Comparison between Codesys and OpenPLC as a Modbus TCP Protocol Integrated Development Environment (IDE)

Andika Bramantio Wicaksono

MTE2019

Abstract

Modbus TCP Protocol is one of the protocols it is applicable in many industrial applications systems that make a system communicate with one another. One of the uses of this Modbus TCP protocol is acting as the bridge of communication between a Programmable Logic Controller (PLC), Human Machine Interface (HMI), the controlled machine, and the integrated development environment (IDE).

The term of the PLC IDE that can work with the Industrial Standard is often associated with exclusive or expensive software/equipment. This can be a challenge for the first-time learner for the new generations of automation engineers learning the basics of automation programs.

There are many choices of software that can be selected when choosing which programs that is suitable for learning the PLC. Preferably is the one that can work with many of the current industrial equipment, is easy to use, and one of the most important parts is the accessibility of the software which is at least free-to-use software.

One of the few selected choices that comes into mind when choosing free software that can work with industrial equipment is Codesys and OpenPLC. Both are powerful program that was designed to be used with many Industrial Equipment and can be equipped with the latest technology in communication and protocol.

Keywords: Modbus TCP, Industrial Internet of Things, Programmable Logic Controller, Automation, Ladder Logic Diagram, FactoryIO

Chapter 1

Introduction

1.1 Background

The Manufacturing Industry in the 21st century relies on certain things. The combination of substantial human resources, computer-oriented integration, and automation. Making the manufacturing operation a sustainable and optimized operation in the company. The main backbone for the machine to be able to be working as it was intended to is the Automation Scripts that run in the background.

Many varieties of software can be used to make the automation scripts that were used in the industrial-grade standard. Such as the Totally Integrated Automation Portal (TIA Portal), Control Development System (Codesys), CX-Programmer (Omron), etc. However, not all of this software came in handy when it comes to being used by a first-time learner of the programmable logic controller (PLC). It is always for some reasons like it's not open to the public, or it is way too expensive to be purchased in the first place which is not suitable for learning it the easier way. In this case, an alternative integrated development environment (IDE) is needed to compensate for such a requirement.

This paper aims to compare two of the most popular, free-to-use programmable logic controller IDE that is popularly used to connect with the Modbus TCP Protocol. Codesys and OpenPLC are the two software that is popular in terms of making industrial automation programming system. Both can also establish a Modbus communication that lets IDE, Human Machine Interface (HMI), PLC, and the automated machine communicate with each other using an ethernet-based network.

This paper will provide a comprehensive analysis of the two IDEs, including their advantages and disadvantages, to help readers make an informed decision when selecting an IDE for their automation program and their integrated development through the Modbus protocol.

1.2 Objective of this Thesis

The objective of this thesis research is to find out which software is the better alternative IDE for making the automation program. In this case, is a comparison between two free-to-use software. The Control Development System (Codesys) developed by the Codesys Group and OpenPLC developed by the OpenPLC Development Group.

1.3 Research Purpose of The Thesis

The purpose of this thesis research is to make a clear difference between free-to-use software that is used in the development of making automation scripts which in this case is the Ladder Diagram/Ladder Logic Diagram (LD/LLD). Although it might be seeming that it has no difference between software usage. But it can be challenging when it is executed, especially when facing difficulties in certain software and its application when applied to real-world manufacturing machinery.

This also aims to help the new engineers/students who might be interested in the automation programming system but might find it challenging when choosing the first software to be used for first-time learning. Counting from user-friendliness, easy use, software documentation, and community support.

1.4 The Scope of the Thesis

The following is the scope of comparing the Control Development System (Codesys) and the OpenPLC

Codesys and OpenPLC are both IDE for making Ladder Logic Diagrams. This thesis would include the overview perspective from the both IDE and the Modbus TCP protocol. Comparison between their feature

1.5 The Limitations of The Thesis

- All of the Factory Simulation would be done in the FactoryIO simulation software and using a custom scene based on the Advanced by Height Scenario.

- The Codesys PLC would be simulated by using the Codesys Control WinV3 x64 virtual PLC and the OpenPLC using the OpenPLC runtime virtual PLC.s
- Communications between the IDE and the FactoryIO would be strictly limited to the Modbus TCP Master/Slave (Server/Client) protocol.

Chapter 2

Literature Review

2.1 Modbus Protocol

Modicon bus communication protocol (Modbus) is a series of communication protocols that were developed by Modicon (now part of Schneider Electric). It has become the standard for the manufacturer to use the Modbus protocol to transmit data from one device to others, for example, PLC, HMI, RTU, and many other smart instruments.

The device can request and transfer data over the serial line of information. In a standard Modbus network, the devices that request the information are called the Modbus Client/Slave, and the device that supplies the information is called the Modbus Server/Master. Some devices also may function as both Client/Slave and Server/Master at the same time.

A Client/Slave device is a device that can process any data or sets of information sent by the Server/Master and later return the output using Modbus. In the meanwhile, a Server/Master is a device that acts as the host computer running the application software. A Server/Master has the right to assign an address to each Client/Slave, initiating broadcast messages to all the slaves in the network. Each Slave is required to respond to all the inquiries and messages that the server has given to individuals. A slave device cannot initiate communication on their own or communicate with each other, they can only respond to inquiries and messages from the Master.

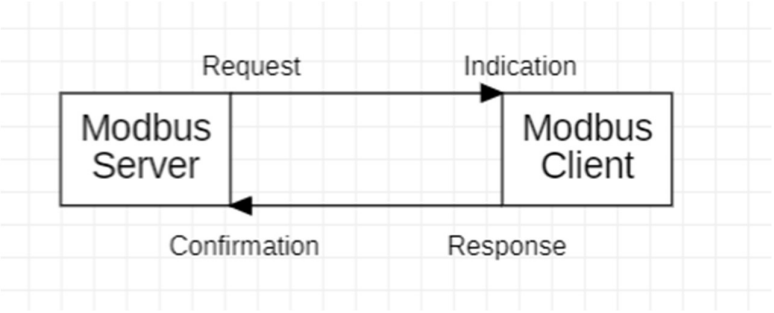


Fig 2.1 Example of Modbus Server and Client Communication

2.1.1 Modbus TCP/IP Protocol

The Modbus TCP/IP is a variant of the Modbus Communication Protocol Family that is intended to use as the controller and supervisor over any automation equipment. The Modbus TCP/IP communicates through many of the devices attached by using the Ethernet Network. The most commonly attached device to this Ethernet network is PLC.

The TCP/IP itself refers to Transmission Control Protocol and Internet Protocol. The TCP/IP protocol allows data in the form of binary blocks transmitted between devices and ensures that the right device receives the correct all sets of data correctly over the Ethernet network. And it has been the worldwide standard that helps to become the foundation of the Internet or the World Wide Web. However, the Modbus TCP/IP combination is just limited to a transport protocol, which means it does not help define how the data is interpreted, which in this case is the Modbus job as the application protocol.

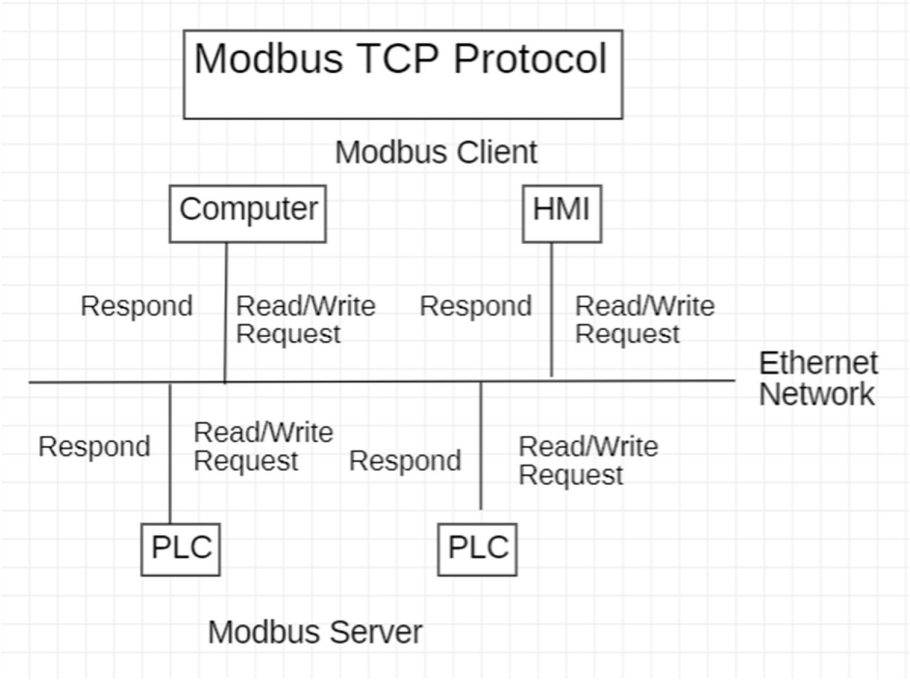


Figure 2.2 Modbus TCP Protocol Diagram

2.1.2 Usage of Modbus TCP Compared to the OPC UA

Modbus TCP over the years has been slowly replaced by a newer protocol called Open Platform Communication Unified Architecture or just simply OPC UA. It is a more advanced protocol when compared to only just using Modbus TCP Protocol. It has many advantages that the Modbus didn't able to do so. Such it can be installed directly on the system or in PLCs, offering more flexibility in terms of device selection, and a more structured data system that allows more efficient and effective data exchanges between one machine to the others.

Although it can offer many strong capabilities Modbus can't afford to do so. In several cases, selecting the Modbus TCP protocol is still a viable choice. Such as installing OPC UA software and hardware can be very expensive to be installed in the devices. Or for a small manufacturing plant that doesn't need expensive equipment yet for their machinery. And compatibility with an older system that might have some problems running OPC UA that might it can be vulnerable since their security system may not have met the modern-day standard for machines security.

2.1.3 Types of Modbus Protocol Used in the Communication

Codesys Supports several types of Modbus Configuration:

1. Codesys acting as the Modbus Master
2. Codesys acting as the Modbus Master (Client)
3. Codesys acting as the Modbus Slave
4. Codesys acting as the Modbus Slave (Server)

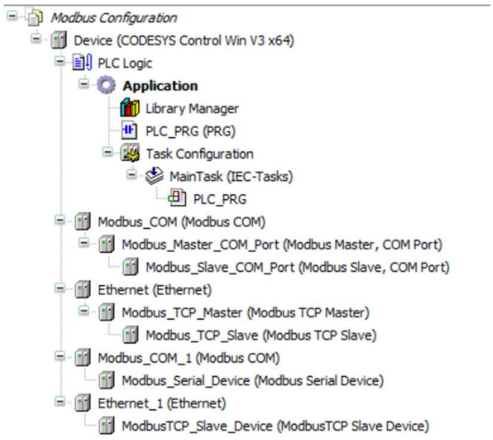


Fig 2.3 Possible Codesys Modbus Configuration

However, for this thesis, Codesys will act as the Modbus Slave (Server)

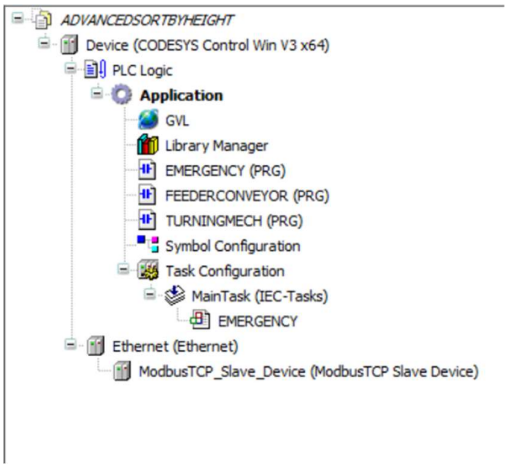


Fig 2.4 Codesys Modbus Slave (Server) Configuration used in the Thesis

The Modbus Slave (Server) configuration is selected because of the simplicity it offers compared to the other configuration available. By only just selecting which network

interfaces would be used for the communication and setting up the tags that match the FactoryIO tags.

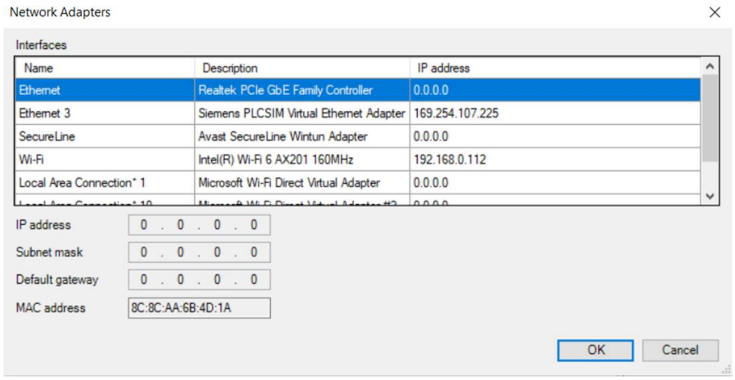


Fig 2.5 Network Adapters Configuration

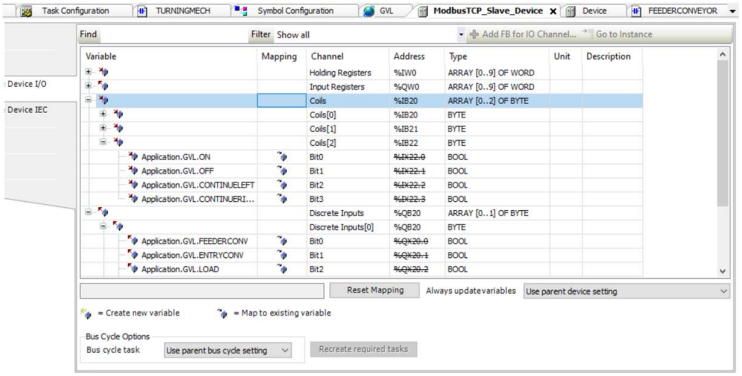


Fig 2.6 Tags Configuration Example

2.1.4 Modbus Function Codes

There are 4 types of Objects Type in Modbus Function Codes with each having its purposes and configuration.

