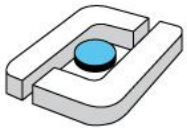


Hochschule Osnabrück
University of Applied Sciences

Fakultät Ingenieurwissenschaften und Informatik

Masterarbeit

über das Thema

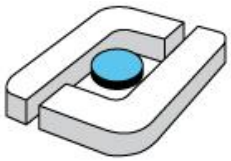
Motion Monitoring of Robots and PLC Controlled Assembly Systems using Process Simulate

vorgelegt durch

Sergio Sóñora Mariño

02 June 2015

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa



Hochschule Osnabrück
University of Applied Sciences

Fakultät Ingenieurwissenschaften und Informatik

Mechatronic Systems Engineering

Masterarbeit

Motion Monitoring of Robots and PLC Controlled Assembly Systems using Process Simulate

Erstprüfer: Herr Prof. Dr. Dirk Rokossa
Zweitprüfer: Herr Dipl.-Ing. Martin Nardmann

Bearbeiter: Sergio Sóñora Mariño
Matrikelnummer: 651395

Geboren am 20.01.1988
In: Boiro (Spanien)

Ausgabedatum: 02.02.2015
Abgabedatum: 02.06.2015

_____
Erstprüfer

_____
Master

_____
Zweitprüfer

Eingereicht am:

Verlängerung genehmigt bis:

Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.
Ferner stimme ich zu, dass eine Software-Kopie meiner Arbeit für einen Plagiat-Test benutzt wird.

Ort, Datum:

_____

_____

(Unterschrift) (Unterschrift)

Abstract

This Master Thesis, carry out at the University of Applied Sciences of Osnabrück, is framed in the Industrial Robotics area, The main objective consist of real-time motion data monitoring of different robots and conveyor systems of an industrial process.

The monitoring application is programmed with the help of the API (Application Programming Interface) Tecnomatix, Process Simulate software from Siemens. This API provides classes and methods needed to program a dynamic library that can represent, in the 3D model Process Simulate, the motion data of the industrial process.

For that, the software PLC (Programmable Logic Controller) is programmed on the server side, which is responsible for capture the different data and transfer through the network, using the TCP/IP protocol. After that, each client can run the monitoring application and connect to the server, visualizing movements in the 3D model.

Thereby, it's possible to monitoring and control an industrial process in real-time, thousands of kilometers away, provided they have a connection to the Internet. Offering the chance to detect faults immediately and track along the whole process.

Resumen

El presente Proyecto Fin de Carrera (PFC), realizado en la Universidad de Ciencias Aplicadas de Osnabrück, está enmarcado en el área de Robótica Industrial, el objetivo principal consiste en poder monitorizar en tiempo real los datos de movimiento de los diferentes robots y sistemas de transporte de un proceso industrial.

La aplicación de monitorización está programada con la ayuda de la API (Interfaz de Programación de Aplicaciones) Tecnomatix, del software Process Simulate de Siemens. Esta API provee las clases y métodos necesarios para programar una librería dinámica que pueda representar, en el modelo 3D de Process Simulate, los datos de movimiento del proceso industrial.

Para ello, se programa el software PLC (Controlador Lógico Programable) en el lado del servidor, que se encarga de capturar los diferentes datos y transferirlos a través de la red mediante el protocolo TCP/IP. Posteriormente, cada cliente puede ejecutar la aplicación de monitorización y conectarse al servidor, visualizando los movimientos en el modelo 3D.

De este modo, es posible monitorizar y controlar un proceso industrial en tiempo real a miles de kilómetros, siempre y cuando se disponga de una conexión a internet. Ofreciendo la posibilidad de detectar posibles fallos de forma inmediata y llevar un seguimiento a lo largo de todo el proceso.

Zusammenfassung

Diese Masterarbeit wurde an der Hochschule Osnabrück angefertigt. Das Hauptziel dieser Arbeit besteht darin, die Bewegungsdaten von verschiedenen Robotern sowie eines Transportsystems in industriellen Prozessen in Echtzeit zu überwachen.

Die Überwachungsapplikation wurde mithilfe einer Programmierschnittstelle (Application Programming Interface), Tecnomatix und Process Simulate Software von Siemens programmiert.

Diese Programmierschnittstelle (API) gewährt alle erforderlichen Klassen und Methoden für die Programmierung einer dynamischen Bibliothek, in der alle gemessenen Bewegungsdaten des industriellen Prozesses in 3D dargestellt werden.

Die Software PLC (Programmable Logic Controller) wird auf der Server-Seite, die für die verschiedenartige Datenerfassung zuständig ist, programmiert. Anschließend übermittelt sie über das Netzwerk mit der TCP/IP-Protokoll. Deshalb kann jeder Kunde später die Überwachungsapplikation verwenden. Erforderlich dafür ist die Anmeldung am Server, auf dem die Bewegungsdaten im 3D-Modell dargestellt werden.

Somit kann die Überwachung der industriellen Prozesse in Echtzeit auch bei einer Entfernung von tausenden Kilometern gesteuert werden, d.h. dass eine Internetverbindung dafür ausreichend ist. Es bietet sich gleichzeitig die Möglichkeit Fehler sofort zu erkennen sowie über den gesamten Prozess zu verfolgen.

Table of Contents

List of Figures

List of Tables

1. Introduction

After factory automation (XX century) is starting a new phase that could be called *Fourth Industrial Revolution* or *Industry 4.0*, is characterized by the inter-connection of machines and systems on the production site itself, and also, a fluid exchange of information to the outside via the Internet.

This thesis focuses particularly on the exchange of information on production systems to the outside. The application allows monitoring the status and position in real time of the machines involved in the manufacturing process.

Additionally, process simulation software of Siemens which are added new functionality is used, thereby taking advantage of existing laboratory resources.

1.1. Objectives

Following the topic of the introduction, the purpose of this project is:

- Study the technical bases of industrial robots and conveyors systems.
- Understand the structure of Process Simulate and how to retrieve and manipulate information regarding locations for resources using the Tecnomatix .NET API.
- Create a plug-in with a GUI and a TCP connection to monitoring the motion data of the whole system.

This raises the level of abstraction since the simulation software is a tool already used in industry. To achieve this objective, a virtual model in Process Simulate is used for motion data monitoring.

1.2. Outline

The rest of the thesis is structured as follows: Chapter 2 discusses the assembly system installed into the lab and all technical bases. Specifications of robots and conveyor system, PLCs and OpenMHS software, Process Simulate

and Tecnomatix are introduced. Chapter 3 reviews the possible solutions for monitoring the assembly system. Advantages and disadvantages of each possible solution are discussed, and why the selected solution is chosen.

Chapter 4 presents the implementation of a model of the conveyor system in Soft-PLC OpenMHS in the server-side, and the development of plug-in for Process Simulate in the client side. It explains the different alternatives to model the conveyor system, also the concepts of client-server communication and TCP protocol. Finally, the operation of monitoring application and its GUI is explained.

Chapter 5 shows the results in different test scenarios. Several monitoring tests are performed and the results of each are discussed. The final chapter presents a short summary along with the conclusions. It also provides recommendations for future enhancements of this research.

2. The Assembly System

The Assembly System consists of five robots (2 Stäubli, 1 Kawasaki, 1 KUKA and 1 Fanuc) and a conveyor system, all controlled by a PLC Software (OpenMHS), created by own University. In this chapter will explain each of parts that form, hardware and software, as well as software used to simulate assembly processes (Siemens Process Simulate).

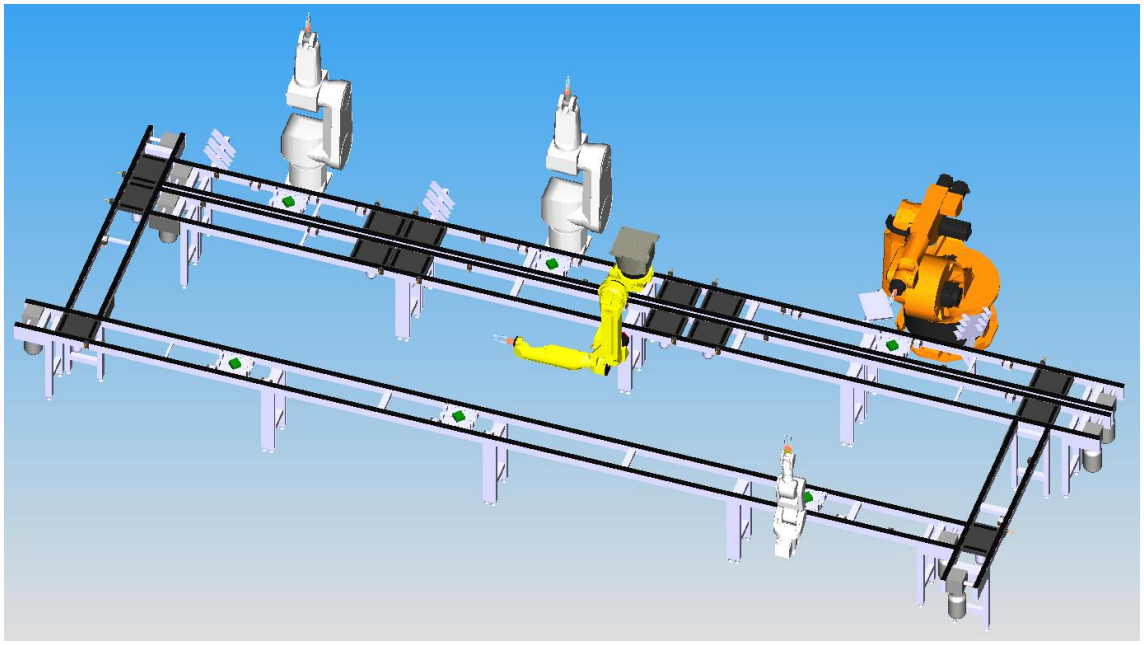


Figure 2.1: Layout of the Assembly System

2.1. Robots

In the following subchapters the robots installed into the Assembly System of Robotics Lab at the University Of Applied Sciences Of Osnabrück will be described. Later, these robots will be monitored with the application developed in this thesis.

All robots have their own controller. These controllers are connected to a central computer where OpenMHS Soft-PLC resides. The central computer works as a client of the robot controllers, and at the same time, as a server for the monitoring application.

2.1.1. Stäubli RX 130

Stäubli is an international mechatronics company, primarily known for its textile machinery, connectors and robotics products. Stäubli Robotics is Stäubli's automation and robotics related division, it produces SCARA and 6-axis robots for industrial automation, including controllers and software.

Figure 2.2: Stäubli RX 130

The assembly system has two robots Stäubli RX130. These robots series feature unique benefits to fit in all environments providing the best possible process quality and increased productivity. In the table 2.1 you can see the specifications of Stäubli RX 130 robot.

Stäubli RX 130 Specifications			
Type	Articulated		
Degrees of Freedom	6 axes		
Maximum load	10 kg		
Maximum reach	2185 mm		
Repeatability	±0.05 mm		
Maximum speed at load gravity center	18.9 m/s		
Work Envelope (degrees) & Maximum Speed (degrees/s)	**Axis**	**Motion Range**	**Maximum Speed**
	JT1	±160°	278°/s
	JT2	±137.5°	278°/s
	JT3	±150°	356°/s
	JT4	±270°	409°/s
	JT5	+120°/-105°	480°/s

	JT6	±270°	1125°/s
Wrist Load Capacity	**Axis**	**Maximum inertias**	
	JT5	1.2 kg · m²	
	JT6	0.3 kg · m²	
Brakes	All axes		
Weight	245 kg		
Installation	Floor/Ceiling		

Table 2.1: Stäubli RX130 specifications

The RX130 robots use the Adept V+ programming language. To monitoring the motion data is necessary to create first a small V+ program that reads the current position of each robot and sends it over the network to the central computer.

2.1.2. Kawasaki RS05L

Kawasaki Heavy Industries, Limited (KHI) is an international corporation based in Japan. Kawasaki develops and builds a vast array of industrial plants and equipment, including industrial robots.

This robot is included inside the R-Series. The new R-Series Robots are setting the benchmark for all small to medium duty industrial robots. The compact design, along with industry leading speed, reach and work range make the R-Series Robots ideal for a wide range of applications throughout a multitude of diverse industries.

Figure 2.3: Kawasaki RS05L

The new lightweight arm along with high-output high-revolution motors provide industry leading acceleration and high-speed operation. The acceleration rate automatically adjusts to suit the payload and robot posture to deliver optimum performance and the shortest cycle times. The slim arm design requires very little floor space. Multiple robots can be installed in *high-density* applications without impeding performance. Specifications could be found in table 2.2.

Kawasaki RS05L Specifications			
Type	Articulated		
Degrees of Freedom	6 axes		
Payload	5 kg		
Horizontal Reach	903 mm		
Vertical Reach	1.484 mm		
Repeatability	±0.03 mm		
Maximum Speed	9.300 mm/s		
Work Envelope (degrees) & Maximum Speed (degrees/s)	**Axis**	**Motion Range**	**Maximum Speed**
	JT1	±180°	300°/s
	JT2	±135° / -80°	300°/s
	JT3	±118° / -172°	300°/s
	JT4	±360°	460°/s
	JT5	±145°	460°/s
	JT6	±360°	740°/s
Wrist Load Capacity	**Axis**	**Maximum Torque**	**Moment of Inertia**
	JT4	12.3 Nm	0.4 kg · m^2
	JT5	12.3 Nm	0.4 kg · m^2
	JT6	7.0 Nm	0.12 kg · m^2
Motors	Brushless AC Servomotor		
Brakes	All axes		
Mass	37 kg (excluding options)		
Installation	Floor, wall, ceiling		
Built-in Harness	Sensor harness 12 inputs, 24DC, GND		
Built-in Utilities	Pneumatic piping (Ø6 x 2 lines)		

Table 2.2: Kawasaki RS05L specifications

2.1.3. KUKA KR 30

KUKA is a German manufacturer of industrial robots and solutions for factory automation. The company name, KUKA, is an acronym for ***K****eller* ***u****nd* ***K****nappich* ***A****ugsburg*. Today, KUKA concentrates on solutions for the automation of industrial manufacturing processes. Most robots are finished in "KUKA Orange" (the official corporate color) or black.

