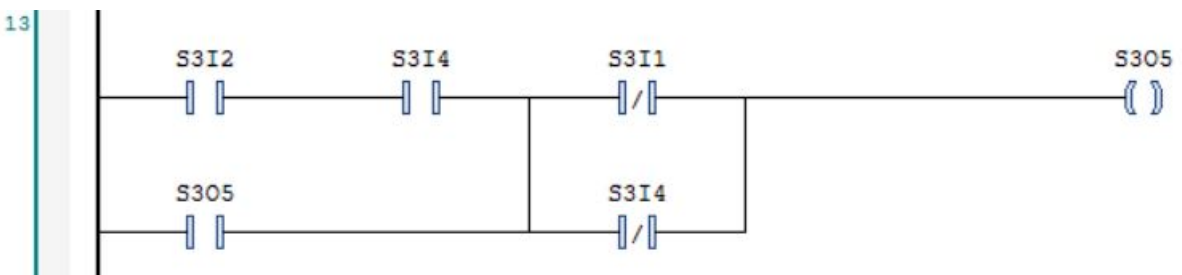


Figure 3.39: RPI Network 13

3.5 Testing and monitoring

3.5.1 Current shortage

Testing the input board was succeeded without any negative feedback. However, in the output board there was a **flaw**. All double solenoid coils did not work accurately. Some may work for couple of minutes then stopped working. Due to the maximum current

can flow in the output side for the optocoupler, these coils did not work appropriately. They need **80 mA**, while the optocoupler enables **30 mA**.

To overcome this issue, Transistor TIP120 was used to bridge the current from the power supply to the outputs. This technique was implemented for all 6 double acting solenoid coils, the rejection coil, and the alarm lamp.

The updated output board scheme is as following:

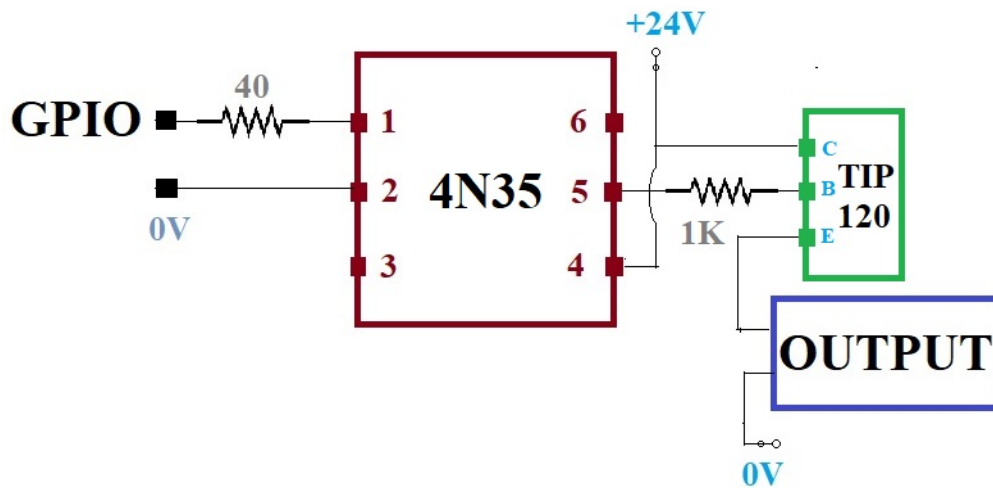


Figure 3.40: Modified opto-coupler for Output

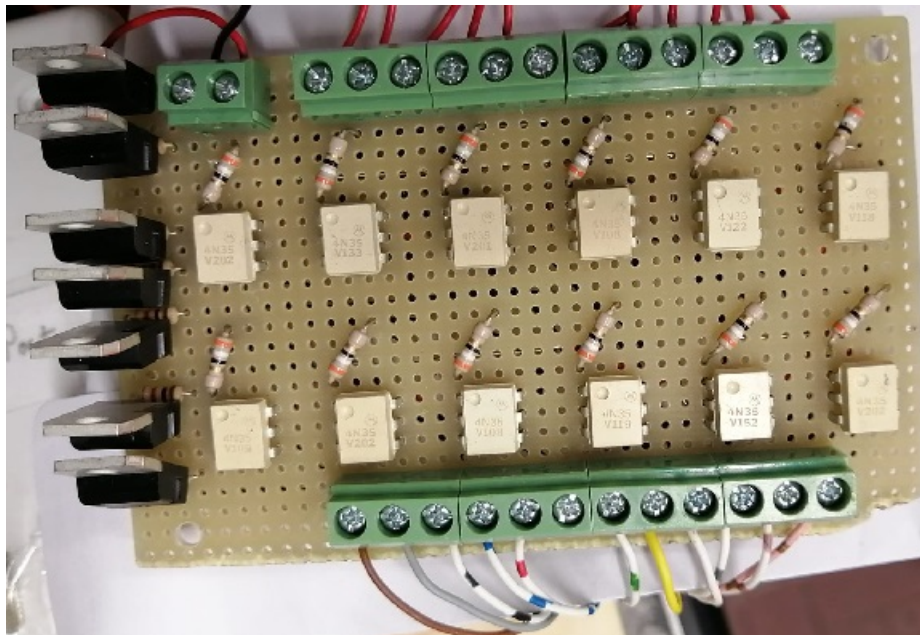


Figure 3.41: Modified Output board scheme

3.5.2 Notes

- Installing Operating system in the Raspberry pi is **mandatory** to use its GPIOs.
- Configuring the RPI to connect to the Wifi is needed only **once**. Then it connects automatically.
- A visual feedback “led” is needed in the control panel to confirm the RPI is ON.
- The three layers of GPIOs extension, Inputs board, and output board were mounted over each other to save space. All cables from these three stations were fixed according to the mapping.
- Some cylinders were working slow or fast. Their pneumatic flow controllers needed adjustment by a screwdriver.
- Timers were set according to the experimental results.
- The online simulation in Codesys was doing great following the real status.
- Without using transistors, double acting solenoids were **trapped** after 2 minutes.
- The 24V power supply is totally **separated** from the 3.3V Raspberry pi, which provides safety factor to the Raspberry pi.
- The code was downloaded only once to the RPI then it is being **saved**.
- Remote visualization is **powerful**, has fast response, and informative.

Chapter 4

LOGO!

4.1 Overview

Siemens AG is a German multinational corporation and the leading industrial manufacturing company in Europe headquartered in Munich city with branch offices spread abroad with many distributors.

LOGO! is a **compact** and **powerful** programmable logic controller (PLC) manufactured by Siemens. LOGO! is an intelligent logic module meant for small-scale automation projects. Widely used in many of industrial processes (conveyor belts, control of compressors, door control, etc.), office/commercial uses and home settings (lighting control, pool-related control tasks, access control, etc.). [5]



Figure 4.1: Siemens LOGO! PLC

As Siemens company is famous for its PLCs, separated modules can be added to establish the full controller depending on the system requirements. Well known modules are CPU module, power supply module, IOs extension module and HMI module.

For this thesis, below components will be used:

1. **24V Power supply:** to supply all the station and the modules.
2. **LOGO! CPU:** It work as the brain for the solution. It also includes number of build-in IOs.
3. **IOs Expansion module:** It is a module to gather IOs.
4. **Communication module:** A module to interact remotely with the system.

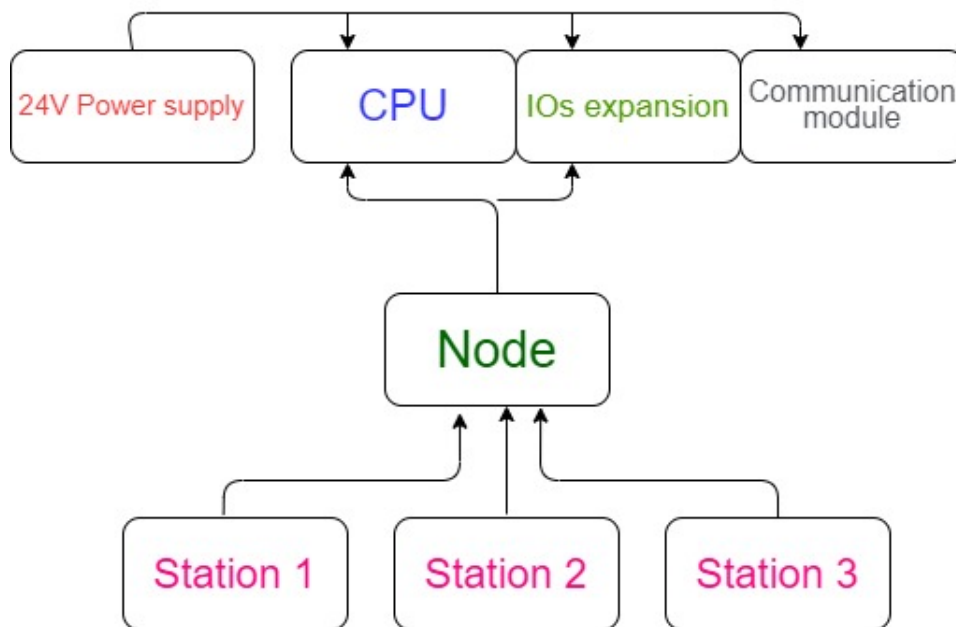


Figure 4.2: PLC components

4.2 IOs assignment

As previously illustrated in table 2.8, for fully and separately control all the IOs, It is needed **8** inputs and **12** outputs. Below table gives all of those IOs.

IOs	Name	COMMENT
Output 1	S1O1	Pushing cylinder advance
Output 2	S1O2	Pushing cylinder retract
Output 3	S1O3	Same as S1O1
Output 4	S2O1	Motor in backward
Output 5	S2O2	Reverse Motor
Output 6	S2O3	Rejection Coil
Output 7	S3O1	Vertical cylinder
Output 8	S3O2	Complement of GPIO21
Output 9	S3O3	Complement of GPIO20
Output 10	S3O4	Horizontal cylinder
Output 11	S3O5	Gripper
Output 12	ALARM	Alarming Lamp
Input 1	S1I1	Pushing cylinder advanced
Input 2	S2I1	Color sensor
Input 3	S2I2	Availability sensor
Input 4	S3I1	Horizontal cylinder retracted
Input 5	S3I2	Horizontal cylinder advanced
Input 6	S3I3	Vertical cylinder retracted
Input 7	S3I4	Vertical cylinder advanced
Input 8	Start	Start Pushbutton

Table 4.1: IOs Assignment for LOGO!

4.3 Hardware implementations

Components were selected to give best performance with less possible cost. Below items were chosen:

4.3.1 CPU Module

LOGO! 24RCE which is Relay based logic module. It has a build-in display. It has also 8 24V AC/DC digital inputs, 4 24V/relay digital outputs, memory of 400 blocks, modular expandable, Ethernet, integrated web server, data log, user-defined web pages, and standard microSD card for LOGO! Soft Comfort.



Figure 4.3: CPU Module

4.3.2 IO Expansion Module

It has a capability of handling up to 8 24V digital inputs and 8 24V/relay digital outputs.



Figure 4.4: IO Expansion

4.3.3 Communication Module

This communication module for connection LOGO! 8 to LTE network has 1 RJ45 port for Ind. Ethernet connection to LOGO! 8, including **2** digital inputs, **2** digital outputs, Write/read access to LOGO! Tags, Text message sending/ receipt, E-mail sending, Position detection GPS, Time synchronization/ forwarding with real-time clock, and Configuration/diagnostics via web interface, Remote access via OpenVPN/HTTPs, DynDNS, observe national approval. [4]



Figure 4.5: Communication Module

A SMA socket (50 ohms) Antenna is needed for this communication module, as well as SIM card to send and receive SMS.



Figure 4.6: Antenna

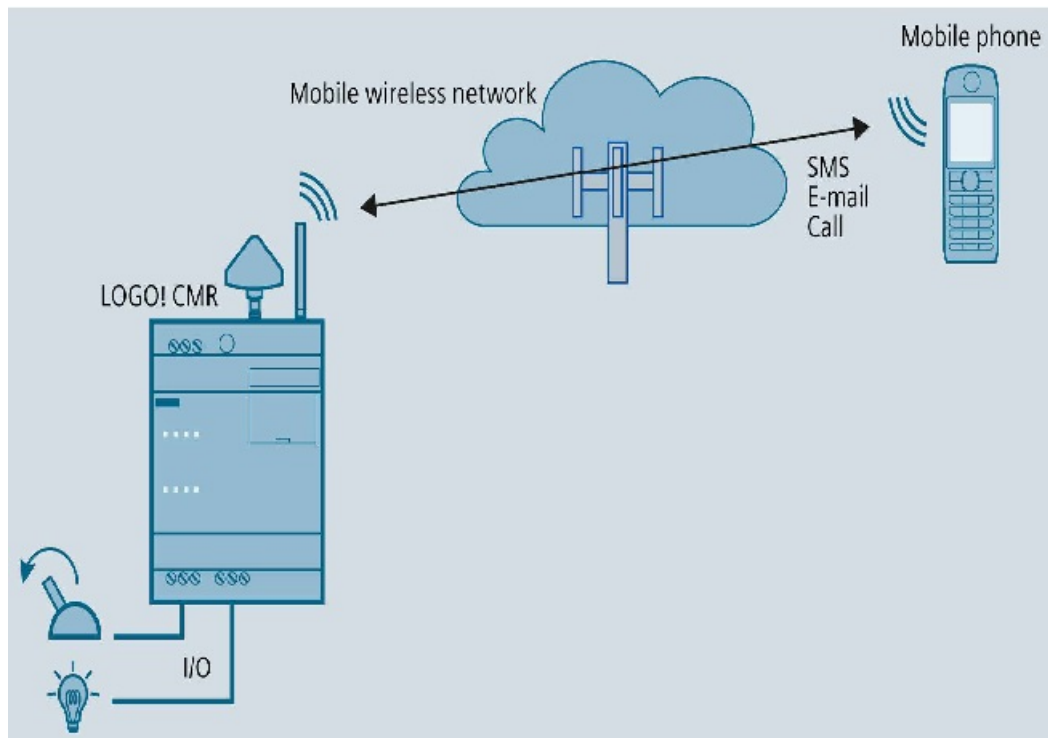


Figure 4.7: Communication Module

Since it has **2** inputs and **2** outputs, One of its Inputs (Input 1) will be acting when the “**Alarm**” is activated. Also, one of its output (output 1) will be activated when activating the “**Virtual Start**” by SMS.

