TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

SAMPO NYYSSÖNEN

CONTROL AND COMMUNICATION OF PEELING MACHINE SIMULATOR

Master of Science Thesis

Examiner: prof. José L. Martinez Lastra
Examiner and topic approved on 2. May 2018

ABSTRACT

SAMPO NYYSSÖNEN: Control and communications of peeling machine simulator
Tampere University of technology
Master of Science Thesis, 56 pages
April 2018
Master’s Degree Programme in Automation Technology
Major: Factory automation and industrial informatics
Examiner: Professor José L. Martinez Lastra

Keywords: Peeling machine, real-time simulation, PLC, control, communications, TCP/IP socket

In this thesis communication between real-time peeling machine simulation and Programmable Logic Controller (PLC) is created. The PLC uses a program which is originally used to control a conventional peeling machine. Thesis describes needed changes in the control program to make the program function with the simulator. The control on the simulation should work so that the simulation imitates the actions of the real machine. Finally, it is shown how this peeling machine simulation could be used from various aspects. One of the goals was to create control for simulation that can be represented on a fair.

The peeling machines are used to peel a 1-4mm thick veneer ribbon from a block of wood. The peeling machine moves the block from block conveyor to lathe spindles in the optimal position to maximize the veneer output. During the peeling various axes are moved simultaneously to support the peeling against forces caused by knife cutting and to improve the quality of the veneer.

The resulting interface uses TCP/IP (Transmission control protocol over internet protocol) communication over Ethernet. The resulting communication interface to simulation initiates various movement commands for each of the axes. The feedback data contains axis status information and relevant sensor status information. Each of the communication clients attempt to send data to its participant every 10ms. The communication interface works reliably and rapidly enough for logic program need and for interactive use. For future work, the simulation environment could be used to test and develop peeling machine control program.

TIIVISTELMÄ

SAMPO NYYSSÖNEN: Simuloidun viilusorvin ohjaus ja kommunikointi
Tampereen Teknillinen Yliopisto
Diplomityö, 56 Sivua
Huhtikuu 2018
Automaatiotekniikan DI-tutkinto-ohjelma
Pääaine: Factory automation and industrial informatics
Tarkastaja: Professor José L. Martinez Lastra

Avainsanat: Viilusorvi, reaaliaikainen simulaatio, PLC, ohjaus, kommunikaatio, TCP/IP socket

Tässä työssä luodaan kommunikaatio reaaliaikaisen viilusorvi simulaation ja ohjelmoitavan logiikan välille. Logiikka käyttää ohjelmaa joka on alun perin tarkoitettu oikean viilusorvin ohjaukseen. Tarvittavat muutokset logiikan ohjelmaan, jotta logiikka toimisi simuloidun järjestelmän kanssa kuvataan tässä työssä. Simulaatiomallin ohjauksen tulee toimia siten että simulaatio imitoi oikean koneen käyttäytymistä. Lopulta näytetään kuinka simulaatiota voisi hyödyntää useista eri näkökulmista. Yksi tavoitteista oli luoda simulaatiolle ohjaus, jotta sitä voitaisiin esitellä messuilla.

Viilusorveja käytetään sorvaamaan 1-4millimetrin paksuista viilumattoa puupöllistä. Viilusorvi liikuttaa pöllin pöllikuljettimelta sorvin karoille optimaaliseen asentoon, maksimoidakseen viiluntuotannon. Sorvauksen aikana useita eri akseleita liikutetaan samanaikaisesti, tukeakseen viilun leikkauksesta aiheutuvia sorvauksen voimia vastaan ja parantaakseen viilun laatua.

Luotu rajapinta käyttää TCP/IP-tiedonsiirtoa (Transmission Control Protocol / Internet Protocol) Ethernetin kautta. Luotu kommunikaatiorajapinta simulaatioon panee alkuun erilaisia liikekomentoja kullekin akselille. Takaisinkytkentädata sisältää akseleiden tilainformaatiota ja tarpeellisten antureiden tilatiedon. Kaikki kommunikaation osanottajat yrittävät lähettää dataa yhteyskumppanilleen 10ms välein. Luotu kommunikaatiorajapinta toimii tarpeeksi nopeasti ja luotettavasti logiikkaohjelman tarpeeseen ja interaktiiviseen käyttöön. Jatkossa simulaatioympäristöä voisi käyttää sorvin logiikkaohjelman testaukseen ja kehitykseen.

PREFACE

I would like to thank Raute for giving me interesting subject and lots of trust and support to work on. I would like to thank Mevea company for making such interesting thesis possible. I want to give personal thanks to Heikki Korpilahti from Raute and Karli Kund from Mevea for good co-operation. Additionally, I want to thank both of my supervisors Antti Pennanen in Raute and José Lastra in TUT for support and trust in the finalizing the thesis. Finally, I want to thank my parents and relatives for support and constant pushing to go forward.

Lahti, 05.04.2018

Sampo Nyyssönen

CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

CAD	Computer Aided Design
DB	Database
RMC	Real-time Motion Control
SFD	Start Frame Delimiter
CRC	Cyclic Redundancy Check
STL	Statement List
SCL	Structured Control Language
LD	Ladder diagram
FBD	Function block diagram
PLC	Programmable Logic Controller
FB	Function block
LVL	Laminated Veneer Lumber
I/O or IO	Input - Output system
CPU	Central Processing Unit
PC	Personal Computer
HMI	Human Machine Interface
TCP	Transmission Control Protocol
IP	Internet Protocol
ISO	International Standardization Organization
OSI	Open Systems Interconnection
MBD	Multi-body dynamics
PID Controller	Proportional-Integral-Derivative Controller
TIA	Totally Integrated Automation

1. INTRODUCTION

Industrial automation has been developing on fast phase for past four decades. The driven factor for fast development has been technological development, higher expectations from the users and maturity of the industrial processing technologies (Mehta and Reddy, 2015). Modern production plants within manufacturing industries are continuously increasing in complexity of the process and the product (Shahim and Moller, 2016). The early test of the engineered automation solution is critical task to reduce the risks during real commissioning and plant operation (Oppelt and Urbas, 2014). It has been proposed that virtual commissioning can reduce the ramp-up time of the actual commissioning (Hloska and Kubín, 2014; Oppelt and Urbas, 2014; Reinhart and Wünsch, 2007; Shahim and Moller, 2016).

There were few parties working on this project: mechanical engineer working on multibody mechanical models, group from Mevea Corporation working on motion controls of the hydraulics, wood process model and visualizations of the model, Mevea also provided the simulation software for the project. In this thesis the logic control and communications between simulation and logics will be discussed.

Objectives of this thesis include the creation of functioning control interface for logic controller to communicate with simulation. Another objective is to modify the original control program to function with the simulation. Final objective is to find possible applications for such simulation environment.

This work can be used as reference for building a real-time simulation control interface. This work will also work as reference for possible user of the program. Information on practical challenges and programming decisions made in the project are also discussed.

1.1 Motivation

In this thesis control interface for real-time simulation of Raute smart peeling machine is created. The simulation model is created using Mevea modeller which provides tools for creating multibody system model for real-time simulation. The simulation uses so called multibody dynamics (MBD) method. This provides means for simulating kinetics, dynamics and forces of multibody systems and creates a visualization of the simulation. This multibody system can be simulated and controlled in real-time. Real-time simulation provides valuable information of the dynamics and control of the system.

Simulations have been traditionally used to research the optimums of sub-processes. These kinds of simulations need deep understanding in the specific sub-process to build mathematical models of the process. For these simulation environments such as Matlab Simulink can be used. These simulations are run *offline,* which means that modelled process events are calculated for certain amount of time and the results of the simulation can be seen afterwards. This way simulating 10 seconds of process can take from 2 seconds to days to calculate. The counterpart for offline simulation is real-time simulation. Real-time simulation refers to computer model that can be executed in same rate as actual *real-world* time. This means that process that would take 10 seconds would also take 10 seconds to simulate. It is easy to understand that modelling such simulation must be simplified to the point that the real-time constraint is achieved.

The Mevea solver, used in this case, calculates the mechanical process in real time. The real-time simulations provide comprehensive data from the system dynamics, which can be used in product development. The real-time simulation environment is claimed to be the next generation for product development and product lifecycle management.

The primary motivation of this work was to create an interactive presentation for wood processing fair called *Ligna*, kept in Hannover Germany 22-26.5.2017. After the fair the work could focus on actual applications of simulation.

1.2 Justification

For the MBD simulation to function as a real time controllable system, the simulation system need to be controlled somehow. The control is desired to be done by the PLC program which would be normally used for the real system. This hardware-in-loop setup with original control program should make the simulated system function as real system would. This hardware-in-loop simulation should allow the testing of logic program through simulation environment. In order to have this hardware-in-loop to control the simulation, the communication between the hardware and the simulation is needed to be defined. For a company to gain benefit of such a virtual machine, the virtual machine applications are studied.

1.3 Problem

The problem that had to be solved this work was to make programmable logic controllers (PLCs) to exchange data with the simulation in rapid and reliable manner to make control system working. Another problem that is solved in this project, is to make peeling machine logic program given by Raute corporation to communicate and cooperate with simulation in a way that the original control software should be modified as little as possible. Additional problem that had to be solved was to study possible uses for virtual machines and consider if they could be applied in this scope.

1.4 Scope

This thesis focuses only on PLCs communicating with simulation and focuses on functional communication interface. Other focus is to use the interface to control the virtual machine using as realistic logic controls as possible. Realistic controls in this context means that the actions in each stage of operation should imitate the operation of the real system. Final focus of the thesis is to research possible applications for the simulation system. The setup used in the thesis uses so called "hardware-in-loop" simulation since the PLC hardware is present and not simulated.

The hardware in the project consists of two Siemens open controller PLCs (s1500 series), Siemens HMI and conventional computer used for simulating. Some switches, network switch and power supplies are also needed for powering the components. Programming is done in TIA v14 environment. The hardware setup used is selected as it is for few reasons; Raute corporation has interest in testing out migration to TIA v14 programming environment for lathe programs, the logics are very compact and can be easily fit inside the simulator bench setup and no excessive electronic cabinet is needed.

1.5 Limitation

The work on this thesis is limited to this specific peeling machine model provided by Raute corporation, although some aspects of the control and communication interface can be used for similar machines models. The controllers are limited to be Siemens s1500 series open controller PLCs, because the open controllers are easy to fit inside small space within the small electric control panel boxes and the original code would be easier to be migrated from siemens environment to another siemens environment. The used programming languages are limited to the ones supported by the Siemens TIA environment, since the Siemens' open controller PLCs can only be programmed by those languages. The simulation is limited to be provided by Mevea corporation and the communications are only considered to the specific application provided by Mevea, however the control interface on PLC side doesn't restrict it only to work with Mevea simulation.

The control of the system that is considered in this work limits to the PLC control that initiates the commands. The actual motion control algorithm is not considered in this thesis. Neither the mechanical modelling or study of real-time multibody mechanical system simulation is not considered.

From the program logic all the safety features are removed from the system since they are not considered interesting in this context. This includes all the pneumatic safety bolts which hold the axes in position when hydraulics is off. Also, the centring of the block is limited to so-called pre-centring, which only roughly centres the block to lathe spindles.

1.6 Outline

Chapter 2 contains literature review on the existing technologies that are used in this work. These technologies are divided in three categories. First literature review on peeling machine as part of plywood industry. Second, TCP/IP over Ethernet protocols are studied. Finally, the virtual machine applications are studied in this context.

Chapter 3 describes the designed structure to make control and communication to work with simulation. The communication is interface structure is explained in detail such that each data field and its purpose in the structure is explained. Also, the commands that are desired to be initiated are explained along with the purpose of the commands. Later in the chapter the needed modifications to original program are discussed along with how the modifications should be done. Finally, chapter 3 describes how the various virtual machine applications could be applied in this context.

In chapter 4 the results of the designed control interface are described. The results are considered through the factors such as protocol stack efficiency, bandwidth usage and average interval between communications. Lastly, in the chapter the control program and the correct functionality are also considered through user experience and through exemplary command.

In chapter 5 the work is concluded by discussing what were the initial goals and how well they were achieved. Chapter also proposes further work based on this thesis and on the simulation environment as whole.

2. LITERATURE REVIEW

For literature review the plywood manufacturing is first studied in chapter 2.1. The applicable communication protocols are studied in chapter 2.2. Chapter 2.3 studies the virtual machine applications. Chapter 2.4 sums up the state of the art of these three fields.

2.1 Plywood manufacturing

Plywood production is done in process-like steps from logs to veneer and from veneer to plywood sheets. In the manufacturing process the yield is tried to be maximized by using all the material as efficiently as possible. Also, the strength and qualities requirements must be met in each of the process steps. (Varis, 2017)

The modern production process is long and complex, but latest machines have been automated so that operators may focus mainly on process monitoring and ensuring fluent process. (Varis, 2017)

Plywood production process is a process of turning a block of wood in to a sheet of plywood or LVL. The outline of plywood production can be seen in Figure 1, however it should be noted that some of the steps may be done in different order. The average yield of each process step can be seen in Table 1. It should be noted the yield drop in from peeling to drying. Just showing the importance of good peeling process in terms of total yield.