Figure 2.4: KUKA KR 30

The optimally adapted motor/gear unit of the KR 30 gives high performance in terms of cycle time and accuracy. The process forces generated are perfectly compensated for by the high stiffness of the FEM-optimized design. The small footprint enables problem-free implementation even in confined cell layouts. In the table 2.3 you can see the specifications of KUKA KR 30 robot.

KUKA KR 30 Specifications			
Type	Articulated		
Degrees of Freedom	6 axes		
Rated payload	30 kg		
Maximum Reach	2033 mm		
Repeatability	±0.15 mm		
Work Envelope	**Axis**	**Motion Range**	**Speed (rated payload)**
(degrees) &	JT1	±185°	140°/s

Maximum Speed (degrees/s)	JT2	+35° / -135°	126°/s
	JT3	+158° / -120°	140°/s
	JT4	±350°	260°/s
	JT5	±119°	245°/s
	JT6	±350°	322°/s
Weight	665 kg (excluding controller)		
Installation	Floor, ceiling		
Robot footprint	850 mm x 950 mm		
Connection	7.3 kVA		
Noise level	< 75 dB		

Table 2.3: KUKA KR 30 specifications

2.1.4. FANUC M-20iA

FANUC is a group of companies that provide automation products and services such as robotics and computer numerical control systems. FANUC is one of the largest makers of industrial robots in the world, had its beginnings as part of Fujitsu developing early numerical control and servo systems. The company name is an acronym for *Factory Automation Numerical Control.*

Figure 2.5: FANUC M-20iA

This robot is mounted on a beam providing a further degree of freedom. The evolutionary design solves two of the oldest problems for industrial robots. The first issue is the time it takes to engineer the robot dressout and the second is the down-time caused when the dressout is disrupted by contact with plant equipment. The innovative M-20iA solves these issues with a hollow upper arm and wrist. This unique design permits the utilities to be neatly contained within the arm, eliminating dress out issues like snagging, tearing and rubbing. The hollow arm and wrist together with the shelf greatly simplifies the integration engineer's job. In the table 2.4 you can see the specifications of FANUC M-20iA robot.

FANUC M-20iA Specifications			
Type	Articulated		
Degrees of Freedom	6 axes		
Payload	20 kg		
Reach	1811 mm		
Repeatability	±0.08 mm		
Work Envelope (degrees) & Maximum Speed (degrees/s)	**Axis**	**Motion Range**	**Motion Speed**
	JT1	340°/370°	195°/s
	JT2	260	175°/s
	JT3	458°	180°/s
	JT4	400°	360°/s
	JT5	360°	360°/s
	JT6	900°	550°/s
Wrist Load Capacity	**Axis**	**Maximum Torque**	**Moment of Inertia**
	JT4	44 Nm	1.04 kg · m^2
	JT5	44 Nm	1.04 kg · m^2
	JT6	22 Nm	0.28 kg · m^2
Mechanical brakes	All axes		
Mechanical weight	250 kg		
Installation	Floor, wall, ceiling and angle		

Table 2.4: FANUC M-20iA specifications

2.2. Conveyor System

The Assembly System of Robotics Lab at the University of Applied Sciences of Osnabrück has a conveyor system from Bosch Rexroth, specifically the model TS2. It is a modular system formed by a conveyor belt, lift-transfer units, stoppers, inductive sensors and work-piece pallets on which the parts are placed.

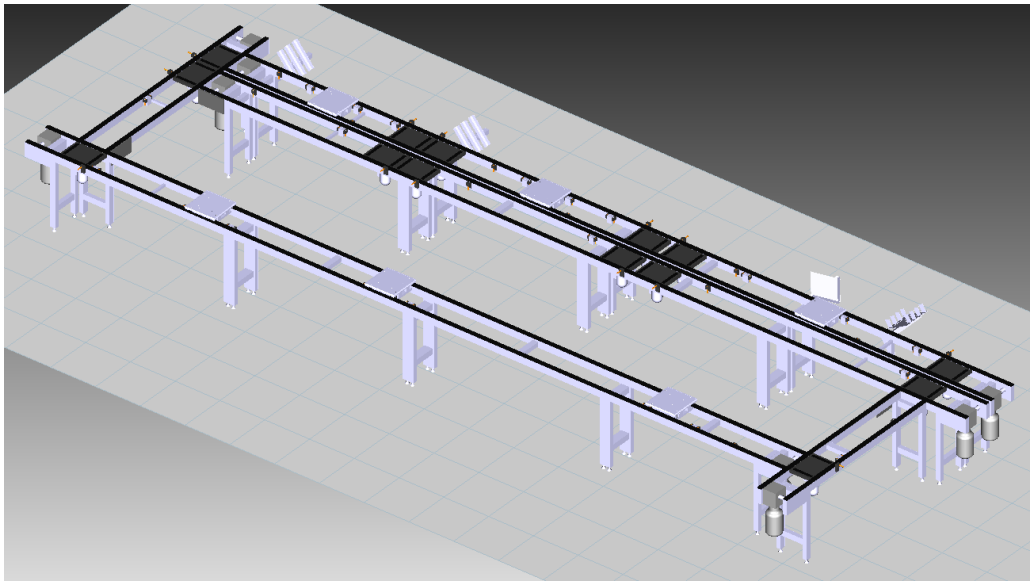


Figure 2.6: Conveyor system

The TS2 assembly conveyor is part of the larger family of TS (Transfer Systems) which include TS1 and TS4plus. TS2 is a work-piece pallet based, non-synchronous, conveyor designed to improve manufacturing productivity and product quality while allowing for maximum assembly flexibility.

2.2.1. Work-piece pallet

The workpiece pallet acts as a carrier for workpieces through the conveyor system. With fixturing, it serves to hold the workpiece for processing at a workstation. Positioning bushings allow the pallet to be located in a station with an accuracy of ± 0.05 mm when used with a lift-position unit.

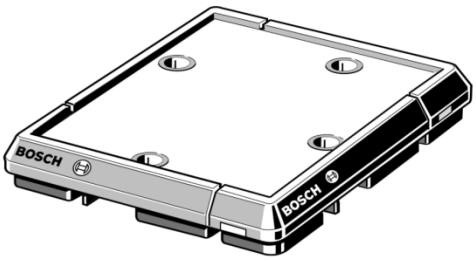


Figure 2.7: Work-piece pallet

The carriers used in the conveyor system of the laboratory have dimensions of 320 mm wide by 320 mm long. They can support a maximum load up to 32 kg. The frame modules are made from electrically conductive polyamide and have built in exciter plates to indicate relative positioning when used in conjunction with proximity switches.

2.2.2. Lift-transfer unit

The Lift-Transfer Unit (LTU) is used to transfer carriers perpendicularly off the conveyor. It is used primarily at corners and intersections, but can also be used for carrier routing changes.

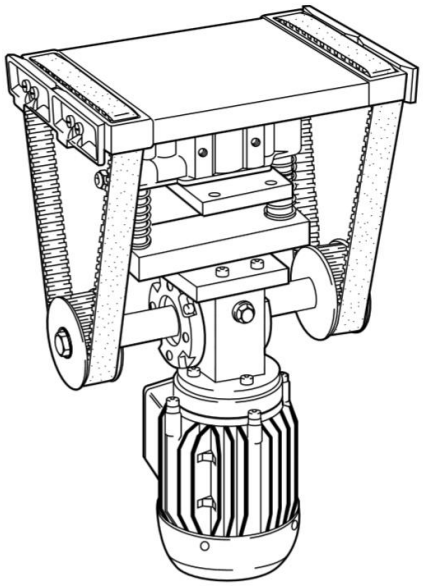


Figure 2.8: Lift-Transfer Unit

The lift plate is powered up and down by a single lift cylinder. In the center, or rest position, the LTU belts are located 1 mm below the bottom of the pallet. A stop bar mounted to the lift plate may be used to stop pallets on the LTU, or inverted so pallets pass through freely.

The LTU is raised by applying air pressure to the bottom of the cylinder. This lifts the LTU to a position 10 mm above the nominal conveyor height. As the LTU rises, the LTU belts engage the pallet and directs (or accepts) the pallet.

2.3. Programmable Logic Controller

A programmable logic controller, PLC, or programmable controller is an industrial computer control system that continuously monitors the state of input devices and makes decisions based upon a custom program, to control the state of devices connected as outputs.

PLCs are designed for multiple arrangements of digital and analog inputs and outputs (see Figure 2.9). Almost any production line, machine function or process can be automated using a PLC. The speed and accuracy of the operation can be greatly enhanced using this type of control system. But the biggest benefit in using a PLC is the ability to change and replicate the operation or process while collecting and communicating vital information.

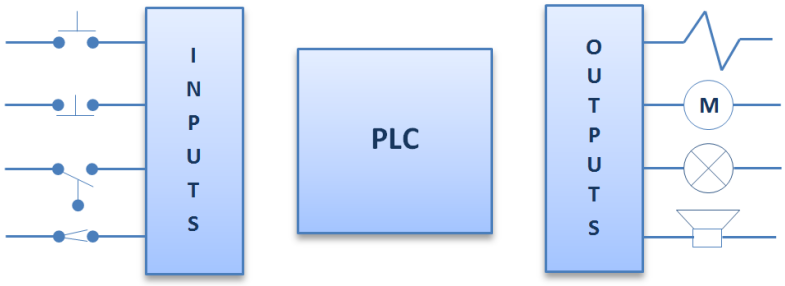


Figure 2.9: PLC main parts

The term logic is used because programming is primarily concerned with implementing logic and switching operations. Input devices (that is, sensors

such as switches) and output devices (motors, valves, etc.) in the system being controlled are connected to the PLC. The operator then enters a sequence of instructions, a program, into the memory of the PLC. The controller then monitors the inputs and outputs according to this program and carries out the control rules for which it has been programmed.

PLCs are now widely used and extend from small, self-contained units for use with perhaps 20 digital inputs/outputs to modular systems that can be used for large numbers of inputs/outputs, handle digital or analog inputs/outputs, and carry out proportional-integral-derivative control modes.

2.3.1. Hardware PLC

Commonly, a PLC system has the basic functional components of Central Processing Unit (CPU), memory, input/output module, communications interface, power supply and the programming device. Figure 2.10 shows the basic layout.

- The **Central Processing Unit** (CPU) is the unit containing the microprocessor. This unit interprets the input signals and carries out the control actions according to the program stored in its memory, communicating the decisions as action signals to the outputs.

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

- The **input/output modules** are where the processor receives information from external devices and communicates information to external devices. The inputs might thus be from switches or other sensor such as photoelectric cells, temperature sensors, induction sensors, or the like. The outputs might be to motor starter coils, solenoid valves, lamps, or similar things. Input and output devices can classified as giving signals that are discrete, digital or analog.

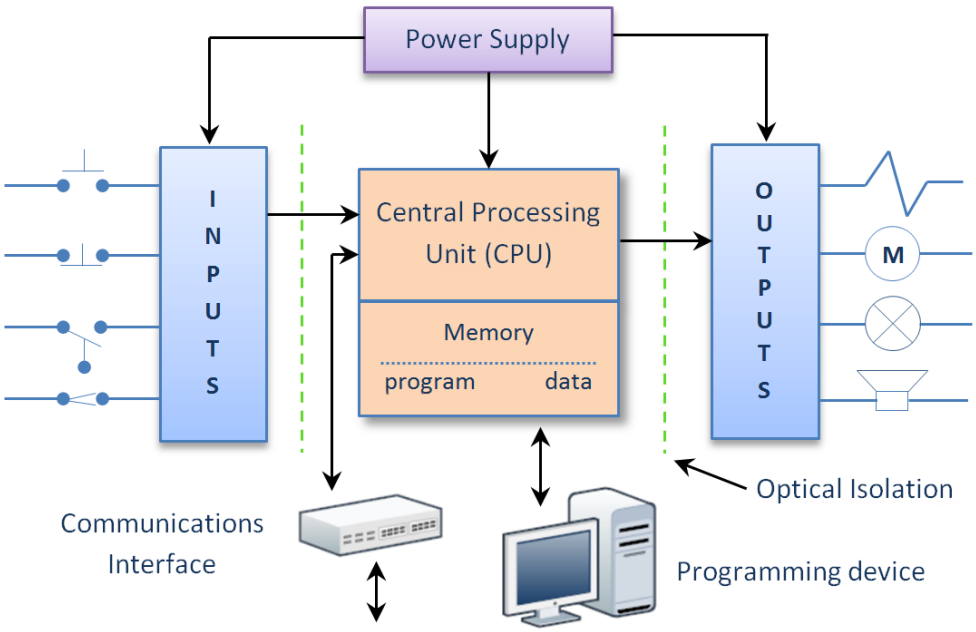


Figure 2.10: Basic layout of a typical PLC

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

- The **power supply unit** is needed to convert the main AC voltage to the low DC voltage (5V or 3.3V) necessary for the processor and the circuits in the input and output interfaces modules.

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

2.3.2. Software PLC

On the other hand, the second type of programmable logic controllers is known as "Soft-PLC". Soft-PLC is a software product, which enabled the industry to build PLC's from standard computer and PLC hardware components. By using this Soft-PLC, users have the flexibility to select hardware according to their priorities: reliability, features, cost, performance, or

vendor relationships. And it could be done independently for CPU and I/O, on a component basis. Soft-PLC was a liberating force for users of proprietary PLC's.

In contrast to a conventional hardware-based control, a Soft-PLC does not have its own hardware. It always has to be integrated in an existing computer system with an Operation System, for example Microsoft Windows. The size of a Soft-PLC load memory can be changed flexibly (up to the maximum available memory size of the PC), in contrast to a hardware PLC memory.