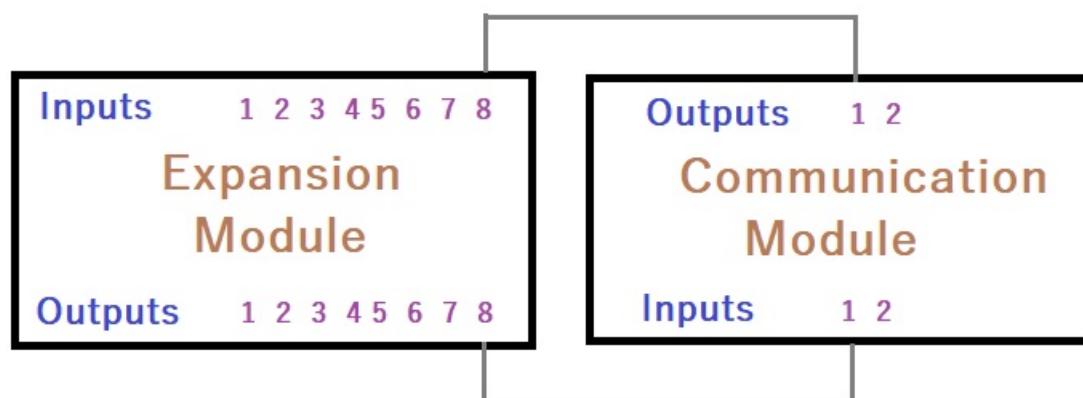


Figure 4.8: IOs connections for Communication Module

4.4 Software Implementation

In order to communicate to the LOGO! Pi via the use of PC, LOGO! Soft Comfort software is needed. Through LOGO! Soft Comfort, all I/Os can be assigned, and monitored. However, there should be always a link between the LOGO! Soft Comfort and the PC via industrial **ethernet cable** to do so.

To reach the final goal, It is necessarily to proceed step by step and solve several steps:

1. **Configuring** the LOGO! Soft Comfort as normal as configuring any software.
2. **Recognition** of the LOGO! by the PC through its IP.
3. **Designing** of the logical circuit.
4. **Downloading** the program to the LOGO!
5. **Configuring** the Communication module.
6. "**Visualization**" interface.
7. **Results**.

4.4.1 LOGO! Soft! Configurations

It is required to create new project using Ladder diagram as below

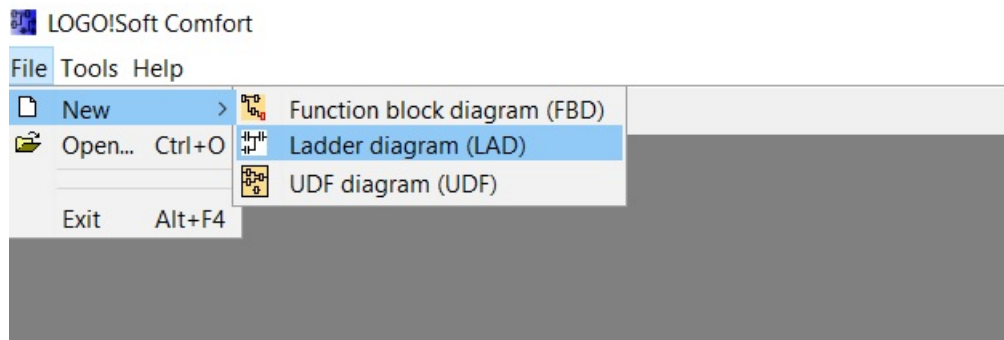


Figure 4.9: Creating new Project

By drag and drop technique, all elements can be chosen. Then can be connected via lines for every network can be created.

I/Os can be assigned to each block as shown below. Also commenting can be done.

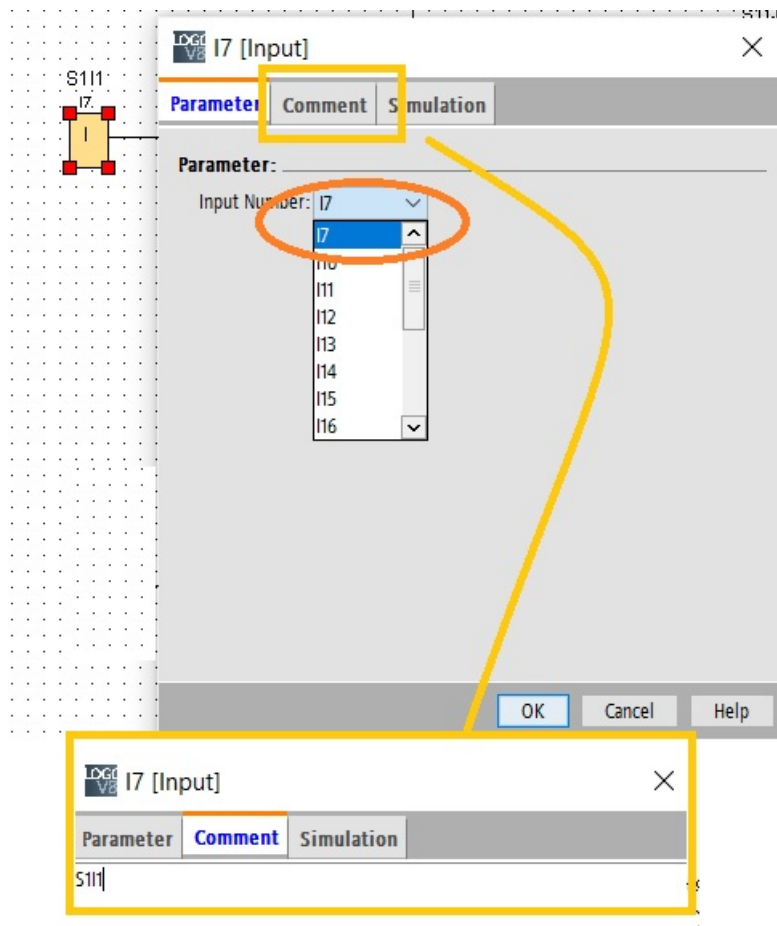


Figure 4.10: Assigning IOs and Comments

Proposed program was created as in **Appendix B 8.2**.

Downloading it to the LOGO! Can be as below

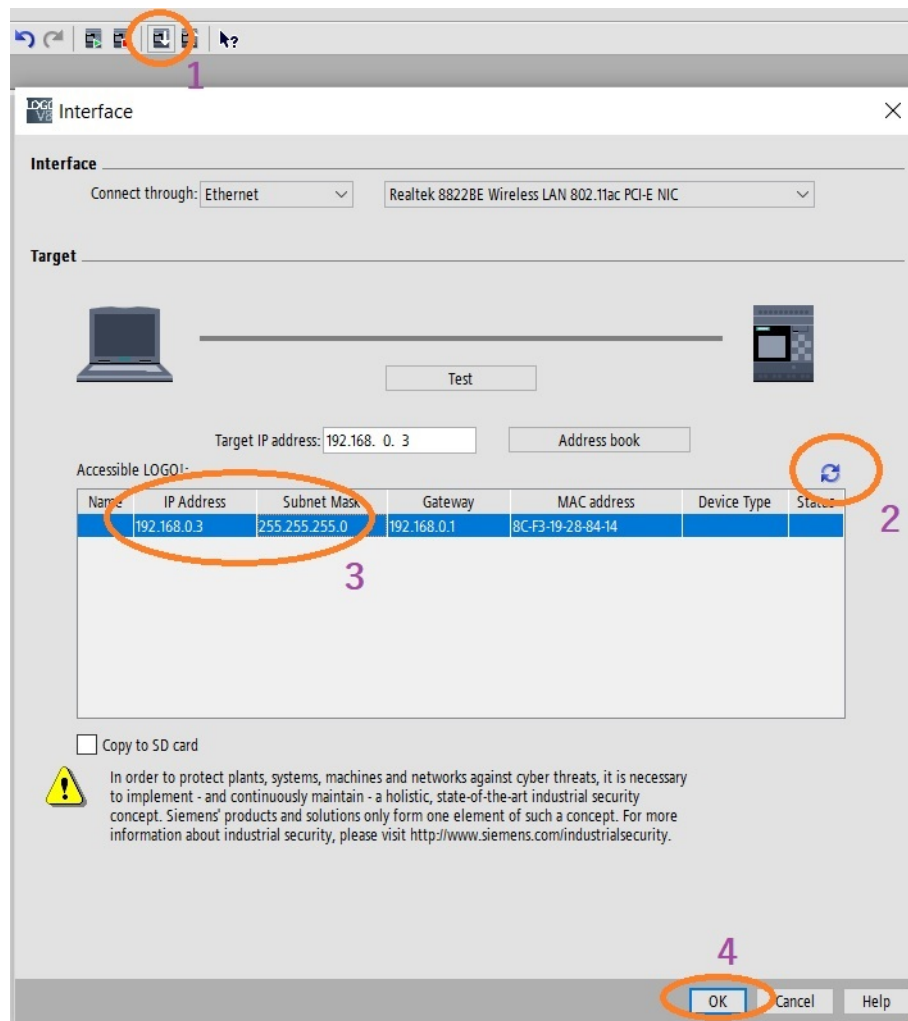


Figure 4.11: Downloading from PC to LOGO!

4.4.2 Simulation

After finishing downloading the ladder diagram, it can be monitored live. Also Inputs can be simulated without the presence of the LOGO! as below.

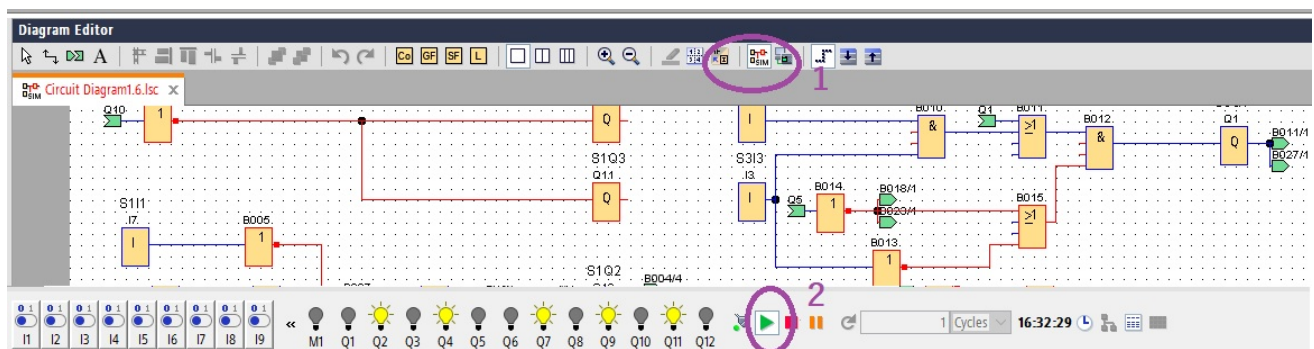


Figure 4.12: Simulation

4.4.3 Remote controlling

Remote controlling can be thanks to the Communication module. However, it needs to be programmed firstly.

The proposed scenario is: When receiving SMS telling “**Start**”, the system starts as a single press on the physical pushbutton. When receiving “**Stop**” SMS, it switches off the this “**virtual Start**”. If the “**Alarm**” goes On, SMS should be received notifying this status.

This can be done as in following steps:

- Inserting the **SIM**, Connecting the Antenna, and supplying the 24V.
- By default, the IP to access the configuration is **192.186.0.3**
- Default username is admin and default password is admin. Later it can be changed. Then Start page will be shown.