OBJECT TYPE	ACCESS	SIZE (BIT)
Holding Register	Read-write	16
Input Register	Read-only	16
Coil	Read-write	1
Discrete Input	Read-only	1

Table 2.1 Modbus Function Codes

In the project used for the Thesis, only the Coil and Discrete Input will be used for communication. Since Holding Register and Input Register are only for working for Analog Devices. The Coil represents the value that can be both read or written by the Master device. Meanwhile, Discrete Input represents either the ON or OFF state of a Sensor

2.2 Programmable Logic Controller (PLC)

Programmable Logic Controllers or simply known as PLCs are computer-based single-processor devices that execute an instruction set that is more known as an Electrical Ladder Diagram. It is also capable to take control and managing a wide range of industrial equipment and some of its type can even take over control of the whole automated systems. PLCs are the main heart and brain of every automatic system in the industry, they are known for being efficient and reliable in industrial applications that involve sequential steps and synchronization between one device with the others.

The Logic in the Programmable Logic Controller is used because the programming language used in the PLC is primarily involving logic and switching operations. The Language that can be implemented in the PLC program is the following:

- Ladder Logic Diagram (LLD)
- Structured Text (ST)
- Function Block Diagram (FBD)
- Sequential Function Charts (SFC)
- Instructions List (IL)

PLCs were first used to replace basic mechanical and electrical relay systems in the 1960s. The automation at the time primarily consisted of electromechanical relays and coils that were hardwired on panels. However, this electromechanical relay system can cause a nightmare if just one relay has issues that could affect the whole system's operation. Nowadays, The Programmable Logic Controller consists of a Power Supply, a CPU, an I/O memory card to store the memories, and a rack where the I/O cards are placed.

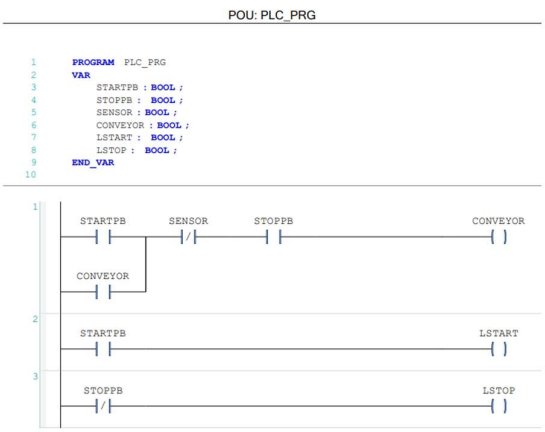


Figure 2.7 Example of Basic Ladder Logic Diagram Language for a Basic Roller Conveyor System

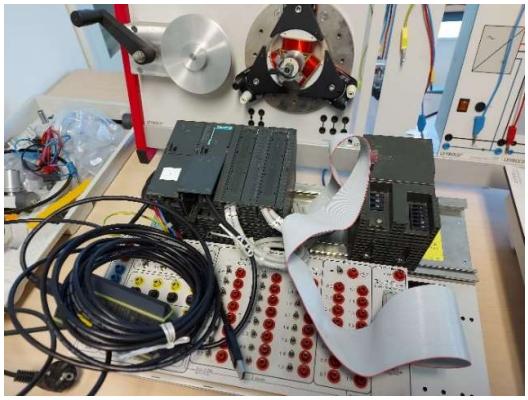


Figure 2.8 A Siemens S7-300 PLC

2.2.1 SoftPLC / VirtualPLC

A soft PLC is a software that can be installed in computer devices that can be used for providing the functionality of a Programmable Logic Controller (PLC). Typically, they're used to simulate the control and process of an industrial automation system. A soft PLC may also be used as a substitute for physical PLCs. They are also known for versatile and cost-effective solutions for small-scale industrial businesses and learning material for new students and engineers.

2.2.2 Advantages and Restriction of Virtual PLC over Real PLC

There are several advantages of using a Virtual PLC over a real physical PLC such as:

1. Costs

   A virtual PLC can be installed in an everyday computer and laptop that people use daily. Some software even offers free installation (no purchases required beforehand).

2. Flexibility

   Virtual PLCs tend to be very flexible when compared to their physical counterpart. It can be modified and hacked into the user's requirement as much as they wish to meet the program's requirements.

3. Easy to set up and learn

   Installing a virtual PLC is as easy as installing regular software on the computer. They also provide a safe environment for learning industrial-grade hardware at zero risking damaging expensive industrial equipment.

Although it has several advantages, Virtual PLC also has several limitations that only Physical PLC possessing it:

1. Simulation Based on Ideal Condition

   Virtual PLCs simulation is always based on a perfect no error simulated environment. They don't have the capability of detecting whenever there are physical wiring errors when I/O doesn't match the assigned address on the computer and the wiring.

2. Performance and I/O limitations

   When it comes to the performance of Virtual PLC. Their performance can depend on the user's hardware specification which can lead to results inconsistencies. Also, most of the available Virtual PLCs have limited I/O that they can support at one time. Which can be affected when Virtual PLC tries to communicate with real word devices.

2.2.3 Ladder Logic Diagram

Ladder Logic Diagram or Simply LLD is a programming language that instead of using text, uses many combinations of symbolic graphics elements that are called symbols. Created to be used by Electricians and Technicians. Since they were made to represent and look like an electrical diagram. Electrical contacts and relays do exist in Ladder Logic (although it is called coil for the relay in ladder logic). The symbol might be different from Electrical Circuit but they have a similar function to do the same job.

Some advantages of Ladder Logic when compared to a similar programming language that is supposed to do the same job:

1. It's easy to read and understand Ladder Logic Diagram
   When it comes to readability and the experience during the programming or learning the Ladder Logic. It is easy since the natural reading sequence of most people that goes from the left side to the right side.
2. Structure of the Execution of the Program

   The ladder Logic Sequence is going from Left to Right and From Top to Bottom, the order of the programming can make the program precisely determine which instructions to be executed first. From one Ladder to another is separated by a line called rungs that will stack on top of each other that somehow will look like a ladder.

3. Flexibility of Programming
   Programming Ladder Logic is pretty easy, especially for those who have Electrical Background or ever working with Relay Logic. For example, you can write the program and debug it on a piece of paper first before implementing it on the computer.

2.3 Software Proposed for the Thesis

For the current thesis, there are several software that is proposed to be used for the research and development of the current thesis topic.

2.3.1 Control Development System (Codesys)

Control Development System or simply known as Codesys is multiple languages, multipurpose integrated development environment (IDE) that is used to program controller and industrial computer programming development. Codesys were developed by 3S-Smart Software Solutions later known as the Codesys Group / Codesys GmbH. This development environment offers user-friendly and free-to-use software to help program the PLCs. However, not all the PLCs can be programmed using this IDE.

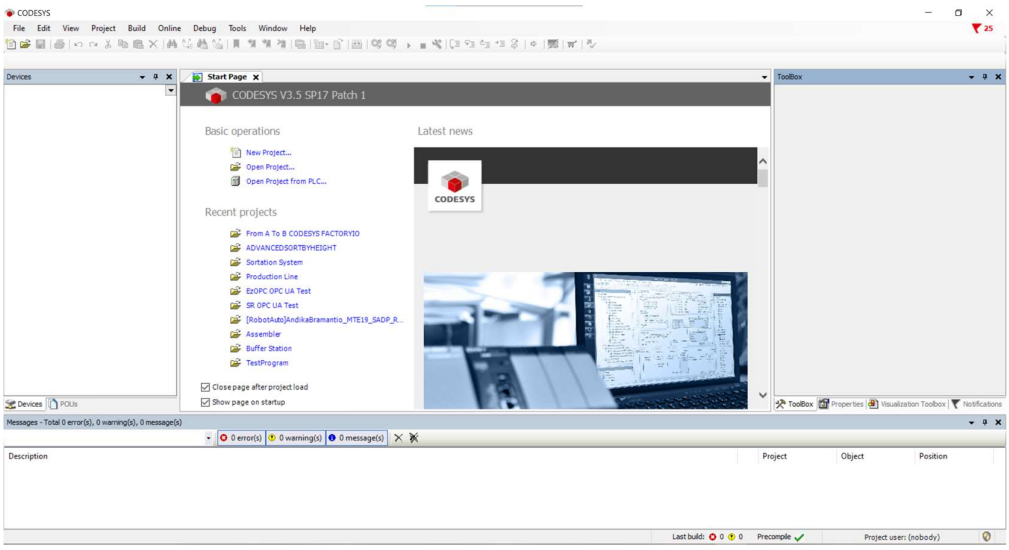


Figure 2.9 The Codesys IDE Home Screen Welcome Page

Codesys offers not also provide the IDE environment to program the language to make the automation system. But also provides the virtual PLC that is called Codesys Control Win V3 x64. It simulates and can act like an actual industrial PLC that can be used to run automation simulations.

2.3.2 OpenPLC

The OpenPLC is an open-source PLC software that offers multi-language, multi-hardware variations that are based on the Beremiz IDE. It is the first of its kind that is a fully functional and industrial standardized open-source PLC that was created by the IEC 61131-3 standard that defines basic software architecture and programming languages to the PLC. The Open PLC splits into two parts: The Editor and the Runtime. The Editor is standard software that was used to create the PLC program. Meanwhile, the OpenPLC runtime is a software designed to be able to be executed on many various devices.

- Arduino Uno / Nano / Leonardo / Micro
- Arduino Mega / Due
- Arduino Nano Every / IoT / BLE
- Arduino RB2040 Connect
- Arduino Mkr / Zero / WiFi
- Arduino Pro (Machine Control and EDGE)
- Controllino Maxi / Automation / Mega / Mini
- Productivity Open P1AM
- ESP8266 (nodemcu)
- ESP32
- Raspberry Pi 2 / 3 / 4
- PiXtend
- UniPi Industrial Platform
- Neuron PLC
- FreeWave Zumlink
- FreeWave ZumIQ
- Windows (generic target as a soft-PLC)
- Linux (generic target as a soft-PLC)

Figure 2.10 List of Devices supported by the OpenPLC Runtime

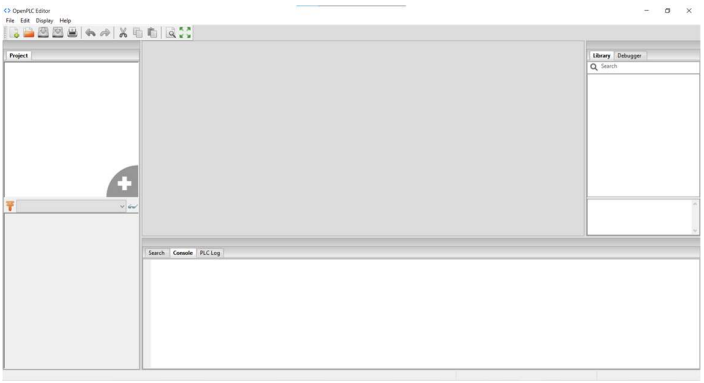


Figure 2.11 OpenPLC Editor IDE Home Screen Welcome Page

For this thesis research project. The Device that will act as the PLC will be the Windows OpenPLC runtime for the simulation and in a more realistic case scenario the PLC would be carried out by using a Raspberry Pi 4B.

2.3.3 Factory IO

FactoryIO is software that was designed to simulate a 3D environment of a manufacturing plant. It allows the users to create a design, test, and simulate many various industrial processes automated system. The software is designed to be able to work with

various devices from different manufacturers such as Siemens, Allen Bradley, MHJ, and even simulated PLC or from a third-party manufacturer through various available protocols.

Figure 2.12 A Typical FactoryIO Scene (Production Line)

2.3.3.1 Factory IO Drivers

IO drivers or I/O drivers is a feature of Factory IO that is acting as the relay bridge between the Factory IO and the PLC Controller. Some brands even have built-in driver support like Allen Bradley and Siemens controllers.

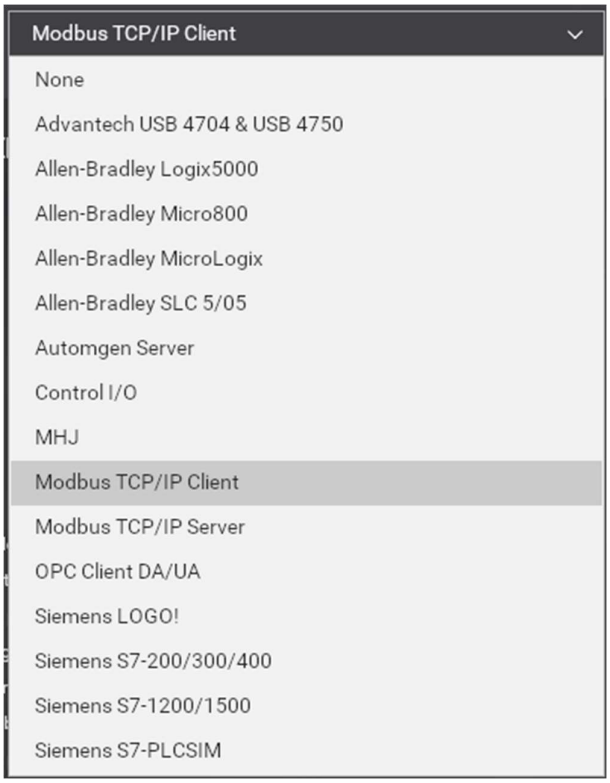


Fig 2.13 List of Drivers supported by FactoryIO version 2.5.2

To assign tags to each address in the FactoryIO it is just as simple as Drag and Drop the desired tags to the PLC address available.