Since there are physical differences between a hardware PLC and a Desktop computer, it is not possible to implement every feature of a hardware PLC in a Soft-PLC. However, existing programs could be executable on a Soft-PLC with small changes.

2.3.3. OpenMHS

2.3.3.1. Overview

OpenMHS is a Soft-PLC with integrated simulator. It was developed in the laboratory for handling technology and robotics at the Hochschule Osnabrück (*Labor für Handhabungstechnik und Robotik*) by Dipl.-Ing. Martin Nardmann.

OpenMHS offers all the functions of a conventional industrial PLC. OpenMHS is even able to operate as a conventional PLC. Through the use of modern computers, it is possible to achieve very short cycle times. In addition, OpenMHS offers the possibility of using interfaces and creating simulation studies.

This Soft-PLC is used to control the assembly system and the industrial robots in the laboratory for handling technology and robotics.

2.3.3.2. Structure

OpenMHS, the PLC control program of the assembly system is divided on some windows. The main window shows one diagram of the whole system (see Figure 2.11). Each work station is represented by a grey and orange block, the rest of sectors are represented by a green and orange block as it can be seen in Figure 2.11.

By clicking in each figure, a new window opens where all the programming of each sectors located. Inside the grey rectangle titled *"Allgemeine Bausteine zur Bearbeitung"* there are four orange blocks. The first orange block titled *"IO_Bausteine"* there are the submodel for the communications (see Figure 2.12).

In this window are all the communications blocks of the assembly system, USB cards for inputs and outputs, RFID readers and connections with PLC controllers of Kawasaki and Stäubli robots. Each module is assigned a separate memory area. For example, USB cards are offset between 0 and 400, the RFID between 2000 and 2200, etc. In this way, it possible to access any bit from the assigned offset.

The fourth block is for conveyor system model created in this work. In Chapter 4.2.1 "Modeling the conveyor system" is explained more in detail.

Going back to the main window shown in Figure 2.11, it contains two round buttons with the name "EIN". By clicking the first button (Bandmotore), the main motors of the assembly line will turn on moving the conveyor belt. The second button is used to put in automatic mode the assembly system.

Below the two buttons "EIN", a green rectangle titled "Manuelle Einstellung RFID" is found. By clicking here, a new window opens. From this new window it is possible to change the initial data written into the RFID system.

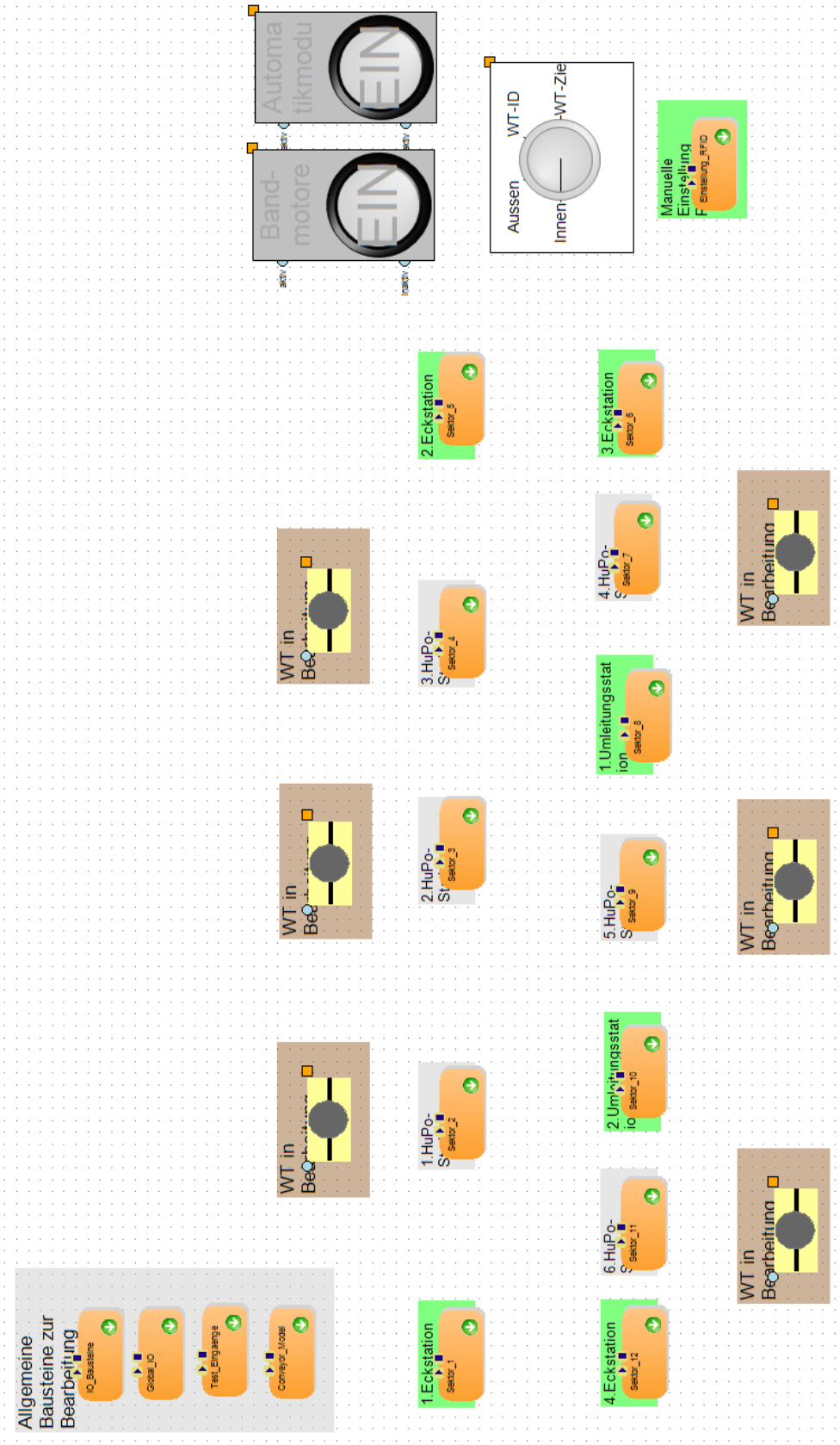


Figure 2.11: Main window panel

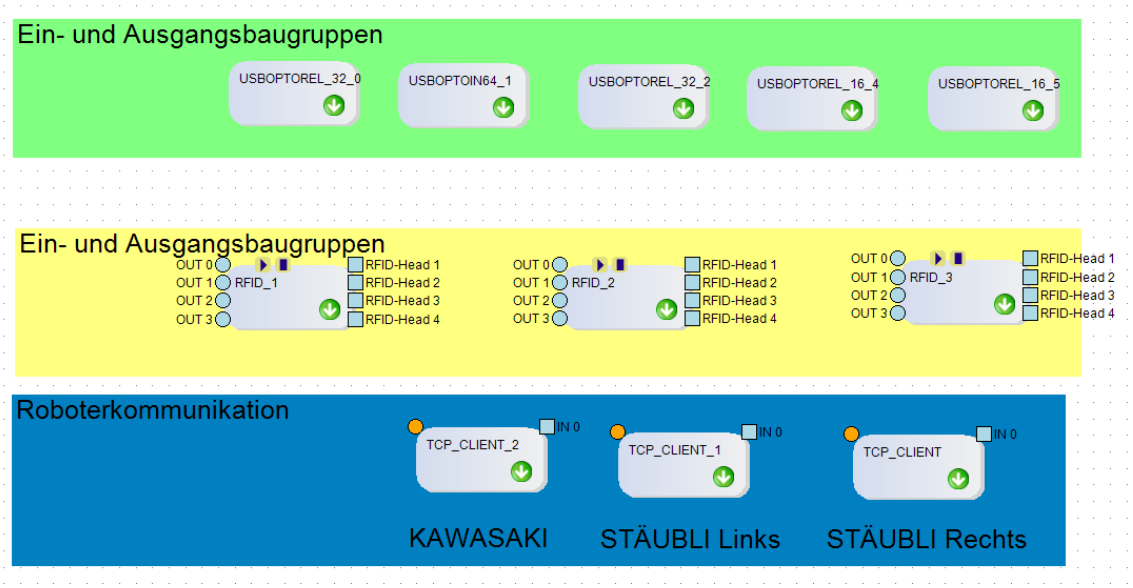


Figure 2.12: Communications window

Finally, Figure 2.13 shows the menu from where the simulation is executed and stopped. This menu is located on the left side of the window and if the program is wanted to be executed the button circled in green must be clicked. On the other hand, if the program is running and it is wanted to stop, the button circled in red must be clicked.

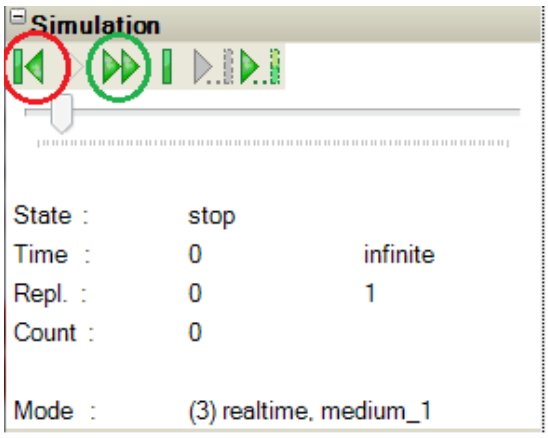


Figure 2.13: Simulation controls

2.3.3.3. Main elements

Create

This element can create parts according to an input signal. The input signal is connected to the first symbol on the left of the element (see Figure 2.14), when a positive pulse arrives a new part is created and then, can be used in the *Conveyor* element.

Figure 2.14: *Create* element

Destroy

The function of this element is to remove the parts that arrive. Once the part arrives at element *Destroy*, this part is removed from the model. Moreover, it allows register how many parts have been removed.

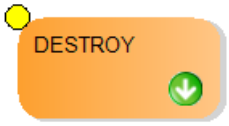


Figure 2.15: *Destroy* element

Conveyor

The *Conveyor* element simulates a queue of parts that crossing a conveyor. Works as a FIFO system (First Input, First Output) where can be define the length of the conveyor and the speed of the parts. In addition, can be used a condition at the beginning or at the end to stop the part and keep it on hold. Another possibility is to connect to the "stop" symbol a signal to stop the conveyor while the signal is active.

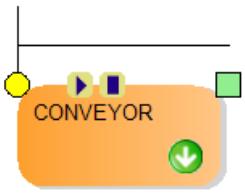


Figure 2.16: *Conveyor* element

Conveyor separator

This element allows distribute parts through different outputs depending on a condition. The parts can be distributed up to three outputs.

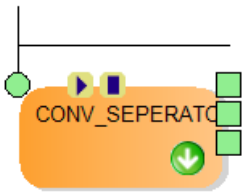


Figure 2.17: *Conveyor separator* element

TCP Server

The *TCP Server* element lets configure a TCP server to allow other applications connected to the network communicate with OpenMHS. It is a TCP socket and uses the Process Image Output of OpenMHS to send information over the network.

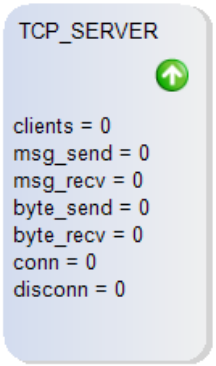


Figure 2.18: *TCP Server* element

Data for Visu

This element is the manager for collect all the parts information that are moving into the *Conveyor* elements. It captures the position, the number of conveyor, an ID and a value for each part. All information captured is sent through the element *TCP Server* to network. In this way, it is possible to use a client application to read the received data.

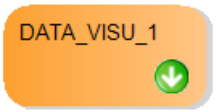


Figure 2.19: *Data for Visu* element

2.3.3.4. Operations Modes

OpenMHS supports the creation of models for different time-dependent problems. Thereby, by using OpenMHS it is possible to solve task of simulation, data acquisition, control and regulation. For this purpose, there are different approaches for calculating the models stored in this software. The calculation of the models is based on time increments.

These time steps would be as:

- Discrete simulation, based on events, generated.
- Continuous simulation, at constant intervals, generated.
- Continuous simulation, in "real-time" pulses, generated.

These different types of calculation can be mixed to some extent application.

The variable calculation models allow it to be used by OpenMHS for:

- Simulation of material flows (frequently event).
- Data acquisition (always real-time, combined with an event-driven).
- Data analysis, offline (constant time steps).
- Soft-PLC (real-time ever, combined with an event-driven).

- PLC simulation (constant time steps, combined with an event-driven).
- Simulation of a contract manufacturing often (frequently event-driven).

Real-time mode

The calculation in this mode is realized at equidistant points in time. The PLC waits between these points the required time until the next cycle. Each calculation involves always the next three steps:

- At first, inputs of the PLC can be queried and written into memory. This memory is called the process input image.

- The second step is the processing of the program. The program picks there exclusively the input values in memory. In this way it is ensured that the input values remain constant during the program run. The calculated output values are written to the process image output table.

- In the third step, the hardware outputs are assigned the values from memory.

Figure 2.20 clarifies the work field.