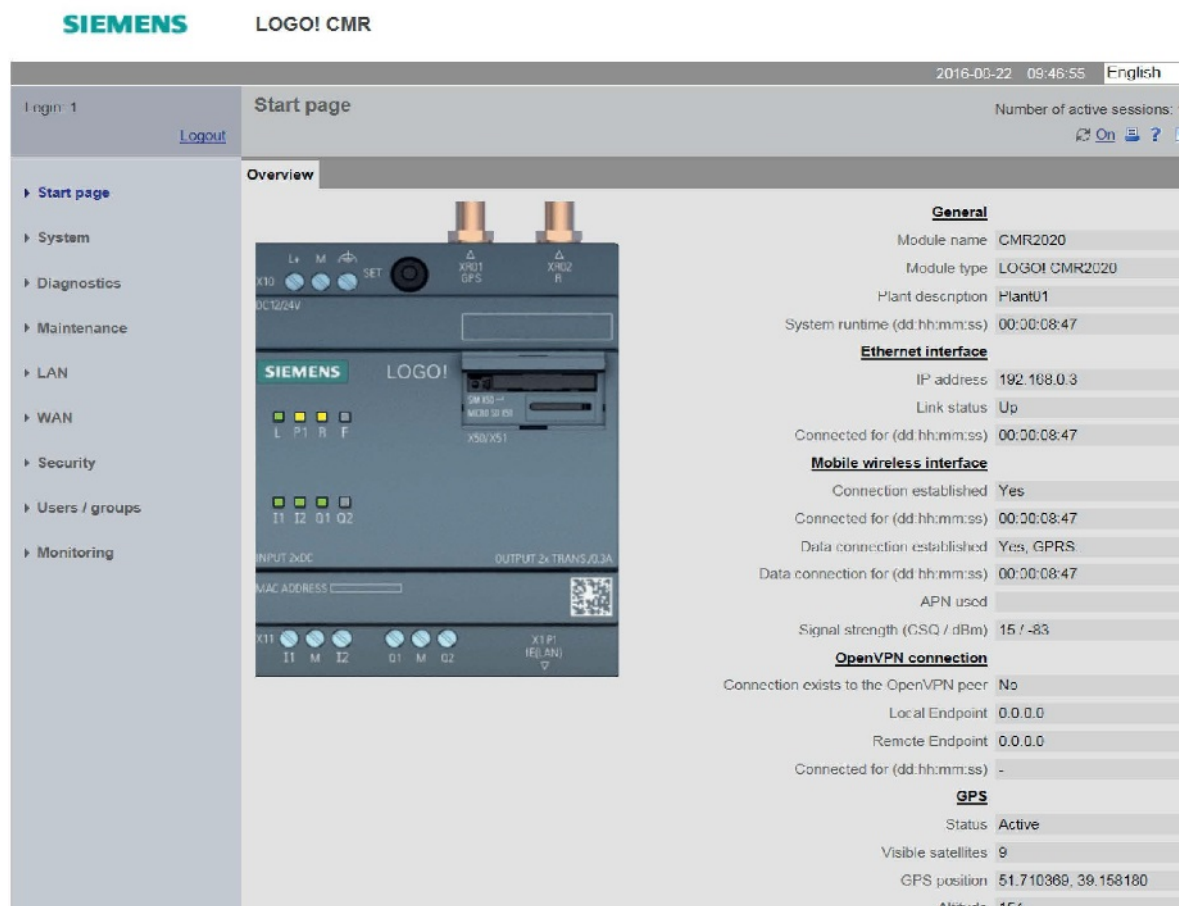


Figure 4.13: Start page

- Changing the IP can be done from LAN tab

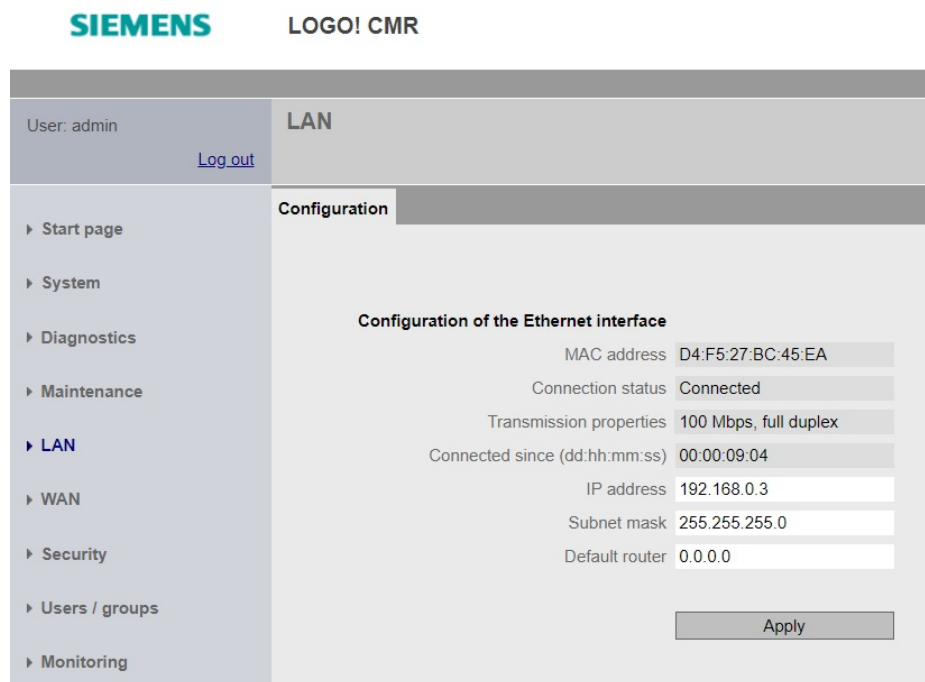


Figure 4.14: LAN

- This is assigning trusted list of numbers which allowed to communicate to the communication module. It is under the tab Users/groups>User

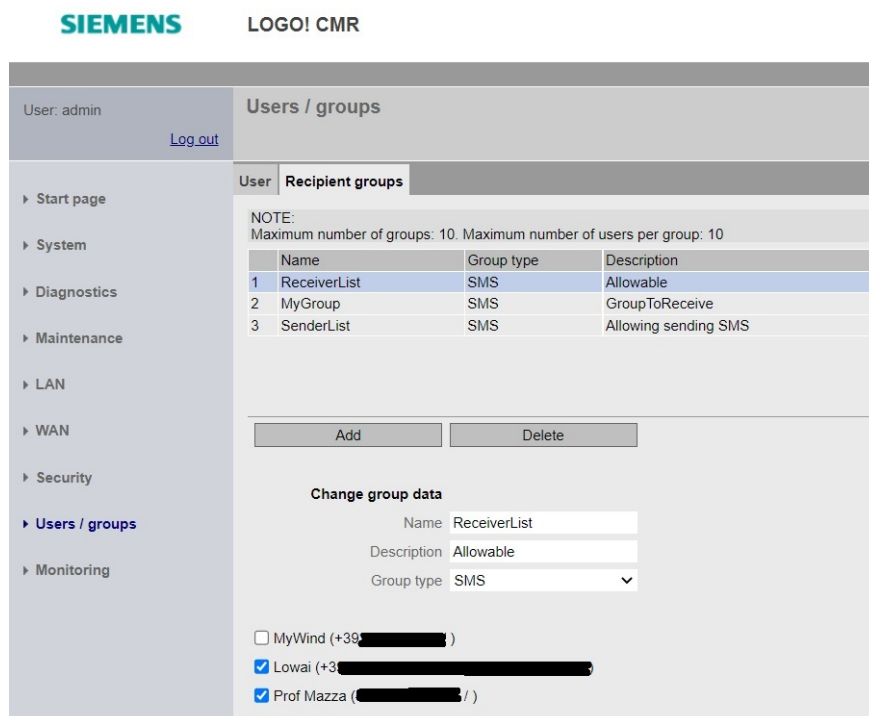


Figure 4.15: Users

- Assigning each user to specific group can be under **Users/groups>Recipient groups**

SIEMENS LOGO! CMR

User: admin [Log out](#)

Users / groups

Start page
System
Diagnostics
Maintenance
LAN
WAN
Security
Users / groups
Monitoring

NOTE:
Maximum number of users: 20

	Name	Description	User name	Phone number	E-mail address	Allow receipt of SMS messages	Phone number can be
1	MyWind	WindSim	adminWind	+39		Yes	Yes
2	Lowai	Mine	admin	+39		Yes	Yes
3	Prof Mazza		ProfMazza	+39		Yes	Yes

Add Delete

Change user

Name MyWind

Description WindSim

Phone number +39

Allow receipt of SMS messages Yes

Phone number can be changed for this user by SMS message Yes

E-mail address

☐ Change user data

User name adminWind

Password

Repeat password

☒ Do not use password rules

Apply

Figure 4.16: Recipient groups

- Monitoring the status of IOs can be through the Home page or **Monitoring>Overview**

SIEMENS LOGO! CMR

User: admin [Log out](#)

Monitoring

Overview LOGO! BM Constants Message texts Signals Events Actions Assignments

LOGO! CMR

CMR_I1	LOGO! CMR / I/O / Input / 1	<input type="checkbox"/> OFF
CMR_I2	LOGO! CMR / I/O / Input / 2	<input type="checkbox"/> OFF
CMR_Q1	LOGO! CMR / I/O / Output / 1	<input type="checkbox"/> OFF
CMR_Q2	LOGO! CMR / I/O / Output / 2	<input type="checkbox"/> OFF

Start page
System
Diagnostics
Maintenance
LAN
WAN
Security
Users / groups
Monitoring

Figure 4.17: Overview

- Specifying the SMS is under **Monitoring>Message text**

SIEMENS LOGO! CMR

User: admin [Log out](#)

Monitoring

Overview | LOGO! BM | Constants | **Message texts** | Signals | Events | Actions | Assignments

NOTE:
Maximum number of message texts: 20

	Name	Content
1	AlarmTesxt	Alarm is ON Day: [DATE] Time: [TIME]
2	StartText	Virtual Start is ON Day: [DATE] Time: [TIME]
3	StopText	Virtual Start is OFF Day: [DATE] Time: [TIME]

[Add](#) [Delete](#)

Edit text

Name:

Content:

Day: [DATE] Time: [TIME]

Number of characters: 38

[Apply](#)

Figure 4.18: Message text

- Choosing the IOs Signals is done by following **Monitoring>Signals**

SIEMENS LOGO! CMR

User: admin [Log out](#)

Monitoring

Overview | LOGO! BM | Constants | Message texts | **Signals** | Events | Actions | Assignments

NOTE:
Maximum number of signals: 32

	Name	Signal configuration
1	CMR_I1	LOGO! CMR / I/O / Input / 1
2	CMR_I2	LOGO! CMR / I/O / Input / 2
3	CMR_Q1	LOGO! CMR / I/O / Output / 1
4	CMR_Q2	LOGO! CMR / I/O / Output / 2

[Add](#) [Delete](#)

Change signal

Name:

Signal source:

Signal type:

I/O type:

Number:

[Apply](#)

Figure 4.19: Signals

- Assigning the event by navigating **Monitoring>Events**

SIEMENS LOGO! CMR

User: admin [Log out](#)

Monitoring

Overview | LOGO! BM | Constants | Message texts | Signals | **Events** | Actions | Assignments

NOTE:
Maximum number of events: 32

	Name	Event configuration
1	AlarmEvent	CMR_I1 Changes to 1
2	Virtual Start is On	CMR_Q1 Changes to 1
3	Virtual Start is OFF	CMR_Q1 Changes to 0

Add Delete

Change event

Name: AlarmEvent

Signal name: CMR_I1

Event: Changes to 1

Apply

Figure 4.20: Events

- Selecting the required action by navigating **Monitoring>Actions**

SIEMENS LOGO! CMR

User: admin [Log out](#)

Monitoring

Overview | LOGO! BM | Constants | Message texts | Signals | Events | **Actions** | Assignments

NOTE:
Maximum number of actions: 32

	Name	Action configuration
1	AlarmAction	Send SMS / ReceiverList / AlarmTesxt
2	VirtualStartOnEvent	Send SMS / ReceiverList / StartText
3	VirtualStartOffEvent	Send SMS / ReceiverList / StopText

Add Delete

Change action

Name: AlarmAction

Target system: Send SMS

Recipient group: ReceiverList

Message text: AlarmTesxt

Apply

Figure 4.21: Actions

- Specifying the Assignments got actions when event happens is under **Monitoring>Assignments**

User: admin [Log out](#)

Monitoring

Overview | LOGO! BM | Constants | Message texts | Signals | Events | Actions | **Assignments**

NOTE:
Maximum number of assignments: 32

	Active	Name	Event	Action
1	Yes	AlarmAssignment	AlarmEvent	AlarmAction
2	Yes	StartAssignment	Virtual Start is On	VirtualStartOnEvent
3	Yes	StopAssignment	Virtual Start is OFF	VirtualStartOffEvent

[Add](#) [Delete](#)

Change assignment

Name:

☒ Activate assignment

When:

Event:

Signal name:

Signal configuration:

Event configuration:

Then:

Action:

Action configuration:

[Apply](#)

Figure 4.22: Assignments

- Under the tab WAN then Mobile wireless setting, enabling the mobile wireless interface and data servicing are needed beside entering the **APN** for the SIM operator.

SIEMENS **LOGO! CMR**

User: admin [Log out](#)

WAN

Overview | **Mobile wireless settings** | Wireless cell | SMS | SMS alias | E-mail | DynDNS

☒ Enable mobile wireless interface

PIN of the SIM card:

☒ No SIM card present.

☐ Allow roaming

☒ Enable data service in the mobile wireless network

APN:

Authentication method:


Name:

Password:

[Apply](#)

Figure 4.23: Mobile Wireless

- Also, Under the tab WAN then SMS, allowing receipt group and text of SMS messages.



LOGO! CMR

User: admin

Log out

WAN

Overview

Mobile wireless settings

Wireless cell

SMS

SMS alias

E-mail

DynDNS

Start page

System

Diagnostics

Maintenance

LAN

WAN

Security

Users / groups

☒ Allow receipt of SMS messages
 ☐ Enable SMS password

Password for writing commands

☐ Send positive acknowledgments

Apply

Test SMS

Recipient group

ReceiverList

Text

AlarmTesxt

Send test SMS

Figure 4.24: Receipt of SMS

- SMS alias is an option found under **WAN>SMS alias**.

[illegible]

Figure 4.25: SMS alias

- Checking the log of diagnostics is under **Diagnostics** tab.

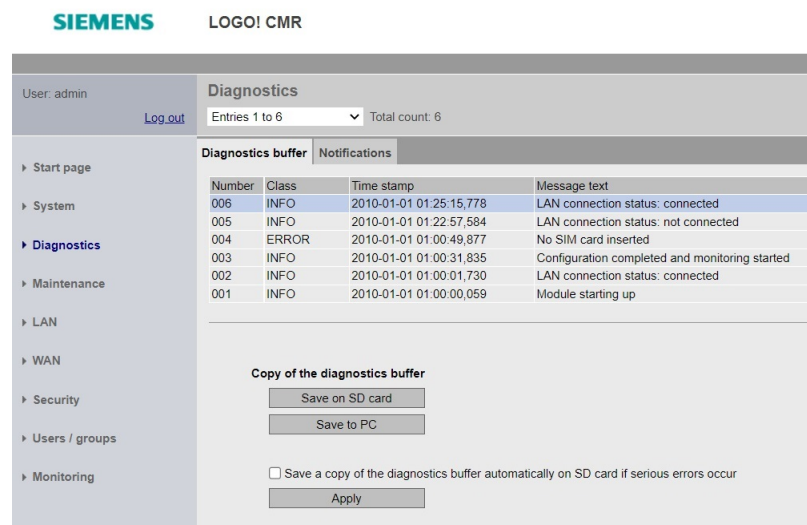


Figure 4.26: Diagnostics

4.4.4 Networks of Logic

Following the proposed scenario in 2.11 “flow chart of the process”, networks will be explained.

- **Network1:** The memory coil “Start” will be ON, if there is any signal from physical start “I8” from the panel or virtual start “I9” from the communication module to start the sequence. OR block has the label “>1”. A pulse block was put to trigger the memory coil only once.

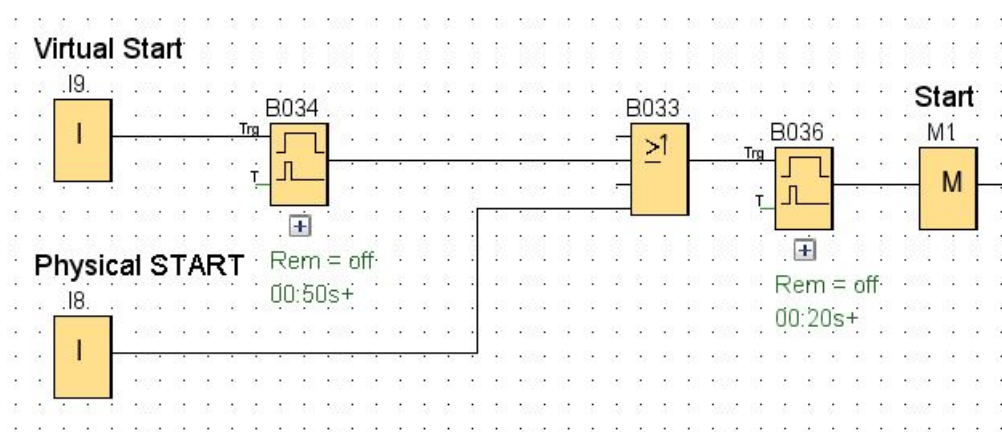


Figure 4.27: LOGO! Logic Network 1

- **Network2:** Whenever “Start” memory is ON, the double acting cylinder in station 1 “S1Q2” advances till it reaches its maximum thanks to the latching. When the

full advance sensor gives signal “**S1I1**”, this coil goes OFF. AND block has the label “&”. Block with the label “1” refers to **NOT** gate.