Table 1. *The average yield of main process steps in modern plywood plant. (Varis, 2017)*

Step:	**Birch**	**Softwood tree**
Log input	100 %	100 %
Debarked logs to cut	89 %	90 %
Cut logs to peeling	85 %	89 %
Wet veneer to drying	57 %	66 %
Veneer from drying	50 %	58 %
Veneer to lay up	43 %	54 %
Compressed veneer	41 %	51 %
Ready veneer	35 %	45 %
Yield	35 %	45 %

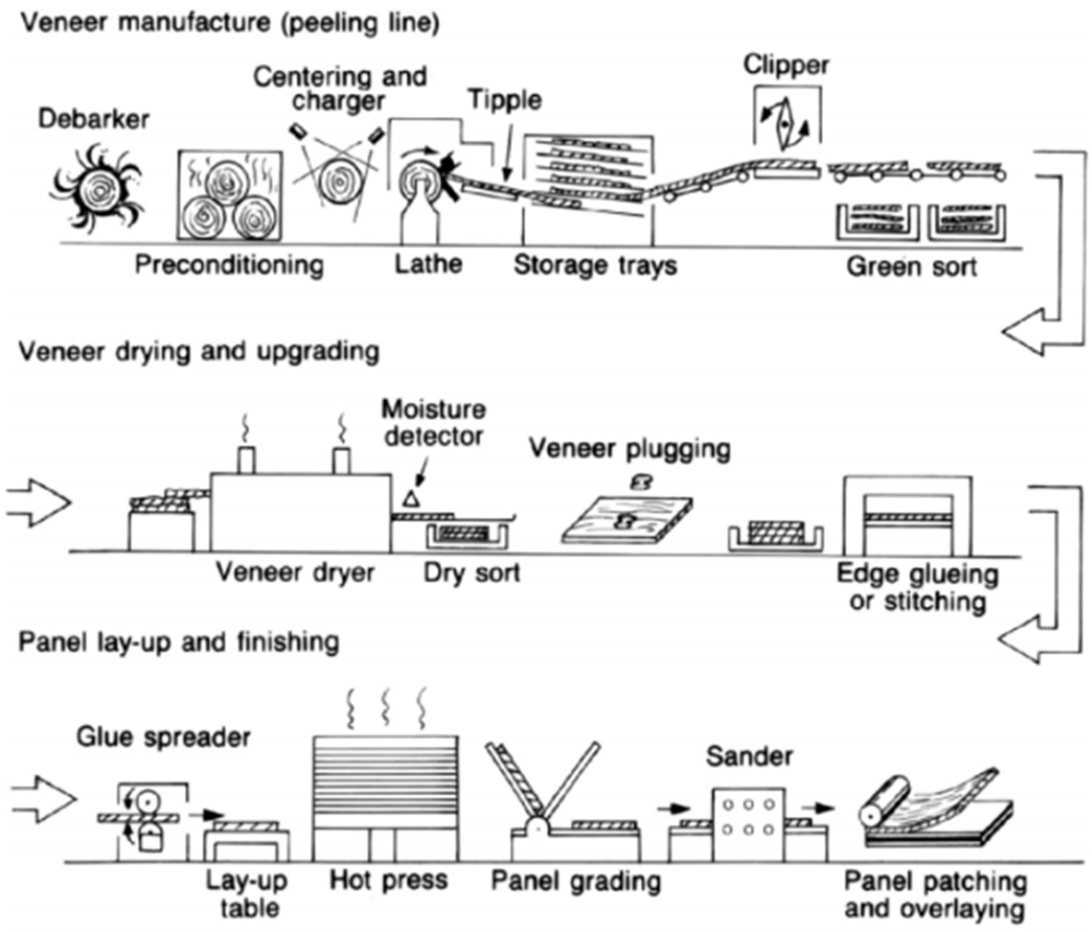


Figure 1. *Outline of plywood production (Shi and Walker, 2006)*

Early mills mid 1960's in the Pacific Northwest of the United States made plywood from flawless, old-growth, large diameter logs (>1,5m) of Douglas-fir, which accounted 90% of North American plywood production. The declining availability of large, high quality logs has caused the need for use of smaller less efficient blocks in plywood manufacturing. The less efficient, such as birch logs averaging 200-250mm are peeled down to 60-65mm in diameter. (Shi and Walker, 2006)

As the log diameters were reduced, they had to be peeled much more efficiently. This requirement in efficiency caused the development of automatic XY-charger (XY originating from the horizontal X and vertical Y movements of the charger) late 1980s'.

The veneer made from smaller diameter logs needs much repair work e.g. patching and jointing represents about third of the work input. To add value two thirds of plywood is processed by scarf-jointing into giant panels, by preservative treatment, by overlaying or by adding a thick textured phenol-resin coat providing a non-slip pattern for flooring. (Shi and Walker, 2006)

Peeling line is composed of XY-charger, veneer lathe, veneer conveyors, veneer clipper, veneer stacker and stack conveyors (Varis, 2017). Veneer peeling line includes operations from after log is cut to block and to drying of the veneer. The peeling line operations can be split in to following steps: move the block to feeder, move the block to centring device, centre the block, move the block from centring to lathe spindles, round-up of the block, peel the block, collect the round-up waste, cut the veneer ribbon and the regular maintenance of lathe knife. Depending on application the veneer ribbon can be dried as one piece directly after the peeling or it can be first cut in to pieces after which the pieces are conveyed to dryer. (Koponen, 1998)

The veneer used in plywood and laminated veneer lumber (LVL)-industry is produced using veneer lathe. In the lathe the block of wood is rotated with constant radial velocity simultaneously as the knife carriage is moving towards the spindle centre. The speed of the knife carriage is relative to rotation speed and thickness of veneer: Knife carriage should move 1,5mm forward each spindle rotation if 1,5mm is the nominal thickness of wet veneer. The nominal thickness of such veneer as dry sheet should be around 1,4mm. The most common peeling thickness of birch veneer is 1,5mm. The goal of peeling process is to gain maximal yield of block while maximizing the capacity of the machine. (Varis, 2017)

XY-Charger plays a crucial part in the plywood factory material consumption. To gain the maximal amount of veneer out of the wood, the block must be centred and placed optimally to lathe spindles. The centring happens in the machine before the lathe, which is called the XY-charger. First is calculated so called pre-centring position, which gives approximate diameter of the block of both ends of the block, the diameter is used to calculate the retrieve positions for XY spindles. Once the XY spindles have retrieved the block, XY spindles move the block under laser scanner, where they rotate the block one round to gain perfect profile of the block. The profile is used to fit maximal spiral inside the wood which is used to place the piece perfectly to the lathe spindles. As the optimal position is known, the spindles are rotated and moved so that transfer arms, which move the block from XY spindles to lathe spindles, move the block to lathe in perfect position. This way minimal amount valuable surface veneer goes to waste and maximal count of full sheets is yielded. (Varis, 2017)

The XY-Chargers have been developed from first laser scanning techniques, which used point lasers with 300mm intervals, to curtain laser, which scanned with 25mm intervals to HD laser fans, which have 3mm interval between lasers. The achievements in laser scanning techniques have greatly improved the yield of the block. Basic principle of centring the block can be seen in Figure 3a, the block is first roughly centred by so-called pre-centring after which the centring is done by spinning the block in centring spindles one rotation to get full 3D scan of the block. Based on this information the block can be placed in optimal position to lathe spindles. (Varis, 2017)

The improved measurement accuracy also provides more accurate position for knife carriage. This position is the position where it waits for the next block. The position should be as close as the block as possible while not touching the block, this way knife instantly starts peeling the block instead of “peeling air” as the knife carriage closes to block. (Varis, 2017)

Latest XY-Chargers can also be combined with autocalibration. The autocalibration is used to verify the measurement data of the scanner. The block diameter is scanned again during the first rotation in the lathe spindles. This data can be used to find error caused by wear of XY-charger components, and the error can be eliminated. It has been studied that effect of autocalibration on yield is around one percent. (Varis, 2017)

The quality of the veneer will affect the following steps of the process and the quality of the final product. The qualities of the veneer that can be affected by peeling process are, toughness, smoothness of the surface, straightness of face of the sheet and compression, just to name a few. For example, if the ply does not have straight or smooth surface the gluing process would need more glue between the sheets. Despite the requirements in the quality of the ply the production speed must not suffer and the yield of the block should be maximized. (Varis, 2017)

Plywood is made by gluing together thin sheets of wood on top of each other with crossing directions of grain. The usability of plywood sheet is mainly decided in veneer peeling and the following steps. (Koponen, 1998)

The peeled ply is given many difficult requirements. Some of the requirements are defined by the quality of the final product and some are set by the following steps of the manufacturing process. The requirements set by quality of the final product can be based on appearance, physical measures such as thickness and the strength of the veneer. Requirements set by following manufacturing process are that ply must not be short or long-ended for it to dry in a uniform manner. The gluing sets high standards for peeling: the ply must have smooth surface and uniform in thickness so that glue spreading is successful, and the usage of the glue is minimal. Also the ply has to endure all the handling in the following steps so ply strength has great economical effect. (Koponen, 1998)

Process of rotary veneer cutting is essentially to cut perpendicular to grain with the knife lying parallel to grain. The block is centred between two spindles on lathe and then rotated against the knife. In so-called spindle-free lathe there is no centring used as there are no spindles where the block should be centred to, the block is dropped on powered back-up devices and the block is rotated by backup devices instead. The knife extends the full length of the block. As the block turns and knife moves towards the block, thin continuous ribbon of veneer is cut through the gap between the flatbar (or some cases rotating nosebar) and face of the knife. The quality of the veneer is determined to great extent by the precise set up of the lathe. The complexity of correct knife setup can be seen in Figure 2. In the figure the spindles (B) are rotating the block (A) against the knife (L) and flatbar

(F) which are moving towards the centre of the block at rate of peeling thickness each rotation. (Shi and Walker, 2006)

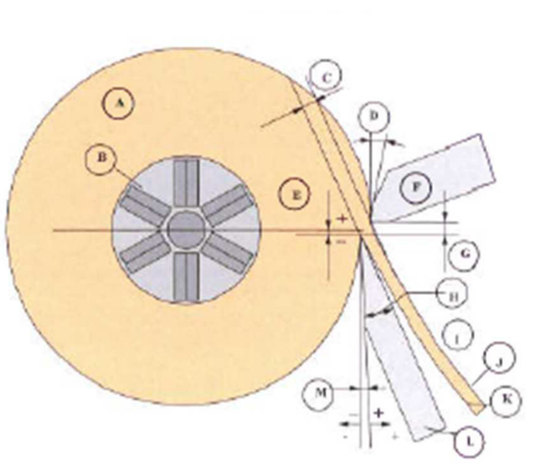


Figure 2. *Knife peeling variables (Varis, 2017)*

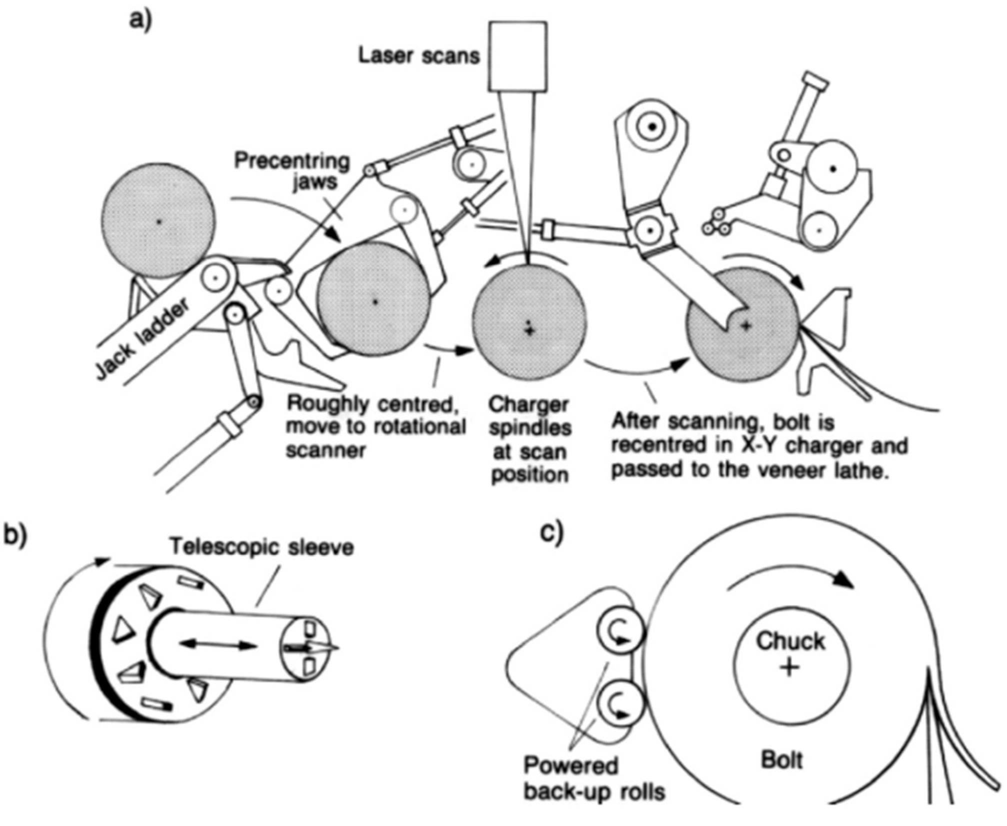

a)
Laser scans
Precentring jaws
Jack ladder
Roughly centred, move to rotational scanner
Charger spindles at scan position
After scanning, bolt is recentred in X-Y charger and passed to the veneer lathe.
b)
Telescopic sleeve
c)
Chuck
Bolt
Powered back-up rolls

Figure 3. *Some features of modern lathes (courtesy of Coe Manufacturing Co., Painsville, Ohio) (Shi and Walker, 2006)*

The knife carriage is moved towards block at the rate of peeling thickness per each rotation of the block as the peel is forced to go through small knife gap, the knife gap is controlled by nosebar. Depending on wood type the knife gap is 10-15% smaller than the desired thickness of the veneer. The compression causes veneer to be more solid and higher quality. The compression of the veneer by nosebar causes a force pushing the block away from the knife. This force causes block to bend so that the block is thinner from middle. To counteract this force backup rolls are used. The backup rolls support the peeling from behind the block, causing opposite force towards the knife, preventing the block to bend. If there is too much pressure from backup rolls however, the block may bend the other way, causing veneer to be thicker from the middle. Uneven thickness of veneer is considered to be poor quality. The rolls that are used as backup device, are powered to rotate. This rotation is to help the spindles rotate the block, or sometimes even rotate the block alone with rotating nosebar. The rotation is needed to counteract the knife cutting force. This knife cutting force can cause so called *spinout*. Spinout causes the spindle to start “drilling” in to block as the knife force causes the spindles to slip from initial grip position. This happens when knife peeling torque exceeds the supporting torque of spindles and back-up rollers. (Koponen, 1998; Shi and Walker, 2006)

2.2 Industrial communication systems

Computer network is a group of computers that share information over communications link (Delamer and Martínez Lastra, 2007). These networks in control systems are used to connect devices that are connected to accomplish a certain task. The tasks require exchange of information between devices to be completed. (Alani, 2014) These devices can be computers, logics, printers, frequency converters, smart-sensors and smart-actuators. The advances in electronics miniaturization and lowered costs of networking infrastructure enable interconnecting even the smallest devices to network (Delamer and Martínez Lastra, 2007). Two systems can be called interconnectable if they share the same standard. However this interconnectability does not guarantee that the systems cooperate. (Wilamowski and Irwin, 2011)

Within the industrial world the networks are called industrial networks as there are various networks in use, each of which are designed for different types of tasks. Some of the networks provide fast and reliable transmission for small amounts of data and some provide high rate of data but unreliable timing. However, the Ethernet based connections promise a single technology that covers the needs of all requirements of industrial applications (Delamer and Martínez Lastra, 2007).

The communication between devices have many similarities to the communication among human. Instead of usually hard-to-understand human logic, controllers have CPU that make the decisions. The CPU is programmed to follow set of instructions, much like human following cooking recipe. However, the controller can’t finish the task by itself, it needs sensors to tell the state of the process and actuators to affect the state of the process, and usually the controllers need to exchange data among other logics to finish the task. To achieve this data exchange with various devices, controllers need various connection mediums.

The information that is exchanged between devices is sequences of bits that are constructed and interpreted by programs that are ran on these devices. In computer networks these bit sequences are called *packets*. Packet contains control information that the network uses to do its job and sometimes also includes user data. Such control information can be packets destination and senders address. (Donahoo and Calvert, 2009)

"The term protocol denotes a set of rules that govern the communication on the same level" (Zurawski, 2015). These rules tell how packets should be structured; where the destination information is in the packet and how big the packet is as well as size of the actual message. A protocol is usually designed to solve a specific problem using given capabilities, however some protocols can be very versatile. As example *internet protocol* (IP) solves the problem of individualizing each device in the network and finding each device in the network. (Donahoo and Calvert, 2009)

Implementing actual network for information transmission requires solving large number of problems. To keep things manageable, different sets protocols are designed to solve different sets of problems. TCP/IP is such set of protocols, often this kind of set of protocols can be called a *protocol suite*. (Donahoo and Calvert, 2009)

If message or a *packet* is wanted to deliver over network, common protocols among participants are needed. The protocols used for specific communication is called *protocol stack*. The generalized stack for communication is defined in the OSI model, name originating from *open system interconnection* model.