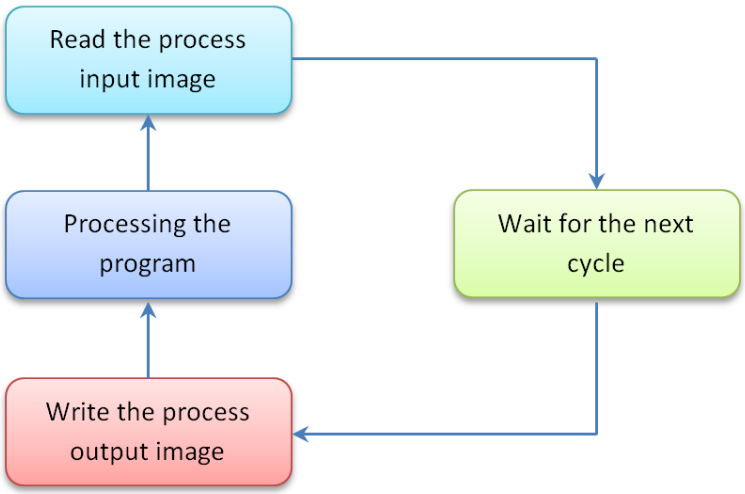


Figure 2.20: Work cycle of a Soft-PLC

2.4. Process Simulate

2.4.1. Overview

Process Simulate is part of Tecnomatix application suites owned by Siemens Product Lifecycle Management Software Inc. It enables process planning, resource planning, part planning and simulation of a manufacturing process in a virtual environment. In order to use Process Simulate for monitoring an industrial process using the Tecnomatix .NET API, it is important to study the structure of the software. The basic configuration of the applications in Tecnomatix is given in Figure 2.21. The software is organized in three layers consisting of a database, a server and clients.

The first layer consists of an Oracle Database Server whose main task is to manage data and ensure the data update is handled correctly. The database is sub-divided into schemas. The eMServer is in the next layer and this is the core element in the configuration. Services are provided by the eMServer to applications/clients, e.g. requests for data from clients are sent forward to the database. The eMServer handles the communication between the clients and the database. The last layer contains the different clients which can represent the applications in the Tecnomatix environment, e.g. Process Simulate.

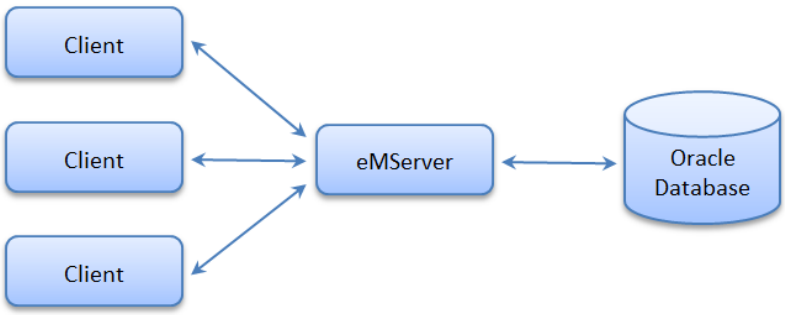


Figure 2.21: Tecnomatix architecture

The data structure of the Tecnomatix clients is displayed in Figure 2.22. As mentioned previously, the database is divided into schemas. These consist of projects which are each made up of objects/nodes structured in tress. These nodes are products, operations, resources and manufacturing features that

together define the manufacturing process. A tree can only contain a group of the same nodes, e.g. resources. Finally, attributes can be attached to the nodes, e.g. files with 3D data or CATIA files.

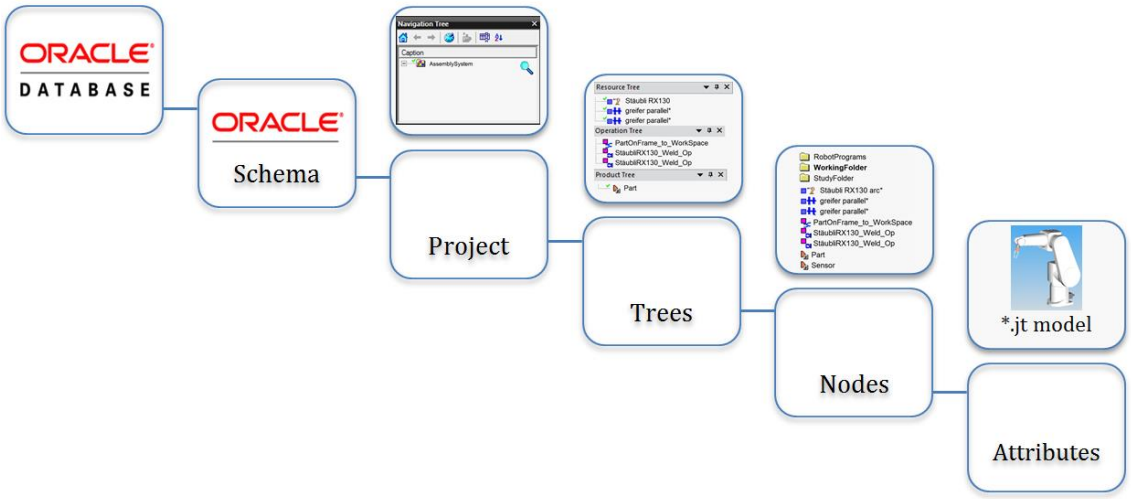


Figure 2.22: Tecnomatix data structure

The environment of Process Simulate is modular and allows creating plug-ins according to Tecnomatix .NET API. Using this API enables the possibility to function as a client for the real-time monitoring proposed in this thesis.

3. Possible solutions for monitoring systems

To achieve the objectives in this work have been proposed several solutions. Two different solutions with their advantages and disadvantages are discussed below, and then, the selected solution is explained.

3.1. IP Camera or CCTV system

One or more IP cameras can be placed into the lab to monitor the whole manufacturing process. Furthermore, the necessary hardware and software will be installed in the client-side, allowing to observing the real-time motion of robots and conveyor system. Otherwise, besides the cost of the equipment, also is required much bandwidth to be acceptable quality. The advantages and disadvantages are listed below.

Advantages

- Easy to implement
- Remote accessibility

Disadvantages

- Need to buy new hardware
- High cost
- High network bandwidth requirements
- Poor process control
- Unable to set alerts to events
- Possibility of blind spots or low visibility areas

3.2. Commercial software and hardware for motion capture

Another alternative solution is to use motion sensors and position in each robot and conveyor. These sensors have their own software that represents the motion captured and allows real-time monitoring of the manufacturing process. One problem, besides the cost, is the difficulty to place the sensors in each of the elements to be monitored.

Advantages

- Remote accessibility
- High control

Disadvantages

- Hard to implement
- High cost
- Need to buy new hardware
- Need to modify the assembly system
- Little flexibility, commercial software.

3.3. Development a plug-in for Process Simulate and use a Soft-PLC

The third alternative consists of developing a plug-in for Process Simulate and use virtual model for monitoring the assembly process. All robots provide real-time joints data, so only are needed obtain this data and represent them over a 3D model.

The conveyor system case is a bit more complicated because this information can't obtain directly. But it can be simulated through PLC software from the own university and then, use this motion data for monitoring.

The advantage of this solution, in addition to the low cost of material, since licenses are available for use Process Simulate, is high customization offered. For example, the application can be programmed to display on-screen alerts based on certain events.

Advantages

- Remote accessibility
- High control
- Low cost
- High flexibility
- Ability for improvements and updates
- Ability to schedule alerts to events

Disadvantages

- Require Tecnomatix software
- Need to know programming
- Trouble to monitoring the conveyor system
- Potential errors or faults, it is not a mature software

4. Real-Time Motion Monitoring Application for Process Simulate

4.1. Client-Server communication

First, it will be explained how to do the communication between the application to be developed and the assembly system. All robots are communicated using the TCP/IP protocol with the PLC software OpenMHS, the conveyor is also controlled from this software via USB cards that allow you to interact with the elements of the conveyor.

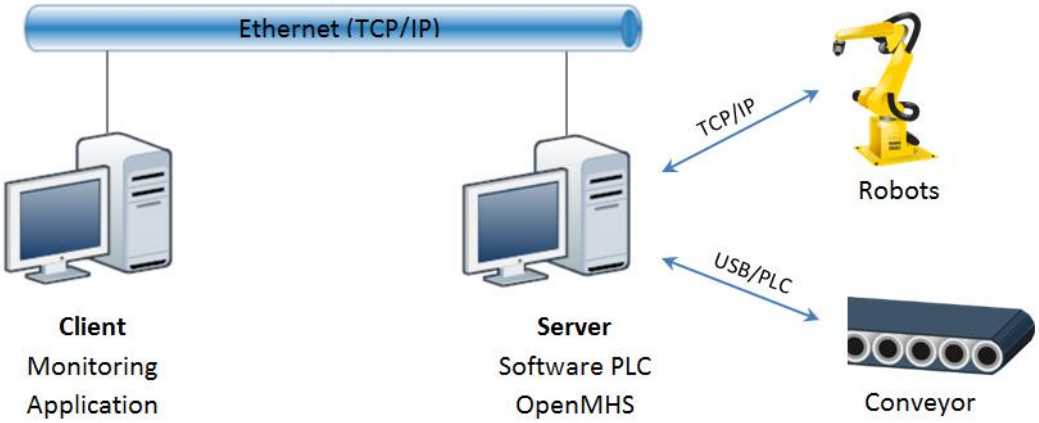


Figure 4.1: Network structure

In the Figure 4.1 it can be seen a diagram where the logical distribution of the various elements of the system is shown. Robots and conveyor communicate with the Soft PLC OpenMHS, and this in turn, works as a server to transmit information to clients with Process Simulate. The communication between client and server is done under the TCP/IP explained in the next chapter, the monitoring application must use a socket programmed in C# to connect to the server and receive the data. This is discussed in more detail in Chapter 4.3.

4.1.1. The TCP/IP protocol

TCP/IP (Transmission Control Protocol/Internet Protocol) is the basic communication language or protocol of the Internet. It can also be used as a communications protocol in a private network (either an intranet or an extranet).

TCP/IP is a two-layer program. The higher layer, *Transmission Control Protocol*, manages the assembling of a message or file into smaller packets that are transmitted over the Internet and received by a TCP layer that reassembles the packets into the original message. The lower layer, *Internet Protocol*, handles the address part of each packet so that it gets to the right destination. Each gateway computer on the network checks this address to see where to forward the message.

Figure 4.2 shows the relationships among the protocols, applications, and sockets API in the client-server, as well as the flow of data from one application (using TCP) to another. The boxes labeled TCP and IP represent implementations of those protocols. Such implementations typically reside in the operating system of a host. Applications access the services provided by TCP through the sockets API. The arrow depicts the flow of data from the OpenMHS application, through the TCP and IP implementations, through the network, and back up through the IP and TCP implementations at the other end.

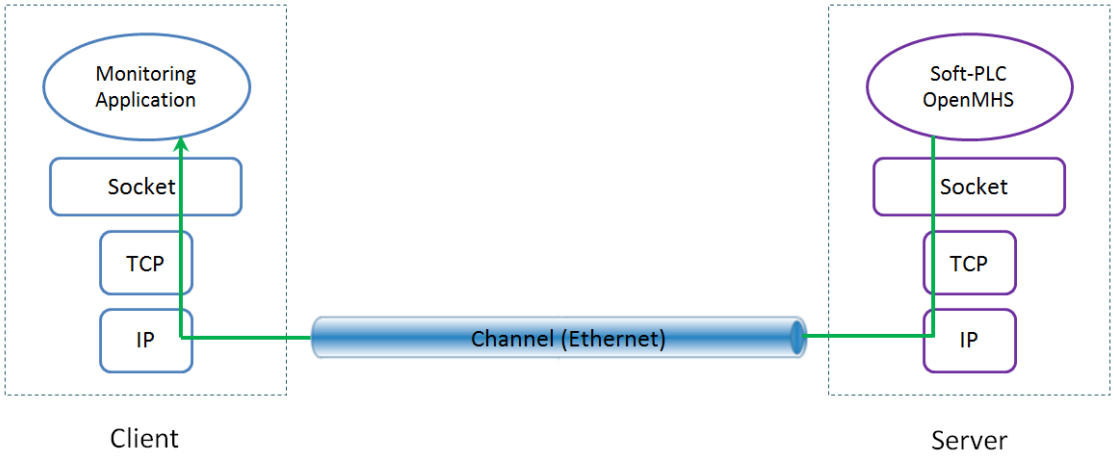


Figure 4.2: A TCP/IP network

TCP is designed to detect and recover from the losses, duplications, and other errors that may occur in the client-server channel provided by IP. TCP provides a *reliable byte-stream* channel, so that applications don't have to deal with these problems. It is a *connection-oriented* protocol: Before using it to communicate, two programs must first establish a TCP connection, which involves completing an exchange of *handshake messages* between the TCP implementations on the two communicating computers. Using TCP is similar to file input/output (I/O). In fact, a file that is written by one program and read by another is a reasonable mode of communication over a TCP connection.

TCP provides reliability by doing the following:

- The application data is broken into what TCP considers the best sized chunks to send. The unit of information passed by TCP to IP is called segment.

- When TCP sends a segment it maintains a timer, waiting for the other end to acknowledge reception of the segment. If an acknowledgment is not received in time, the segment is retransmitted.

- When TCP receives data from the other end of the connection, it sends an acknowledgment (see Figure 4.3). This acknowledgment is not sent immediately, but normally delayed a fraction of second.

- TCP maintains a checksum on its header and data. This is an end-to-end checksum whose purpose is to detect any modification of the data in transit. If a segment arrives with an invalid checksum, TCP discards it and does not acknowledge receiving it (it expects the sender to time out and retransmits).

- Since TCP segments are transmitted as IP datagrams, and since IP datagrams can arrive out of order, TCP segments can arrive out of order. A receiving TCP re-sequences the data if necessary, passing the received data in the correct order to the application.

- TCP also provides flow control. Each end of a TCP connection has a finite amount of buffer space. A receiving TCP only allows the other end to send as much data as the receiver has buffers for. This prevents a fast host from taking all the buffers on a slower host.