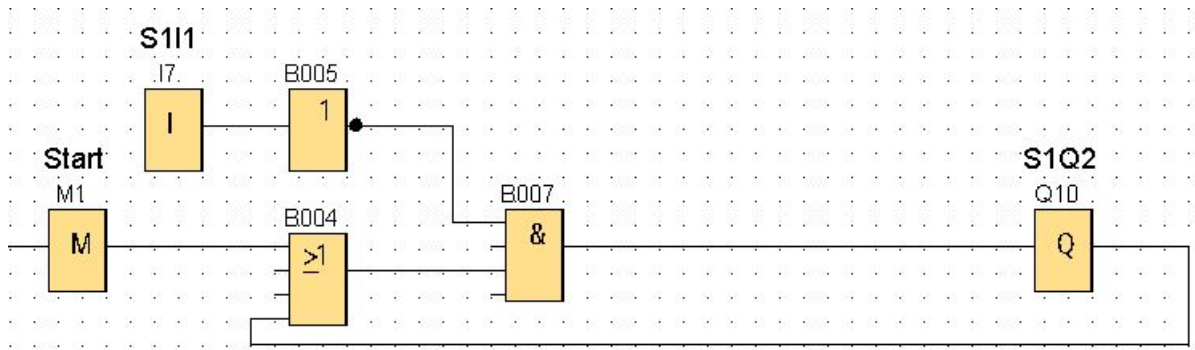


Figure 4.28: LOGO! Logic Network 2

- **Network3:** When the double acting cylinder goes advancing, its retracting solenoid coil “**S1Q1**” goes OFF. Also the single acting cylinder does the same. This is been implemented by a **NOT** gate of “**S1Q2**”.

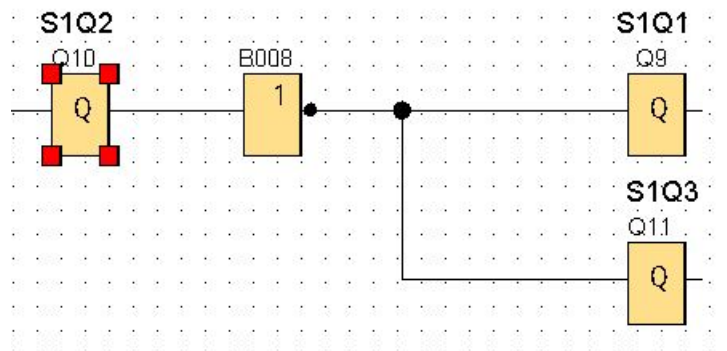


Figure 4.29: LOGO! Logic Network 3

- **Network4:** Whenever the color sensor “**S2I1**” detects the silver color, the counter counts up. It was specified as counter up by assigning the LOW block. PV is the preset value for the counter. When the current value equals to PV, “Alarm” goes ON. The counter is reset when there is Start memory goes ON while the Alarm is ON.

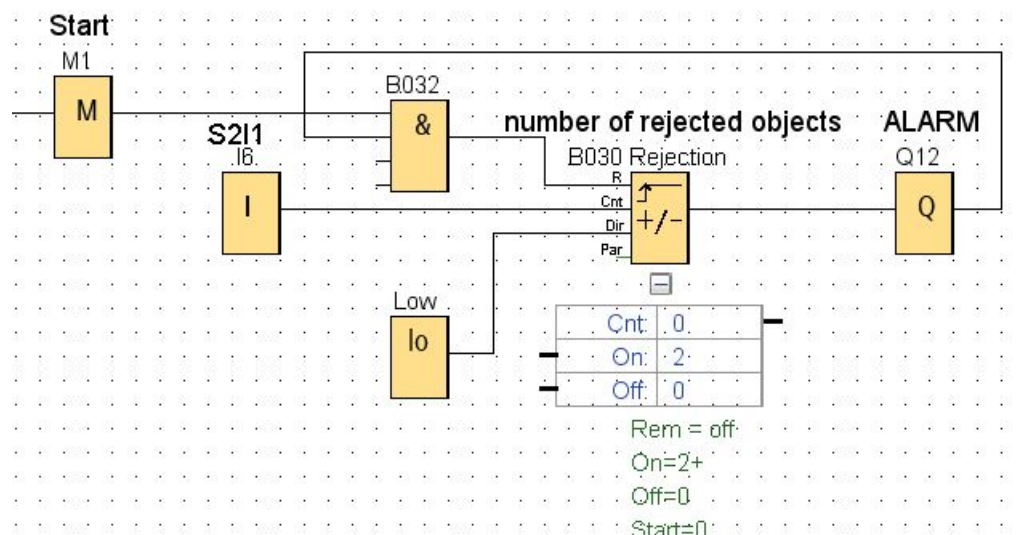


Figure 4.30: LOGO! Logic Network 4

- **Network5:** When the color sensor “**S2I1**” detects the silver color, it goes ON to enable timer “**T1**” which goes ON till it reaches its preset time in “**PT**”. Which enables the rejection coil “**S2Q3**”. Also it disables the motor directional coil “**S2Q2**” enforcing the motor to rotate backward.

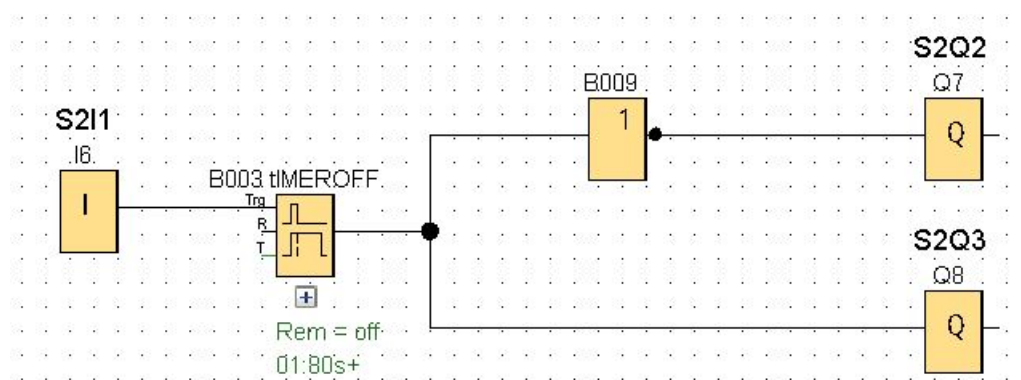


Figure 4.31: LOGO! Logic Network 5

- **Network6:** If the Start memory goes ON, immediately the motor “S2Q1” must rotate for preset time “PT”. After this time, the motor stops to save energy.

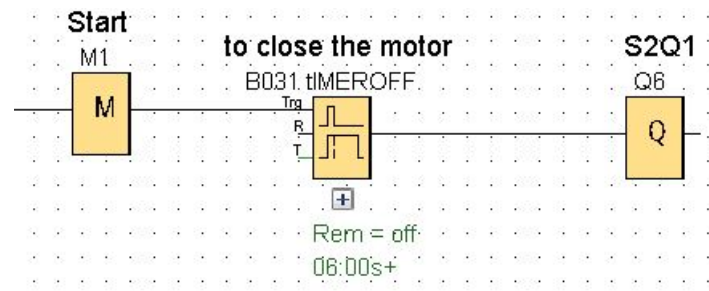


Figure 4.32: LOGO! Logic Network 6

- **Network7:** The retracting coil for the Horizontal double acting cylinder in station 3 “S3Q2” is the complement of the advancing coil “S3Q1”.

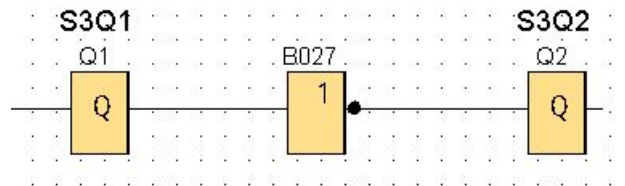


Figure 4.33: LOGO! Logic Network 7

- **Network8:** The retracting coil for the Horizontal double acting cylinder in station 3 “S3Q4” is the complement of the advancing coil “S3Q3”.

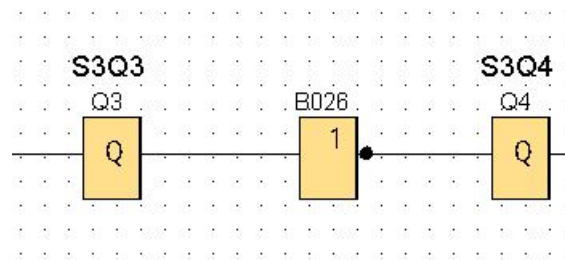


Figure 4.34: LOGO! Logic Network 8

- **Network9:** The advancing coil for the Horizontal double acting cylinder in station 3 “S3Q1” is being set by the availability sensor “S2I2” and when its retracting sensor “S3I3” is ON. It retracts or resets when it reaches the full advancing sensor “S3I3” and the gripper “S3Q5” is OFF.

Thus: $S = S2I2.S3I3$ $R = S3Q5 \cdot S3I3$

Since the future state “X” can be controlled through the old state “x” as:

$$X = (S+x).R'$$

$$\begin{aligned} \text{Then } S3Q1 &= (S2I2.S3I33 + S3Q1) \cdot (S3Q5 \cdot S3I3)' \\ &= (S2I2.S3I3 + S3Q1) \cdot (S3Q5' + S3I3') \end{aligned}$$

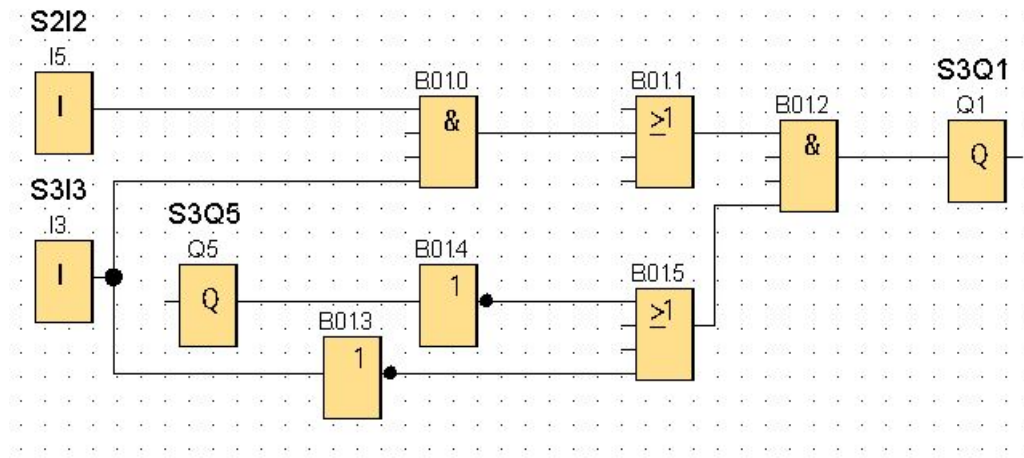


Figure 4.35: LOGO! Logic Network 9

- **Network10:** The advancing coil for the Vertical double acting cylinder in station 3 “S3Q3” is being set two times:
 - When the gripper “S3Q5” is ON and retracting sensor “S3I1” is ON.
 - When the gripper “S3Q5” is OFF and retracting sensor “S3I2” is ON.

It retracts or resets two times:

- When it reaches the full advancing sensor “S3I2” and the gripper “S3Q5” is ON.
- When the full retracting sensor “S3I1” is ON and the gripper “S3Q5” is OFF.

$$\text{Thus: } S = S3Q5.S3I1 + S3Q5'.S3I2 \quad R = (S3Q5 \cdot S3I2) + (S3Q5' \cdot S3I1)$$

Since the future state “X” can be controlled through the old state “x” as:

$$X = (S+x).R'$$

$$\text{Then } S3O3 = (S3Q5.S3I1 + S3Q5'.S3I2 + S3Q3) \cdot ((S3Q5 \cdot S3I2) + (S3Q5' \cdot S3I1))'$$

$$= (S3Q5.S3I1 + S3Q5'.S3I2 + S3Q3) \cdot (S3Q5' + S3I2') \cdot (S3Q5 + S3I1')$$

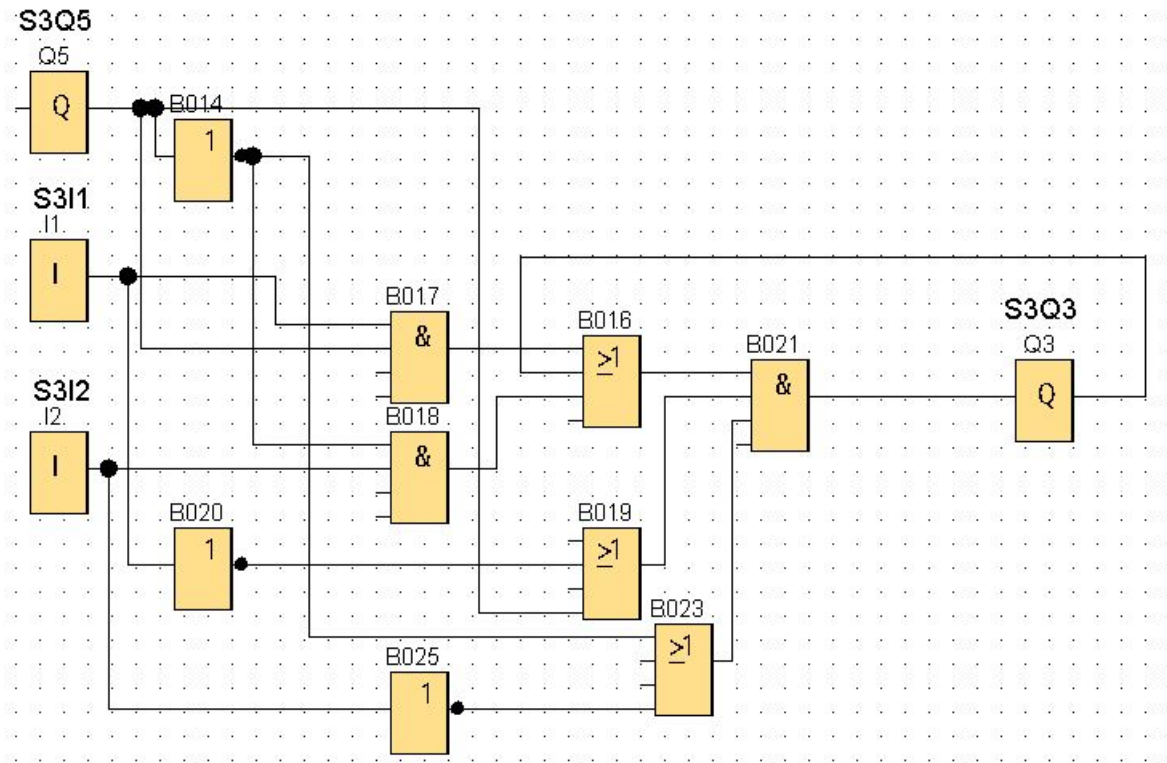


Figure 4.36: LOGO! Logic Network 10

- **Network11:** The gripper in station 3 “**S3Q5**” is being set when both the full advancing sensor “**S3I2**” and full advancing sensor “**S3I4**” are ON. It retracts or resets only when both full retracting sensor “**S3I1**” and full advancing sensor “**S3I4**” are ON.