The OSI model was introduced by ISO foundation. The OSI model was created to counteract the lack of interconnectability among different fieldbus standards. To simplify the handling of complex task of data communication it was decided in the committee to partition it to hierarchical layered model. The significance of the OSI model and its value for practical use came from the consistent implementation of three essential concepts; *protocol*, *service* and *interface*. The definition of protocol has been described in previous chapter. The *service* represents any service that's made available by one layer to layer above it. The layer below is called service provider and layer above is called service user. OSI model only defines their functionality, not how they are implemented. The *interface* is between every two layers, this specifies which services are provided by lower layer to upper layer. (Zurawski, 2015)

In the OSI model the protocols in the stacks are called *layers*. Seen from the logical perspective, each of the layers communicates horizontally with corresponding layer in the other end. Layers don't communicate vertically with layer above or below. They take request from upper level layer, process and possibly add its own overhead to it and make request to pass the message to lower level layer. This happens until the lowest, physical layer, is reached and the data transferred. Upper level n+1:th layer is called service user for n:th layer, whereas the n:th layer is called service provider for the upper layer, this applies to all layers except for the physical layer, illustrated in Figure 5. (Wilamowski and Irwin, 2011)

The packet can go through hubs and switches where the three lowermost layers are decrypted to read information of the receiving end, then decrypted again (see Figure 4). Once the message has been received in the other end, the packet is passed from "down to up". The protocols parse their own overhead from the message and pass it onward to upper layer until application layer is reached, where the data packet is read. (Wilamowski and Irwin, 2011)

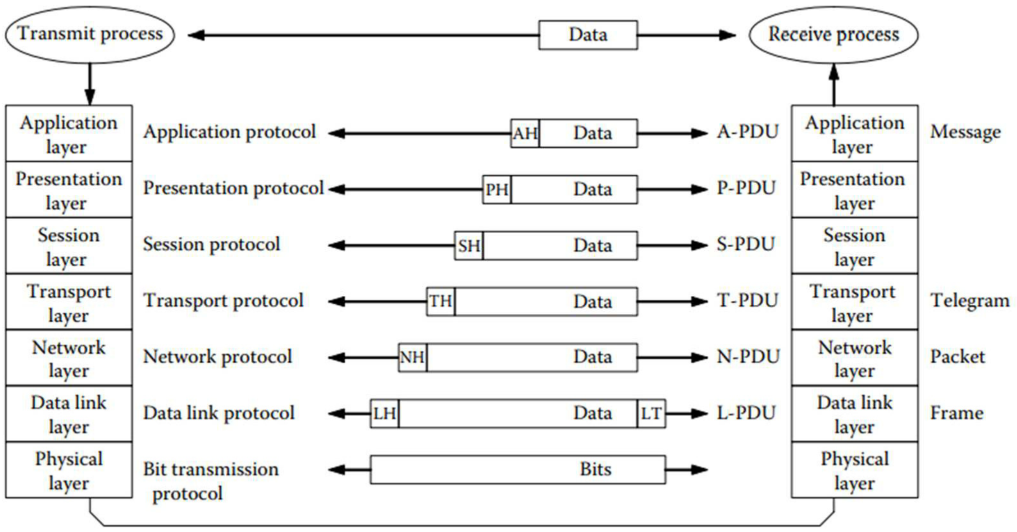


Figure 4. *Communication in OSI model (Zurawski, 2015)*

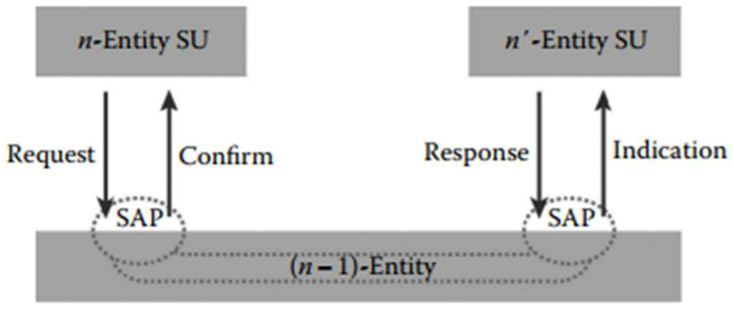


Figure 5. *Predefined service primitives (Wilamowski and Irwin, 2011)*

2.2.1 Ethernet

Ethernet covers the layers 1 and 2 of the OSI model. It defines the physical transfer characteristics, such as type of cable, connectors and signal modulation. Ever since the first invention of Ethernet by Xerox in 1970s the physical layer has been evolved to provide faster connection (higher baud rate), but most implementations still utilize the same data link layer implementation that was standardized in 1982. (Delamer and Martínez Lastra, 2007)

Standard Ethernet is a so-called packet switching network (not to be confused with switched network later in chapter). This means that data is transmitted in smaller packets or frames. Each of which contain all required information, such as receiver and sender address. The packet can be from 64 to 1526 bytes long. The Ethernet / IEEE 802.3 frame is composed of 7 parts (Figure 6) described in detail below:

1. Preamble, 7 bytes for synchronizing sender and recipient (1010…1010)
2. Start Delimiter, 1 byte for signalling that message starts, ends with two ones (10101011)
3. Destination address, 6 bytes for destination MAC address
4. Source address, 6 bytes for sender MAC address
5. Length / Type, 2 bytes describing frame length in IEEE 802.3 or Type in case of Ethernet II frame (0x0800 for IPv4 datagram)
6. Data block, 46-1500 bytes of user data, which contain the data and headers of higher layer protocols. If the data is less than 46 bytes, the data block will be "padded" to be 46byte long.
7. Checksum, 4byte checksum is generated to check the correct transmission of data.

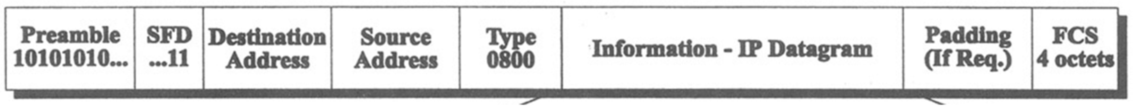


Figure 6. *Ethernet frame split in parts. (Miller and Cummins, 2000).*

After transmission of one packet on wire there must be a gap of 12 bytes called interframe gap (IFG for short, also called interpacket gap by some sources) before sending another packet on wire.

The packet parts apart from data block are called "overhead" and data block is the actual message. The DIX Ethernet frame format is the same with the difference that instead of 2 bytes of length there is be 2 bytes for data type (Pigan and Metter, 2008).

The MAC (Media Access Control) address is used for reaching stations in the network. A MAC address is given by hardware vendor to each Ethernet interface. First 3 bytes of the MAC address are used to identify the vendor and the rest can be freely assigned by the vendor. This address is unequivocal worldwide. MAC address is also known as hardware, station or Ethernet address. This address usually coded in hardware and usually cannot be changed to avoid address conflicts in a network. (Pigan and Metter, 2008)

In the physical layer the Ethernet needs to convert the bits to electrical signals on the coaxial cable. In the original 10BASE5 version of Ethernet which transmitted 10Mbit/s used Manchester encoding to encode the original data to signal, the Manchester encoding uses rising and falling edges of signal to represent logic 1 and 0, requiring frequency double of the baud rate (i.e. 10Mbit/s requires 20MHz frequency). Manchester encoding solves the issue of the long "silences" during long sequences of ones or zeros, these long sequences are easily misinterpreted by the receiving end. (Delamer and Martínez Lastra, 2007)

With higher baud rates, reduced bandwidth is desired while still avoiding the silences caused by long sequences of ones or zeros. In later version, such as 100BASE-TX use 8-wire cable with twisted pairs instead of coaxial cable. The 100BASE-TX uses 4B5B to encode the data. 4B5B modulates the 4bit sequences to 5 bits, each of the 16 variations of the 4bit sequence having unique 5bit code, which are selected so that the change of value happens for certain. (Miller and Cummins, 2000)

As major problem associated with twisted pair wiring scheme is that as the signal levels need to be strong enough to be reliably interpreted over twisted cable pair, the wire radiates electromagnetic interference (EMI). If one attempts to use 125Mhz signal directly on twisted pair media, the specifications for energy radiation would exceed the limits of Federal Communications Commission (FCC) and CENELEC. To overcome this EMI radiation issue 100BASE-TX introduces a sublayer which employs MLT-3 (Multi Level Transmission – 3 level) as another encoding before transmission. MLT-3 modulates the signal so that it has three states (Positive, zero and negative) and the signal changes states only if "1" occurs. As the transition happens only with "1" and there are four transitions (positive to zero, zero to negative, negative to zero and zero to positive). This means that MLT-3 modulates the message so that the message has maximum fundamental frequency of 1/4th of the original. As the 4B5B would increase the frequency from 100MHz to 125MHz as it increases the bit count by one fourth and MLT-3 divides it to fourth of the modulated signal. The resulting signal has maximum fundamental frequency of 31,25MHz. (Miller and Cummins, 2000)

The collisions within the Ethernet are caused by devices trying to send messages simultaneously and causing the messages to become corrupted. The collisions happen more often the more devices are connected to same "collision domain". These collision domains were defined by the number of devices connected to the same bus. In the early versions of Ethernet used thick coaxial cable for transmission. Current fast Ethernet uses Cat 5 cables, which uses two cables for sending and two for receiving, providing full duplex features and changing the topology from bus to star. The older coaxial cable versions of Ethernet needed to solve these collisions by using Carrier Sense Multiple Access with Collision Detection (CSMA/CD). To avoid collisions and resolve collisions CSMA/CD defines a set of rules on how to act to avoid and to resolve collisions (Delamer and Martínez Lastra, 2007). These collisions were caused the Ethernet to be non-deterministic and thus not desirable for industrial applications. However, introduction of switched Ethernet allowed to use Ethernet on industrial applications as it provided needed determinism (Henning, 2016). The switch will keep queue of outgoing frames for each connected endpoint in case frames from multiple sources are directed to same destination arrive at the same time. This avoids collisions and lost packets, providing efficiency of nearly 100% of the available bandwidth (Delamer and Martínez Lastra, 2007). As the Siemens Profinet devices provide integrated switching, meaning that each device may function as a switch, the Profinet network (or industrial Ethernet network using Profinet devices), can be set up as fully switched network (Pigan and Metter, 2008). Fully switched network means that network is set up so that there is only a single device in each collision domain (one device can't collide with itself), thus providing fully collision free network even with bus and tree topologies if switching devices are used (Pigan and Metter, 2008; "PROFINET System Description, System manual," n.d.).

2.2.2 TCP/IP protocol stack

The TCP/IP (Transmission control protocol/Internet protocol) protocol family was originally developed by DARPA (Defence Advanced Research Projects Agency) in the seventies. Original objective was to allow computer systems to communicate freely, regardless of the location. TCP/IP protocol is currently most widely used protocol family in offices, homes and industrial environments. (Pigan and Metter, 2008) The TCP/IP protocol suite is also known as the *Internet protocol suite*, as TCP/IP is most widely used standard in the Internet (Edwards and Bramante, 2009).

TCP/IP is a suite of protocols which consists of two main parts. Transmission control protocol (TCP) controls the transmission and the actual data transfer. Internet protocol (IP) is used to uniquely address a computer in a network. The TCP/IP is considered as a suite of protocols that work in concert to provide wide range of functionalities that we now take for granted. Figure 8 shows some of the TCP/IP related protocols and the interdependencies of the protocols. (Miller and Cummins, 2000) Also UDP is considered to be major part of TCP/IP protocol family. (Donahoo and Calvert, 2009; Pigan and Metter, 2008)

The TCP/IP protocol suite is mapped to four layers each of which can be cross-referenced to seven-layer OSI model. Figure 7 illustrates the mapping of the TCP/IP model to OSI reference model. (Edwards and Bramante, 2009)

- The *network interface layer* corresponds to the physical and data link layers of the OSI reference model. The network interface layer is responsible for the device drivers and hardware interfaces that connect node to transmission media.

- The *Internet layer* corresponds to the network layer of the OSI reference model. Internet layer is responsible of delivery of packets through network, nodes that perform functions on this layer are responsible for receiving a datagram, determining where to send it and forwarding it towards its destination.

- The *transport layer* corresponds to the transport layer of the OSI reference model. Two primary protocols function on this layer, TCP (as the name implies) and UDP. These layers are responsible for successful data flow between nodes within a network.

- Application layer corresponds to the application presentation and session layers of the OSI reference model. Users initiate a process that use an application to access network services and on the other end, lower layers receive the data and pass it up to application for processing for the user. This layer concerns with the details of the application itself and not so much about moving the data.

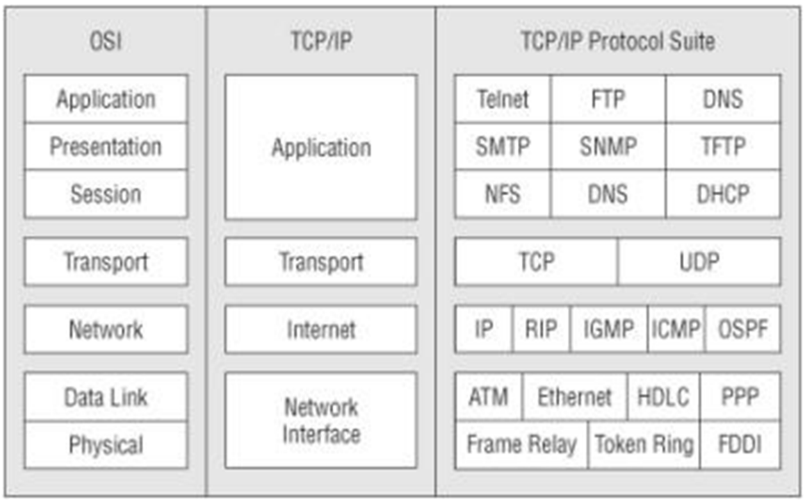


Figure 7. *TCP/IP suite and OSI model (Edwards and Bramante, 2009)*

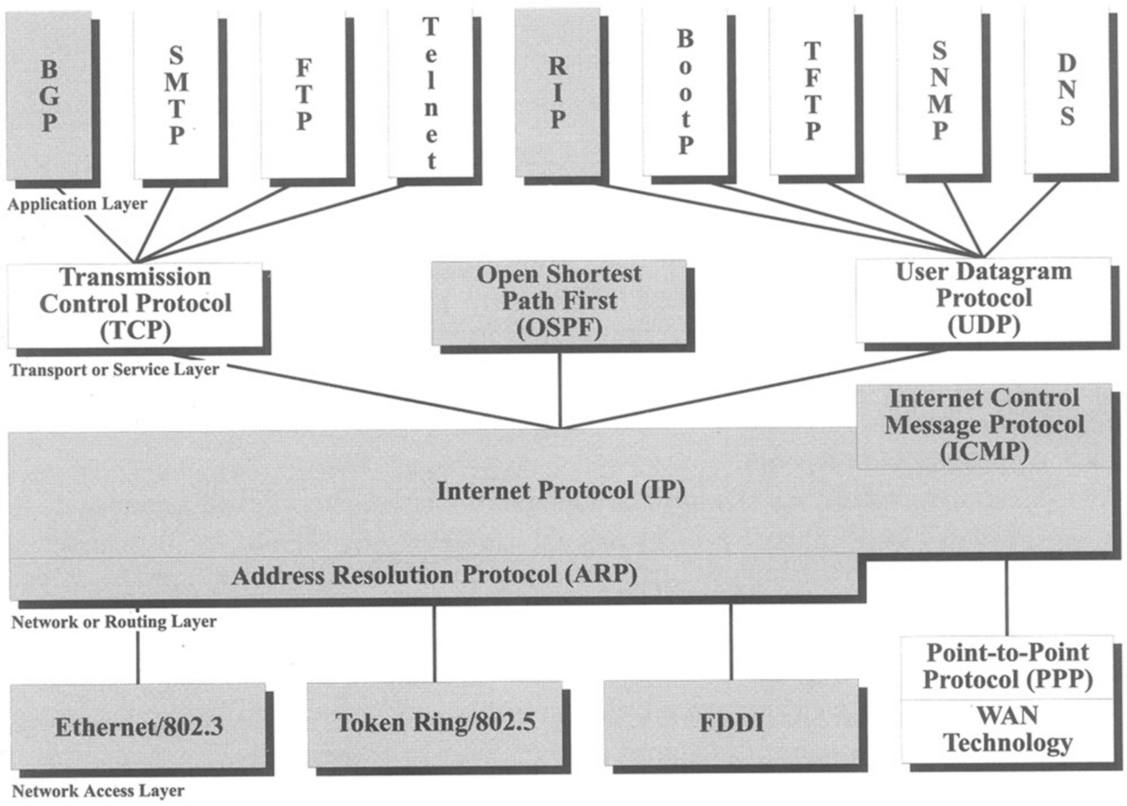

BGP
SMTP
FTP
Telnet
RIP
BootP
TFTP
SNMP
DNS
Application Layer
Transmission Control Protocol (TCP)
Open Shortest Path First (OSPF)
User Datagram Protocol (UDP)
Transport or Service Layer
Internet Control Message Protocol (ICMP)
Internet Protocol (IP)
Address Resolution Protocol (ARP)
Network or Routing Layer
Ethernet/802.3
Token Ring/802.5
FDDI
Point-to-Point Protocol (PPP)
WAN Technology
Network Access Layer