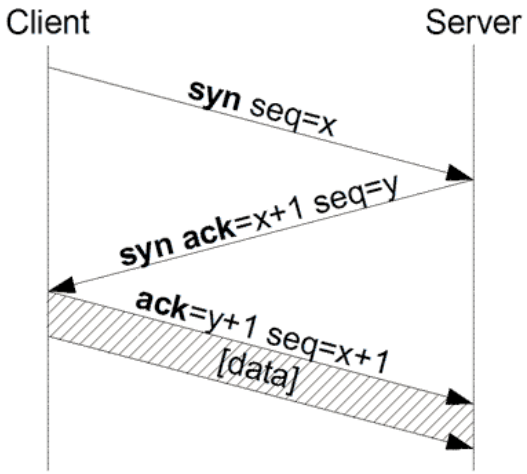


Figure 4.3: Protocol operation

4.2. Programming the server side

4.2.1. Monitoring the conveyor system

To monitoring conveyor state is necessary to know the position of each tray at each instant of time. The conveyor does not provide this information, because for the correct working is not necessary. To control it, the PLC uses different inductive sensors located throughout the conveyor system. This system is divided into 12 sectors. At the beginning of each sector it has a stopper to stop the trays when the PLC program decides. Through these sensors and the stoppers, Soft-PLC can control the whole transport process.

After analyzing the conveyor working, three possible solutions to monitoring the position of each tray at each instant of time are reached. Then, each solution with its advantages and disadvantages are described.

4.2.1.1. Possible solutions

1. Implement the conveyor logic into the monitoring application

Reading the status of each sensor at the beginning of each sector is possible to determine if a tray enters that sector. As the speed is constant, the position can be calculated once the tray is detected. This makes it possible to program the logic states reading sensors and calculate the approximate position of the tray over the conveyor.

This solution has a disadvantage. It needs instantly to know the change of state of each sensor. Due to the small delay between request and response data to the server, not always the status change is detected in the sensors, triggering errors in the visualization.

2. Collect all data of the whole conveyor system into OpenMHS

Another possible solution is to model the trays motion inside the software PLC. Each sector is divided into segments the size of each tray, and a list is generated with the information of all segments that will be sent to the monitoring application (see Figure 4.4).

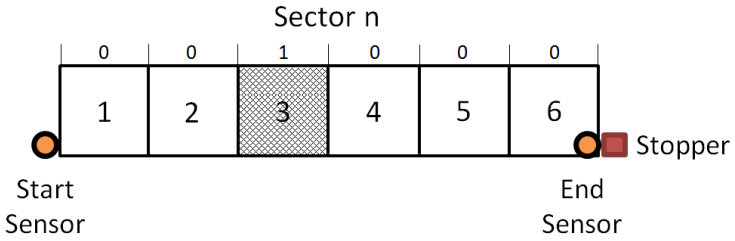


Figure 4.4: Segments diagram

This list contains a segment identifier and the information of if have or not tray at that time. In the case of have a tray, is displayed on the 3D model by the monitoring application.

This solution permitted real-time motion monitoring of all trays, because all the logic is in the PLC software on the server.

A disadvantage of this method is the large amount of information transmitted per unit time, because the information is being sent to all segments in which it has divided the conveyor. In addition, the movement of the trays on the 3D model is not continuous, jumps to switch segment appears (see Figure 4.4).

3. Collect only the data of each tray into OpenMHS

The third solution is a mixture of the other two. On one hand, all the motion logic of the trays is modeled in the Soft-PLC from server. On the other hand, a list is generated with the information of each tray moving (the ID, position, sector number) and is sent to the monitoring application.

In the Figure 4.5 can be seen one of the sectors, once the tray comes in into the sector its position can be determinate, because the movement speed is constant. At the end of the sector another sensor detects the tray, if the tray stops at the stopper the sensor continue active, indicating that the tray remains in that position. In this way if another tray enters in the sector, it will stop just behind the first tray, following the methodology of a FIFO system (First Input, First Output).

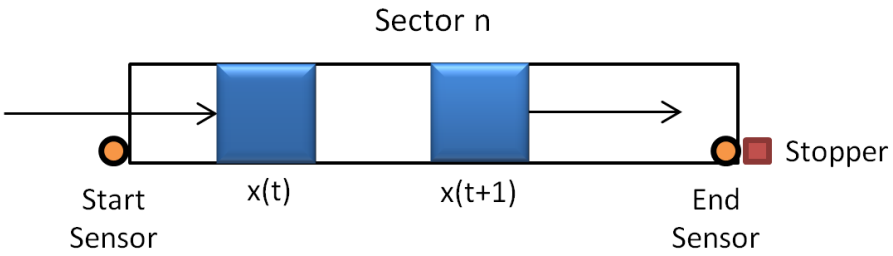


Figure 4.5: Solution three diagram

This solution permits work in real-time, because logic is implemented from the server side, and allows to use a smaller size buffer than solution 2, since only each moving tray sends data. **Therefore, this is the selected solution to monitoring the conveyor system**.

4.2.1.2. Modelling the conveyor system into OpenMHS

After analyzing the possible solutions and select the best one, implementation begins. First, it needed divide the conveyor system in sectors or segments. In Figure 4.6 can be seen the layout of different segments.

Each segment is modeled in OpenMHS, and is composed at least of the following blocks:

- **Creator block**: It is responsible for generating a carrier each time it receives the input signal from the sector.

- **Conveyor block**: it is a model from conveyor with a specified length, with this block is possible to model the carrier motion as if was an object of a queuing system.

- **Destroy block**: at the end of segment, the carrier is removed in this block.

- **Switch block y logic block OR**: these blocks are used only for testing, during normal operation are not necessary.

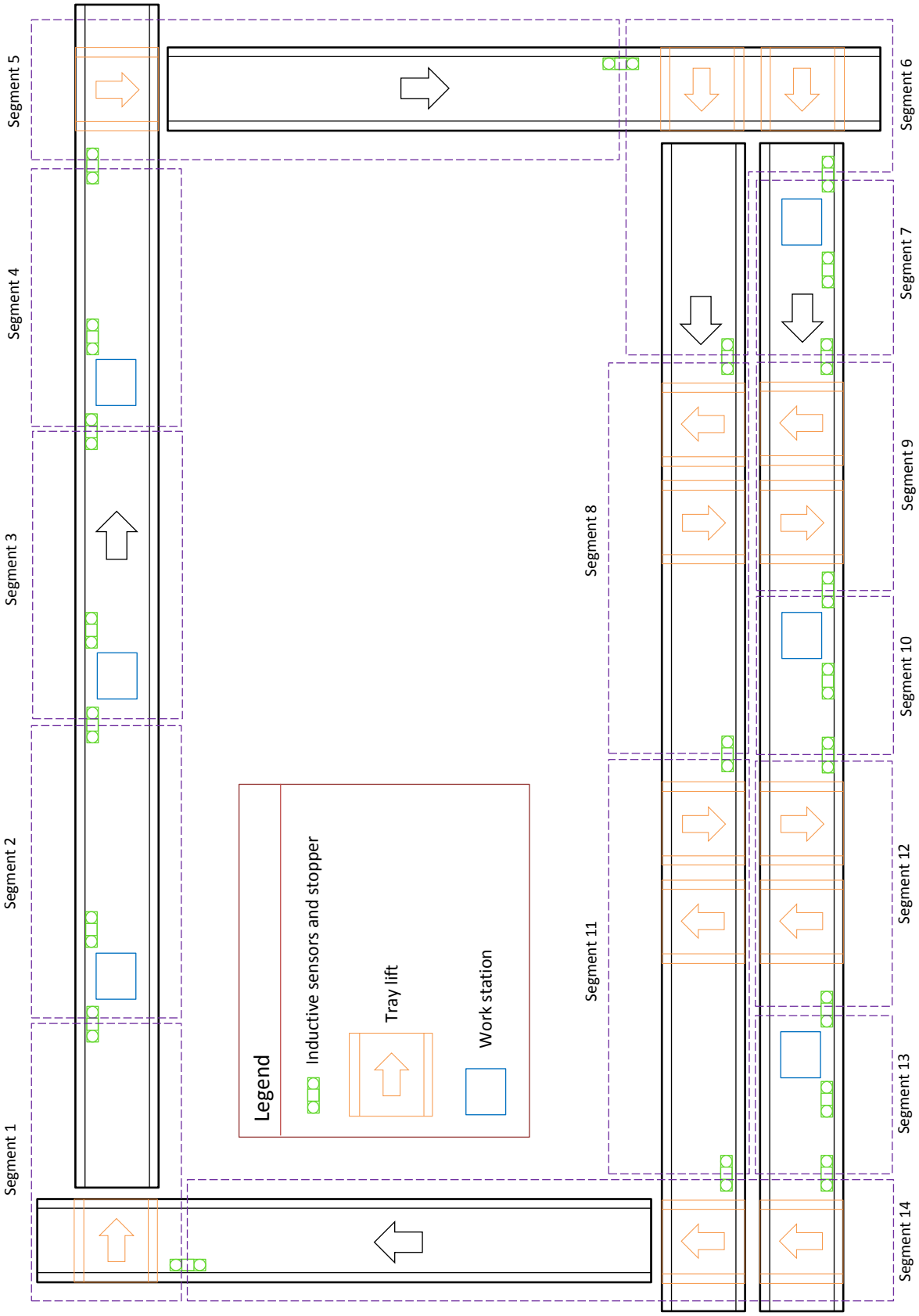


Figure 4.6: Segments layout

Figure 4.7 corresponds to the segments 2, 3, 4, 7, 10 and 13. These segments are modeled in the same way, a *Creator* block, two blocks *Conveyor* and a *Destroy* block. Each *Conveyor* block has a stop condition at the end, and it is activated depending on the status of the sensors of that segment.

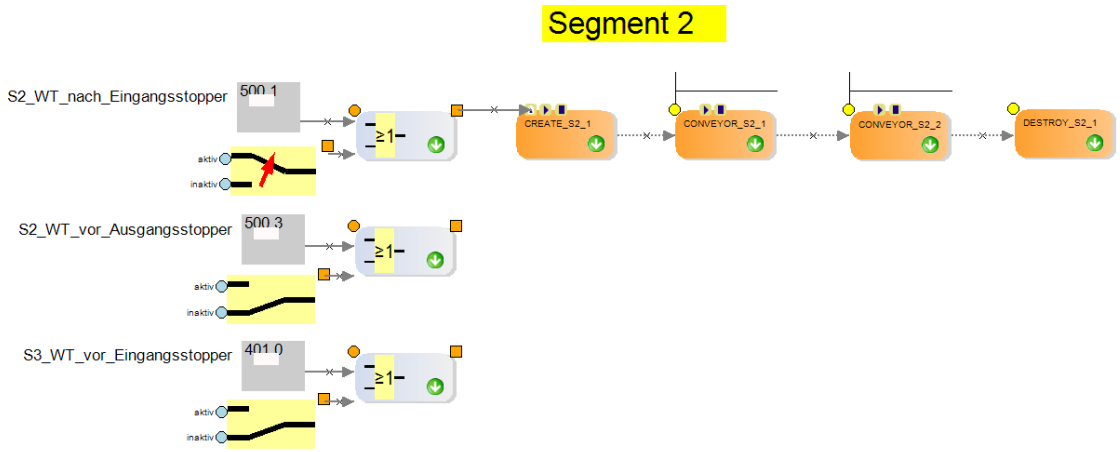


Figure 4.7: Blocks diagram segment 2

The corners for the segment 1 and 5 are modeled according to Figure 4.7. In this case only a *creator* block, a *conveyor* block and a *destroy* block are needed. When the carrier is placed over the lift, is necessary to pause for a few seconds therefore, the *conveyor* block receives a signal that allows stopping the carrier while the lift is working.

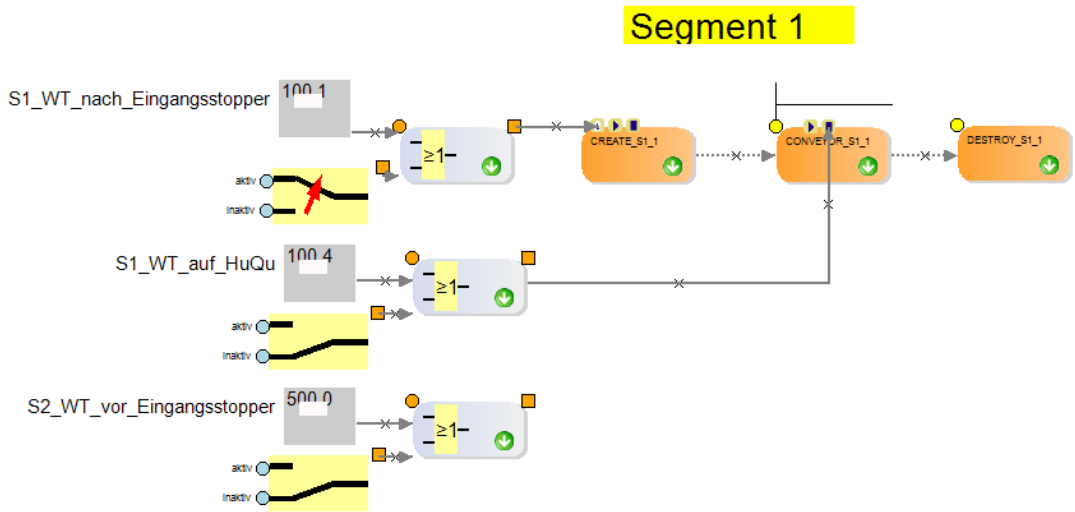


Figure 4.8: Blocks diagram segment 1

The corner for segment 6 is a special case, because the carriers can follow two paths. To model this situation is necessary to use a new block called *conveyor separator*. With this block the carrier path can be chosen. This decision is based on the position of the first lift, whether it is in the "down" position the outside line is taken, if it is in the "up" position the inside line is taken. In Figure 4.9 can be seen this configuration.