Thus: $S = S3I2.S3I4$ $R = S3I1 \cdot S3I4$

Since the future state “X” can be controlled through the old state “x” as:

$$X = (S+x).R'$$

$$\text{Then } \mathbf{S3Q5} = (S3I2.S3I4 + S3Q5) \cdot (S3I1 \cdot S3I4)'$$

$$= (\mathbf{S3I2.S3I4} + \mathbf{S3Q5}) \cdot (\mathbf{S3I1}' + \mathbf{S3I4}')$$

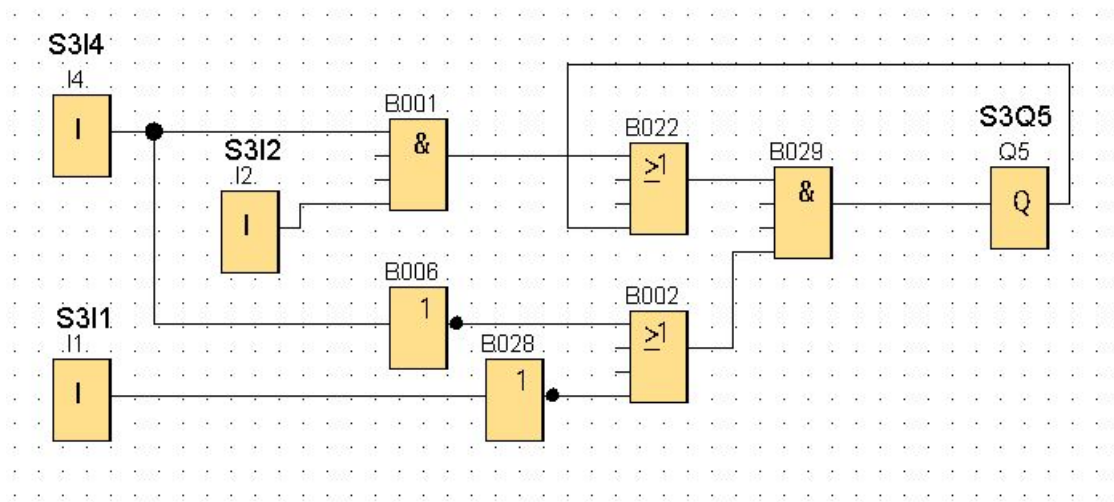


Figure 4.37: LOGO! Logic Network 11

4.5 Testing and monitoring

- LOGO! supports many modules including HMI.
- LOGO! Soft Comfort is **NOT** a free software. It needs a license, which is over cost.
- LOGO! Soft Comfort is easy to handle but it needs much focus for many networks to avoid mistakes.
- Timers and counters can be adjusted from the LOGO!'s **screen**.
- **Monitoring** the status of all IOs can be done through the LOGO!'s screen also.
- Although the **price** of the CPU is expensive, the communication module is doubling its price. However, the expansion module is in the middle.
- **Less wires** are needed in general which makes the less maintenance effort and easier debugging.
- A **delay** was countered due to the delay of receiving or sending the SMS because of signal weakness.
- Status of IOs can be provided thanks to the communication module.
- CMR communication module supports GPS, calls, and emails. However, they were disabled in this thesis.
- To have better **visualization**, HMI is needed.

Chapter 5

EasyPort

5.1 Overview

The EasyPort process interface implements bi-directional transmission of process signals between a real control process using standard low voltage technology (24 V) and a PC. In order to communicate with the PC, OPC is used.

Open Platform Communications (**OPC**) is a series of industrial specifications and standards for industrial telecommunication. In 1996, under the name of OLE for Process Control (Object Linking and Embedding for process control) was developed the original standard by an industrial automation task force. OPC specifies the communication of real-time plant data between control devices from different manufacturers.[7]

The EasyPort USB basic module has **16** digital and **4** analogue inputs as well as **16** digital and **2** analogue outputs. The link between the EasyPort process interface and the PC is set by means of an electrically isolated USB interface or an electrically isolated V.24 interface. However, the V.24 interface allows the use of one module only. The drivers for USB operation are required to identify it for the PC.



Figure 5.1: Easyport

Using an USB hub, Up to **4 EasyPort modules** can be connected together. However, the addresses of each module must be different. [8]

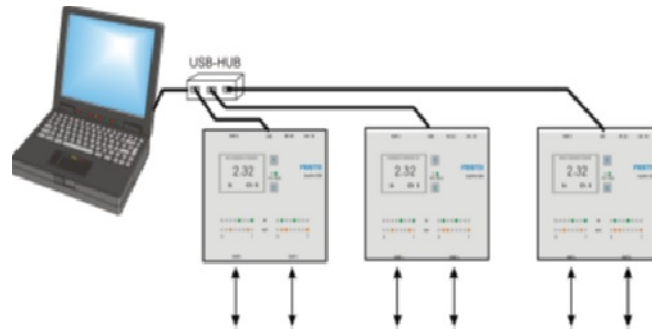


Figure 5.2: Easyport connection

5.2 IOs assignment

As previously illustrated in table 2.8 , for fully and separately control all the IOs, it is needed 8 inputs and 12 outputs. Easyport combines each 8 Inputs and 8 Outputs in a port. Each Easyport has 2 ports (port 1.x and port 2.x). Below table gives all of those IOs.

IOs	Name	COMMENT
Port 1 Output 0	S1O1	Pushing cylinder advance
Port 1 Output 1	S1O2	Pushing cylinder retract
Port 1 Output 2	S1O3	Same as S1O1
Port 1 Output 3	S2O1	Motor in backward
Port 1 Output 4	S2O2	Reverse Motor
Port 1 Output 5	S2O3	Rejection Coil
Port 1 Output 6	S3O1	Vertical cylinder
Port 1 Output 7	S3O2	Complement of GPIO21
Port 2 Output 0	S3O3	Complement of GPIO20
Port 2 Output 1	S3O4	Horizontal cylinder
Port 2 Output 2	S3O5	Gripper
Port 2 Output 3	ALARM	Alarming Lamp
Port 1 Input 0	S1I1	Pushing cylinder advanced
Port 1 Input 1	S2I1	Color sensor
Port 1 Input 2	S2I2	Availability sensor
Port 1 Input 3	S3I1	Horizontal cylinder retracted
Port 1 Input 4	S3I2	Horizontal cylinder advanced
Port 1 Input 5	S3I3	Vertical cylinder retracted
Port 1 Input 6	S3I4	Vertical cylinder advanced
Port 1 Input 7	Start	Start Pushbutton

Table 5.1: IOs Assignment for Easyport

5.3 Hardware implementations

Easyport was supplied by 24V from the power supply. Then Inputs and outputs were connected from the multi-wire cable to the Easyport according to the following diagram.

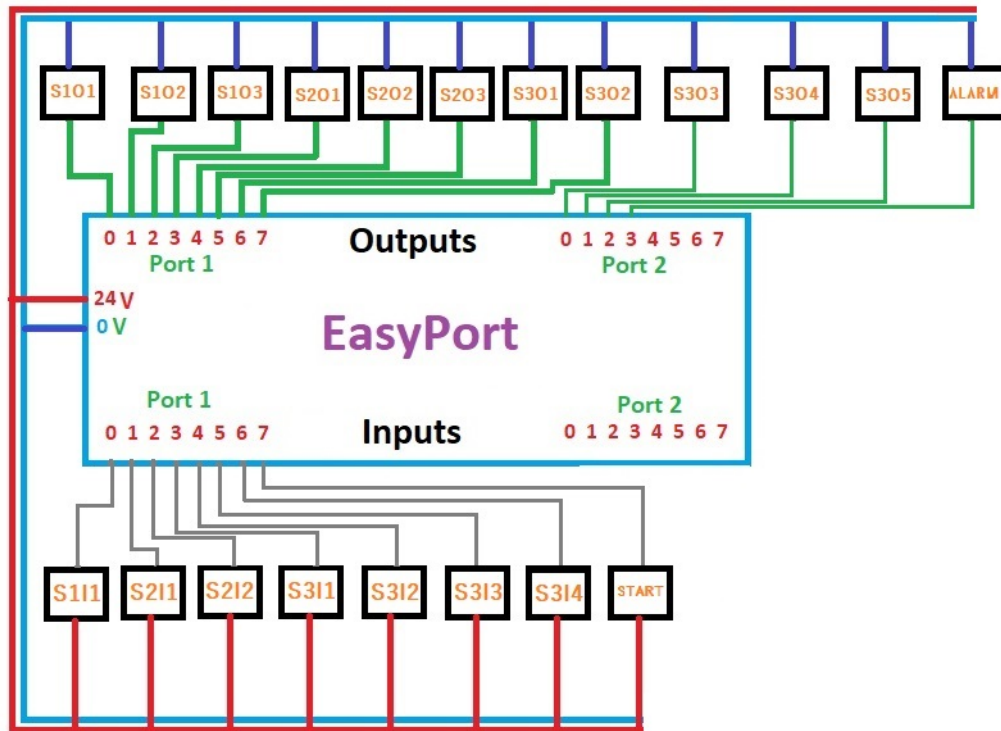


Figure 5.3: Easyport hardware

5.4 Software Implementation

5.4.1 Software Initialization

Four softwares are required to be installed:

- **Easyport driver:** A software required to identify the Easyport by the PC. It can be downloaded under the name of “Setupeasyport”.
- **Easyport USB driver:** A software to identify the USB cable used to link the Easyport with the Pc. It can be downloaded under the name of “setup_usb”.
- **OPC driver** which is required to define the OPC. It is found under the name setup_ezopc.
- **FluidSim software** to code the ladder diagram and handle the IOs.

5.4.2 Coding

After downloading all previous programs, Easyport must be supplied by 24V power supply. Also, USB cable must connect the EasyPort to the PC.

To check the connection between the PC and the Easyport, EasyPort demo Application software must be launched.

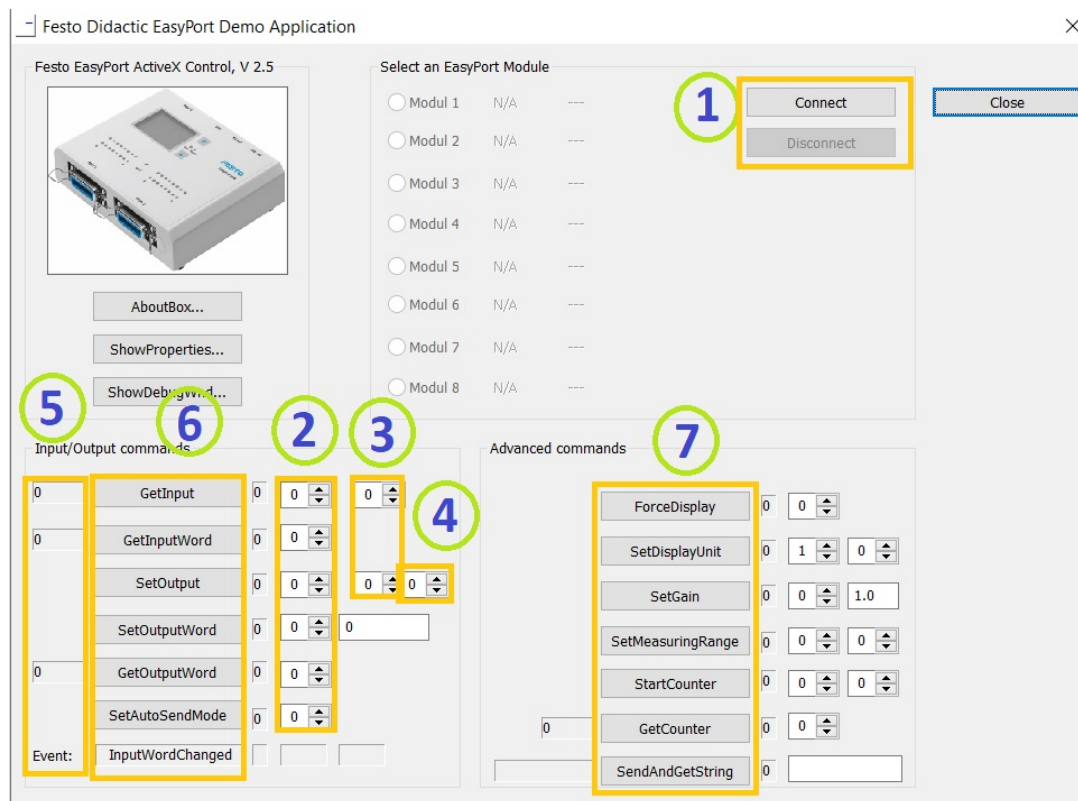


Figure 5.4: Easyport Demo

Below numbers illustrate the meaning of each:

1. To establish connection between the Easyport and the PC.
2. **0** refers to Port1 and **1** refers to Port 2.
3. To check exact I/O 0-7.
4. **0** for reset and **1** for set.
5. Status of I/O.
6. Commands for testing.
7. Advance control.