Figure 8. *Major protocols of the TCP/IP protocol suite. (Miller and Cummins, 2000)*

Internet protocol, is used to transfer datagrams or "data packets" from endpoint to endpoint across one or more intermediary nodes. In contrast to Ethernet where frames are transferred from endpoint to endpoint directly through LAN, an IP packet is expected to go through several LANs before reaching its destination. An IP packet can even use different layer 1 and 2 networks along its path. Station which wishes to communicate with another station needs to be identified with unique IP address. Unlike in Ethernet, this IP address is not hardware dependant. (Delamer and Martínez Lastra, 2007; Pigan and Metter, 2008)

The Internet protocol is connectionless service with unreliable datagram service, meaning that not correctness of the data nor sequence, completeness or unambiguity of the datagrams is checked at the IP level. The correctness and acknowledgements are done in TCP which is typically used together with Internet Protocol. (Pigan and Metter, 2008)

Similarly, as the Ethernet frame, IP has its own frame, which has data and overhead. Frame has minimum length of 20 bytes and its most vital parts are the 32bit source and destination IP addresses. The packet structure of IPv4 frame can be seen in total in Figure 9. (Delamer and Martínez Lastra, 2007; Pigan and Metter, 2008)

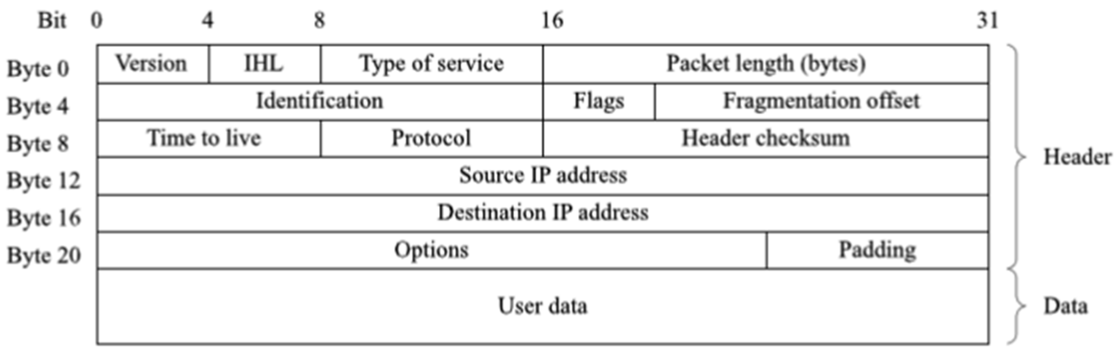


Figure 9. *Structure of IPv4 packet (Pigan and Metter, 2008)*

The IP addresses defines logical network addresses for the TCP/IP protocol suite. This address must uniquely identify an IP endpoint. Unlike the MAC address used in Ethernet the IP address is not hardware dependant. Sometimes IP address can be also identified by more familiar name called domain name which is more commonly known as “internet address” as the name is sequence of text strings separated with dots such as “www.CompanyName.com”. As the internet protocol needs 32bit address, DNS (Domain Name System) can be used to determine 32bit IP address for given domain name. (Delamer and Martínez Lastra, 2007; Pigan and Metter, 2008)

Two major extensions to IP are the Internet Control Message Protocol (ICMP) and Address Resolution Protocol (ARP). As the TCP/IP doesn’t introduce underlying network technologies and existing technologies such as Ethernet and Token Ring are used, there is need for address translation since these rely on addressing structure that is incompatible with address structure of IP. In the case of Ethernet, ARP resolves the MAC address of the corresponding IP address. The ICMP provides an error messaging capability and rudimentary routing and reachability function. ICMP provides two important diagnostics tools, ICMP “Echo Request” – Ping and Traceroute. Receiver of ICMP Echo Request packet must send it back, making it Echo reply. The mechanism provides easy checking of availability of certain address. Traceroute provides a mechanism to figure out how many intermediary stations are needed to be crossed before the destination is reached.(Delamer and Martínez Lastra, 2007; Miller and Cummins, 2000; Pigan and Metter, 2008)

Transmission control protocol, or TCP for short is the part of the TCP/IP protocol suite that controls the transmission of data. It takes place on 4th or the transport layer of OSI stack, above the network layer where internet protocol lies. The TCP provides reliable data transmission to higher layers. The purpose of the TCP is to segment the data into units that fit in to IP packets and may be retransmitted lost packets without the two endpoints using TCP not knowing that the data is split in to smaller packets. Two endpoints using TCP simply see continuous flow of data, which is guaranteed to be reproduced at the receiving end without loss of information. (Delamer and Martínez Lastra, 2007)

TCP passes the data in units called segments to IP layer. When TCP splits a message, it uses the sequence number to reconstruct the message at the receiving end. If one packet of 500 bytes is set to be sent as two 250byte packets TCP sends first message with sequence number of 1, and the second packet with sequence number 251. Depending on the case the receiver will respond to first message with ACK 251 and to second ACK 501, or simply acknowledge both packets by ACK 501. (Delamer and Martínez Lastra, 2007)

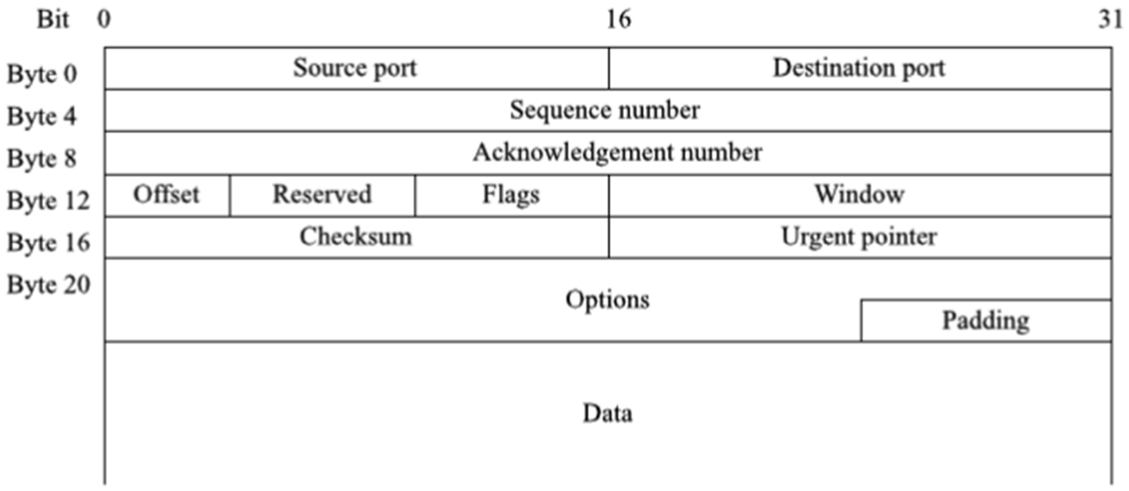


Figure 10. *TCP packet structure (Pigan and Metter, 2008)*

In Figure 10 the TCP packet structure can be seen. The source and destination port describe the source and recipient (application) in the target host provided by IP address. The 32bit sequence number ensures the correct order of the data stream and acknowledgement number is used to acknowledge the sender of successful sending of packets. Flags are used to inform the recipient of what the message contains (example: does the package acknowledge something or want to start or close the TCP connection). The window field specifies the maximum amount of data that can be sent without acknowledgement. (Delamer and Martínez Lastra, 2007)

TCP defines concept of port to distinguish between different connections in the same device. Each connection is assigned a port number at the sending host and at the receiving host, the port numbers don't need to match in both ends but need to be constant during the lifecycle of TCP connection. As an IP packet arrives to host containing TCP segment, port number is used to place the TCP packet to correct *stream.* The TCP port is often addressed using combination of IP address and port number (for example 192.168.0.12: 8080). The combination of IP address and port number can also be referred to as a socket. (Delamer and Martínez Lastra, 2007; Pigan and Metter, 2008)

A socket is an abstraction which an application may send and receive data through, similar as open-file allows application to read and write data to storage. A socket allows an application to connect to the network and communicate with other applications that are connected to the same network. Information written to socket by one application can be read by another application and vice versa. (Donahoo and Calvert, 2009)

Socket itself is not a protocol nor does take a part in the OSI model. Socket provides an interface from transmission control layer to higher level layers. Socket is used to abstract the lower layer protocols to be easily used by application layer. This abstraction allows programmer to generate an object with parameters that define which protocols are used and where to find participant. In the case of TCP/IP protocol suite, IP address and port number are needed to find participant. Once the object of socket is created, it can be referred as object and called to connect, send and receive messages or to disconnect. The server needs to be started and listening to the port before the client can connect. Once the client sends connection request and first package, the server will, in this case reply with package to each package sent by client. Client and server can be configured so that both replies to each message sent by other, so client nor server can be distinguished one from another after the first message. The classical server – client roles can be distinguished by client being the first one to start the conversation or to end the conversation. (Donahoo and Calvert, 2009)

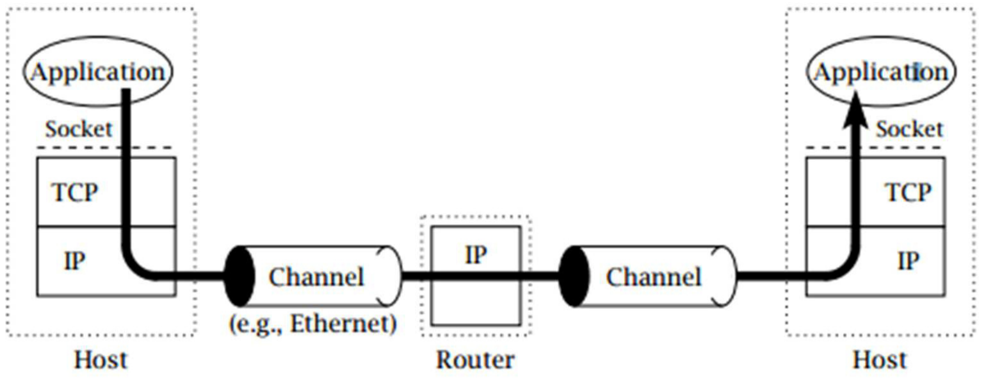


Figure 11. *TCP/IP protocol with socket interface (Donahoo and Calvert, 2009)*

When talking about socket, we are talking about file descriptor that allows us to read and write data from. This file descriptor needs parametrization to function correctly. As the socket is using TCP/IP stack, it needs to be parametrized accordingly. Server needs to know which port it is listening to, so it can separate incoming messages from others. Secondly server needs to know the socket type. Socket type is usually stream or datagram, the one is for TCP and another is of UDP in that order. The socket needs to know if it is the active or passive participant of the connection. Server is set to be passive since it typically is not shutting down the connection. Once the server is parametrized, it needs to bind the port to be listened. Once the binding of the port is successful the file descriptor is created, and the port cannot be bound by other applications. Once the file descriptor is created, it can be set to listen to incoming connections and messages to its port. Client side will need this port and address of the server to connect to the server. When client creates a socket for communicating with other socket, it will open a free port where it can receive messages itself. Upon sending a message to server, the client will send its own address and port to the server, so the server can respond to correct participant. Depending on how the client and server are built both sides of the connection can send and receive messages.

2.3 Virtual machine applications

As the simulation system is developed, the possible applications for virtualization of machine can be discussed. Ideally the virtual machine could be used throughout the whole product lifecycle. Early in the design phase of new machine, the concepts of virtual prototyping can be used. Before actual commissioning the concept of virtual commissioning can be used to make the actual commissioning go faster. During and after commissioning, virtual training environment can be used to train operators of the machine learn to operate faster. In the following chapters various VM application concepts are discussed.