Fig 2.14 Tags assignment in FactoryIO

Chapter 3 Research Methodology

3.1 Technical Information Regarding the FactoryIO Scene

The scene used for the FactoryIO testing for this thesis is a modified advanced sort-by-height default scenario. The modification is made because of the need of simplifying and adjusting accordingly to the programming that has been made before the scenario. The modification is including adding 2 sensors that let the machine knows when to continue feeding the pallets to the sortation machine.

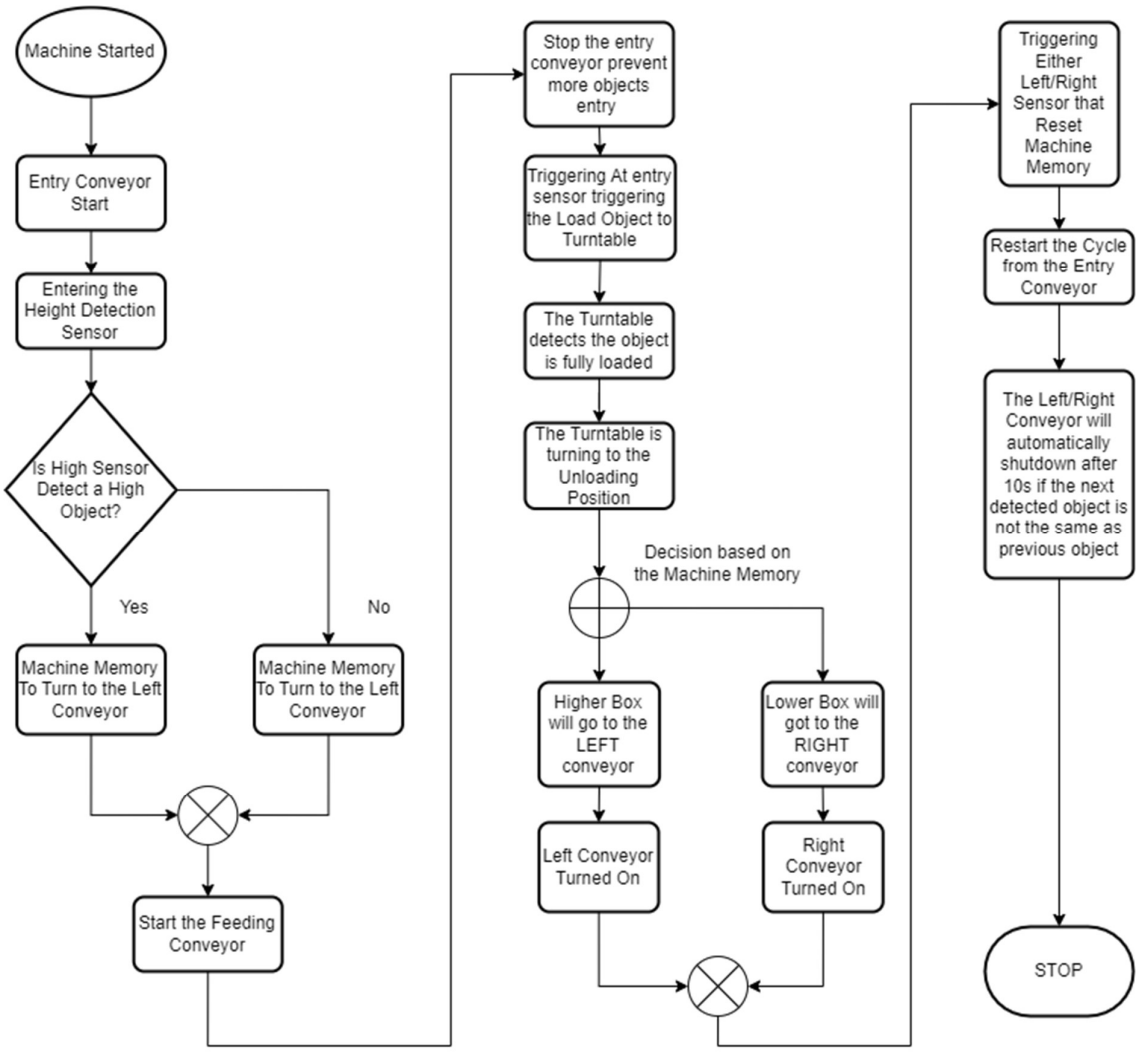


Fig 3.1 Scenario workflow flowchart

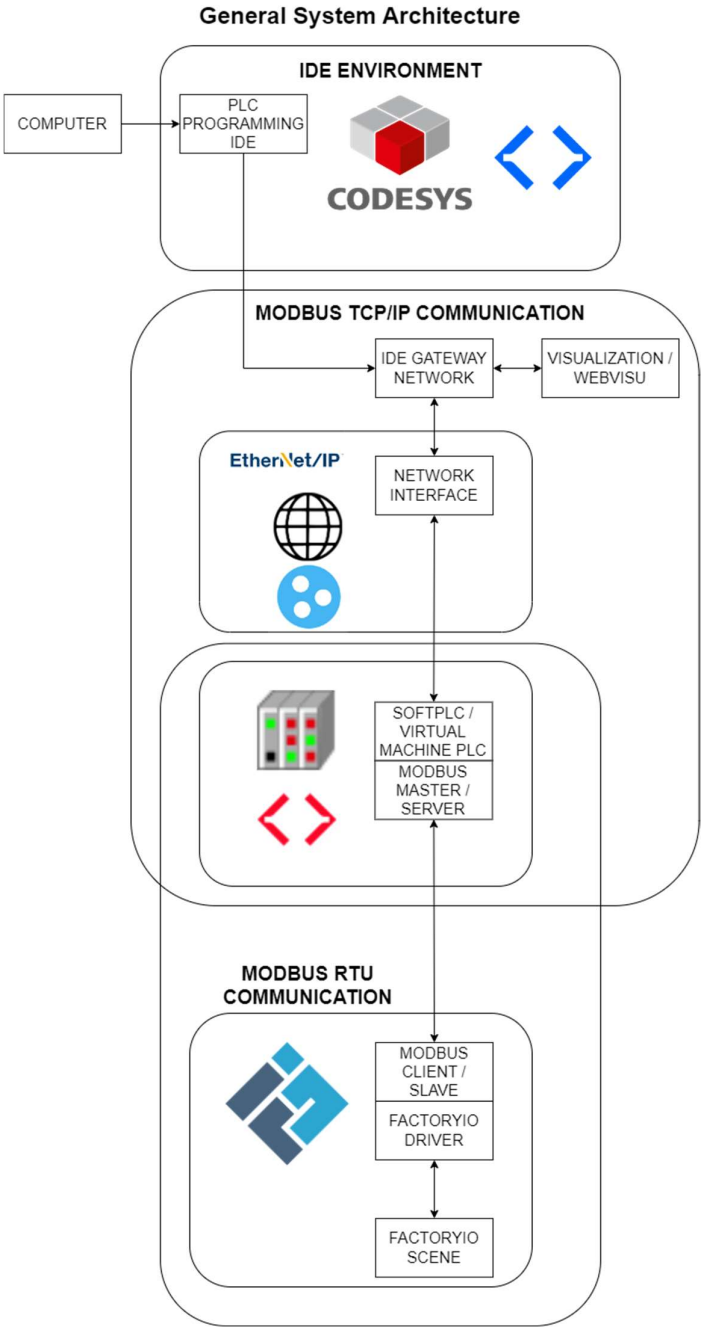


Fig 3.2 General PLC System Architecture

Components used in the custom scenario in FactoryIO include:

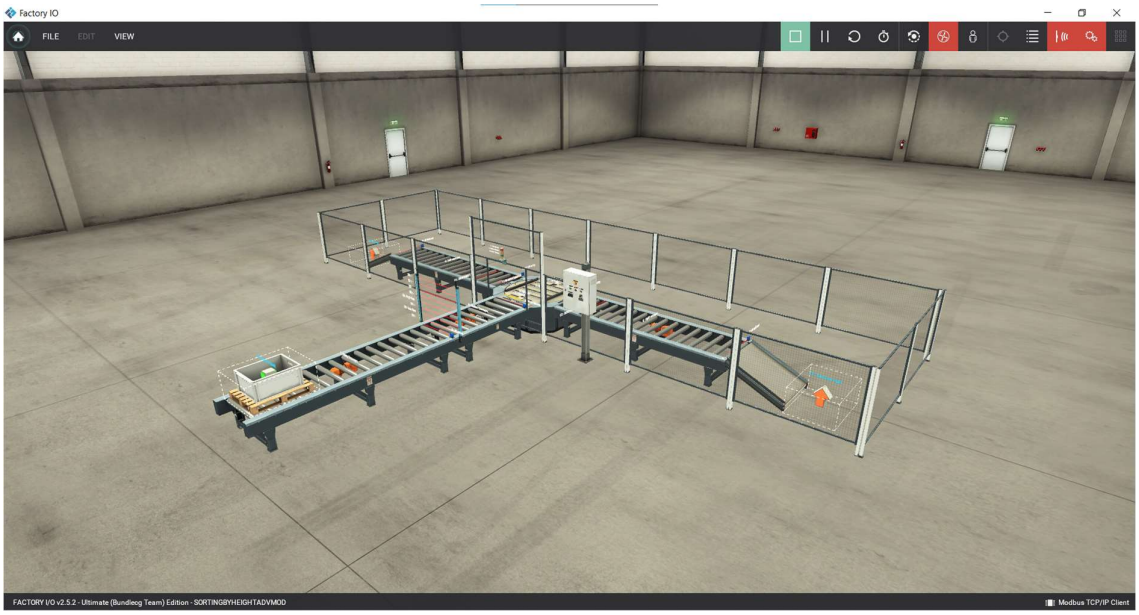
- 3x Roller Conveyor 4M
- 1x Roller Conveyor 2M
- 1x Light Array Emitter
- 1x Light Array Receiver
- 8x Diffuse Sensor
- 1x Turntable
- 1x Control Panel
- 2x Chute Conveyor Low
- 1x Stack Light
- 

Fig 3.3 The Scenario Used

An interval of 2s between emitting from one object before letting another one spawn/enter into the system is required to make the system not confused with the current object that is already in the system or make the system memory error conflict with a new one.

List of Spawned Objects: Box (S Size, M Size, L Size), Stackable Box, with Wooden Pallet below each object.

3.2 I/O Assignment

Input Output Assignment or simply I/O is the list of tags that were used to make the address for each variable in the system. For this thesis, the following I/O address assignment is used:

COILS	INPUT / INPUT REGISTER
At Entry	Entry Conveyor
At Back	Feeder Conveyor
High Box	Load
Low Box	Unload
At Turntable Entry	Turn
At Load Position	Left Conveyor
At Unload Position	Right Conveyor
At Front	Green Indicator
At Left Entry	Yellow Indicator
At Right Entry	Red Indicator
At Left Exit	Start Light
At Right Exit	Reset Light
Start	Stop Light
Reset	Counter (INPUT REGISTER)
Stop	
Emergency Stop	
ON	
OFF	
CONTINUE LEFT	
CONTINUE RIGHT	

Table 3.1 List of I/O Addressing

3.3 Programming Used in the Thesis

3.3.1 Codesys-Based Program

Global Variable List: GVL

```
{attribute 'qualified_only'}
VAR_GLOBAL
    ATENTRY : BOOL ;
    ATBACK : BOOL ;
    LOWBOX : BOOL ;
    HIGHBOX : BOOL ;
    TURNTABLEENTRY : BOOL ;
    LOADPOS : BOOL ;
    UNLOADPOS : BOOL ;
    ATFRONT : BOOL ;
    RIGHTENT : BOOL ;
    LEFTENT : BOOL ;
    RIGHTEXIT : BOOL ;
    LEFTEXIT : BOOL ;
    START : BOOL ;
    RESET : BOOL ;
    STOP : BOOL ;
    EMERGENCY : BOOL ;
    ON : BOOL ;
    OFF : BOOL ;
    FEEDERCONV : BOOL ;
    ENTRYCONV : BOOL ;
    LOAD : BOOL ;
    UNLOAD : BOOL ;
    TURN : BOOL ;
    LEFTCONV : BOOL ;
    RIGHTCONV : BOOL ;
    GREENIND : BOOL ;
    YELLOWIND : BOOL ;
    REDIND : BOOL ;
    LSTART : BOOL ;
    LRESET : BOOL ;
    LSTOP : BOOL ;
    CONT : BOOL ;
    CONTINUERIGHT : BOOL ;
    CONTINUELEFT : BOOL ;
END_VAR
```

Fig 3.4 Codesys Global Variable List

The Codesys Program is set by using a Global Variable List (GVL). Which is used for making the variable can be used globally in the project and not just specified to just one of the local programs (PRG).