To simulate controlling the Easyport, below steps to be followed:

- Firstly Clicking on “**Connect**”. Then setting any Output to 1 and observe it in both the EasyPort LEDs and Digital module LEDs.
- To observe an Input, After connecting 24V to the specific Input- use Command **GetInput** to check it in the interface.
- When Closing DEMO Application, the EasyPort will return to its initial conditions.

After launching FluidSim, new project would be created. By dragging and dropping, ladder diagram can be created as in **Appendix C 8.3**.

To Initialize FluidSim for OPC mode, it can be done through navigating (**Options>OPC mode>OK**).

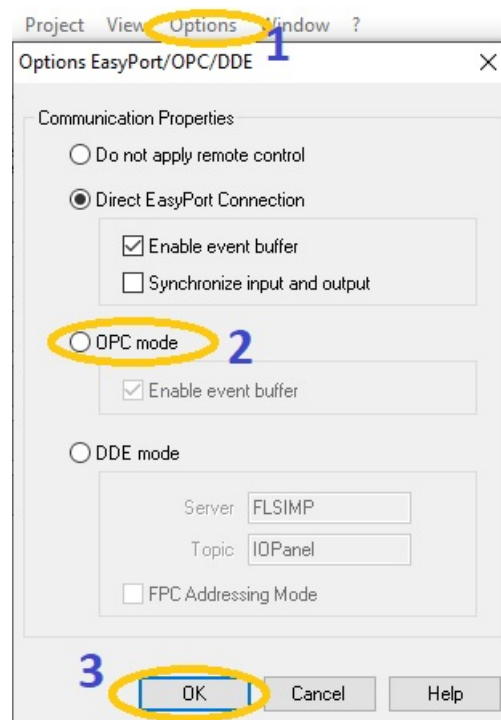


Figure 5.5: FluidSim OPC

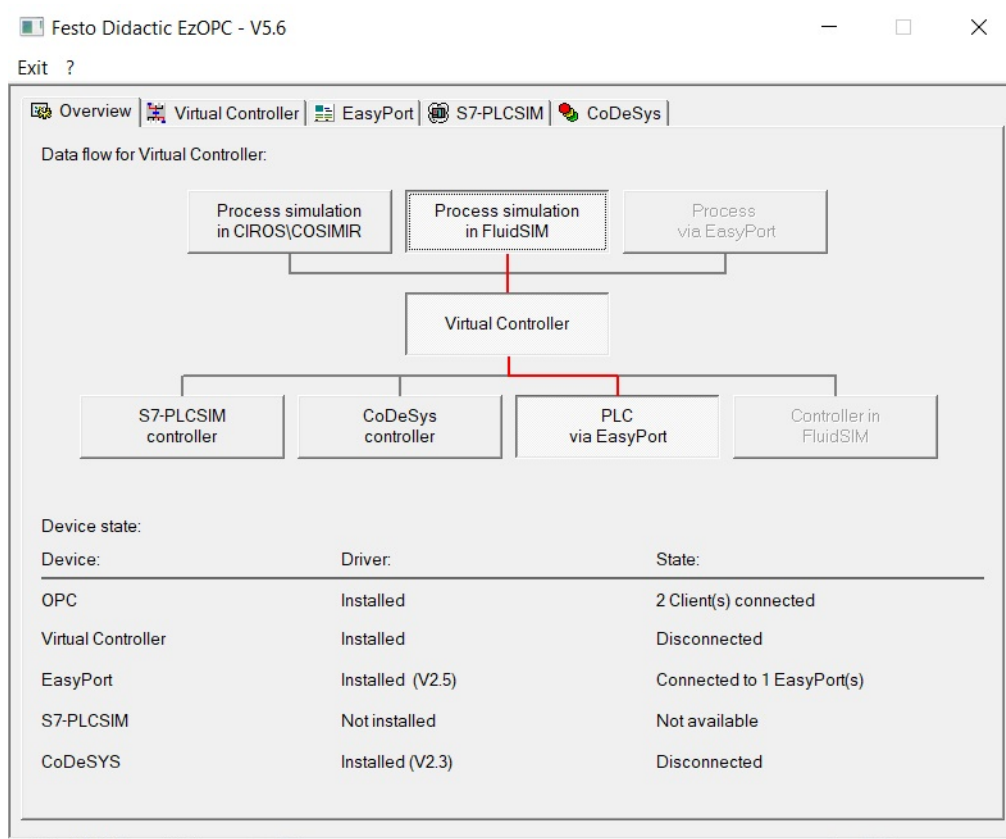


Figure 5.6: OPC Overview

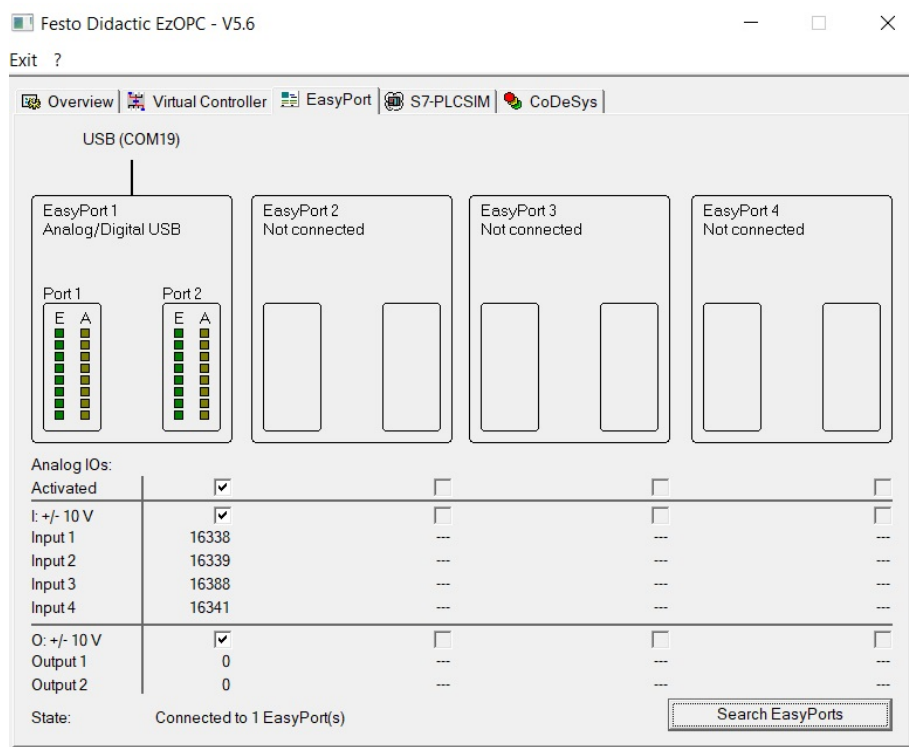


Figure 5.7: OPC IOs Status

5.4.3 Remote controlling

Easyport has the ability to show the status of each IO, thanks to its build-in LEDs. This can provide local feedback. However, remote access can not be done directly. But, indirect remote control through controlling the PC where the FluidSim is installed can be done. **Virtual Start** can be implemented by a push button switch.

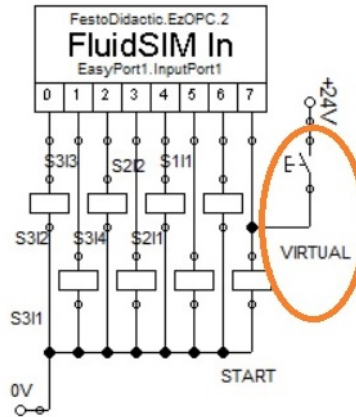


Figure 5.8: Virtual Start

5.4.4 Connections of the networks

Following the proposed scenario in 2.11 “flow chart of the process”, networks will be explained. All **pneumatic** items are being simulated as in below figure:

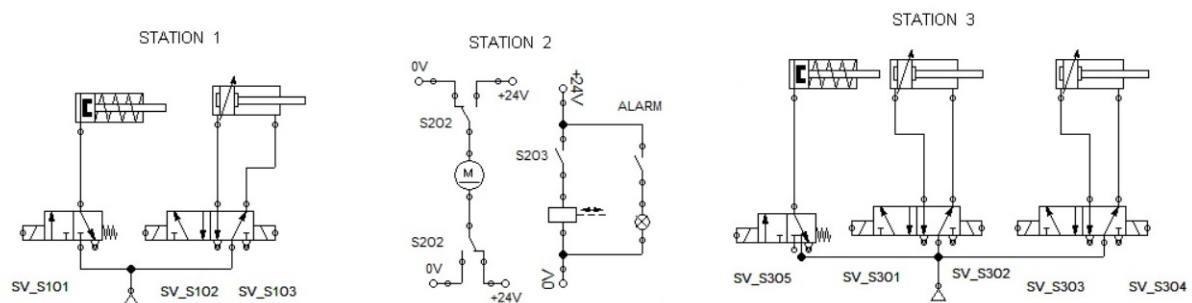


Figure 5.9: Pneumatic items

All **electrical** items including the IOs modules are being simulated as:

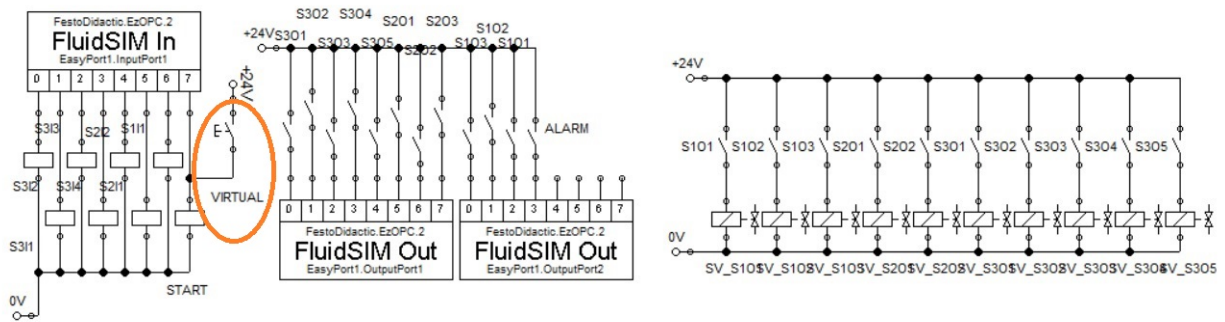


Figure 5.10: Electrical items

- **Network1:** Whenever “**Start**” coil is ON from the panel or “**Start**” from FluidSim to start the sequence, the double acting cylinder in station 1 “**S1O2**” advances till it reaches its maximum thanks to the latching. When the full advance sensor gives signal, this “**S1I1**” coil goes OFF.

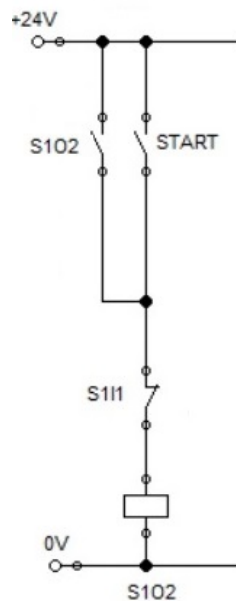


Figure 5.11: FluidSim Network 1

- **Network2:** When the double acting cylinder goes advancing, its retracting solenoid coil “**S1O1**” goes OFF. Also the single acting cylinder “**S1O3**” does the same. This is been implemented by a NOT gate of “**S1O2**”.

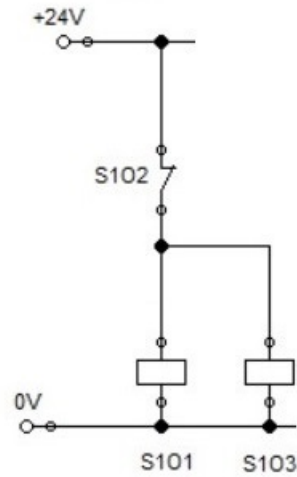


Figure 5.12: FluidSim Network 2

- **Network3:** When timer **T1** has signal of ON, the motor “**S2O1**” becomes ON to rotate backward.

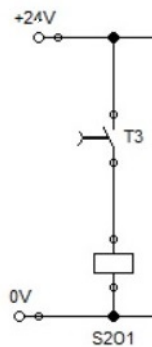


Figure 5.13: FluidSim Network 3

- **Network4:** Whenever the rejection coil “**S2O3**” goes ON, a counter counts up. Preset value must be set to the counter. When the counter reaches the preset value, “**Alarm**” goes ON. The counter is reset when there is coil Start goes ON while the Alarm is ON.

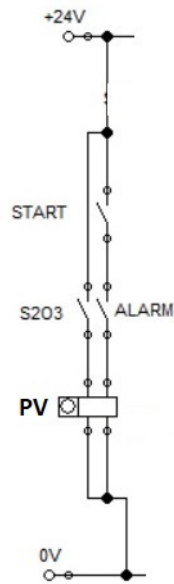


Figure 5.14: FluidSim Network 4

- **Network5:** When the color sensor “S2I1” detects the silver color, it goes ON to enable timer OFF “T1” which goes ON till it reaches its preset time.

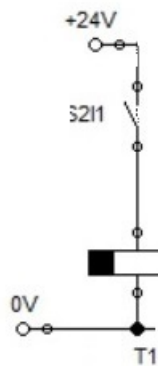


Figure 5.15: FluidSim Network 5

- **Network6:** When timer T1 has signal of ON, the motor directional coil “S2O2” becomes OFF forcing the motor to rotate backward and vice versa.

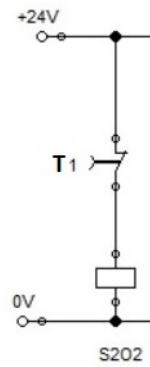


Figure 5.16: FluidSim Network 6

- **Network7:** When timer T1 has signal of ON, the rejection coil “S2O3” becomes ON.

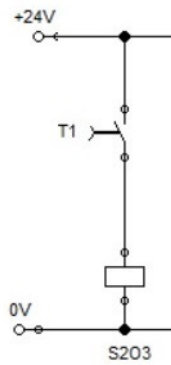


Figure 5.17: FluidSim Network 7

- **Network8:** The retracting coil for the Horizontal double acting cylinder in station 3 “S3O2” is the complement of the advancing coil “S3O1”.