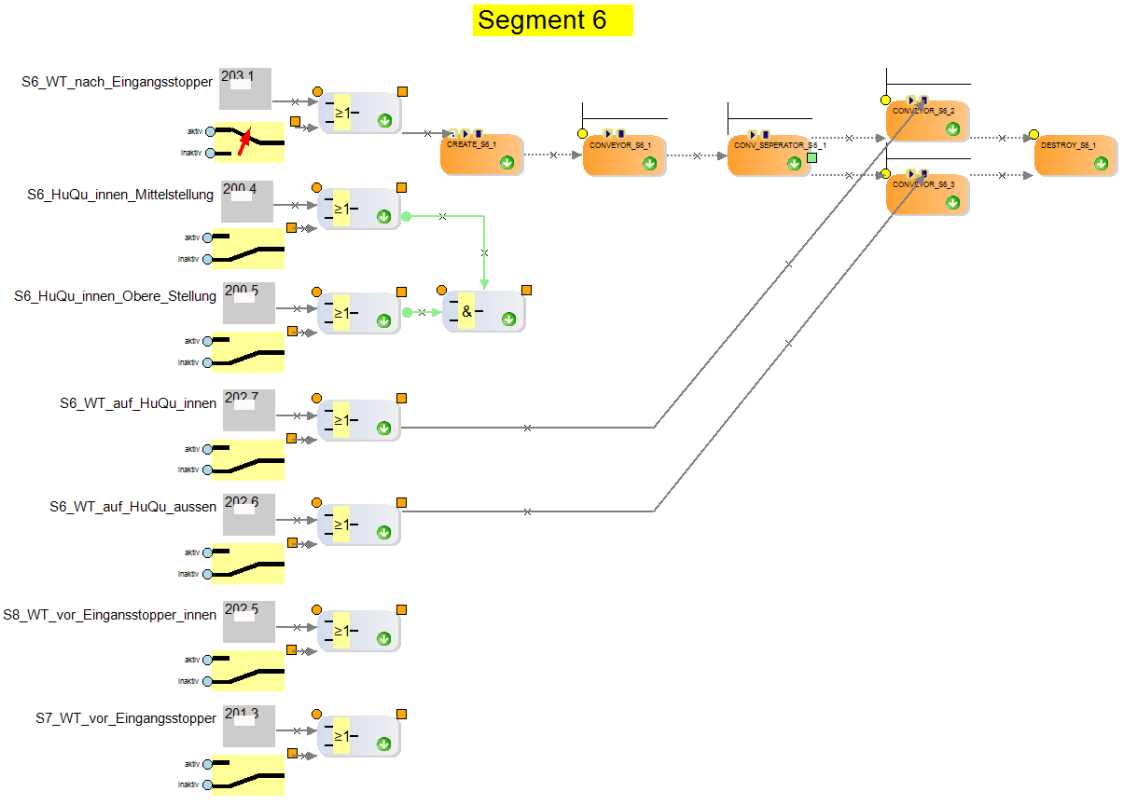


Figure 4.9: Blocks diagram segment 6

In this case, a signal is also used to wait the time it takes the lift to change position. A logic gate is included to determine the position of the elevator, because it only has information of the "up" and "center" position. Therefore, when no signal, it means that the lift is in the "down" position.

Last corner, segment 14 (see Figure 4.10), is a different case. It is the opposite case of segment 6. It is necessary that the carriers that come from the inside or outside line come out to single output. To model this operation mode two *creator* blocks and two *conveyor* blocks are used. This makes it possible to model the two situations, since the path length of each *conveyor* block is different. Elevators signal is also used to pause the movement during operation.

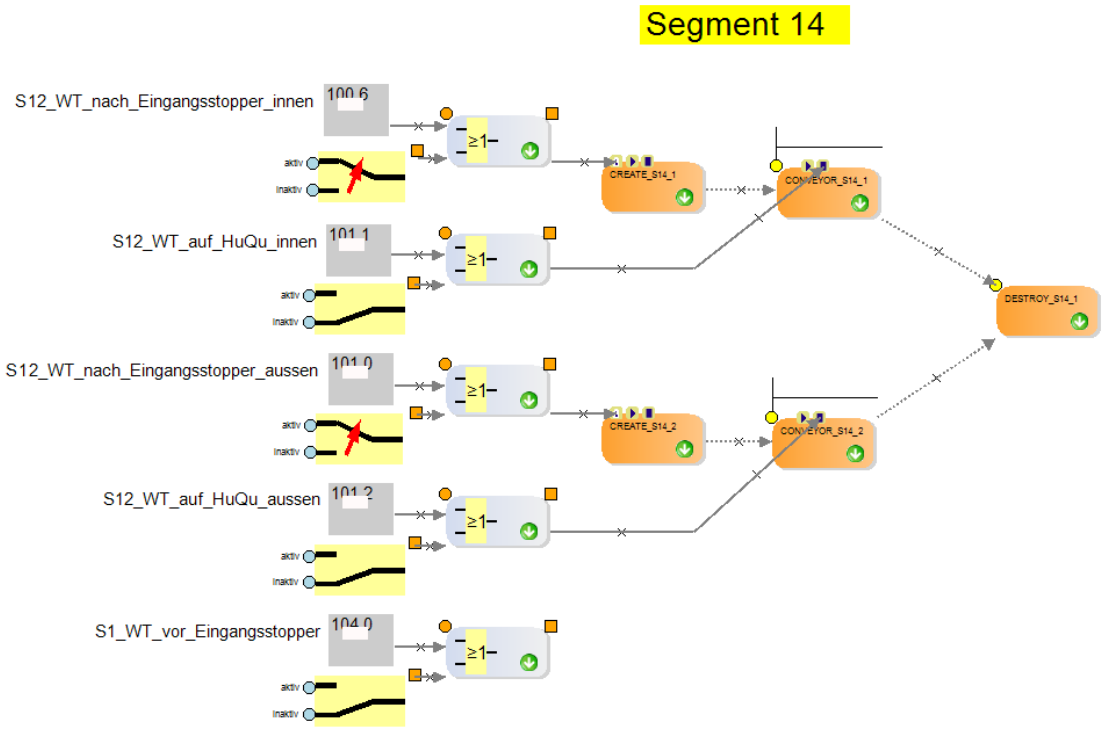


Figure 4.10: Blocks diagram segment 14

Then the segments 8, 9, 11 and 12 are modeled. In these segments the carriers can change the direction from the inside to the outside line and back again. *Conveyor Separator* blocks and conditions of the position of the respective lifts are used to model this situation. If the lift is in the "down" position, carrier continuous into the same line, else if it is in the "up" position, carrier must change line. Due to lack of information directly from the "lower" state, is also used a logic gate AND. It is very similar to the case of segment 6. In Figure 4.11 can see the segment 8, the same design method can be applied to 9, 11 and 12.

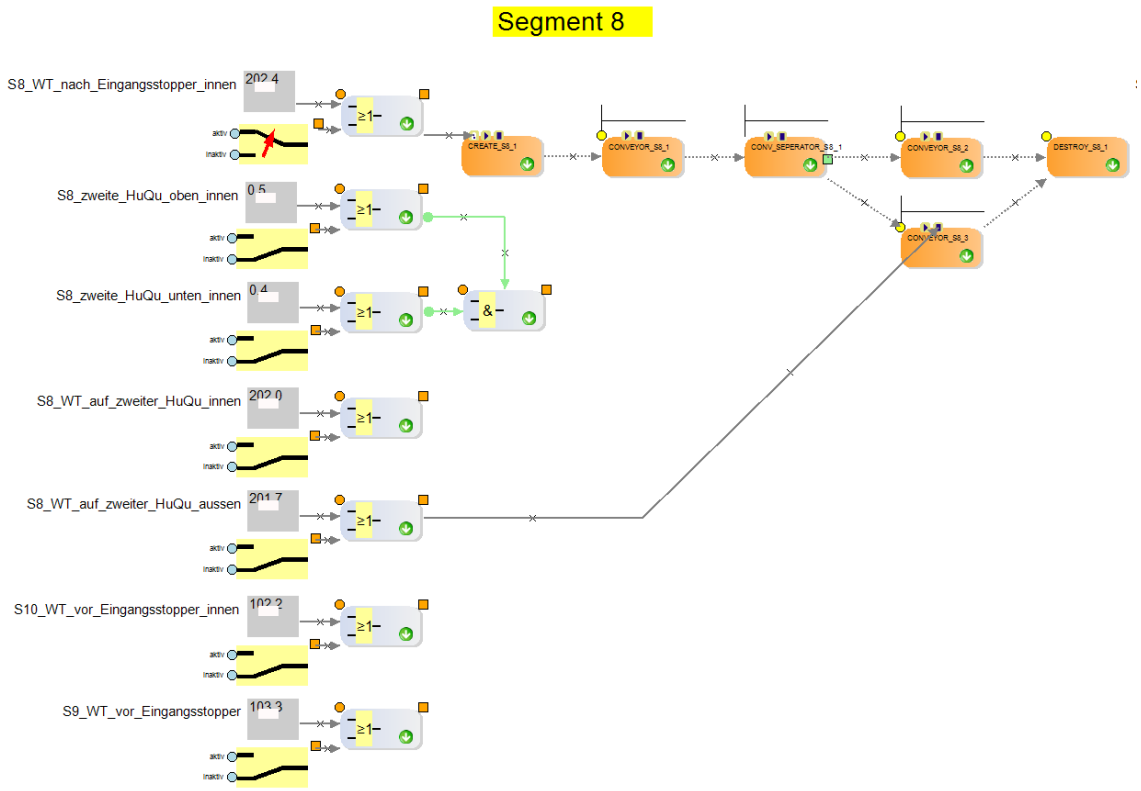


Figure 4.11: Blocks diagram segment 8

Finally, in Figure 4.12 the whole conveyor system model in OpenMHS shown, with all segments explained above.

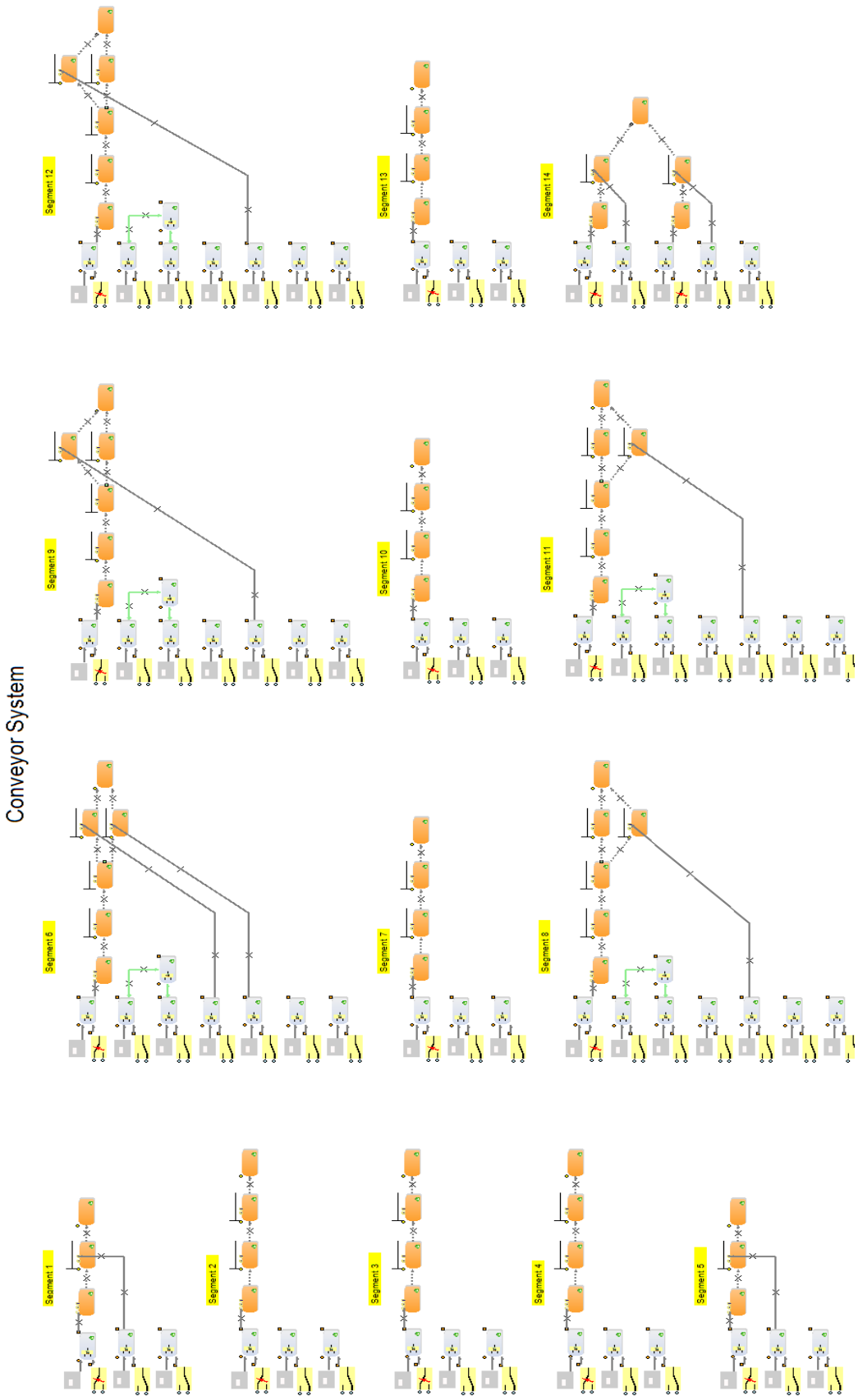


Figure 4.12: Conveyor system model

4.2.2. Process Image Addresses

All information collected from OpenMHS is organized in different directions of the Process Image, which is then sent to the network. To read the information from the monitoring application is necessary to know the position of each data. Each data assigning with its corresponding direction is shown in Table 4.1.