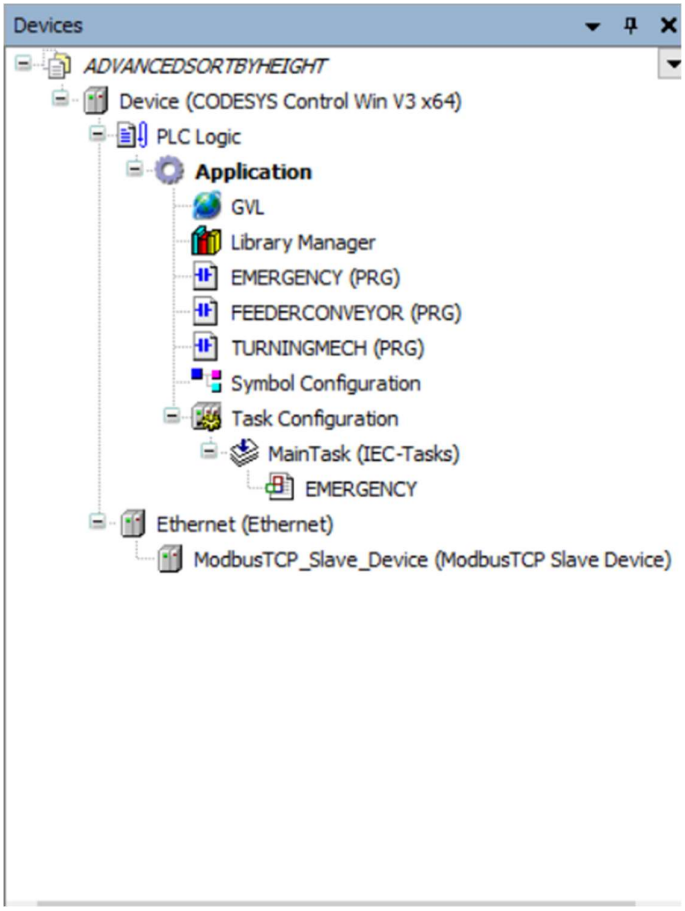


Fig 3.5 Codesys Project Environment, can consist of multiple programs at one project

As Shown in the picture above, for the Codesys there are 3 main parts for the program: The Emergency, The Feeder Conveyor, and The Turning Mechanism.

Fig 3.6 The Emergency Program

The Emergency program consists of two ladders which are Global Variable Emergency and equipped with the Feeder Conveyor and Turning Mechanism Program hooked into it. This program is used for stopping the other program when the Emergency button has been pressed by the Machine Operator.

The first program in Codesys is the Feeder Mechanism of the Scenario. Consisting of how the objects will be fed into the sensor and entering the memory to the machine of which type of object that is used in the mechanism.

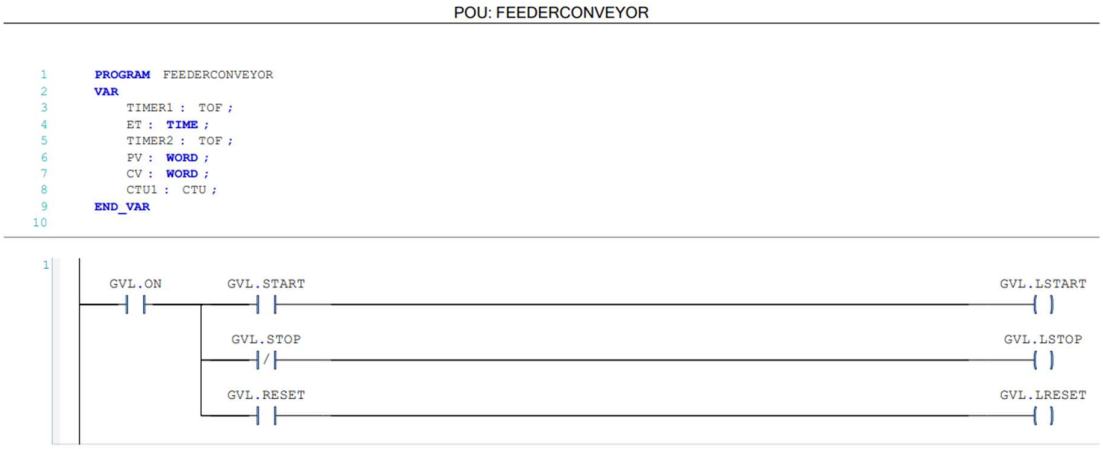


Fig 3.7 Feeder Conveyor (PT1)

In this part, the first program is to set the buttons for controlling the machine which is consisting of the Start Push Button indicated by the Green Button, the Stop Push Button indicated by the Red Button, and the Reset Button indicated by the Yellow Button

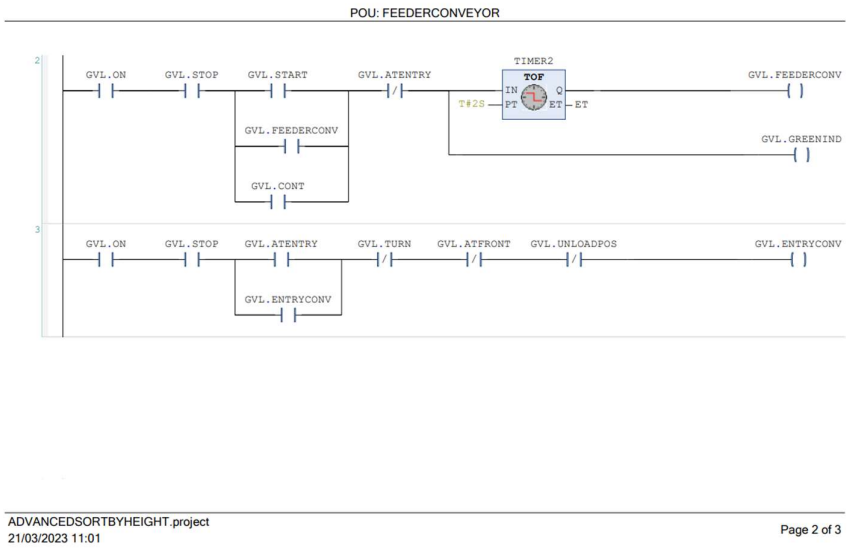


Fig 3.8 Feeder Conveyor (PT2)

The second rung is used when the start push button has been pressed for the first time after the machine has been turned on, and also when the part is finished being sorted. Is to start the feeder conveyor feeding with the new item. The mechanism is set to be stopped when a sensor labeled ATENTRY in the front of the Sorting Sensor has been triggered by the object.

The Third Rung is the mechanism to trigger the Entry Conveyor. To make the Conveyor turned on, some prerequisites must be fulfilled. In this case, the Turntable must be in the ready position. The object has been sorted by the Sensing Mechanism, and not in the Unloading Position.

The CONT label on the second rung is used to make the machine can continue on its own after the Part has been sorted and left the sorting station. Otherwise, without it being implemented in the programming, the machine would stuck after sorting just one item and require an Operator's manual assistance to continue sorting.

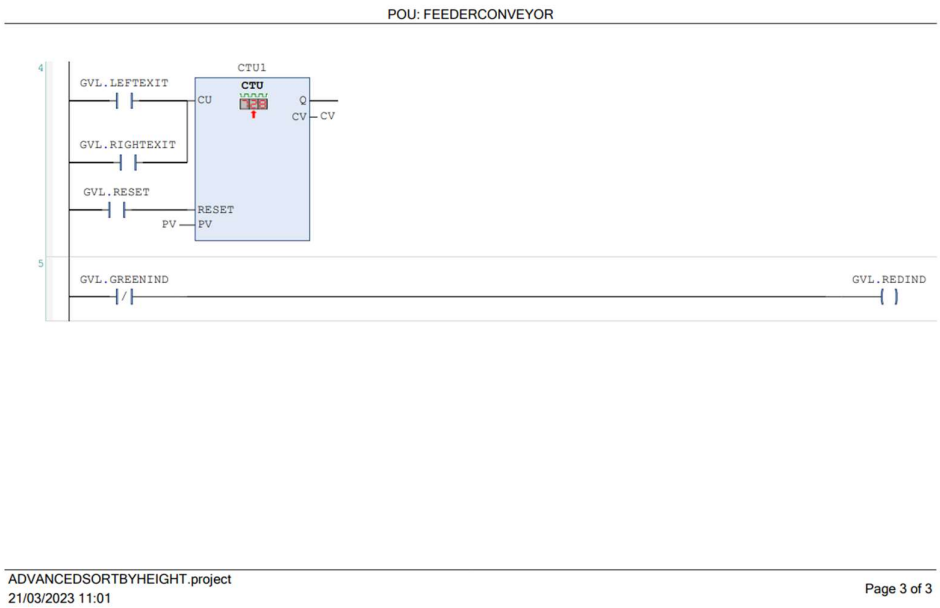


Fig 3.9 Feeder Conveyor (PT3)

The fourth rung is used for the counter that counts how many items that has been sorted through. And the last rung is used to indicate when the machine is busy sorting the machine by using the Light Indicator.

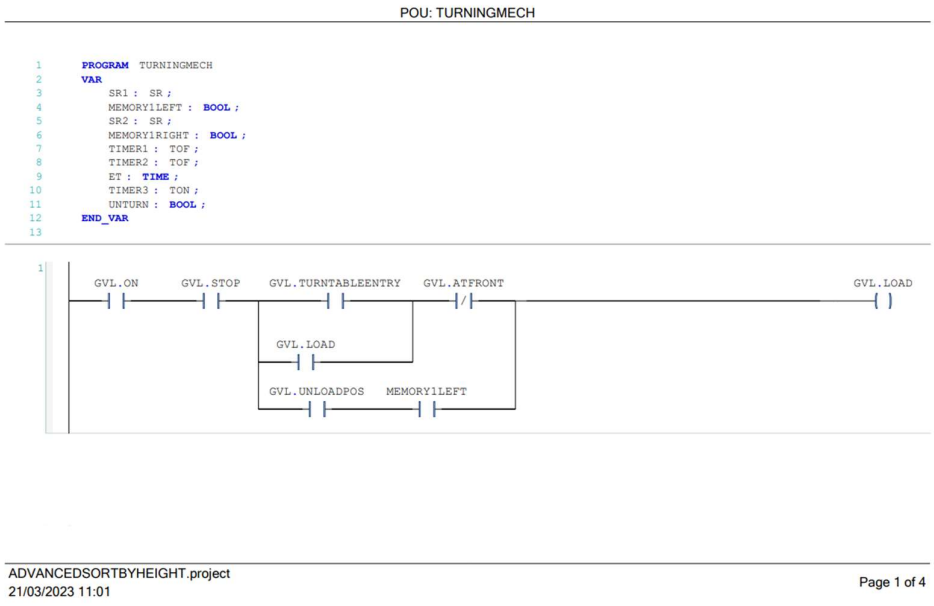


Fig 3.10 Turning Mechanism (PT1)

The First part of this program is to detect if the object is already in front of the Turntable. If the object had reached the end of the Entry Conveyor and the machine is already in the Loading position it will immediately load into the Turntable.

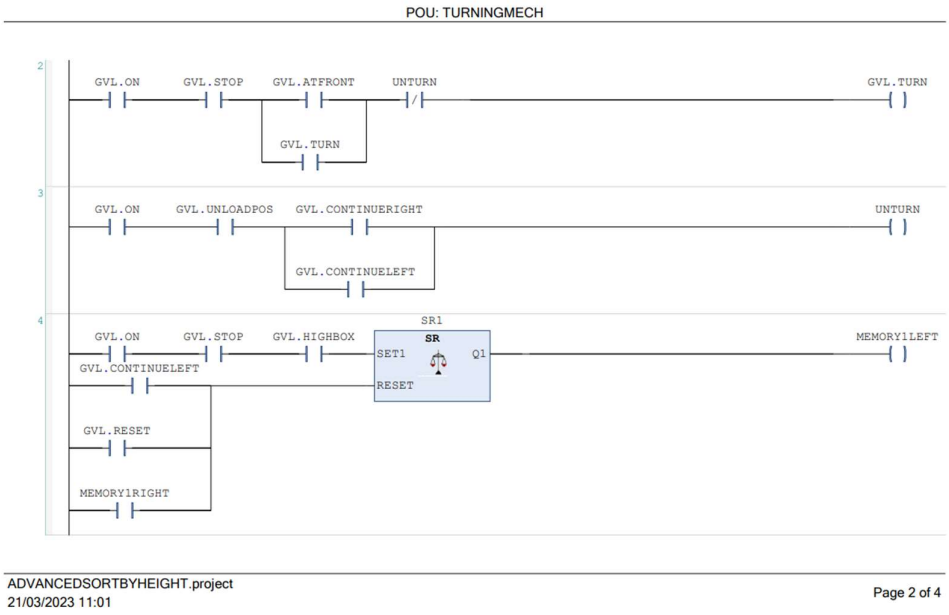


Fig 3.11 Turning Mechanism (PT2)

The second rung on this program is used when the object has been fully loaded into the turntable position and not in the turned position (yet), it will turn to the turned position. Meanwhile, the third rung is used to turn back the turntable to the unturned position after the object has passed the sensor to make the mechanism continue its job by feeding another object into the system.

Rung numbers four and five are both unique rungs. In this case, they both used the SR latch to make a memory about the height of an object detected by the Height Sensor. To make the system free of conflicting memory about deciding which object is correct. They are both set to cancel each other program when the other's prerequisite has been fulfilled first.