"Virtual prototype is a computer simulation of a physical product that can be presented analysed and tested during design, engineering, manufacturing, service and recycling as if on a real physical model." (Wang 2002, ix. sit. Leino 2015)

All industrial sectors are confronted by challenges to design better high variety of products to variety of customers (Ameri and Dutta, 2005; Auweraer et al., 2008). These products need to be delivered in shorter time and in lower product and design cost (Auweraer et al., 2008). As the need for development grows, so needs the knowledge and expertise for development grow as well (Ameri and Dutta, 2005). Monolithic design teams can no longer efficiently manage the product development so to avoid lengthy product development systems, higher development costs and quality problems (Ameri and Dutta, 2005). Collaboration across distributed and multidisciplinary design teams has become necessity (Ameri and Dutta, 2005). It has been claimed that virtual prototyping shortens the product development cycles, reduces product development costs, enables better decision making and higher quality of the product. (Leino, 2015)

In virtual prototyping virtual product models or computer simulation models of the product are used instead of or in addition of the physical prototype. Virtual prototype is computer simulation model of the product prototype that is used in the virtual prototyping. This work is focused on real-time interactive models that can be controlled by external logic. The problem of virtual prototyping is to justify the decision to go through virtual prototyping since it is difficult to measure quantitative value such as money. There are too many variables affecting the benefits and costs of virtual prototyping and resources create value only when they are put into use and combined with skills and knowledge. The quantitative objects that add to the value could be number of engineering changes, cost of physical prototyping, calendar time of product development and number of safety problems. The more problems are identified and changed in virtual prototype, the less problems occur in physical prototype. Similarly the cost of creation of virtual prototype will be quantified. (Leino, 2015)

Quantitative methods for measuring value of virtual prototyping can be difficult to measure, so qualitative measures might be interesting. The main proposition by work of Leino, 2015, is that value of virtual prototyping is manifested by its position as an intermediary object and a medium for improved communication and knowledge creation. The added value of a virtual prototype is derived from the facilitation human interventions, opportunity to make mistakes and learn with immaterial prototypes, compared to conventional trial and error physical value chain. (Leino, 2015)

Usual problem with logic programming is that it can only be tested once the plant has been installed. This causes longer times on site on actual commissioning. (Ko and Park, 2014) The three most significant drivers in automation industry that have higher impacts on automation industry are “Deadline control”, “Risk of delay and interruption during ramp up” and “Software quality in relation to future operation”. (Shahim and Moller, 2016)

Virtual commissioning involves replicating the behaviour of one or more pieces of hardware and software environment. The virtual commissioning is the action of commissioning the PLC code against simulated system. The simulation requires the same behaviour as the real production system, this behaviour includes movements, signals, material flow and safety parameters. The simulation can confirm that the robots and other automated machines work as expected which will reduce commissioning time. (Markovič et al., 2015)

If the program could be tested before it is used on site, the commissioning time could be potentially reduced. The virtual commissioning is proposed to do just that, as the virtual machine (or even plant) is made to imitate the actions of actual machine. (Ko and Park, 2014) A project going through virtual commissioning (VC) has been studied to experience less interruptions in the ramp up and make it possible to hand over the entire system to customer on time. (Shahim and Moller, 2016)

Human errors often occur when operator is working under high time pressure and when performance requirements are high (Vieira et al., 2010). The staff-training objectives include, better operating of the system, increase safety, anticipate incidents and reduce risks of faulty procedures and manage efficiently in emergency situations (Bartak et al., 2000). Operating training simulators use mathematical models capable of simulating the real process in a realistic manner to allow efficient training in the virtual environment (Gerlach et al., 2016). The practical operator training is fundamental and critical in terms of effort, time, costs, hazard and to the equipment itself (Vergnano et al., 2017). It has been suggested, that interactive training introduced via a computer has been reducing training time and been more cost efficient compared to traditional classroom lectures (Stone, 2001).

For a system to work correctly in faulty scenarios, all the fault scenarios and responses to fault scenarios must be tested. Testing of the faulty scenarios in PLC programming is often done by hand and without systematic workflow. To find a way for systematic testing, one can look in to conventional software development. In conventional software development, there are something called *asserts* are used during unit testing. These asserts are used only during software development, and they can be used to tell if the block is working incorrectly in certain scenario. These asserts are done by giving a partial program called *unit* a various sets of input parameters and checking if the unit outputs a correct response to given set of inputs. In PLC programming there is hardly any common practice for such testing, possibly due to the fact that asserts are not supported by development platforms. (Rösch and Vogel-Heuser, 2017)

For similar kind of testing fault injection principles could be implemented for testing in simulator environment. In simulator environment three kinds of fault injections can be implemented, software-implemented fault injection (SWIFI), Hardware-implemented fault injection (HWIFI), model implemented fault injection (MIFI). While SWIFI and HWIFI are used on prototypes or system testing, MIFI is used in conceptual design phases to give early feedback to engineers. (Rösch and Vogel-Heuser, 2017)

Fault injection is composed of four parts:

- Fault(s) *"F"* that are injected
- A set of activations *"A"*
- The readouts *"R"*
- The actions or measures *"M"* that are derived from F.A.R.

Faults are usually based on hardware or human error. Hardware errors can be implemented on simulator side and software errors can be implemented in PLC software. The simulator makes it possible to check how the logic works in those scenarios without breaking the machine or risking the safety of workers. Even automated fault testing could be implemented to check all the interesting scenarios in the system every time program is changed. (Rösch and Vogel-Heuser, 2017)

As example, SWIFI means injection of error to software, HWIFI can be a forced pin for example and MIFI can be implemented to simulator model as faulty hose for example (Rösch and Vogel-Heuser, 2017).

2.4 State of the art

There is constant development on plywood manufacturing, and the development is going towards better yield and lower downtimes of the machines. The better yield is gained through better centring optimization of the block, reducing the core size of the block and trough development of the blade settings to increase the peel quality and reduce the spin-outs of the blocks. Current modern peeling machines typically peel from 5 to 20 blocks a minute, this rate depends highly on the block size and the type of the peeling machine.

The industrial communication is going towards Ethernet based communication protocols. The lean towards Ethernet based communications can be explained by the higher data rate provided by the Ethernet which is needed to excavate more data from the system to cloud applications and monitoring. These applications typically use TCP/IP protocol family for communication.

The simulations systems and applications are rapidly growing for industrial applications. Various vendors provide different platforms for mechanical modelling as more and more companies are interested in creating simulation of their machines. The reason for the interest is the claimed benefits of the simulation, the system can be prototyped in safe environment to avoid collisions, the operators can be trained faster, and the program can be tested before commissioning to verify correct functionality.

Table 2. *State of the art in virtual machine industry, industrial communication system and plywood industry*

	Virtual machines	**Industrial communication systems**	**Plywood machine industry**
State of the art	Building a virtual machine is costly but promises of beneficial applications (Shahim and Moller, 2016)	Various industrial buses and protocols. Protocols generalized by OSI model.	Highly automated high efficiency machinery (Varis, 2017)
Current trends	Using virtual machines, for prototyping and for virtual commissioning.	Ethernet based communications give promise to meet all the needs (Delamer and Martínez Lastra, 2007).	Higher personnel efficiency and higher quality of the end product (Varis, 2017)
Current challenges	Technological adaptation in the companies. Companies need new type of expertise. (Reinhart and Wünsch, 2007)	Changes to new technologies in industries is slow.	Alternative products such as OSB (Oriented Strand Board) (Varis, 2017)

3. DESIGNING CONTROL SYSTEM FOR SIMULATION

In this chapter the work to be done is designed and the taken decisions are reasoned. Chapter 3.1 describes decisions made to the control interface and how the interface should be composed and how it should be used. Chapter 3.2 considers the needed changes that are to be done to the control program for it to work with simulation. Chapter 3.3 considers how the virtual machines could be applied in this context.

3.1 Defining interface

The interface in this case means the set of data that is exchanged between the two participants. So altogether there should be four data sets defined in this project one incoming and one outgoing for both PLCs (from PLCs perspective). The outgoing interfaces are used to control the axes and the incoming interfaces are used to give measurement information from the simulation system.

This interface can be defined in multiple ways for same application and the same interface can be used differently. As example the interface could only send one axis data at a time making the interface somewhat small. Also, the same interface could be used differently; the data could only be transmitted only when a value is changed, instead of periodical data exchange. From the examples it can be easy to understand, that changing the interface definition or the way it is used can change the program that use these interfaces, proving the earlier made claim that the interface should be defined early in the development process.

The environment where the communication interface resides can be seen in Figure 12. The data is exchanged with the Mevea IO clients which map the data to Mevea IO pool which is read by the simulation and the real-time control script (python code) and which write the status information back to the output pool which is mapped and sent back to the PLC by the Mevea IO client.

Although the interface should be defined early in the development process, doesn't mean that it cannot be altered later in the process to get more information from simulation or to add more axes to be controlled by the interface. Although in this case, all the new features are suggested to be added to the end of the interfacing data structures, as the data added to middle of the structures is more prone to human error.

The communication interface selection is simple, since the only communication protocol each of the devices support is TCP/IP socket. The Mevea IO interface supports CAN protocol and the use of TCP/IP socket. The Siemens' systems don't support CAN without external devices. Since the TCP/IP is believed to provide sufficient information exchange rate through Ethernet, the TCP/IP socket is selected to be used over Ethernet.

Although the TCP/IP socket doesn't provide robust communication in timely manner, the deviation is believed to be sufficiently small since there is no hard-real-time requirement for the communication.

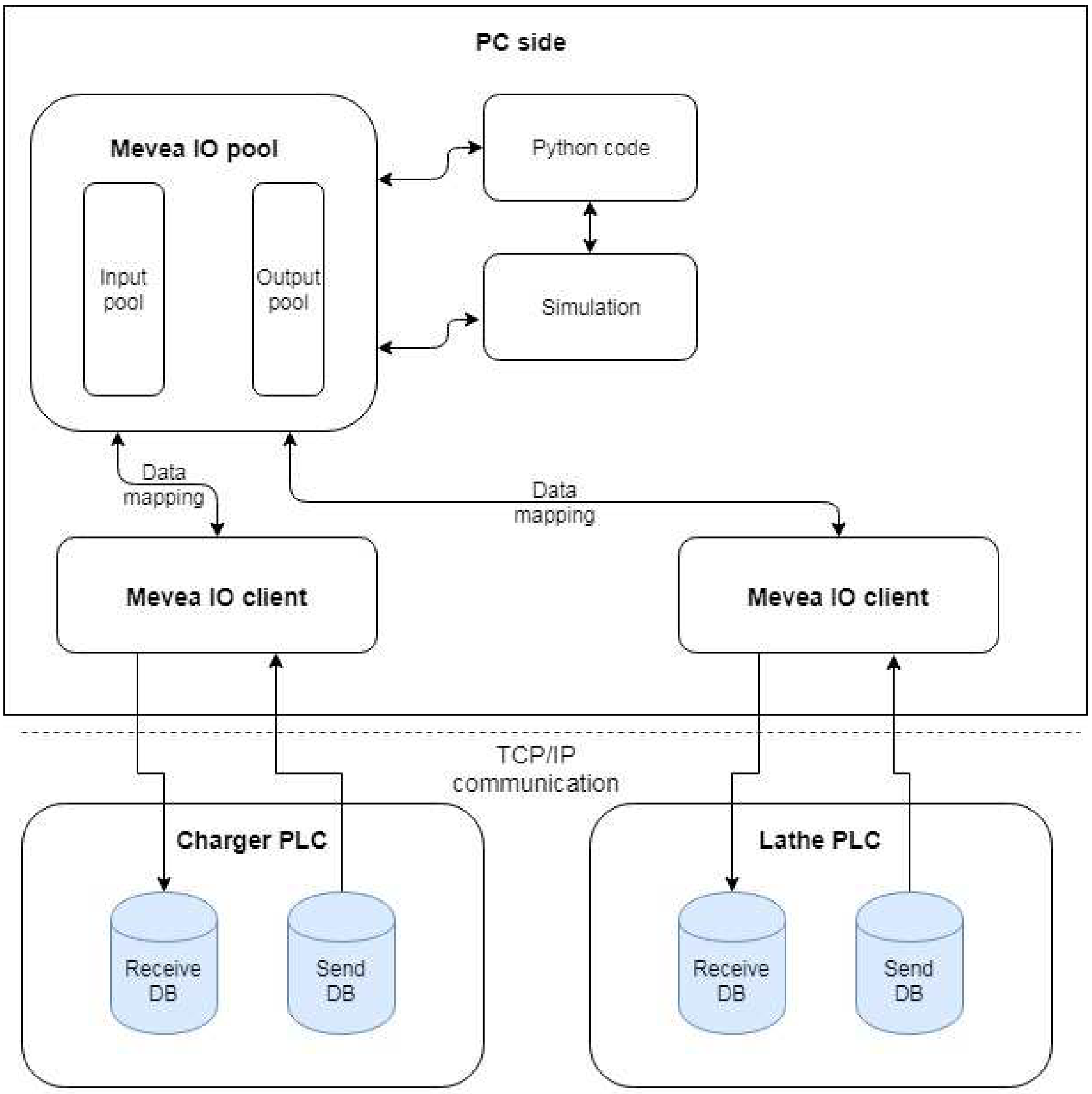


Figure 12. *Communication system where designed interface resides*

3.1.1 Sending command to simulator

The interface that is defined between PLCs and simulation is used to control the axes in the simulation. The motion control of the axes is done by scripts. The scripts are written in python language. The embedded python environment to Mevea solver allows the script to read and write to variables in the model. The script files are included in the simulation model and linked to the variables, which they are controlling. These scripts are called every simulation cycle and thus can be used for real-time motion control of the simulation. Each controlled axis has its own script for motion control.

The control system interface is designed to imitate the motion control program used in the real machine, in our case the real-time motion controller is Delta RMC. The commands in Delta RMC are initiated by command number. In the controller this command number is the key value to structure which contain command specific data, such as command type, control parameters and control values. The Delta RMC takes command number as input along with possible variables if for example the target position of the command varies. Delta RMC receives the axis position typically from absolute position sensor for the feedback, which is needed for control.

Instead of the command number, several control data structures are sent to simulation IO interface. One control data structure contains 5 bits and 5 control values, this control structure is described in detail in the following chapter. There are also reserve bits and reserve values in the interface, which make interface modifications easier to manage on PLC side. Bits in the control data structure are used to indicate the type of command needs to be carried out. Real values in the control data structure describe the parameters the command needs to use.

The commands always start with rising edge of execute bit. Rising edge of signal means the change of signal state from logic “0” to “1”. In principle, it should be enough for this execute bit to be on until the next TCP packet is sent. As the functioning of the script is non-deterministic the signal is held as logic “1” for 500ms to ensure that the signal is received. As a result, one cannot initiate commands to one axis at higher rate than 2/s.

3.1.2 Control parameters

To make correct movement in right situation the system needs to send right parameters. In this chapter each bit and control value are explained in detail.

Commands are composed of two parts. Control bit frame and target value frame. The control bit frame is composed of five bits that initiate and describe the type of the command. The value frame is composed of five real values each describing *how* the motion should be accomplished. The designed composure of one control message for one axis can be seen in Figure 13.

Execute bit is used to signal of new command to axis. Execute bit takes place as first bit of the control bit sequence of the axis. Basically, the rising edge of execute bit is meaningful but new rising edge (thus new command) is unable to come before previous has fallen.

Tracking bit is used in case axis needs to follow certain curve or to move relative to another axis movement. Tracking bit is in the same slot of the interface for some axes there is reserve bit (as seen in Figure 18), in cases of open close style axes the reserve bit is controlling the movement to "close" direction movement of "open / close" type of movement used in spindles. Tracking bit is set to be the second bit in the control bit frame of each axis.

Torque control bit is meant for cases when the axis is desired to move with desired torque or pressure. The torque control bit works like reserve bit for "open / close" type movements, with the difference that it controls open direction. Reason to use two bits for two directional movement is to have three different states for axis movement: open, close and stop. Torque control bit is the third bit of control bit frame.

Speed control bit is used mainly for joystick movements. During joystick movement, i.e. operator turns joystick left and holds it there, the speed control bit should be true during the movement and movement should stop when speed control bit goes to false or end position is reached. Speed control bit is the fourth bit in control bit frame.

Enable bit is used to enable any of the commands for the axis. This means that in normal operation all the enable bits should be true all the time and the axis shouldn't move if enable bit is off. Enable bit is the fifth and last used bit of the control bit frame, the unused reserve bits are not considered to be part of control bit frame.

Target position is a real value which defines the target position where the axis should move. Target position is the position of cylinder stroke, which can be defined to start from start or from the end of the cylinder. Position directions are supposed to be chosen so that zero is always towards centre of lathe spindles or towards ground. In case of tracking the position gives the offset value to tracking. All the positions are typically given in 1/10th to 1/100th of millimetre accuracy. All the target speeds, accelerations and decelerations are given in same scale with target position of that axis. Target position is the first value of the target value frame.