OpenMHS Process Image		
Offset	**Data**	**Comment**
0 to 499	Conveyor System	ID, position, conveyor and value
500	Stäubli 1 State 1	Overall state
501	Stäubli 1 State 5	Mode Selected
502	Stäubli 1 Joint 1	Joints compressed (+120/3)
503	Stäubli 1 Joint 2	
504	Stäubli 1 Joint 3	
505	Stäubli 1 Joint 4	
506	Stäubli 1 Joint 5	
507	Stäubli 1 Joint 6	
508 to 511	NC	Not use
512	Stäubli 2 State 1	Overall state
513	Stäubli 2 State 5	Mode Selected
514	Stäubli 2 Joint 1	Joints compressed (+120/3)
515	Stäubli 2 Joint 2	
516	Stäubli 2 Joint 3	
517	Stäubli 2 Joint 4	
518	Stäubli 2 Joint 5	
519	Stäubli 2 Joint 6	
520 to 523	NC	Not use
524	Kawasaki State 1	
525	Kawasaki State 2	
526	Kawasaki Joint 1	Joints compressed (+120/3)
527	Kawasaki Joint 2	
528	Kawasaki Joint 3	
529	Kawasaki Joint 4	
530	Kawasaki Joint 5	
531	Kawasaki Joint 6	
532 to 535	NC	Not use

Table 4.1: Process Image Addresses

4.3. Programming the client side

When studying the documentation of the Tecnomatix.NET API, a class was found that enabled the creation of a command button integrated into Process Simulate. The general steps used for creating a plug-in can be described using a flowchart diagram, see Figure 4.13. The first step is to connect to the server; if connection fails, it is attempting again. When connected to the server, the next step is to read and write all data in a buffer. Once the data are obtained, conveyor trays and robots move. This sequence is repeated every 50 milliseconds.

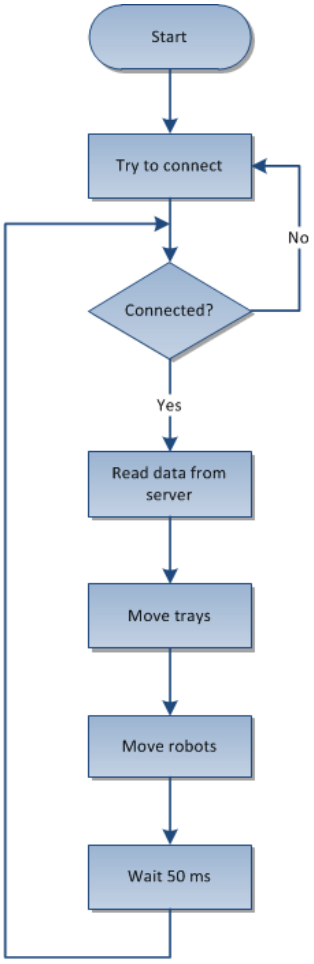


Figure 4.13: The flowchart shows an overview of how the application works

In the following sections the monitoring application will be described in more detail. For development the monitoring application has used the programming language C# object-oriented, because the Tecnomatix .NET API is ready for this programming language.

4.3.1. The TCP Socket

To connect to the server and receive the data is necessary to open a TCP socket, which provides methods to manage the connection, sending and receiving through the network. To create the TCP socket is needed to know the IP address of the server, the port used and also memory space to store the received data in the buffer. In Figure 4.14 the operation of this object is described using a flowchart.

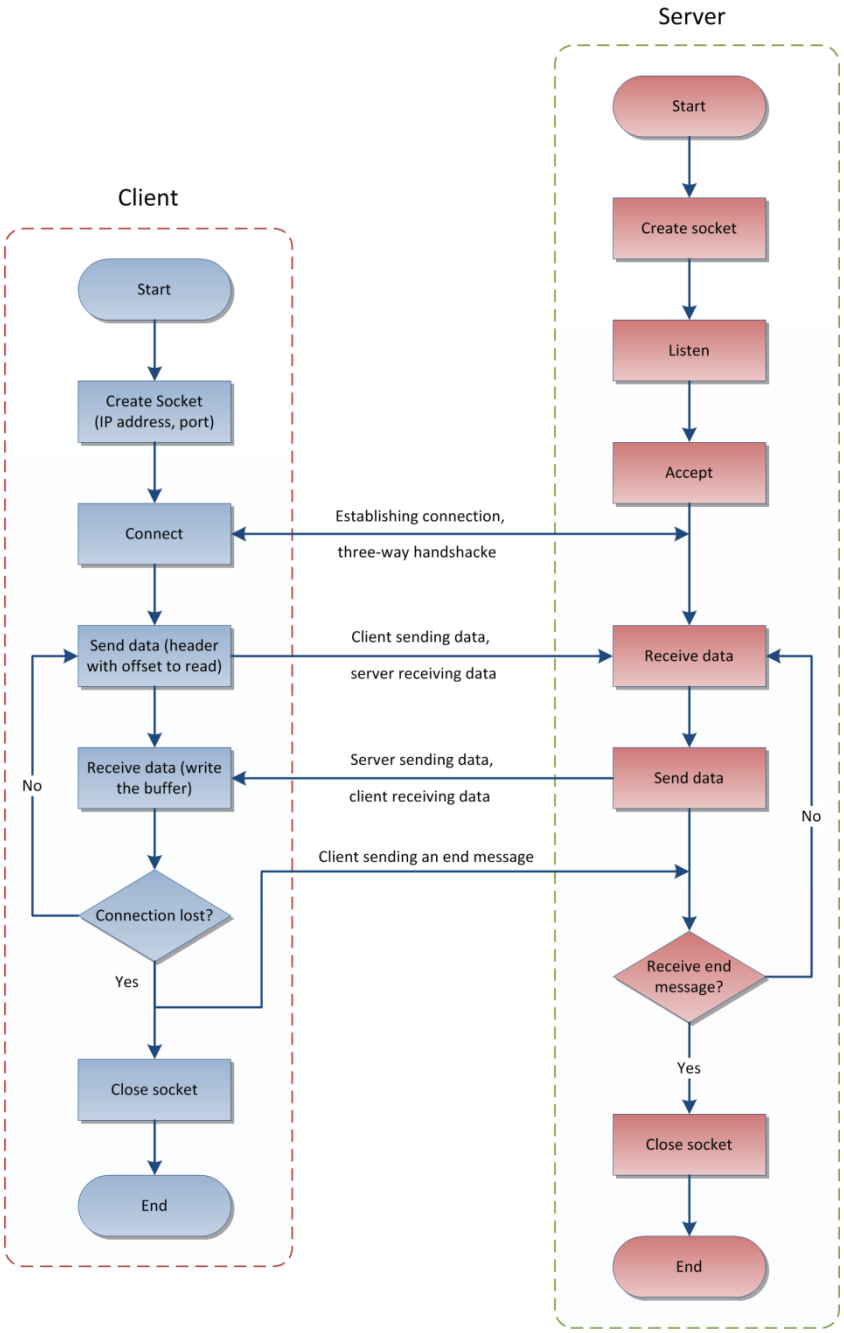


Figure 4.14: Flowchart for sockets communication using TCP

After creating the socket (the IP address of the server and port is required), it tries to connect to the server, once connected it sends a message requesting data from the server Process Image. The server responds by sending the data to the client, and then the socket writes data to the buffer. If the connection is still active the procedure is repeated again, updating the information in the buffer. If the connection is terminated, it informs the server to close the communication with the client.

4.3.2. Representation of motion data from the robots

First, collect the information of robot objects in the virtual model in Process Simulate is needed. Using *TxApplication.ActiveDocument.PhysicalRoot.GetAllDescendants* it is possible to access all physical objects in the virtual cell in Process Simulate. In order to retrieve the nodes displayed in the folder *Resources* in the Object Tree, all objects are filtered using a class *TxTypeFilter*. If *TxRobot* is used as a filter, the resulting objects are robots and are saved for later use.

The flowchart in Figure 4.15 shows the sequence to move all robots from virtual model.

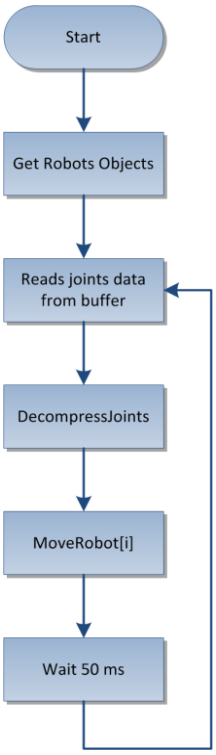


Figure 4.15: Flowchart for move robots

After obtaining the objects "robot" from virtual model into Process Simulate, the data of the six joints are read from the buffer. Joints values are compressed to use less bytes in the data transfer. Then, it calls the *DecompressJoints* method, which subtract 120 and multiply by 3 each joint, thus turns to obtain the real value of the joint. Finally the *MoveRobot[i]* method updates the values of the joints in the "robot" object. Once values are updated, *TxApplication.RefreshDisplay* method is called to update the graphical view.

4.3.3. Representation of motion data from the conveyor

First, collect the information of carrier objects in the virtual model in Process Simulate is needed. Using *TxApplication.ActiveDocument.PhysicalRoot.GetAllDescendants* it is possible to access all physical objects. In order to retrieve the conveyor object, all objects are filtered using a class *TxTypeFilter* with *TxDevice* as filter. Then using the method *GetAllDescendants* again over the conveyor object it is possible to access to the carriers. If *Tx2Or3DimensionalGeometry* is used as a filter, the resulting objects are the carriers.

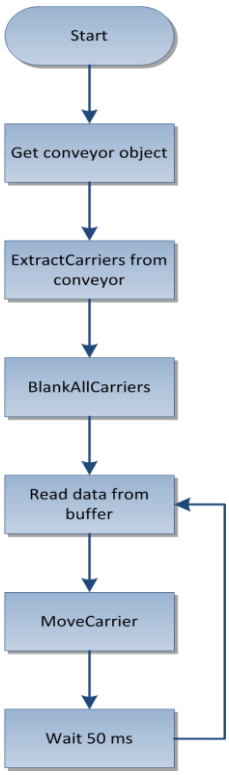


Figure 4.16: Flowchart for move carriers

Once obtained all the carrier objects, they hide from the 3D model using the method Blank. When the server connection is established, the ID, segment number and position of each carrier moving read from the buffer.

With this data is called *MoveCarrier* method. With the ID number is checked if the current carrier is already in motion, if it is not assigned a new carrier. If the carrier is already in motion, the location vector with the new position is updated in the object "carrier". After upgrading the position data, the graphic view is refreshed by the method *TxApplication.RefreshDisplay.*

In the event that a carrier is removed from the conveyor system, its ID does not reappear. In that case the object "carrier" used the ID that is hidden, and is released for use in a new carrier.

This process is repeated every 50 ms while the connection to the server is maintained. Because the high refresh rate, the sensation of movement in objects that change position is produced.

4.3.4. Simple Button Command

The simple button command to execute de application was named *MotionMonitoring* and was placed in the tool bar of Process Simulate, see Figure 4.17. Once the user clicked on the button the GUI displayed in Figure 4.18 appeared.

Figure 4.17: Command button *MotionMonitoring*

4.3.5. The Graphic User Interface

A simple GUI was developed in C# to make it easy for a Process Simulate user to connect to a server. The steps previously mentioned in Section 4.3.1, 4.3.2 and 4.3.3 have been covered in the GUI in Figure 4.18. This is the main window for the plug-in and is named *MotionMonitoring*. If the checkbox

"Try to connect" is checked the application try to connect to the server with the IP address and Port assigned. Once connected, if data movement and has loaded the 3D model will start monitoring.

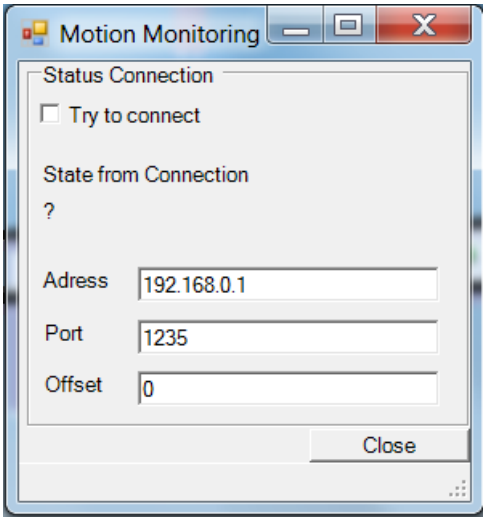


Figure 4.18: The GUI for setting the connection data

5. Test scenarios

5.1. Test one robot from local server

The first test is done by simulating the motion data of a robot from the same computer, server and client on the same computer. Adjustments are made in the application to get a good response. All joints are tested and satisfactory results are obtained. Figure 5.1 shows a screenshot of the test.

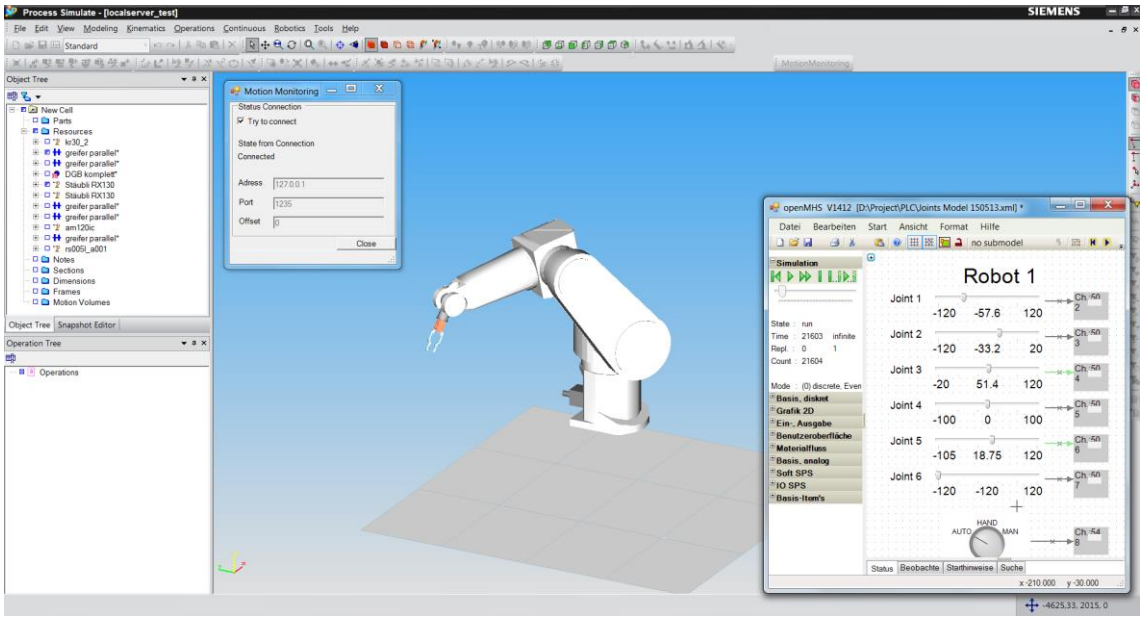


Figure 5.1: Monitoring one robot from local server

5.2. Test one robot in real world

The above test is repeated in a completely different scenario. Now the communication with the lab server is tested, the speed of response is also checked with the real robot. The results are very good and the refresh rate improved from the previous test. In Figure 5.2 a screenshot of real robot monitoring can be seen.