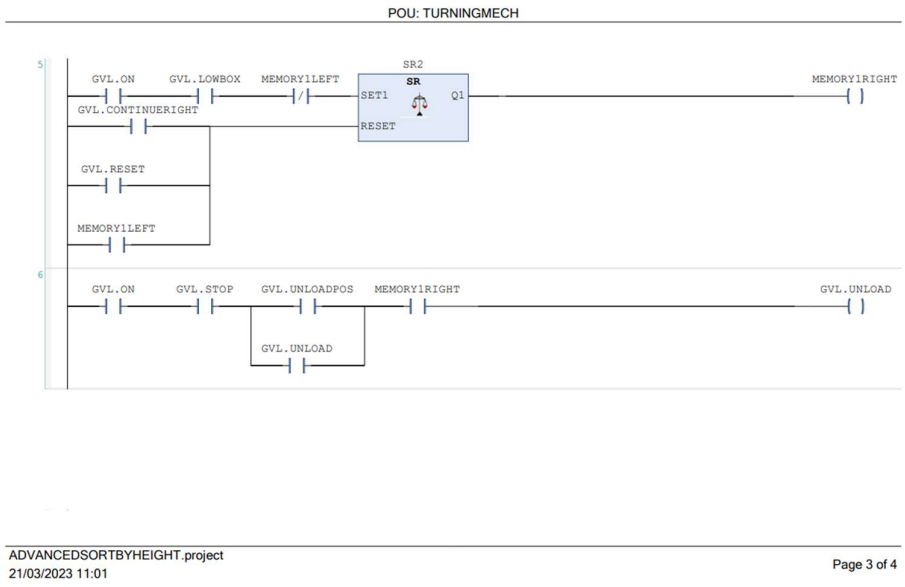


Fig 3.12 Turning Mechanism (PT3)

The sixth rung is used for Unloading the Lower Box if it's recorded in the Machine Memory. Meanwhile, the High box is attached to the first rung.

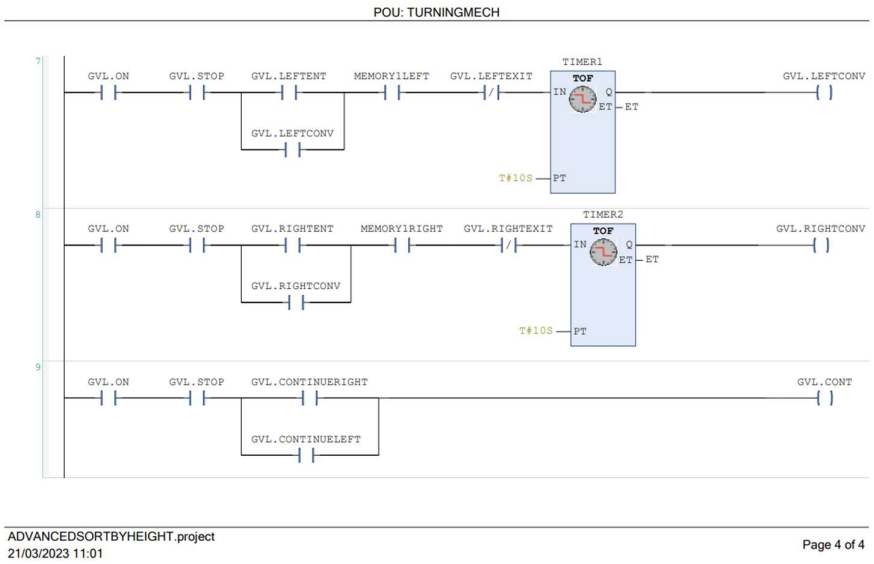


Fig 3.13 Turning Mechanism (PT4)

The seven and eighth rung is used to control the exit conveyor for each left and right direction respectably. It will run for 10 seconds and will continue if the next object sensed by the sensing mechanism is the same as the previous one. Otherwise, it will automatically turn off after the Object left the conveyor.

3.3.2 OpenPLC Based Program

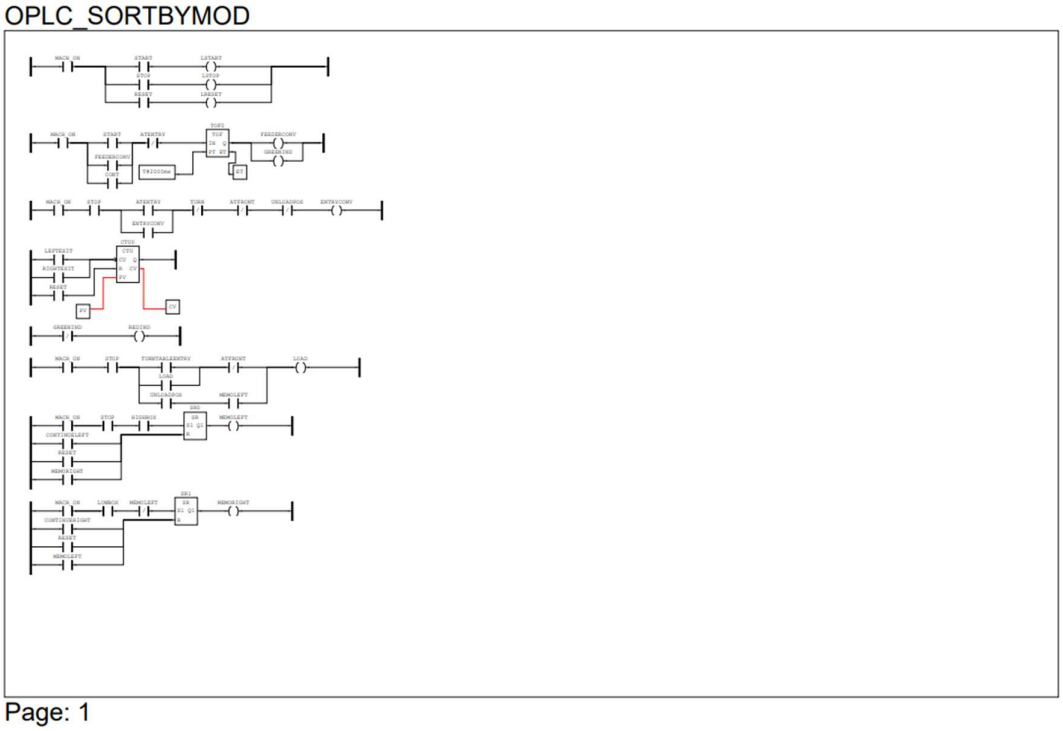


Fig 3.14 OpenPLC Program (Part 1)

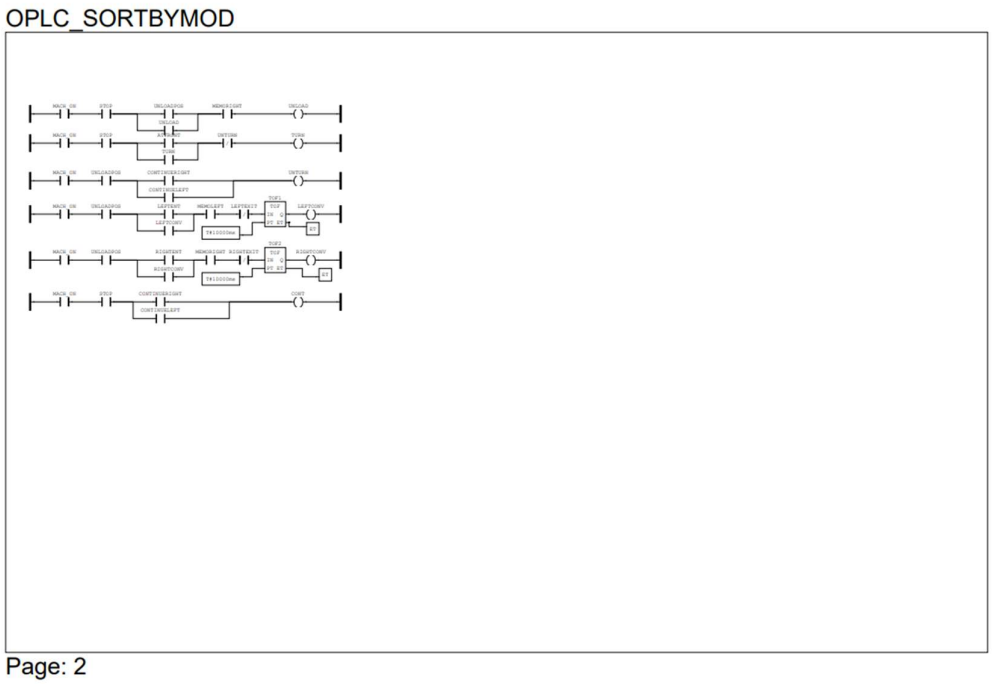


Fig 3.15 OpenPLC Program (Part 2)

When compared to the program that was made in Codesys, programs that are made in OpenPLC can be a little bit of a mess since it works as one file and with a local variable and in one program environment rather than a global variable list and multiple programs like the Codesys Counterpart. However, the function of the code will be the same since the basic Ladder Logic Program is identical for both IDE.

3.3 Establishing Modbus Communication with FactoryIO

3.3.1 Codesys Modbus

To begin the communication between Codesys and FactoryIO, first we need to add the Ethernet device to the Environment. By Right clicking on the current device which is the Control WinV3 x64 > Add Devices > Fieldbuses > Ethernet/IP > Ethernet Adapter > Ethernet.

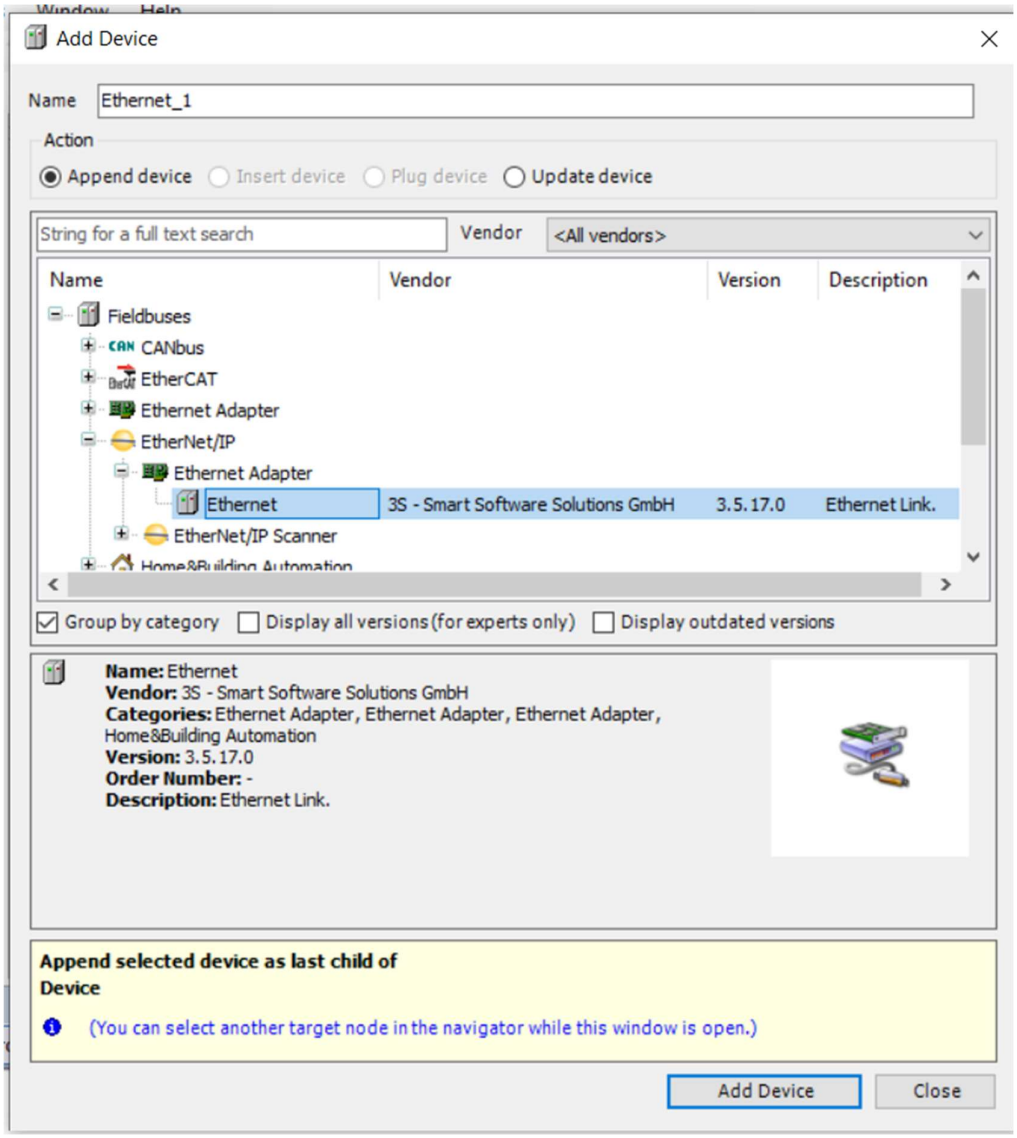


Fig 3.16 Adding Ethernet Device

After adding the Ethernet Device, select it and a new screen will pop up out of the window.

Select Modbus > ModbusTCP Slave Device > Modbus TCP Slave Device.