Target speed defines the maximum speed axis is allowed to move during action. Target speed is not necessarily reached if target acceleration and deceleration are not high enough, or the target position is very close. Target speed should be set to be higher in normal operation and slower or ramped in joystick operated movement. Target speed is the second value of the target value frame.

Target acceleration defines maximum acceleration for the movement to reach the target speed. Target acceleration should be defined high enough so that the target speed can be reached. Target acceleration is the third value in target value frame.

Target deceleration defines deceleration used to start decelerating the movement to full stop at target position. Very high deceleration value might cause end effector to jerk in end position, low deceleration makes the movement slow as the deceleration needs to start much earlier. The low deceleration in joystick control might cause it to feel sloppy, as the end effector starts slowing down with given deceleration as the joystick is released, causing end effector to move even after the release. Target deceleration is the fourth value in target value frame.

Target torque, or target pressure, is for those open close type movements where we set certain pressure and we move axis without conditions to either of the end positions with the given hydraulic pressure. In the real system, too high grab pressure could break the wood block and low pressure may not provide enough grip strength during peeling. Target torque is last value in control value frame, as the reserve values are not considered to be part of the control value frame.

Parameters explained here are configured separately for each different command for each axis. For example, for one axis there might be 5 different position control commands each with different position, speed, acceleration and deceleration. This allows moving the axis to various positions but also with various speed profiles. The structure of the send DB is illustrated in Figure 13.

3.1.3 Feedback values

As the real system needs information of the system states for control and diagnostics, so does the control system of the virtual machine. The simulation needs to send only a few diagnostics of the axes. In addition to axis information some sensor data is also sent.

Enable bit is sent from each axis to tell that the axis is doing well. The enable bit can be used for diagnostics on the PLC side, or to test operation on actuator failure.

In position bit is set to be true once the target position of the desired movement is reached and reset to false as the new command is initiated.

Actual Position is the sensor data of the axis position. The axis position is measured using absolute position sensor. Actual position is returned as the same scale as the target position is given.

Actual Speed is the actual piston movement speed of the axis. The speed is returned in same scale as the target position is given. Actual speed isn’t normally measured by any actual sensors in any but rotational axes, in which the speed is calculated from pulses using rotational encoders. But as the interface is desired to be uniform for all the axes, the actual speed is measured for all the axes (since simulated sensors are free and easy to implement).

These 2 bits and 2 real values are returned for each axis of the machine. There are also reserve bits which could be used for more specific status information of the axes.

Sensor data and simulation data are additionally returned from the simulation. As most of the sensor data is already returned as position data of the axes and no safety sensors are not needed, only few additional sensor information is needed. Only the most crucial sensor data for operation is gathered from simulation. These sensors measure pre-centring distance for pre-centring calculation and increment sensors for detecting the block in the feeder steps. The simulation data consists only of simulation time of the system. This information is used in the system to recognize that the simulation is not running, and the states of the state machines are reset once the simulation restarts.

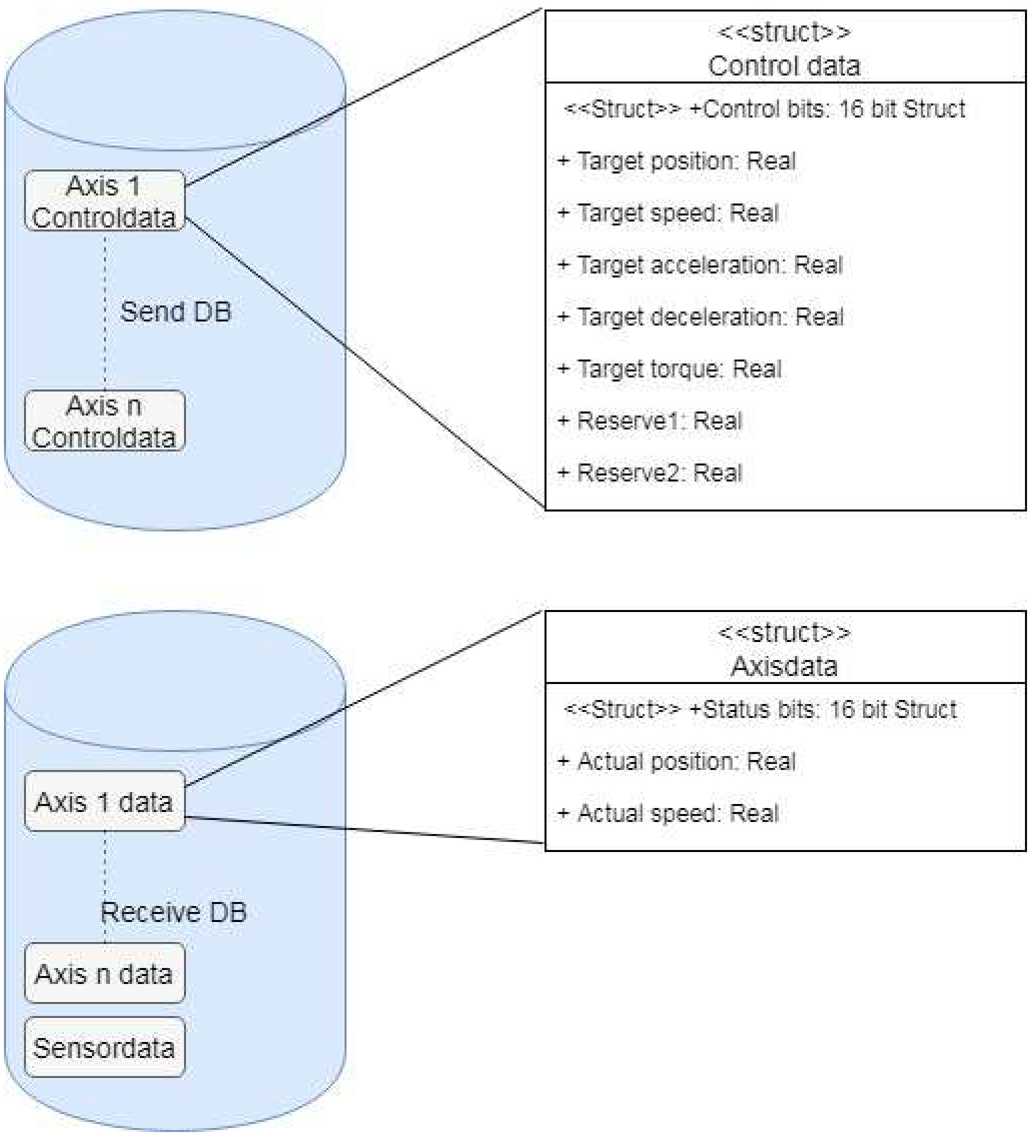


Figure 13. *Figure showing database structures used for communication*

3.1.4 Command types

Various command types are needed to control axes in different ways. In simple cases the axis is ordered to move from one position to another or is ordered to move until is ordered to stop. Each of the needed command type need to be defined in the interface for the controlling system to understand which type of command to initiate. The needed command types can be generally divided to four different categories: position control, speed control, torque control and tracking.

Position control is most commonly used of the control commands. It moves the axis to target position with given target speed, target acceleration and target deceleration discussed in previous chapter. Position control is initiated by raise of execute bit while enable is true. Actual speed curve of position control command can be found in chapter 4.2. Figure 24, a control voltage curve in Figure 23 and example control bits and control values for the movement in Figure 22.

Position control is used for all normal go-to type movements. By parametrizing the deceleration and speed can accuracy be increased and vibration in end position reduced, this comes with obvious cost of slowness of movement. On the other hand, speed of the movement can be increased simply by increasing all the values. All the desired speed and acceleration values are given as absolute values. The motion controller interprets the direction of the movement from current position and target position. From the direction of the movement, the controller decides if the speed should be negative or positive.

Once the parameters with command bits are sent to interface, the simulator script generates a quadratic or triangle shaped target curve for the speed in which to move to target position, area of this quadratic speed curve being the moved distance. This target speed curve is then tried to be followed by the control script. The control script has table of control values, flowrates of the valve and the area of the piston. With this knowledge the script can calculate the feedforwarded control value of the virtualized control signal to achieve desired movement speed at any given moment.

Speed control is mainly used for joystick movements by operator. Joystick movements typically drop the automation cycle of the program and hence only should be used in special cases. Such special cases can be spin-outs, maintenance or other non-predefined situations.

Speed control is initiated by raising edge of execute bit while holding speed control bit true throughout the joystick movement. Speed control needs all the same parameters as the position control. Like in position control the speed control takes speed and acceleration values as absolute values. The direction of the speed is decided in controller which decides the movement speed from the target position. Target positions for these movements should be unreachable end positions of the axes.

Similar as position control the speed control curve might look quadratic or triangle shaped even though the internal implementation is different. Internally the control tries to follow given acceleration until target speed is reached and when the joystick is released it should follow given deceleration until stopped.

Torque control command is used to move axes from end position to end position, or at least try to move from end to end, with given pressure set by proportional valve. Torque control enables us to move axes to squeeze the log with pressure and by doing so grabbing it firmly to spindles or clamps. Even though the variable is named torque, it is the control value for maximum pressure in cylinder.

The torque control is only enabled to use on few axes, clamps and spindles. These axes are used to clamp the log by the ends, as the log length is not always known, and the axes need to hold the log with some pressure, it cannot be controller purely by position, but rather by pressure or torque depending on actuator. Torque control doesn't care about reaching positions, although the logic following the position might care, torque control only sets certain pressure on either side of the piston and hopes for the best.

Torque control is used so that reserve or "close" bit is *true* and torque control bit *false* while we are moving the axes towards the log. The bits are inverted when unclamping. This movement, like others, is initiated by rising edge of execute bit. The usage of two bits for two direction movement is reasoned by detecting the power out situation if both bits are 0.

When using torque control only target torque is needed. The directions are set by the control bits. The speed of the movement is decided by the nominal flows of the hydraulic components. The pressure is often wanted to be lowered after the initial stroke, since the grip is achieved from the initial stroke. Long duration high pressure might crack the log in the real world.

Tracking is used by few axes during peeling. These axes need to follow the distance of the knife carriage from the middle of the spindle centre. The knife carriage distance from spindle centre is assumed size of the log. Each of these axes has different kind of action during tracking.

Knife carriage tracks the rotation of lathe spindles and moves forward target peeling thickness forward for each full rotation of lathe spindles. Peeling thickness is given as *target torque* parameter to knife carriage interface. Peeling is the only time that the feed-back loop duration is desired to be sufficiently short in the system. In the feedback loop the PLC reads the position of the knife carriage and controls the rotation speed of the spindles to keep the peeling speed (peeling speed is the linear speed of the resulting ve-neer mat) as static as possible.

Backup rolls (Top and bottom roll) distance should be about the same from the spindle centre as the knife carriage during peeling, as the log peels in spiral-like fashion and the distance to centre of log from different parts of the log is different. Backup rolls should support the peeling and prevent the log from bending from the middle. The bending of the log cause lower quality veneer as the log peels differently from ends and from middle. Backup rolls receive possible offset value to *target position* value in case the log is desired to be pressured towards the knife with even greater force or if the force is desired to be lowered.

Backup roll rotation during tracking causes backup rolls to rotate so that linear speed in the surface is the same or slightly more than the linear speed of the block surface speed (or peeling speed). This should support the rotation of the spindles against peeling forces.

Pitch angle (the cutting angle of the knife) follows given curve during peeling. Different kinds of curves can be used for different kinds of wood as the wood core and outer wood layer properties depend on the species. Also, there is an offset to the angle which can be adjusted depending on the wear of the knife.

3.2 Debugging the original program for simulation use

The principle in the program debugging is that the changes made to the code could be easily separated from the original program code. If there are changes to be done to program blocks, the changes should be made in the beginning of block to be easily separated from the code. Ideally the changes to program should be done in separate program blocks which could be turned on for the simulation use and off for real machine control. This allows the program blocks to be easily reused in other programs and projects with some changes if the function blocks are made to be reusable.

Original program is used for real machine, not virtual one, which can cause some issues if one desires to use the same program for virtual machine. As there are some natural simplifications in simulated system, there some inputs which don't exist in simulation. There is also some code that needs wants to know the states of the inputs. The inputs that are simplified out of the simulation program check if the axis is in end position, check if motor is running, monitor the oil level or temperature. The inputs that sense axis end position need to be replaced with the interpreted end position from the position information of the axis. The difficulty is that the axis might be “in position" earlier or later than they would be in real system and initiate a next step too early or too late, so the position needs to be carefully adjusted, much like sensor position is adjusted within the real system. The faults with motor run information can be fixed by either forcing the input bit of the motor, or by bypassing the input each time the it is used. The bit forcing is easier to implement and to manage, since it doesn't need as much code to be changed and it can be changed in one position of the code. In the Siemens TIA development environment, the forcing can be done anywhere in the code (even the input bits can be forced!), as long as the bit forcing function is called before the bit is used the first time. For that reason, the bit forcing function is called in the beginning of the program cycle.

Peeling line has two separate logics, one controlling charger and another controlling lathe. The practical separation to two logics has been made on the basis that these can be delivered separately, so the program has been separated in to two parts. This also allows faster deployment as the programming and commissioning can be done by two people easier, also this allows easier separation of programs if either of the machines is sold separately.

Totally integrated automation portal (TIA-portal) is selected to be used to program logic code, as it is only development environment supported by given logics. Programming languages in logics are all written in IEC61131-3 standard based languages: ladder diagram (LAD), function block diagram (FBD), in statement list (STL) and structured control language (SCL).

Program code for charger and lathe was given by Raute Corporation from a program which is used to control the real version of the simulated machine. Although the code is the same, the real project is done with two Siemens s400 series logic, whereas this project uses next generation s1500 series logics. Both logics support same programming languages, but some standard library functions are changed or removed, and the internal operation of the PLCs is changed.

The actual interface that connects logic to simulation has already been discussed previously. As the logic program receives the data from simulation, the data should substitute the actual data from the real interface.

The actual data transfer between simulation and logic is done using buffer databases (DB). There should be *data to simulation DB* and *data to simulation buffer DB*. Only the data to simulation block is manipulated in various parts of the program and before the sending the data is copied to buffer block which is then sent to simulation. Same goes for received data, data received is captured to *data from simulation buffer* which is then copied to *data from simulation* DB which is then read in the program.

Each axis position data is read in individual function block inside the PLC code, these blocks read status and position information of the axis and write the information to DB for other functions use. For simulation usage, the position data is overwritten in these function blocks as well as the necessary status bits are forced to correct states. As the data is overwritten in these functions, should the axis information spread to all other blocks correctly.

As the PLC program uses control and feedback values as integers and the interface has the values as real values. The real values have much wider value range compared to integers so possible overflow in the conversion must be taken in consideration. Fortunately, in this case the overflow is already considered in the interface definition. In the interface definition the maximal usage of integer scale is desired with easy-to-understand scale. As example axis of 650mm in length is given in 1/100th mm precision, making the axis use value range of 0-65000, as unsigned integer scale isn't exceeded (16bits provides range from 0 to 65535 in unsigned format). Axis of 750mm uses 1/50th mm precision as the 1/100th exceeds the integer value range (1/50th precision creates range from 0 - 37500 and 1/100th scale create range of 0-75000 causing overflow in integer).