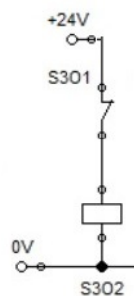


Figure 5.18: FluidSim Network 8

- **Network9:** The retracting coil for the Horizontal double acting cylinder in station 3 “S3O4” is the complement of the advancing coil “S3O3”.

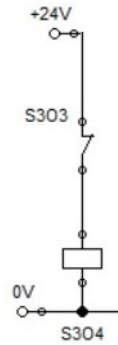


Figure 5.19: FluidSim Network 9

- **Network10:** The advancing coil for the Horizontal double acting cylinder in station 3 “S3O1” is being set by the availability sensor “S2I2” and when its retracting sensor “S3I3” is ON. It retracts or resets when it reaches the full advancing sensor “S3I3” and the gripper “S3O5” is OFF.

Thus: $S = S2I2.S3I3$ $R = S3O5 \cdot S3I3$

Since the future state “X” can be controlled through the old state “x” as:

$$X = (S+x).R'$$

$$\text{Then } S3O1 = (S2I2.S3I3 + S3O1) \cdot (S3O5 \cdot S3I3)'$$

$$= (S2I2.S3I3 + S3O1) \cdot (S3O5' + S3I3')$$

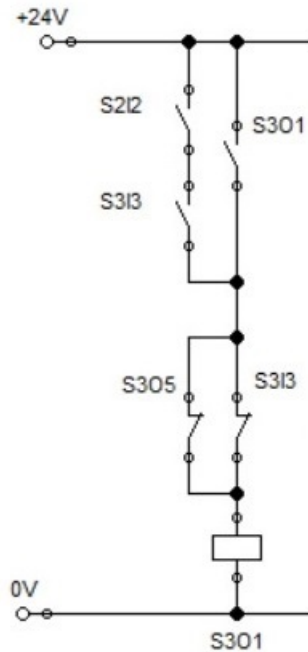


Figure 5.20: FluidSim Network 10

- **Network11:** The advancing coil for the Vertical double acting cylinder in station 3 “**S3O3**” is being set two times:

- When the gripper “S3O5” is ON and retracting sensor “S3I1” is ON.
- When the gripper “S3O5” is OFF and retracting sensor “S3I2” is ON.

It retracts or resets two times”

- When it reaches the full advancing sensor “S3I2” and the gripper “S3O5” is ON.
- When the full retracting sensor “S3I1” is ON and the gripper “S3O5” is OFF.

Thus:

$$R = (S3O5 \cdot S3I2) + (S3O5' \cdot S3I1) \quad S = S3O5 \cdot S3I1 + S3O5' \cdot S3I2$$

$$R = (S3O5 \cdot S3I2) + (S3O5' \cdot S3I1)$$

Since the future state “X” can be controlled through the old state “x” as: $X = (S+x) \cdot R'$

$$\begin{aligned} \text{Then } S3O3 &= (S3O5 \cdot S3I1 + S3O5' \cdot S3I2 + S3O3) \cdot ((S3O5 \cdot S3I2) + (S3O5' \cdot S3I1))' \\ &= (S3O5 \cdot S3I1 + S3O5' \cdot S3I2 + S3O3) \cdot (S3O5' + S3I2') \cdot (S3O5 + S3I1') \end{aligned}$$

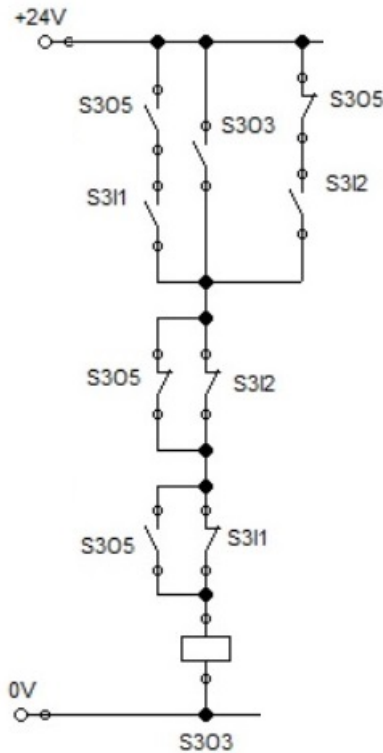


Figure 5.21: FluidSim Network 11

- **Network12:** The gripper in station 3 “S3O5” is being set when both the full advancing sensor “S3I2” and full advancing sensor “S3I4” are ON. It retracts or resets only when both full retracting sensor “S3I1” and full advancing sensor “S3I4” are ON.

Thus: $S = S3I2.S3I4$ $R = S3I1 \cdot S3I4$

Since the future state “X” can be controlled through the old state “x”
as: $X = (S+x).R'$

Then $S3O1 = (S3I2.S3I4 + S3O1) \cdot (S3I1 \cdot S3I4)' = (S2I2.S3I3 + S3O1) \cdot (S3I1' + S3I4')$

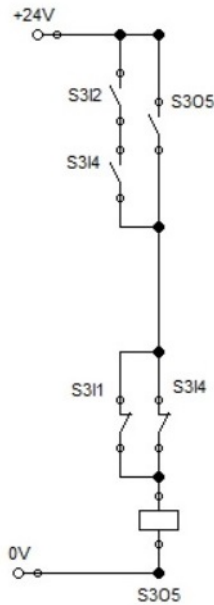


Figure 5.22: FluidSim Network 12

5.5 Testing and monitoring

- FluidSim can be a free demo software with limited time. While the unlimited version needs a license, which is over cost.
- FluidSim is easy to handle but it needs much focus for many networks to avoid mistakes.
- FluidSim has the option of implementing both the **pneumatic** and **electric** circuits.
- Timers and counters can be **ONLY** adjusted offline from the FluidSim **only**.
- **Monitoring** the status of all IOs can be done through the FluidSim and through the Easyport also.
- Although the price of the Easyport is not cheap, it is the only required component.
- Least wires are needed which makes the least maintenance effort and easiest debugging.
- Since Easyport is acting as a gate, it does not have a **memory**. FluidSim is always needed to operate the system.
- Status of IOs can be provided through the FluidSim interface only. While there is **NO** direct remote control to the system.

Chapter 6

The Panel layout

Two switches were created to control the +24V power supply; one for the LOGO! and the second one for the Easyport.

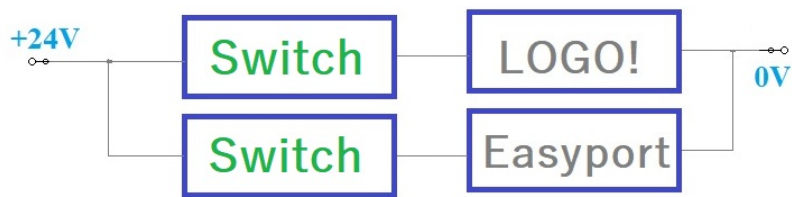


Figure 6.1: Power Supply switches

“**Start**” pushbutton was made to supply its signal to all three controllers. Its connection as in below figure.

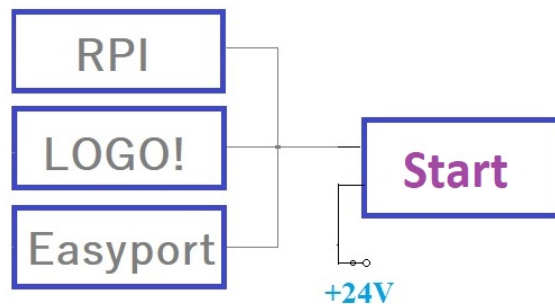


Figure 6.2: Start pushbutton wiring

“**Alarm**” was established by a 24V led. Its ground was directly connected to the 0V, while the other part was connected to all three controllers as illustrated in below figure.

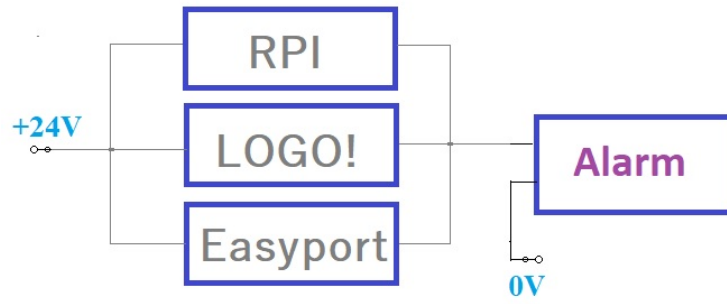


Figure 6.3: Alarm led wiring

The panel was created to include all three solutions together with their external IOs.

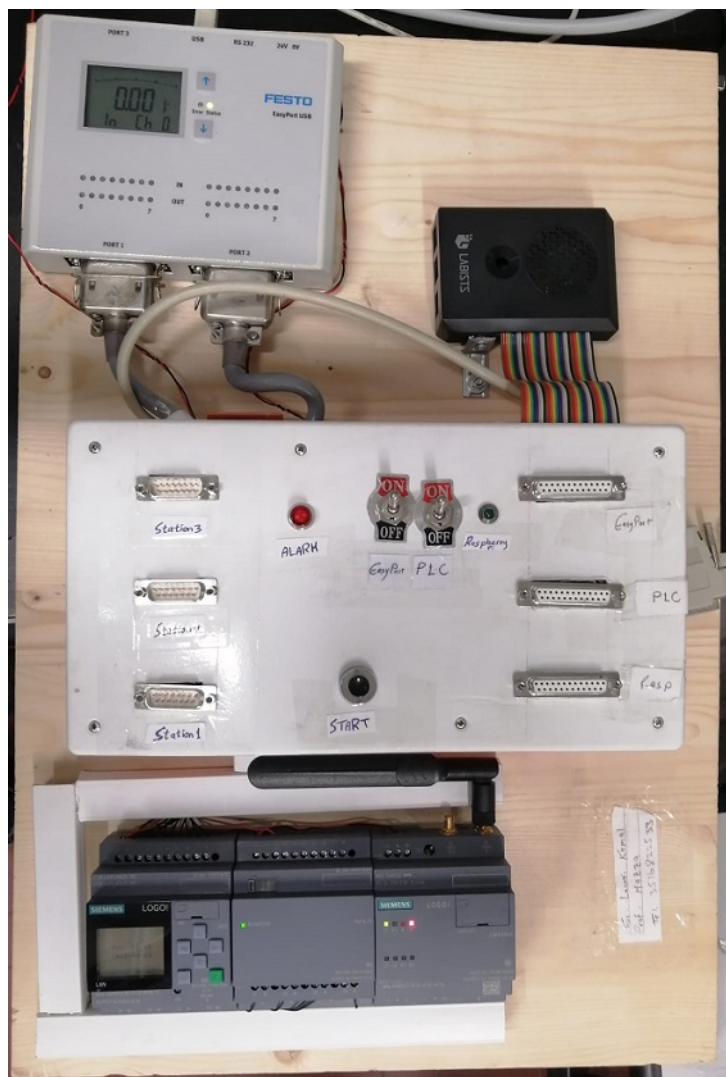


Figure 6.4: Panel layout

Chapter 7

Conclusions and Future Work

7.1 Conclusions

Festo Meclab, a Small-scale **electropneumatic assembly line** was the system under the study to differentiate three methods of low-cost controllers. Same assembly line setup was applied for each controller and notes were collected.

As regards for the behaviors studied, all inputs and outputs were **totally controlled** according to the proposed scenario. Therefore, good results were reported for each type of control **successfully**.

In particular, as mentioned in the previous chapters, each controller has its own **pros and cons**. Through the fig. 7.1 it is possible to see all the experimental notices performed with respect to each controller under same testing conditions. These three systems were compared in a **0 to 3 scale**; which **0** indicates to the **minimum** indicator incurred and **3** refers to the **maximum**.

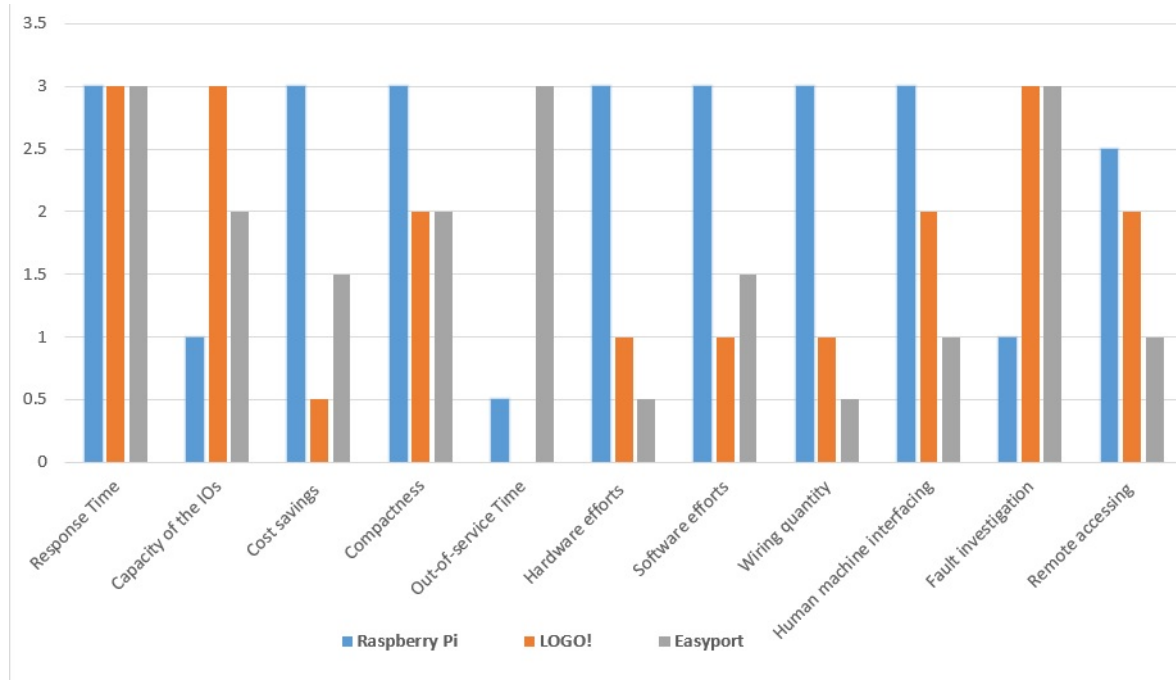


Figure 7.1: Final results

Based on the figure it is possible to note the following:

- **Raspberry Pi** is the most cost-effective controller for limited IOs due to its restricted number of GPIO. Although it is compact, it needs great efforts to build its IOs modules, which demands more wires, mapping, and harder fault tracing. It worth to mention Codesys has the ability to create the best **user-friendly** interface, but the Raspberry pi should be connected to the same interface. Thanks to its operating system and RAM, Raspberry pi can be a **stand-alone** controller without the need to re-launch the program. Thanks to its GPIOs flexibility, each GPIO can be programmed to be input or output.
- **LOGO!** is a modular controller with ability to handle great number of IOs with fault investigation in less wiring and faster response time. These features come with the **highest cost** for the hardware and the software as well. Communication module is a great method to control the system through the SMS from SIM card. However, Network signal **delay** may occur.
- **EasyPort** is a great IOs gate for FluidSim program. It has fast response time, informative fault tracing ability in minimum wires for up to **32** inputs and **32** outputs in user-friendly software. From the other side, it does not support direct remote access since it depends on FluidSim for operation. **No memory** means offline control is not an option at all times. Although its local online monitoring is not attractive, it comes with relatively low cost for both software and hardware.