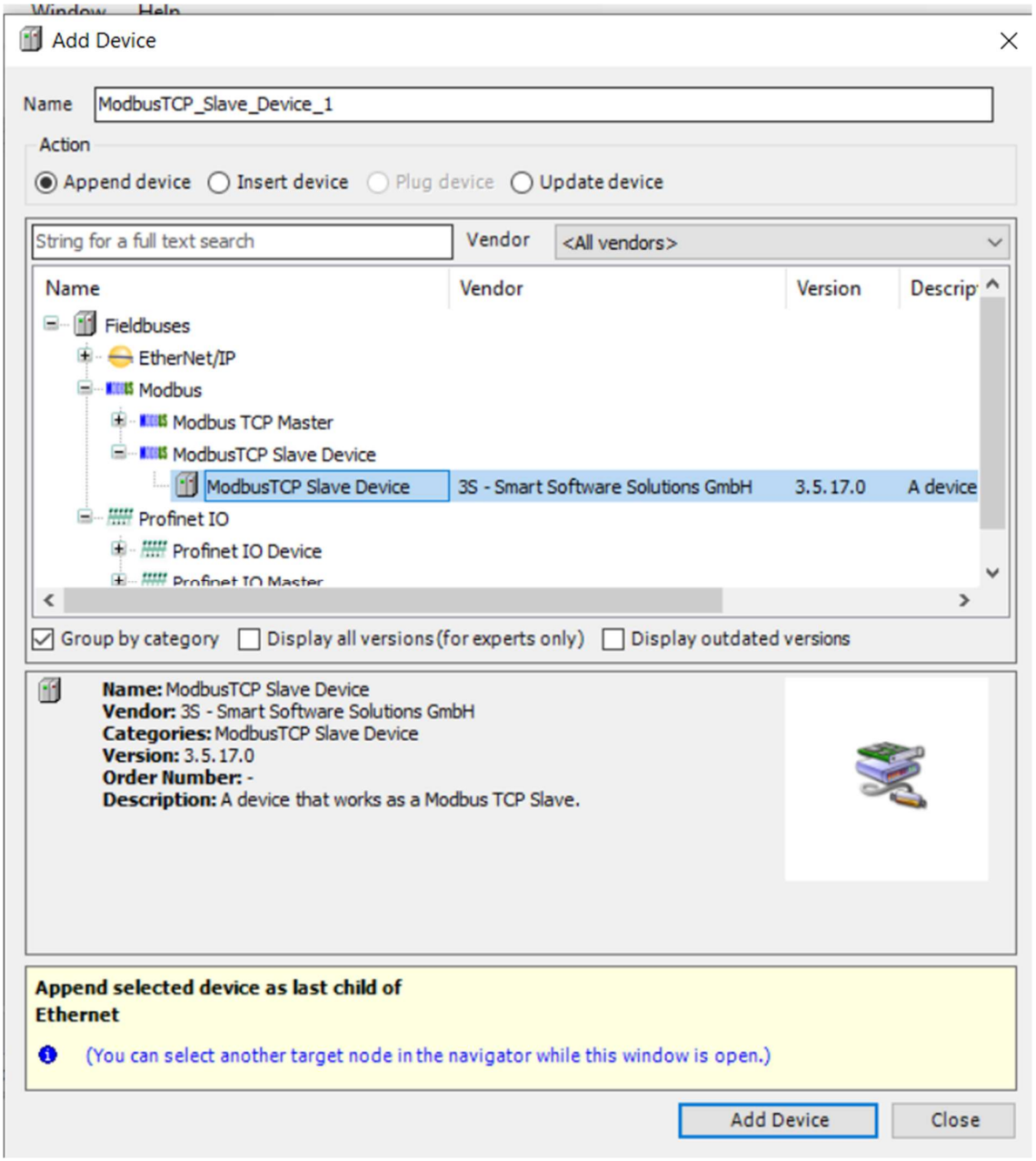
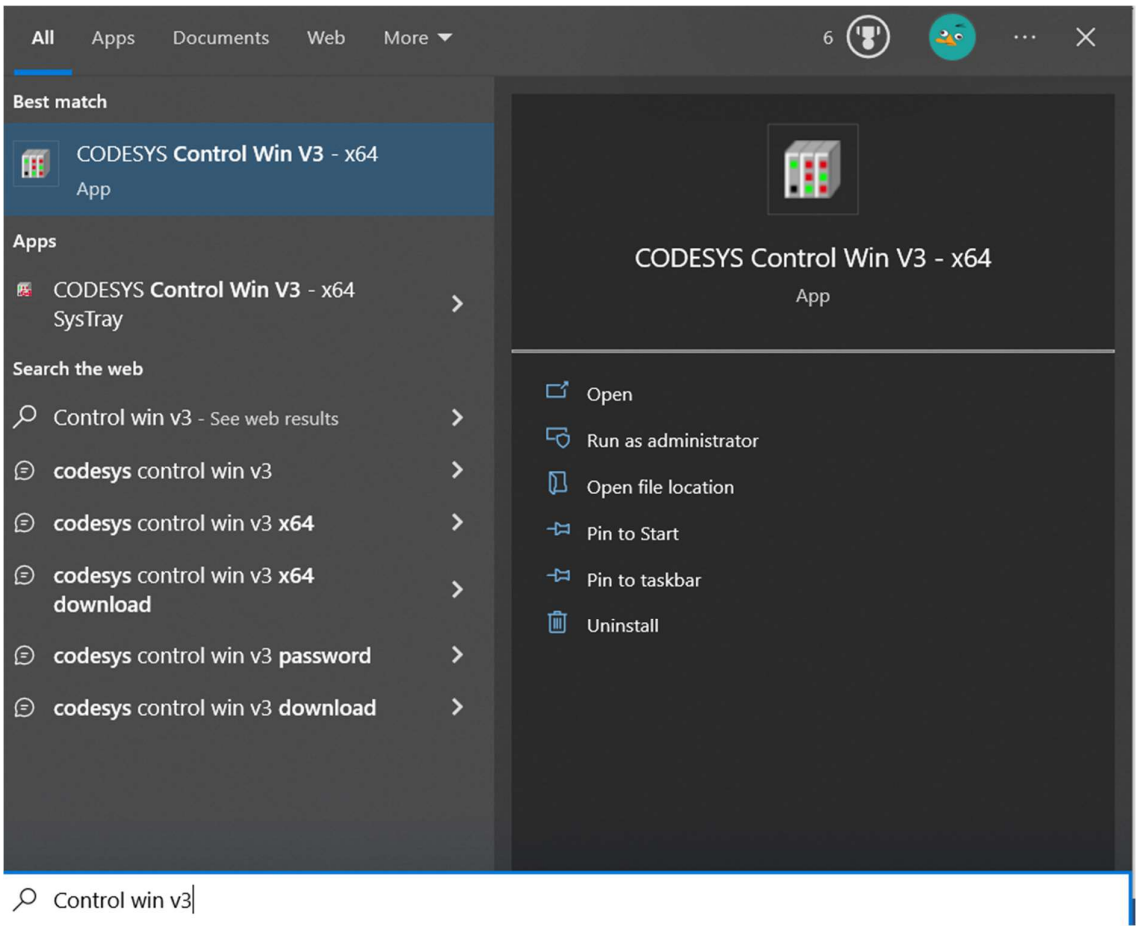


Fig 3.17 Adding Modbus TCP Slave Device

After the previous step is done, now it's time for configuring which network interface would be used for the PLC to be connected to. First, begin by configuring the Network Gateway to the PLC device. Start the device by typing Control Win V3 in the Windows Search bar or by using the small icon on the right side of the taskbar, right-click on it, and click the start PLC button.



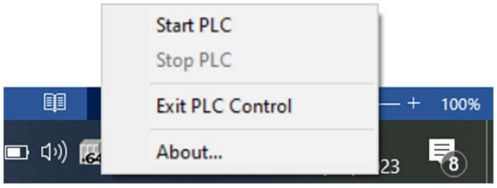


Fig 3.18 Start the Control Win V3 Virtual PLC.

After the device started, go back to the IDE then Select the Device > Device User Logon > Enter the Login Credentials. When it successfully logon, a green indicator should appear.

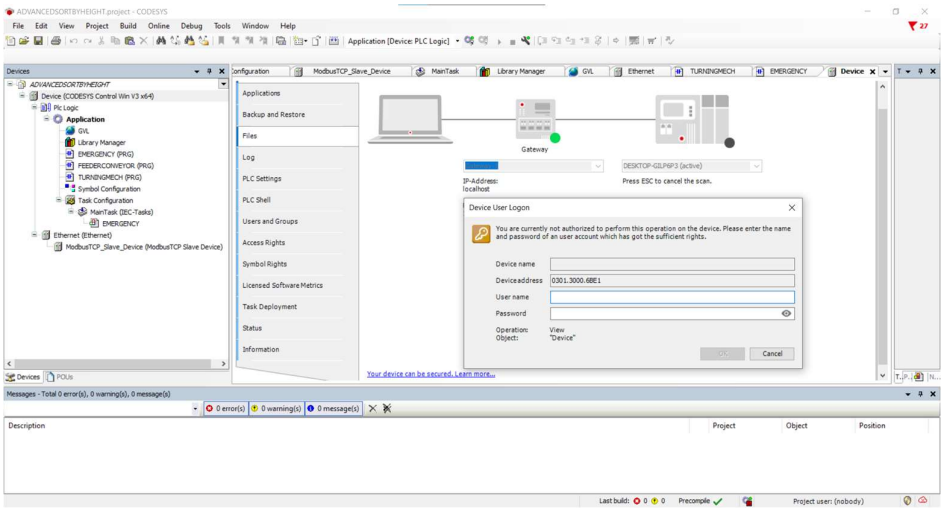


Fig 3.19 Device User Logon Interface

After that, click on the Ethernet > General > Network Interface > Browse. A new screen will pop up showing all the available networks to be used with the program. It can be varying from a Wi-Fi network, Ethernet connection, or a VPN if the device is located in a remote location

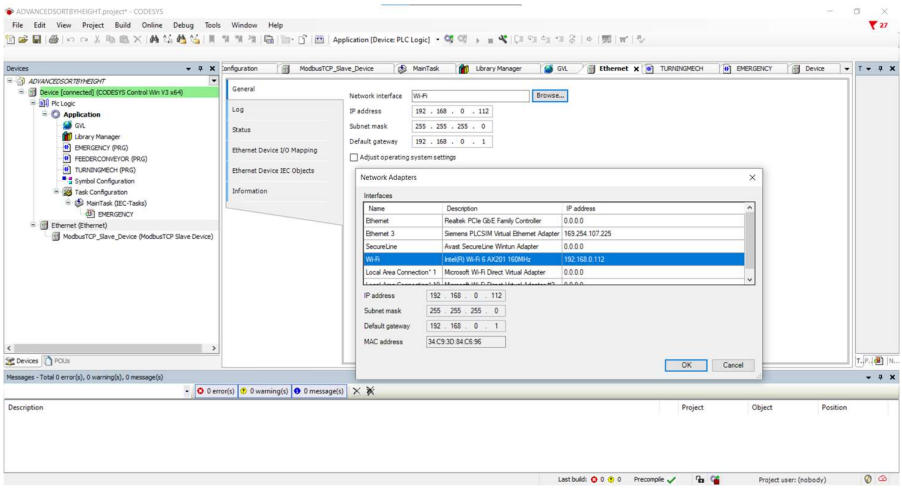


Fig 3.20 Network Selection

After all of the previous steps are done, select the Modbus Slave device > Modbus TCP Slave Device I/O Mapping. These steps are required to assign each ladder logic to each Input Output mapping to match the one that has been assigned in the FactoryIO Input Output assignment. The Input which is the FactoryIO sensor entered in the Coils section and the Output tags are entered in the Discrete Inputs section.

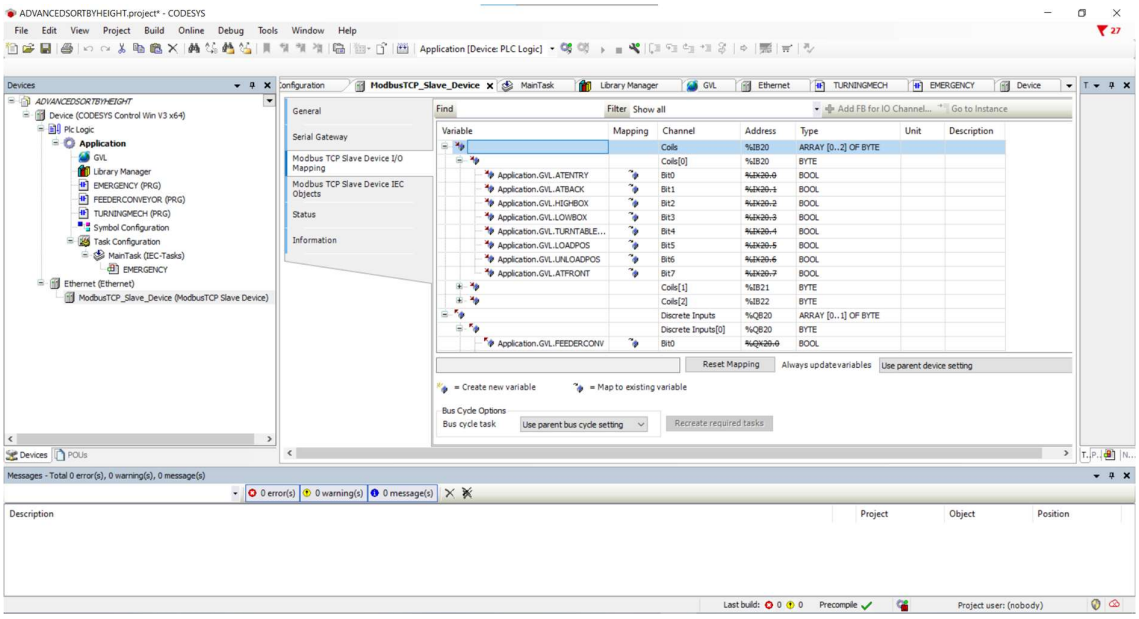


Fig 3.21 Input Output Mapping

And then to begin the Program go to the Online > Login to start the program. Alternatively, it can just use Alt + F8 hotkey and then click the Start button

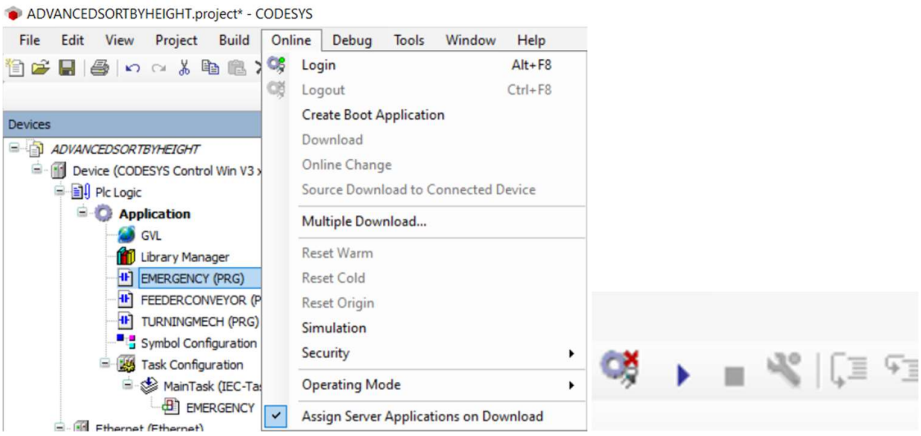


Fig 3.22 Login and Start Procedure

Then continue to the FactoryIO > Driver > Select the Modbus TCP/IP Client > Configuration and Input the Host Address.