One problem with this structure is that the PLC program use integer values to save unsigned integer values for some reason. These two datatypes share the same bitlength and format, but integer value has most significant bit (MSB) as negative value. This results so that integer value flips to -32768 after 32767 and continues to positive direction, where unsigned integer continues with positive values. This easily causes confusion as the values are monitored. Also, the conversion from integer format to real value format requires one intermediary step to unsigned integer to convert the initial value to real value correctly. If 50000 is to be saved in integer and directly converted to real, this real value would then contain -15536. This doesn't cause problems in communication with Delta RMC as the Delta uses unsigned integers as data type and data type is lost in the transmission and the value is understood "correctly" as 50000.

The previously mentioned conversions must be done somewhere. Previously mentioned buffer DBs provides good interface to convert the data. The data could be received to interfacing "receive buffer DB", and after that the data is moved to the actual receive DB after which the datatype is converted, and the axis direction is corrected. Respectively as the data is sent, the data is first moved to buffer DB, after which the data is converted, and the axis direction is corrected. The solution provides that the interfacing buffer DB has data which is in correct format for the simulation to understand, and the normal DB has the data in format that is good for the PLC. Other option is to change the axis directions in the simulation to correspond the directions of the PLC program.

Motion control blocks are used to generate commands and send them to interface. Normally in the logic there are function blocks that send command number to Delta real-time motion control (Delta RMC). For point-to-point movement delta RMC has *move absolute* command, which moves axis in closed loop to requested position from where axis happens to be, using requested speed, acceleration rate and deceleration rate. The axis stops at the requested position and hold it in closed-loop control. ("Delta RMC User Manual," 2017)

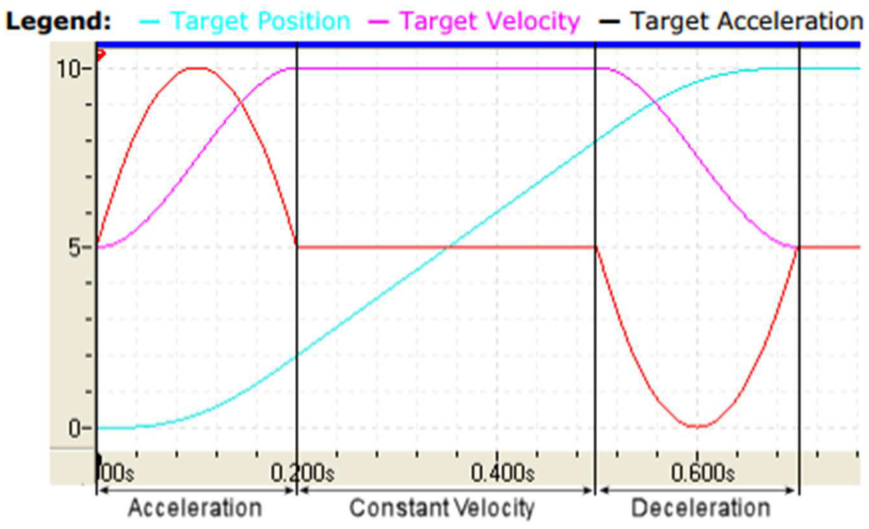


Figure 14. *Example movement control curve, using 2nd order acceleration coefficient ("Delta RMC User Manual," 2017)*

Since the project is not using Delta motion control in between the simulator and PLC the control done in Delta RMC had to be done somewhere else. As mentioned in chapter 3.1 axes control data imitate the data that is used by Delta RMC. The simulator side has the actual motion control programmed inside, scripted in python. Unlike in the motion control of the Delta which uses 2nd order acceleration coefficient (see Figure 14), the acceleration used by python script uses 0th order coefficient but provides satisfactory accuracy for fair presentation simulation.

Each axis has control function which write data corresponding to desired action to its own section of the control interface DB. These are simple functions that are ran under main program. Each function takes command pulses from motion control block which used to send commands to Delta RMC. Each command pulse runs one line of code where it sets new target values to communication interface block. Once values are changed, it goes to set the execute bit for a short period of time. Depending on command type the function block also sets the control bits according to the command. These bits encrypt the command type, such as speed control for joystick movement or tracking for following curve.

The simulation model itself does nothing with previously mentioned inputs from the PLC even though the inputs are defined in the model, but python scripts which are called by the model do. In most cases the python code is used to control the input voltage to 4/3 valve controlling the flow to axis cylinder chambers since the hydraulics are simulated within the simulation.

3.3 Considerations on virtual machine applications

In chapter 2.3 are discussed potential applications for virtual machines. The initial goal is to create a showcase simulator for a fair. Here further goals can be discussed. As mentioned in chapter 2.3 the virtual machine has many applications each of which has been proposed to improve the product quality, reduce the product lifecycle or increase product value. In this chapter applications are discussed and how each aspect could be put in use in this scope.

The virtual prototyping is an action of generating virtual model of the proposed machine. The virtual prototype can be used in product development and it can reduce the through time of product development as the mishaps of mechanical design can be spotted in a glance.

The virtual prototyping can be used in case of peeling machines to reduce the product development time. Although the device considered in this thesis already existed and no prototyping is needed, the virtual prototyping has already been proven to be useful in another peeling machine project.

Virtual commissioning is an action of generating PLC program against virtual model and commissioning the PLC program against the simulation model. The virtual commissioning has been proposed to reduce the ramp up time on site and improve the PLC code quality.

Applicability of going through virtual commissioning for a project can be debated. The company needs fair amount of expertise to generate simulation models for each project. The expertise would most likely require hiring of new personnel or using subcontractors to generate new models. Also, the project lifecycle management would need to be changed, the personnel and the management needs to adopt to new project lifecycle. On the other hand, if each of the new machines go through virtual prototyping, the models are ready for virtual commissioning afterwards. Also, the virtual commissioning could be tested on one project and the benefits and costs of virtual commissioning could then be reported.

Possibly the most obvious use for simulator is training purposes. Training simulators have been implemented multiple fields; in airplane pilot training, unmanned aerial vehicle (UAV) operator training, some lifting crane companies have developed simulators for training purposes. These have few things in common: the machines are expensive, they need expertise to control and faulty operation can cause severe damage and costs. Peeling machine is no different in this respect. Peeling line is costly, the control variables need deep understanding of the process and the faulty operation can cause damage to machine. Although, unlike many other training simulators which control mobile machines, peeling machine sits on foundation.

Even though the real machine sits on foundation, it doesn’t reduce the need for operator training. The machine is expensive, collisions are possible, downtime is not desired and product quality has high requirements. Even with automatic control, the machine still needs to have some parameters operated manually and faulty situations handled correctly and preferably quickly to maximize the production. To reduce these expenses in production, operator training via simulation can be implemented. The operator training via simulation can be split in few objectives, each of which have different requirements of the simulation system:

- The first and easiest objective is to train the operator the functionalities of the controls interactively, this gives safe environment for operators to get familiar with the controls.
- Second objective is to practice certain faulty scenarios, in our case these could be such as spin-outs and splinter stuck in knife gap, more of faulty scenario implementation in following chapter.
- Third objective, as the peeling process is simulated, the effects of control variables can be seen in peeling process simulation. The process simulation gives operator comprehensive understanding of variables, and effects of variables on process and resulting veneer.

Applying one or more of these layers will add to operators' comprehension of the machine. The comprehension should reduce operator training time with the actual machine and so should increase value to customer.

Implementing different layers for simulation requires different levels of reality of the simulation system. For the first objective the simulation system doesn't need to be realistic, the system simply needs to be visualized and the controls need to be interactive. Additionally, the system dynamics should function correctly. For the second layer, all the requirements of first layer are needed as the system should be ran normally before the error occurs. Additionally, these error cases need to be implemented somewhere, typically in the model or in the controlling software. More of the fault injection in the following chapter. Finally, the third objective for operator training requires the actual process simulation, in addition to the mechanical simulation. The process model simulation could possibly give actual information of the resulting veneer and generate errors of the second layer by itself. As the process could cause errors itself, the human generated errors by faulty parametrization can be reduced as the reasons for the faults are learned.

Fault injection is an action which observes the operation of a system after a different kind of fault. The fault injection is claimed to improve the PLC code quality and the product safety. Fault injection defines three types of injected faults: model implemented (MIFI), software implemented (SWIFI) and hardware implemented fault injection (HWIFI).

Each of previously mentioned fault injections could be implemented in our case. Software implemented faults could be forcing various bits to faulty state and to see if the state of the machine stays manageable. Software injected errors give critical data of the program to see which sensor faults could cause critical errors in the system functionality. Hardware implemented fault injection could be done by forcing a pin of desired input to see if the response is correct. Forcing pins in hardware can be redundant as this forcing can also be done in programming environment. If hardware implemented faults are desired to be tested, more actual hardware should be used. Model implemented faults could be very interesting as the hardware faults, that are normally nearly impossible or too expensive to test, could be tested in simulated environment. The more sophisticated the tests for faulty scenarios are, the more specific error messages can be given. The more informative error message is, the faster the fault can be localized and fixed.

4. TESTING THE CONTROL AND INTERFACE

In this chapter the control and communication system designed in chapter 3 is tested and proven that the communication and the control system works as it is desired. In chapter 4.1 the communication is studied through the overheads, efficiency, cycle time and bandwidth usage. Chapter 4.2 the correct functionality of the logic control is studied among with the correct functionality of control interface through example of command execution.

4.1 Communicating through TCP/IP

The control interface stack uses socket interface on transmission control protocol over internet protocol (TCP/IP) which is transferred over Ethernet. The socket communication is used between Siemens PLCs and Mevea-IO (input-output) interface.

In our case the socket connection is in place to replace the conventional IO interface and fieldbus interfaces of the real physical system. Fieldbuses are used to connect peripheral IO devices, motor drives and controllers to main PLC.

The socket interfaces need to gather all the communication interfaces together and communicate with that information between simulation instead of peripheric devices. There is also need for conventional IO for joystick, lights, switches and buttons that are connected to PLCs IO cards, as the operators bench is used to operate with the simulation.

The most data consuming part of the interface is the control interface that control each of the axes. The control data sent to each axis contain information of the desired movement. This way the data sent to interface has relatively large payload, especially as the integers are sent as real values. Real value has length of 4 bytes where integer has length of only 2 bytes. Control data is sent to simulation every 10 milliseconds for each axis. This data is sent even if no change in control values has happened. Also, the usage of the *only-send-data-as-a-reply* cycle failed as the Mevea IO client sent messages every 10ms (or any other given interval) and it seemed to break the normal receive-respond cycle within the PLC program.

In chapter 2.2 are considered the communication protocols used for this application. The environment is TCP/IP over Ethernet. To look in to the transmitted bits and bytes, either an oscilloscope to listen to actual changes of voltage level over cables is needed, or software to show the captured messages on our network card is needed. Due to practical reasons, such as, that the transmission is difficult to read from 4B5B coded and MLT-3 modulated message and the lack of oscilloscope resulted in the use of software to capture the messages. The software used to capture messages from our network card is WinPcap and software used to analyse and show the messages is Wireshark, both of which are freeware software.

Figure 17 shows various messages that are captured, each row marks one message and information such as message type, length (actual length and data length), time (relative to start capture time), protocol, sender and receiver IP and port as well as flags (PSH, ACK) used. These messages can be opened to be analysed, the analysis tool shows the message in binary or hexadecimal format (Figure 16). For convenience, the hexadecimal format is used for it is shorter and easier to read by human eyes, in hexadecimal format each byte is represented by two digits from 0 – F (0-9 then letters A-F, letters A-F represent decimal numbers from 10-15). The message analysis can be seen in second picture of Figure 16. In Figure 16 there are marked different overheads of the different protocols. Overhead of Ethernet II frame shown on orange, IP overhead on blue and TCP on green and the actual data is marked on red. The Ethernet II overhead is incomplete, the Ethernet II frame overhead should be 18byte long whereas the figure shows only 14 bytes. The cyclic redundancy check (CRC) bytes are missing from the end of the message, also the preamble and SFD are missing. The reason for missing bytes in header is simply due to network card, which strips these parts from the message before passing it on to application. Additionally, the Ethernet II physical layer protocol has defined inter-frame gap which is defined as minimum gap to wait before sending another packet, the IFG is 12 bytes long.

In Figure 15 the nesting of the message inside of other protocols can be seen. As the message is passed along to lower protocol, an overhead is also added to the message. In the figure, there is a Telnet message, in our application there is the command data to simulation or status information from simulation to PLC, but the message is encapsulated in similar manner. The result of the encapsulation can be seen in Figure 16 as the outermost layer is Ethernet II frame, then IP and finally TCP as the innermost layer.

Upped part of Figure 18 showing the resulting data structure on PLC side of one axis information that is according to the designed structure of chapter 3.1. First the control data bits are in structure, named accordingly. After control bit frame there are control values containing the information how the command should be executed. Lower in the Figure 18 the same structure is parsed accordingly so that the data can be stored in the Mevea IO pool.