7.2 Future Work

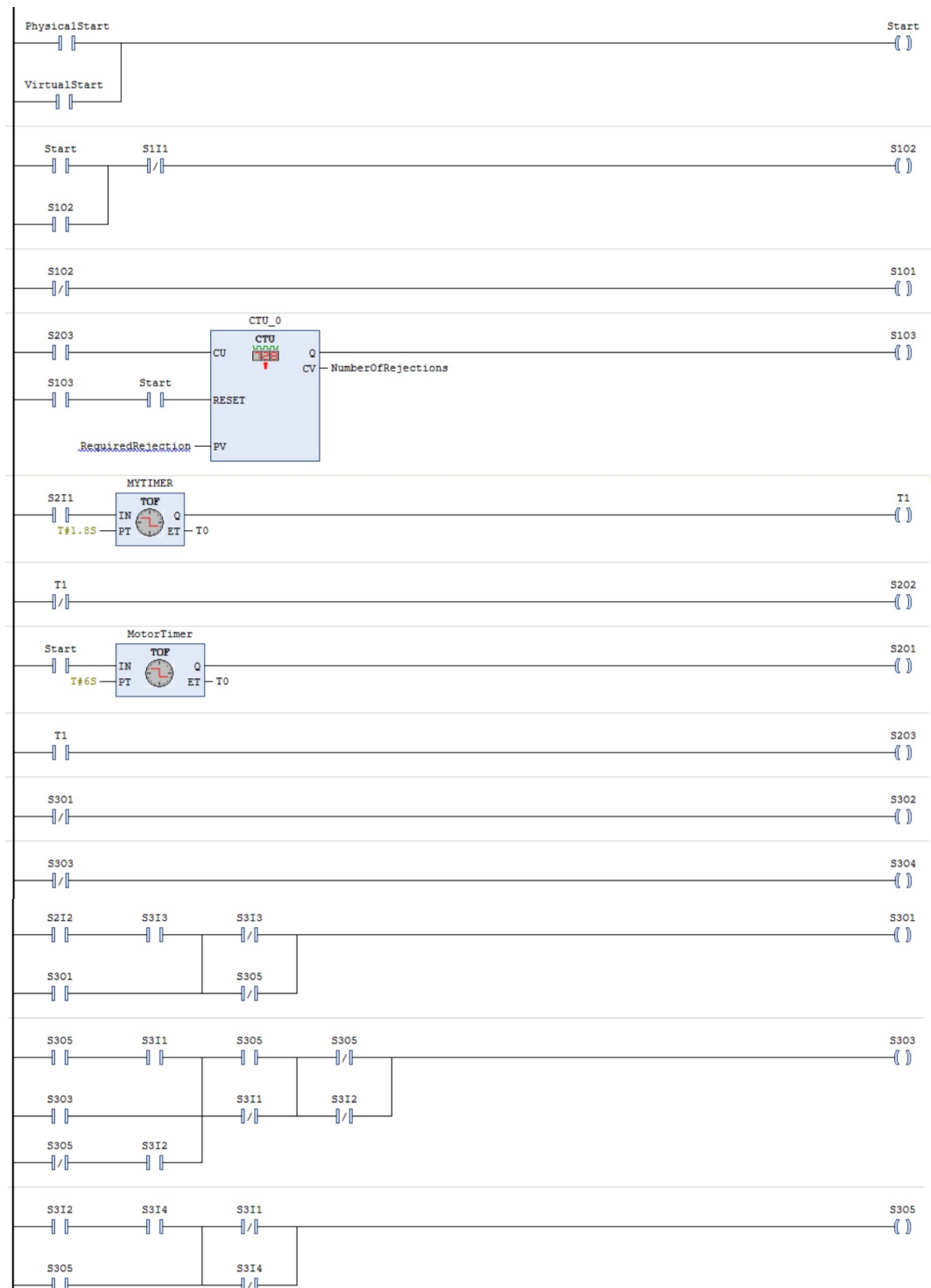
Possible future developments are:

- **Eliminating** the communication module can reduce the cost for the LOGO!. Then, connecting it through local network can provide **cheaper** accessibility.
- Using **more** than one Raspberry pi could be a good option for limited number of IOs.
- Using **sharing** software to control the computer where FluidSim is installed can be affordable solution to control remotely the system.
- **Installing** FluidSim on a Raspberry pi to substitute the need for a **computer**.

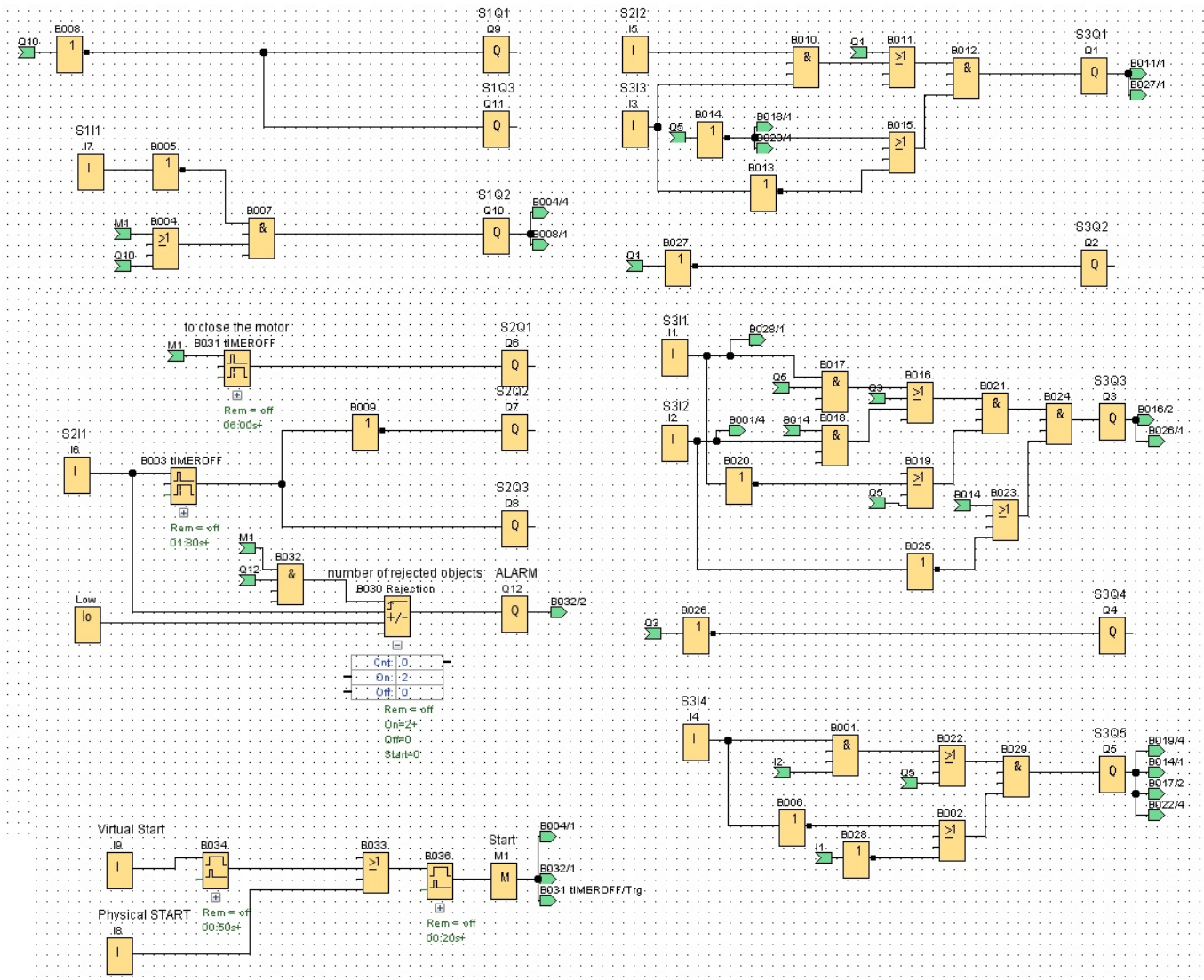
Chapter 8

Appendix

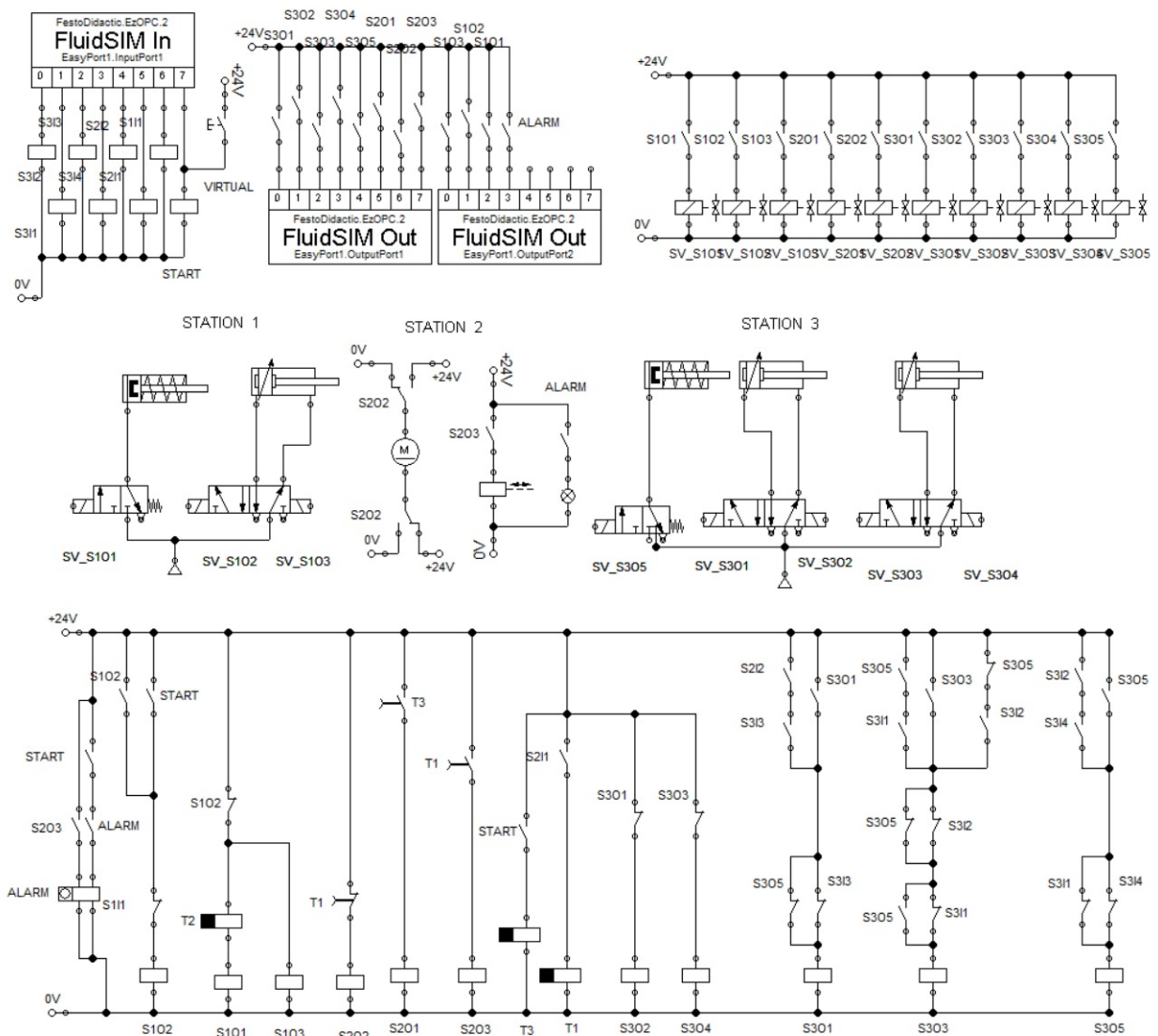
8.1 Appendix A



8.2 Appendix B



8.3 Appendix C



Bibliography

- [1] Vishay Semiconductors, Document Number: 81181, Rev. 1.2, 07-Jan-10. 3.2.1, 3.2.1
- [2] Standard IEC 61131, March 2017.
- [3] Festo MecLab stations
FESTO Comp. <https://www.festo.com/us/en/e/technical-education/learning-systems/stem/meclab-r-id32631/>, 2021. 2
- [4] LOGO! CMR manual:
Siemens Comp. <https://cache.industry.siemens.com/dl/files/268/103657268/att850514/v1/BALOGO-CMR2020-CMR204076en-US.pdf>, 2019. 4.3.3
- [5] Alireza Qadiri. Automatic Control, PLC Programming and Simulation of Pneumatic Cylinders' Lifetime Testing. Rel. Luigi Mazza. Politecnico di Torino, Corso di laurea magistrale in Mechatronic Engineering (Ingegneria Meccatronica), 2021. 4.1
- [6] Ankush Das., «Single Board Computers: Alternatives to Raspberry Pi» July 21, 2020. 3
- [7] Nishchal K. Verma; Teena Sharma; Seetaram Maurya; Dhan Jeet Singh; Al Salour, 2017 IEEE International Conference on Prognostics and Health Management (ICPHM). DOI: 10.1109/ICPHM.2017.7998316. Date of Conference: 19-21 June 2017 5.1
- [8] Manual EasyPort USB 721876, Festo © 2022. 5.1