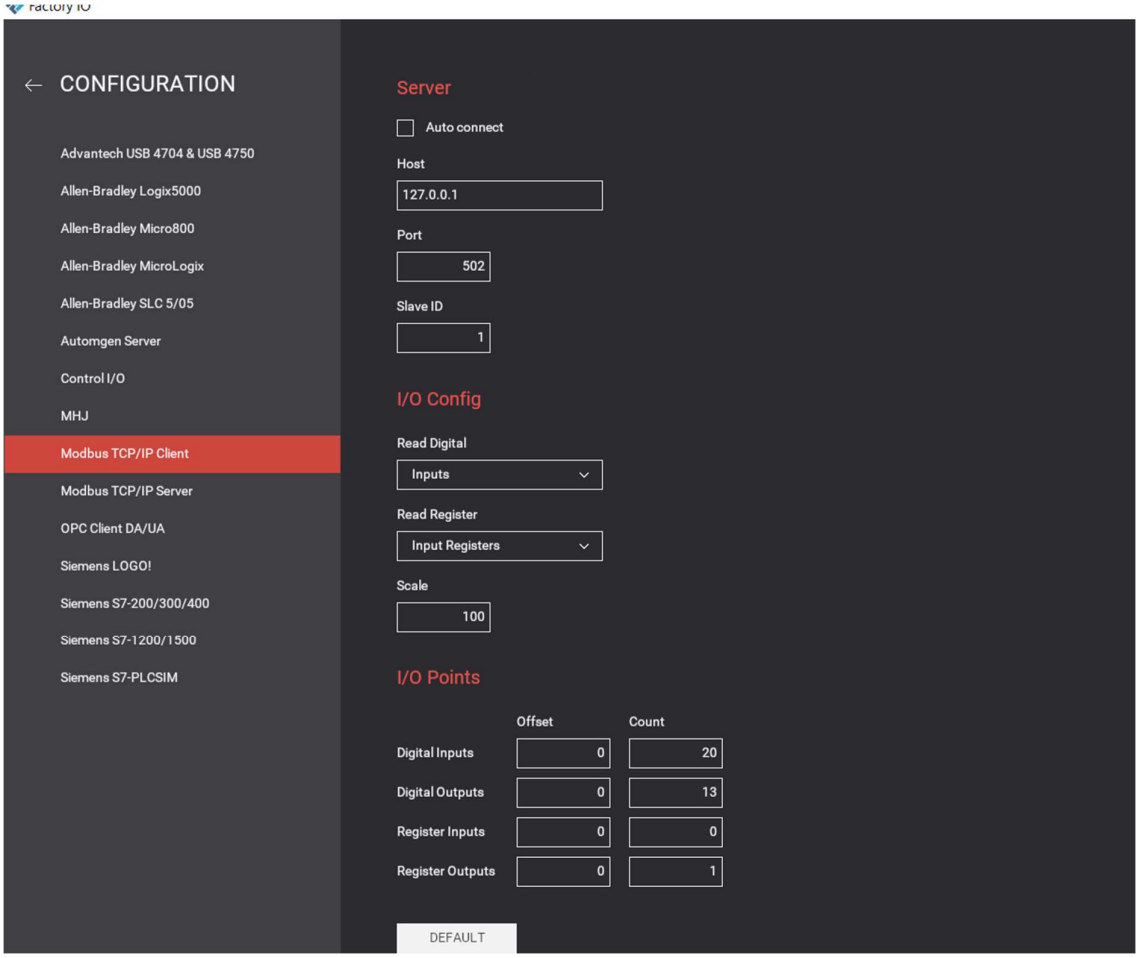


Fig 3.23 Modbus TCP/IP Configuration in FactoryIO

If everything is correctly set up, go back and click Connect. A green checkmark should appear right next to the Driver Selection

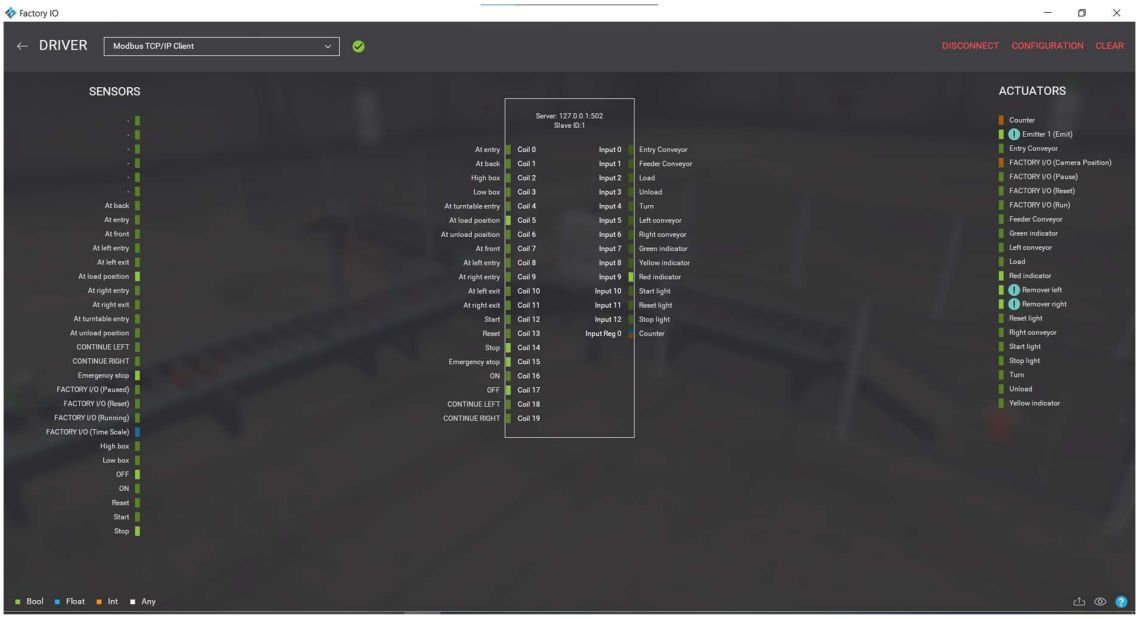


Fig 3.24 The Modbus Configuration has been set up for Codesys

3.3.2 OpenPLC Modbus

Different from setting up in Codesys. OpenPLC requires its user to first download a program in .st extension format and save it to a user-specified directory in the computer first and name it accordingly to the project.

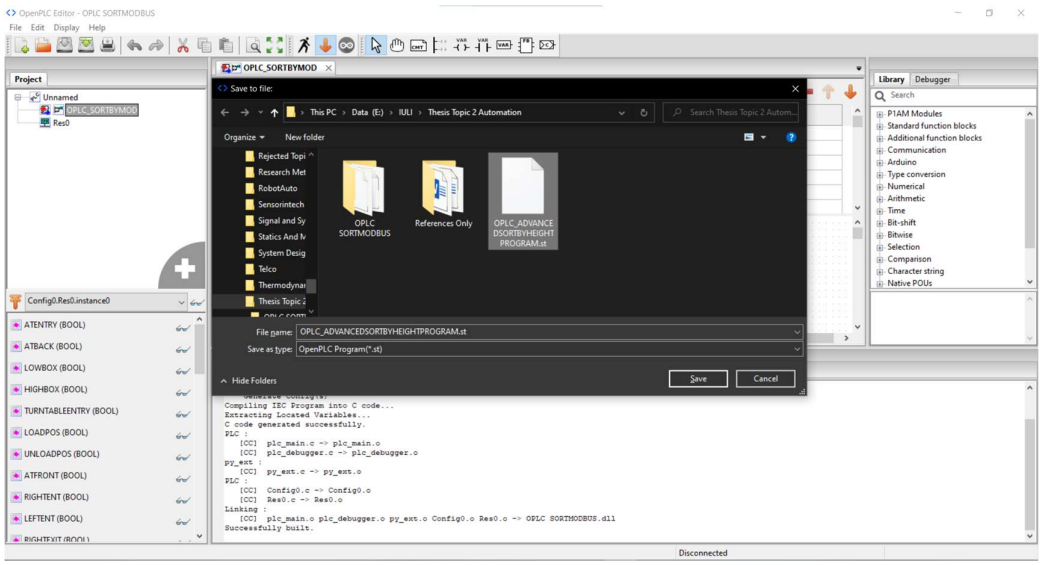


Fig 3.25 OpenPLC Save the file to the Directory

After the file has been successfully saved into the computer directory, now it's time to run a separate program called OpenPLC Runtime. This is a Web-server-based interface that will both act as a virtual PLC and the HMI for the Operator/User. To start it click on the Search Bar > OpenPLC Runtime. Or if it's running on a Raspberry Pi that already had an OpenPLC installed it can directly just enter the Address Running for OpenPLC

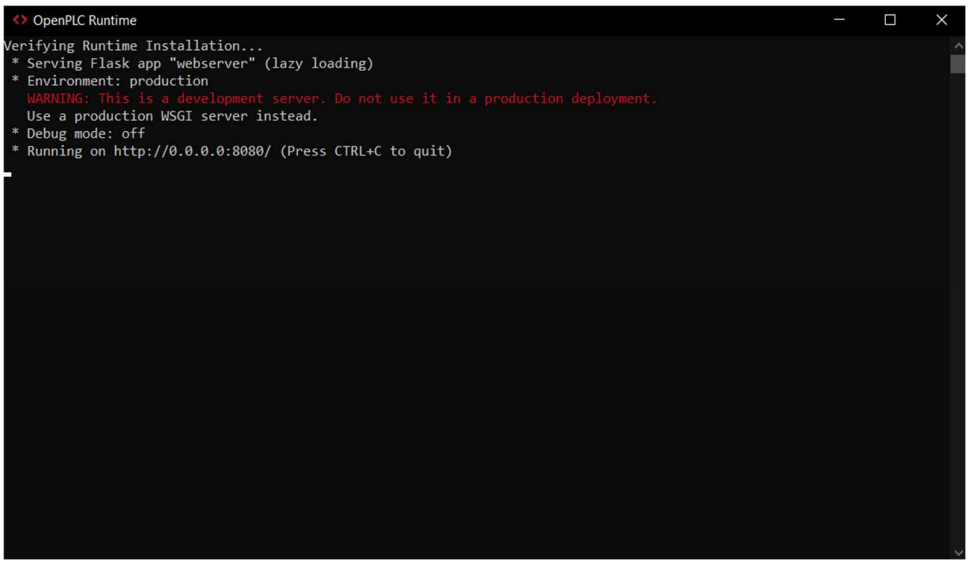


Fig 3.26 OpenPLC Runtime

Since the test was conducted by using a Windows Computer, Head to the Browser and type the 127.0.0.1:8080 address into the Browser. If typed correctly it should show this page.