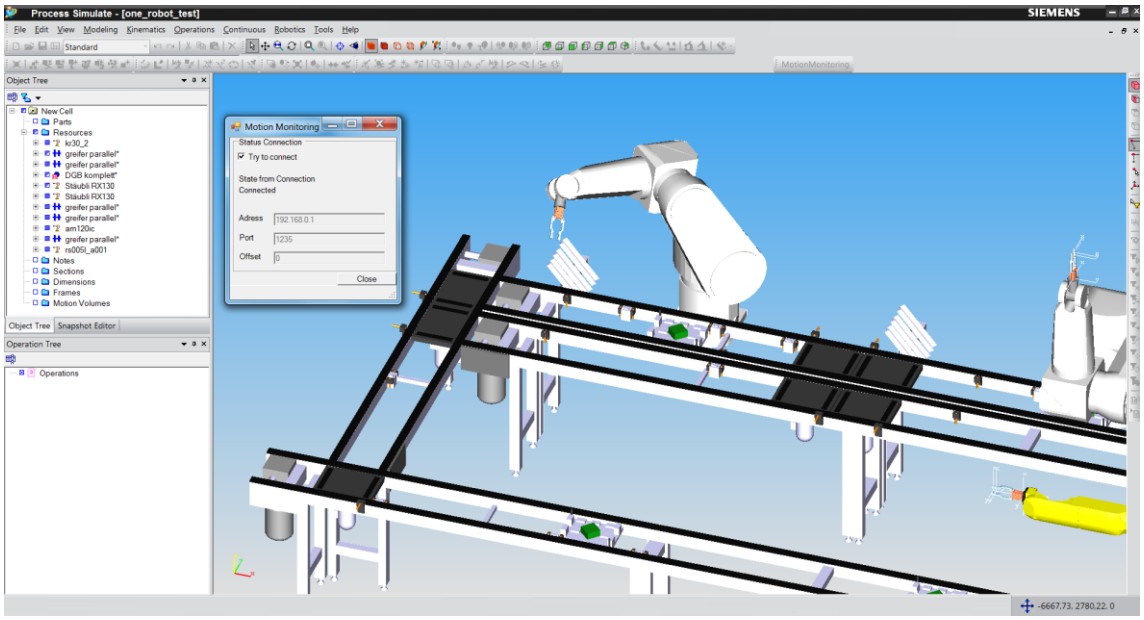


Figure 5.2: Monitoring one robot from real world

5.3. Test conveyor system with one carrier

This test is operated conveyor system with a single carrier. It tests the model implemented in OpenMHS and the first attempts an error is discovered. After correcting small errors in the block model starts again the monitoring application.

As seen in Figure 5.3, the carrier moves through the transport system. The position is not exact, there are small differences. This is due to the approximate calculation is made of the position when the carrier enters a sector.

After adjusting well the lengths of each segment, position gets closer to reality.

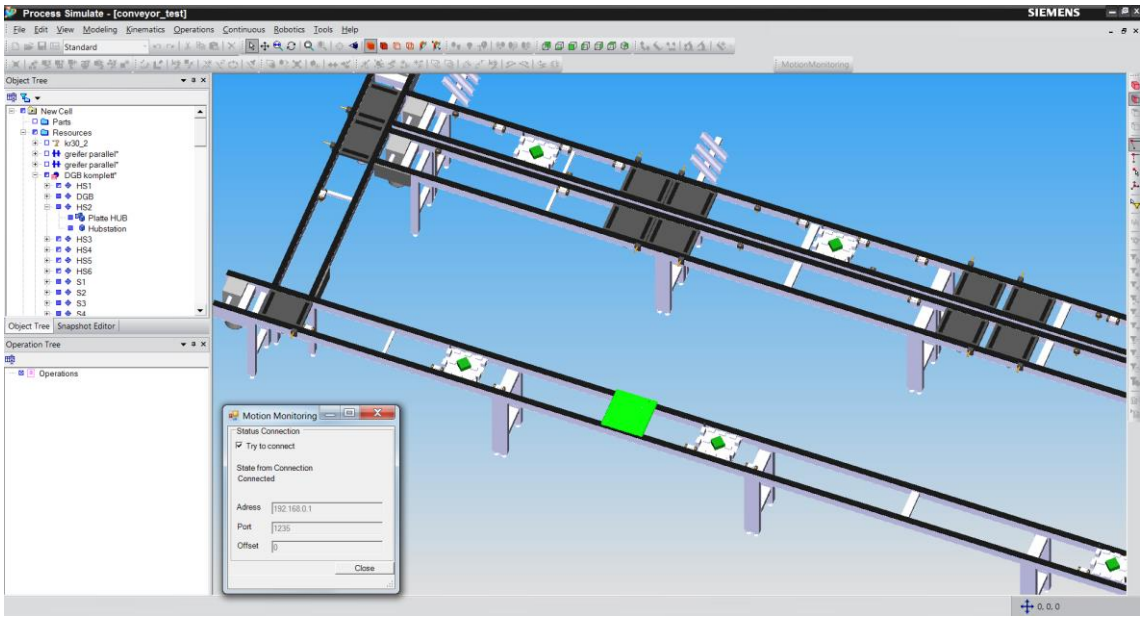


Figure 5.3: Monitoring the conveyor system with one carrier

5.4. Test conveyor system with several carriers

After testing the transport system with one carrier and calibrate the position, is necessary to monitor more carriers at the same time. In early tests are discovered a small programming error that caused to stop some carriers. After solving the problem everything is working properly and the same results of one carrier are obtained.

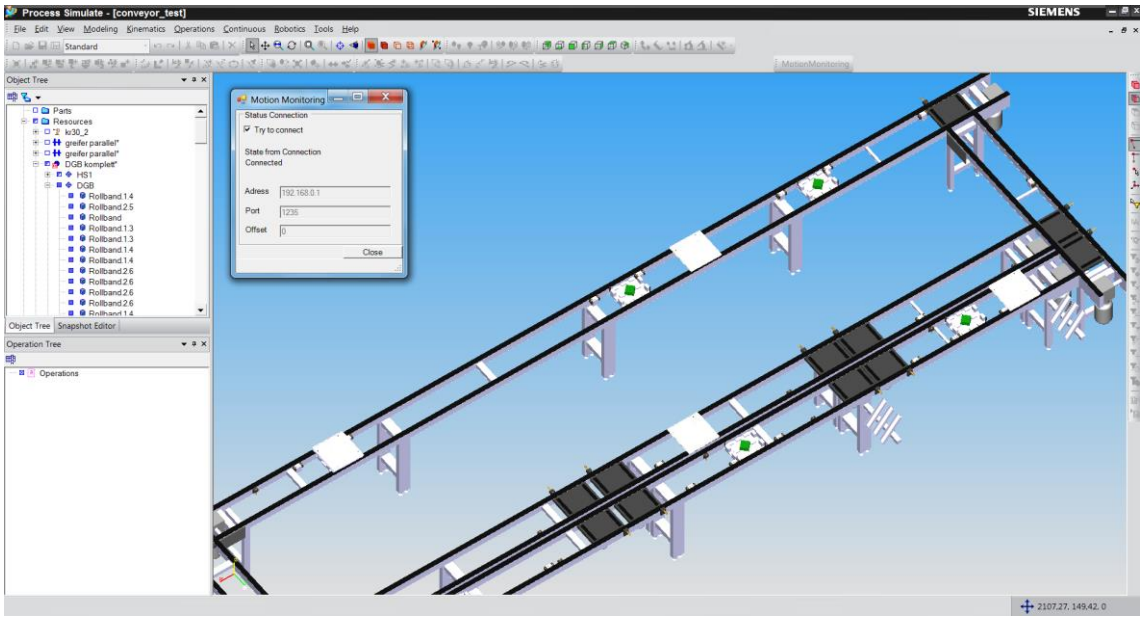


Figure 5.4: Monitoring the conveyor system with several carriers

5.5. Test conveyor system and robots

In the last test, the conveyor system and a robot are operated in the same time. Robots and carriers are monitored simultaneously; it is obtained refresh times of the order of tenths of a second. The tests of monitoring application are finished. In Figure 5.5 a screenshot of the test is shown.

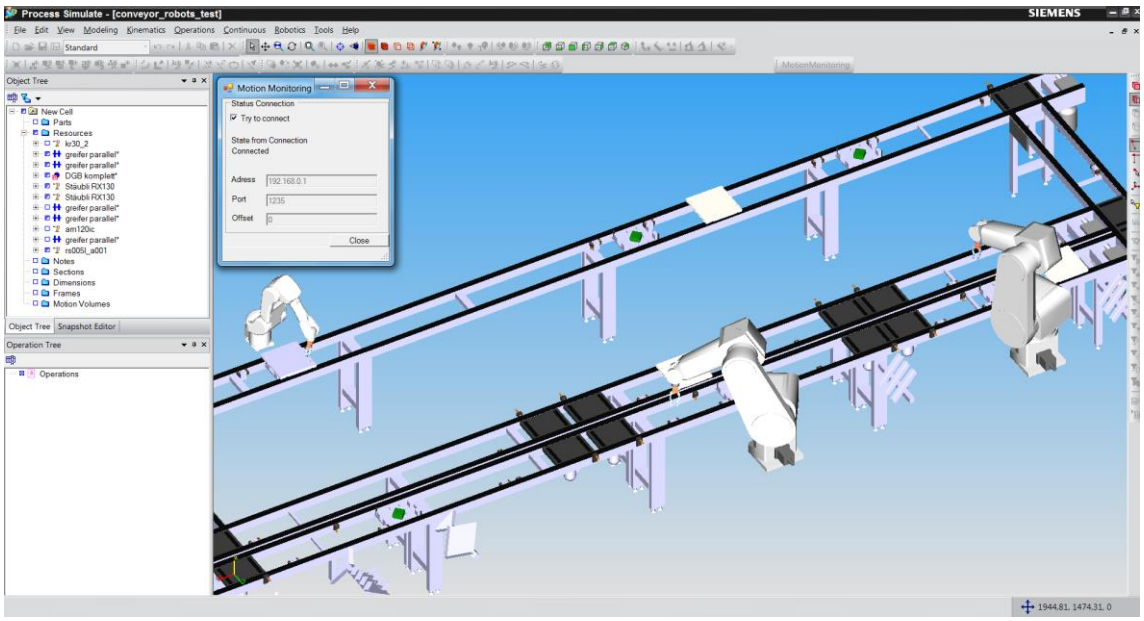


Figure 5.5: Monitoring the conveyor system and robots

6. Conclusion and possible improvements

During implementation of this master thesis I used many different technologies. First, I learned a lot about the simulation tool Process Simulate. It is a very comprehensive tool to simulate complex industrial processes and allows creating new features.

To develop the monitoring application, I had to investigate many different areas. Learn more about industrial robots, operation and programming, also conveyors systems and especially the PLC-Software OpenMHS by Dipl.-Ing. Martin Nardmann.

Monitoring the conveyor system was a challenge due to the complexity of the system and the lack of information in order to implement it. It was necessary to develop a model inside the software PLC.

To develop the plug-in for Process Simulate had to learn more about object-oriented programming. The entire application was developed in C# with the help of the Tecnomatix .NET API supplied by Siemens.

Also deepen knowledge network sockets and protocols to understand the operation of the client-server communication. Once defined the concepts was possible to implement the proposed solution.

Now is possible from any computer with Internet connection and Process Simulate, visualize real-time motions of robots and carriers that are being monitored.

After observing the result has been very rewarding, as they have fulfilled all the objectives set at the beginning and the application is fully functional to monitoring the robotics lab at the university.

Besides, some improvements Could Be Performed in order to include a larger number of robots, since at the time of writing this thesis could only be monitor three robots, the two Stäubli and Kawasaki.

The monitoring of the conveyor system can also be improved. At the moment it is not possible to distinguish between carriers, but the conveyor system has RFID readers that could be used to read and identify carriers.

Finally, this work has been very exciting because of the challenge posed do that with new technologies that were never used. I think it's a very important complement to my studies in Electronics and Automation Engineering, since in most of the topics discussed in this paper has not tried during the bachelor.

7. References

[1] D. Makofske, M. Donahoo, K. Calvert. *TCP/IP Sockets in C#: Practical Guide for Programmers*. May, 2004

[2] 2015 Microsoft Corporation. MSDN Library. *http://msdn.microsoft.com /en-us/library/ms123401(v=MSDN).aspx*. 2015-03-02

[3] Siemens PLM. *Tecnomatix.NET Manual*.

[4] Siemens PLM. *eMServer Data Importing Student Guide,* January 2008.

[5] Siemens PLM. *Process Simulate Basic Student Guide*, January 2008.

[6] Soft PLC corporation. *www.softplc.com*. Texas. 2015.04.12

[7] Kawasaki robotics, INC (Web). *www.kawasakirobotics.com*. USA. 2015.05.12

[8] FANUC robotics, (Web). *www.fanucrobotics.com.* 2015.04.21

[9] Boschrexroth, (Web). *www.boschrextoth-us.com*. USA. 2015.05.12

[10] Adept Technology (Web). *www.adept.com* 2015.04.18