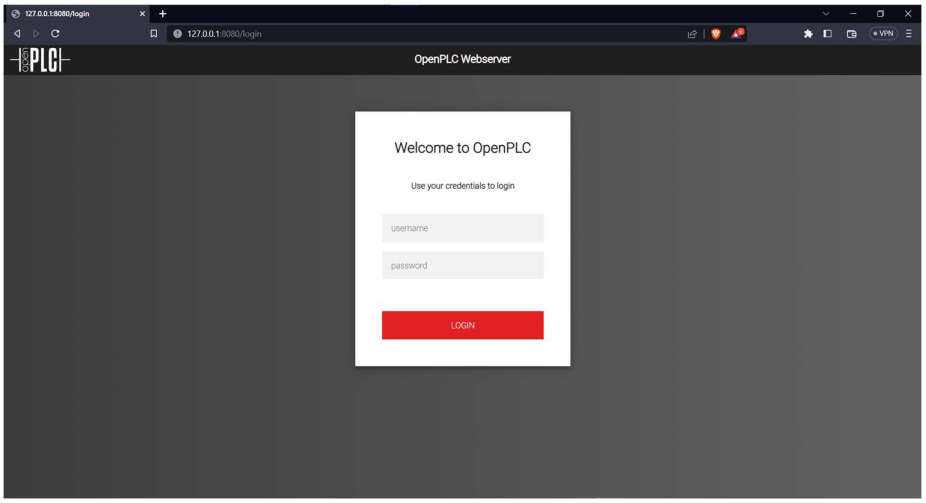


Fig 3.27 OpenPLC Web Interface

The default username and password by default for the first-time login should be:

Username: openplc

Password: openplc

After login, it should show a Dashboard menu alongside all the features including starting and stopping the virtual PLC

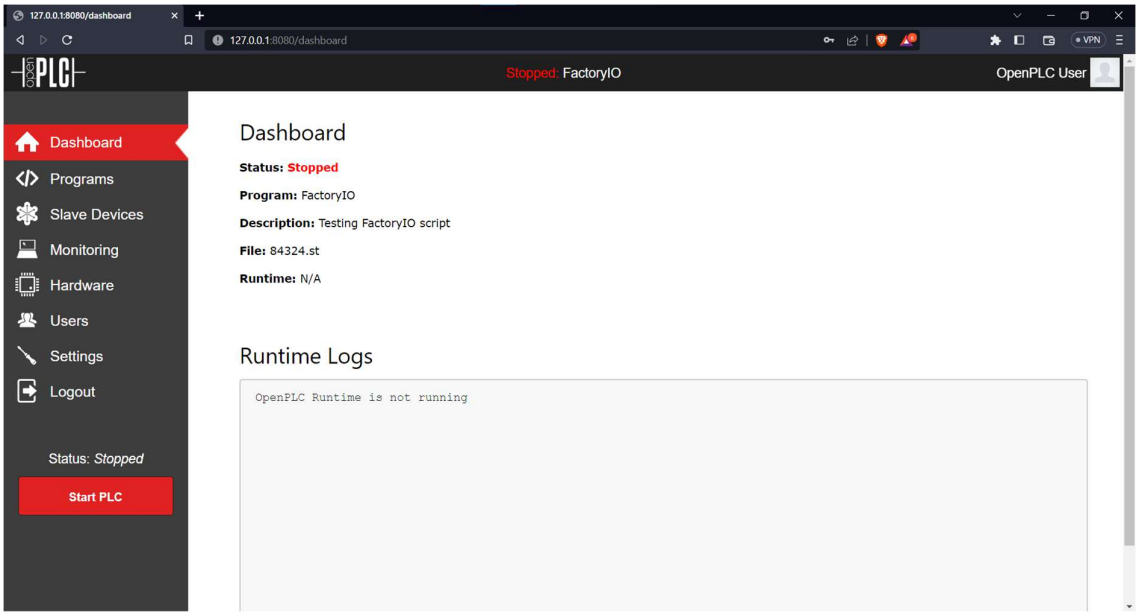


Fig 3.28 OpenPLC Dashboard Page

To upload the program that has been saved previously in the computer. Go to Programs > Upload Program > Choose File > Upload the File to the Cloud Server > Name the Program and add the program project description (optionally) then click on the Upload Program Button.

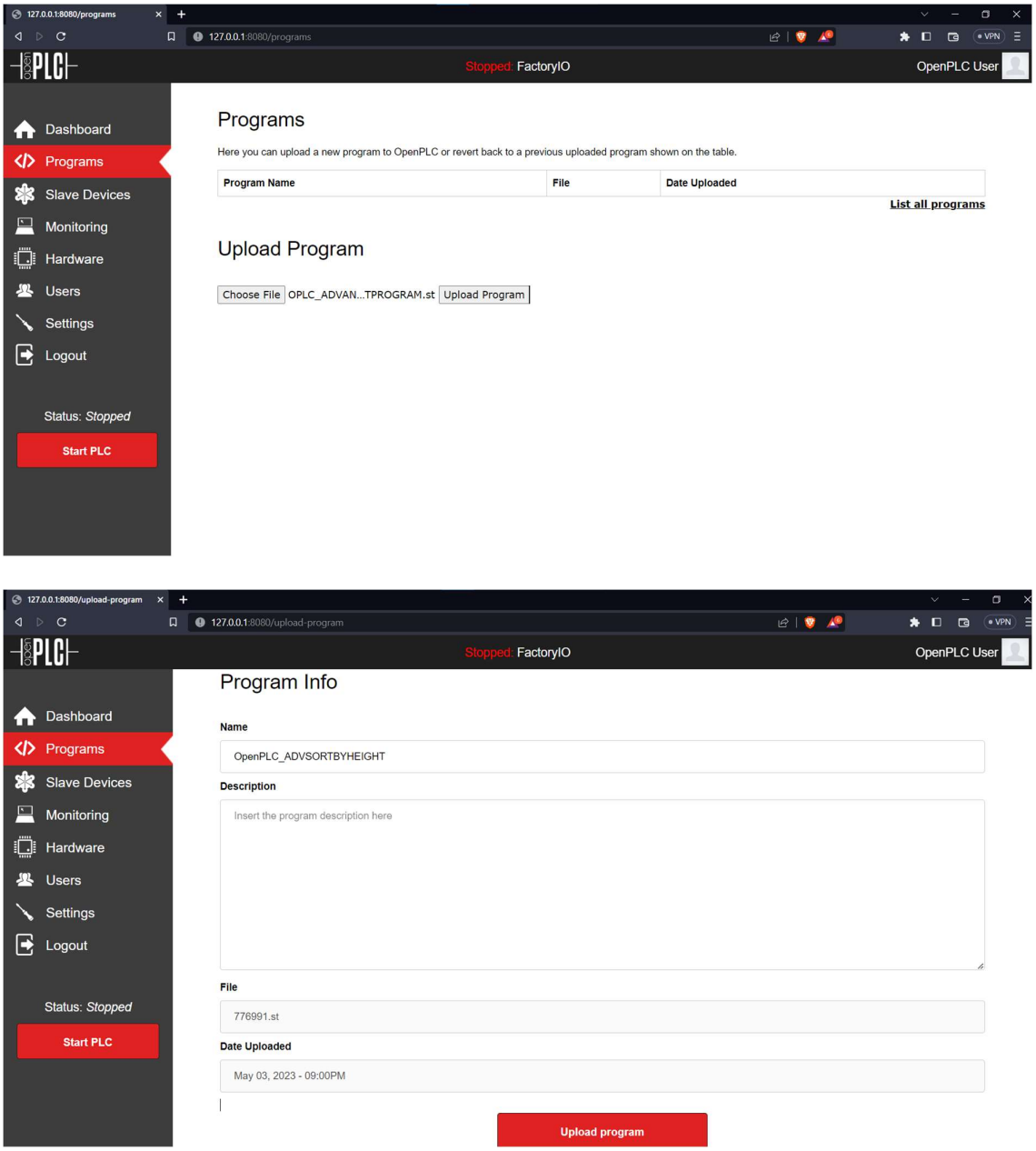


Fig 3.30 Name the Program and The Program Description

When the Program has been successfully uploaded into the Virtual PLC Server, it should show a screen that tells that the program has been compiled successfully.

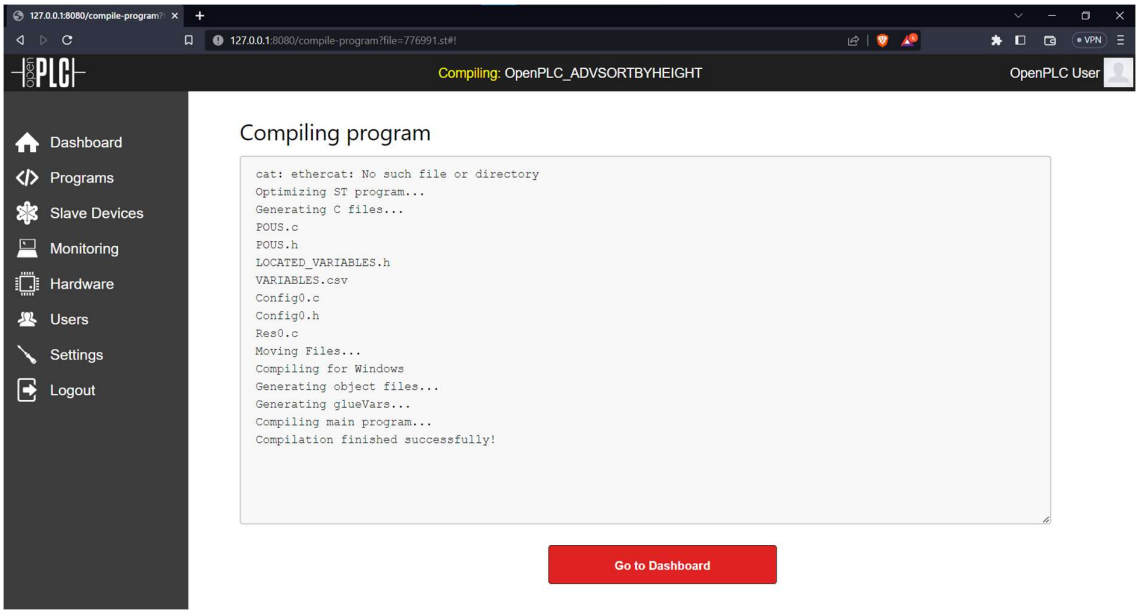


Fig 3.31 Program Compiled Successfully

Then head to FactoryIO. Click on Scenes > My Scene > Driver > Modbus TCP/IP Client > Configuration > Set it to 127.0.0.1 on IP Address and 8080 as the Port > Click on Connect than it should show a green mark right next to the driver selection.

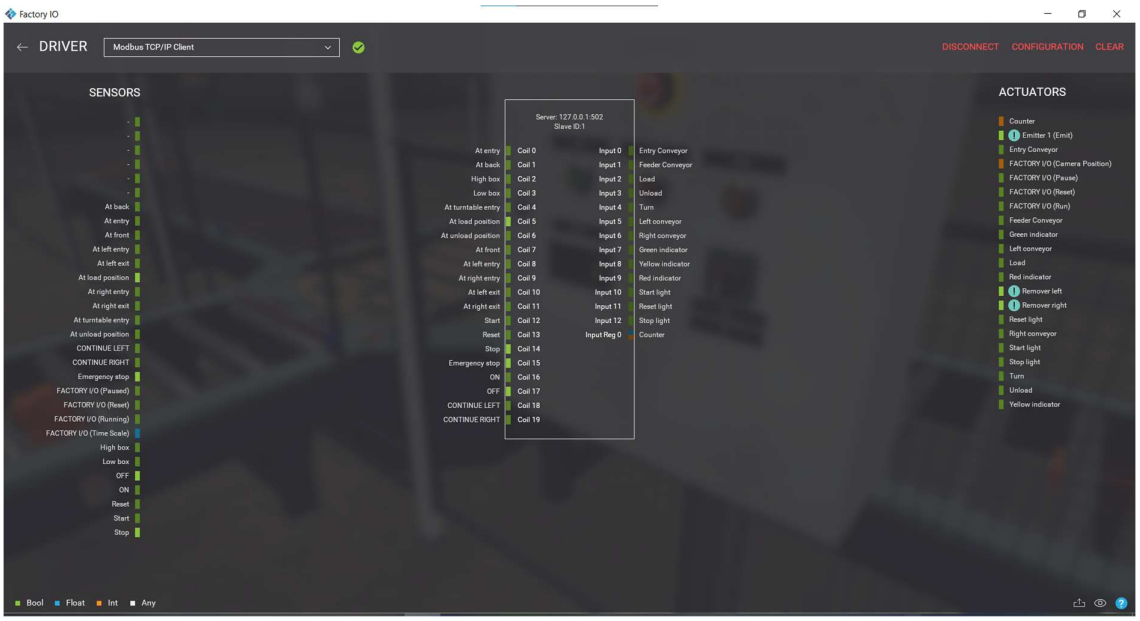


Fig 3.32 Connection established successfully

3.4 Block Diagram

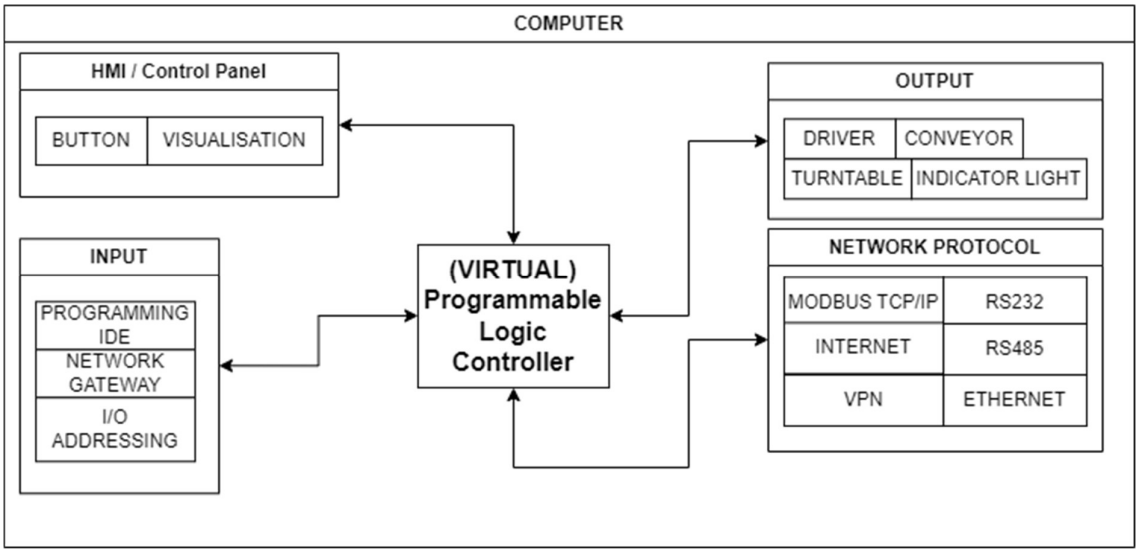


Fig 3.33 Block Diagram of the Program

The Block Diagram above is just a simplified representation of what's happening in the system during the Sortation Progress.

Chapter 4

General Overview and Results

4.1 Introduction to the Results

The results of the research that were presented in this thesis conducted by self.