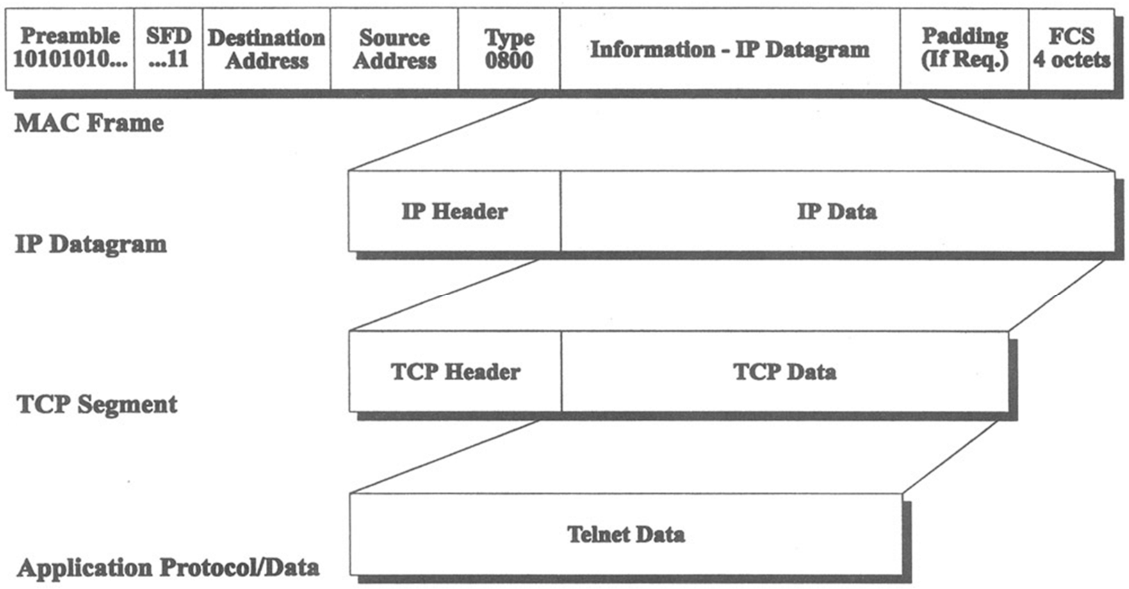
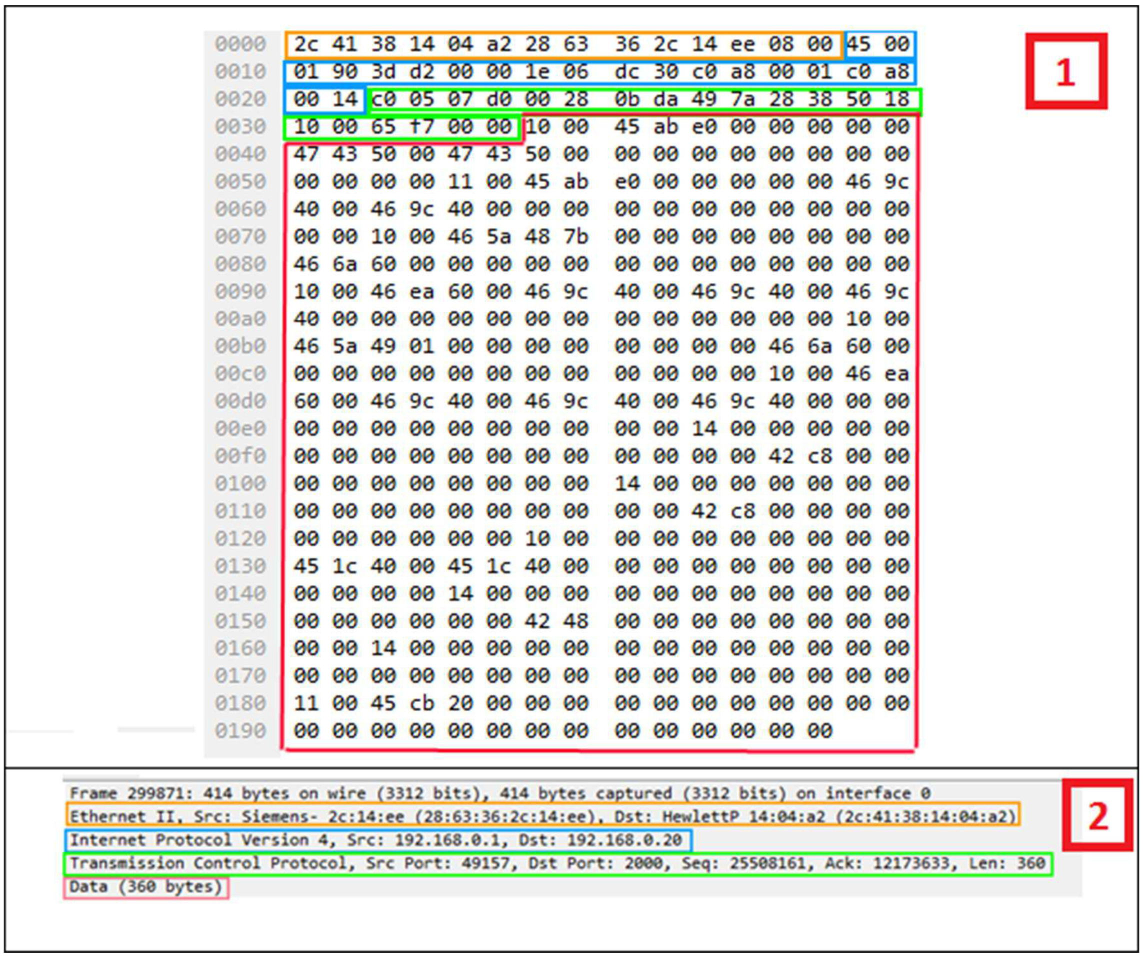


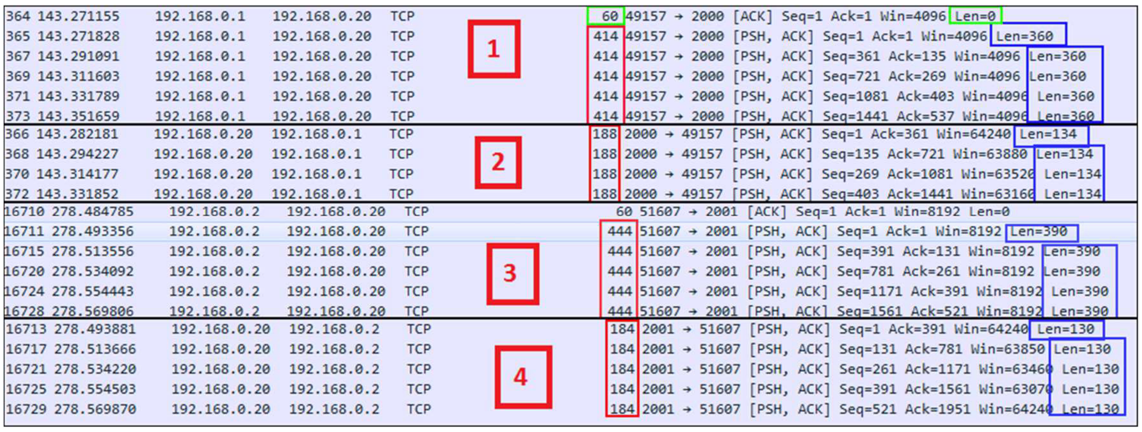
Figure 15. *Encapsulation of Telnet data inside TCP/IP inside Ethernet II frame (Miller and Cummins, 2000)*



```
0000  2c 41 38 14 04 a2 28 63  36 2c 14 ee 08 00 45 00
0010  01 90 3d d2 00 00 1e 06  dc 30 c0 a8 00 01 c0 a8
0020  00 14 c0 05 07 d0 00 28  0b da 49 7a 28 38 50 18
0030  10 00 65 f7 00 00 10 00  45 ab e0 00 00 00 00 00
0040  47 43 50 00 47 43 50 00  00 00 00 00 00 00 00 00
0050  00 00 00 00 11 00 45 ab  e0 00 00 00 00 00 46 9c
0060  40 00 46 9c 40 00 00 00  00 00 00 00 00 00 00 00
0070  00 00 10 00 46 5a 48 7b  00 00 00 00 00 00 00 00
0080  46 6a 60 00 00 00 00 00  00 00 00 00 00 00 00 00
0090  10 00 46 ea 60 00 46 9c  40 00 46 9c 40 00 46 9c
00a0  40 00 00 00 00 00 00 00  00 00 00 00 00 00 10 00
00b0  46 5a 49 01 00 00 00 00  00 00 00 00 46 6a 60 00
00c0  00 00 00 00 00 00 00 00  00 00 00 00 10 00 46 ea
00d0  60 00 46 9c 40 00 46 9c  40 00 46 9c 40 00 00 00
00e0  00 00 00 00 00 00 00 00  00 00 14 00 00 00 00 00
00f0  00 00 00 00 00 00 00 00  00 00 00 00 42 c8 00 00
0100  00 00 00 00 00 00 00 00  14 00 00 00 00 00 00 00
0110  00 00 00 00 00 00 00 00  00 00 42 c8 00 00 00 00
0120  00 00 00 00 00 00 10 00  00 00 00 00 00 00 00 00
0130  45 1c 40 00 45 1c 40 00  00 00 00 00 00 00 00 00
0140  00 00 00 00 14 00 00 00  00 00 00 00 00 00 00 00
0150  00 00 00 00 00 00 42 48  00 00 00 00 00 00 00 00
0160  00 00 14 00 00 00 00 00  00 00 00 00 00 00 00 00
0170  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0180  11 00 45 cb 20 00 00 00  00 00 00 00 00 00 00 00
0190  00 00 00 00 00 00 00 00  00 00 00 00 00 00
```

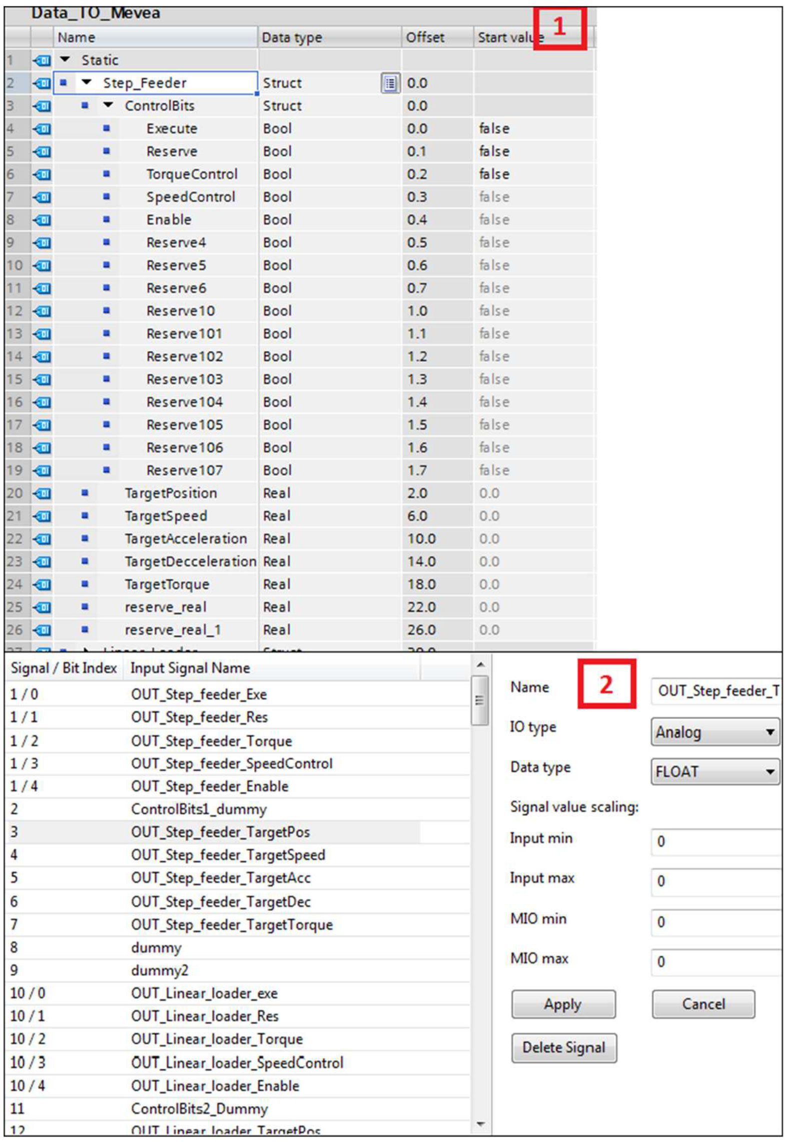
```
Frame 299871: 414 bytes on wire (3312 bits), 414 bytes captured (3312 bits) on interface 0
Ethernet II, Src: Siemens- 2c:14:ee (28:63:36:2c:14:ee), Dst: HewlettP 14:04:a2 (2c:41:38:14:04:a2)
Internet Protocol Version 4, Src: 192.168.0.1, Dst: 192.168.0.20
Transmission Control Protocol, Src Port: 49157, Dst Port: 2000, Seq: 25508161, Ack: 12173633, Len: 360
Data (360 bytes)
```

Figure 16. *Raw data capture from Lathe PLC to PC*



```
364 143.271155    192.168.0.1    192.168.0.20   TCP    60 49157 → 2000 [ACK] Seq=1 Ack=1 Win=4096 Len=0
365 143.271828    192.168.0.1    192.168.0.20   TCP   414 49157 → 2000 [PSH, ACK] Seq=1 Ack=1 Win=4096 Len=360
367 143.291091    192.168.0.1    192.168.0.20   TCP   414 49157 → 2000 [PSH, ACK] Seq=361 Ack=135 Win=4096 Len=360
369 143.311603    192.168.0.1    192.168.0.20   TCP   414 49157 → 2000 [PSH, ACK] Seq=721 Ack=269 Win=4096 Len=360
371 143.331789    192.168.0.1    192.168.0.20   TCP   414 49157 → 2000 [PSH, ACK] Seq=1081 Ack=403 Win=4096 Len=360
373 143.351659    192.168.0.1    192.168.0.20   TCP   414 49157 → 2000 [PSH, ACK] Seq=1441 Ack=537 Win=4096 Len=360
366 143.282181    192.168.0.20   192.168.0.1    TCP   188 2000 → 49157 [PSH, ACK] Seq=1 Ack=361 Win=64240 Len=134
368 143.294227    192.168.0.20   192.168.0.1    TCP   188 2000 → 49157 [PSH, ACK] Seq=135 Ack=721 Win=63880 Len=134
370 143.314177    192.168.0.20   192.168.0.1    TCP   188 2000 → 49157 [PSH, ACK] Seq=269 Ack=1081 Win=63520 Len=134
372 143.331852    192.168.0.20   192.168.0.1    TCP   188 2000 → 49157 [PSH, ACK] Seq=403 Ack=1441 Win=63160 Len=134
16710 278.484785    192.168.0.2    192.168.0.20   TCP    60 51607 → 2001 [ACK] Seq=1 Ack=1 Win=8192 Len=0
16711 278.493356    192.168.0.2    192.168.0.20   TCP   444 51607 → 2001 [PSH, ACK] Seq=1 Ack=1 Win=8192 Len=390
16715 278.513556    192.168.0.2    192.168.0.20   TCP   444 51607 → 2001 [PSH, ACK] Seq=391 Ack=131 Win=8192 Len=390
16720 278.534092    192.168.0.2    192.168.0.20   TCP   444 51607 → 2001 [PSH, ACK] Seq=781 Ack=261 Win=8192 Len=390
16724 278.554443    192.168.0.2    192.168.0.20   TCP   444 51607 → 2001 [PSH, ACK] Seq=1171 Ack=391 Win=8192 Len=390
16728 278.569806    192.168.0.2    192.168.0.20   TCP   444 51607 → 2001 [PSH, ACK] Seq=1561 Ack=521 Win=8192 Len=390
16713 278.493881    192.168.0.20   192.168.0.2    TCP   184 2001 → 51607 [PSH, ACK] Seq=1 Ack=391 Win=64240 Len=130
16717 278.513666    192.168.0.20   192.168.0.2    TCP   184 2001 → 51607 [PSH, ACK] Seq=131 Ack=781 Win=63850 Len=130
16721 278.534220    192.168.0.20   192.168.0.2    TCP   184 2001 → 51607 [PSH, ACK] Seq=261 Ack=1171 Win=63460 Len=130
16725 278.554503    192.168.0.20   192.168.0.2    TCP   184 2001 → 51607 [PSH, ACK] Seq=391 Ack=1561 Win=63070 Len=130
16729 278.569870    192.168.0.20   192.168.0.2    TCP   184 2001 → 51607 [PSH, ACK] Seq=521 Ack=1951 Win=64240 Len=130
```

Figure 17. *Collection of captures of sent and received data*



Data_TO_Mevea

	Name	Data type	Offset	Start value
1	Static			
2	Step_Feeder	Struct	0.0	
3	ControlBits	Struct	0.0	
4	Execute	Bool	0.0	false
5	Reserve	Bool	0.1	false
6	TorqueControl	Bool	0.2	false
7	SpeedControl	Bool	0.3	false
8	Enable	Bool	0.4	false
9	Reserve4	Bool	0.5	false
10	Reserve5	Bool	0.6	false
11	Reserve6	Bool	0.7	false
12	Reserve10	Bool	1.0	false
13	Reserve101	Bool	1.1	false
14	Reserve102	Bool	1.2	false
15	Reserve103	Bool	1.3	false
16	Reserve104	Bool	1.4	false
17	Reserve105	Bool	1.5	false
18	Reserve106	Bool	1.6	false
19	Reserve107	Bool	1.7	false
20	TargetPosition	Real	2.0	0.0
21	TargetSpeed	Real	6.0	0.0
22	TargetAcceleration	Real	10.0	0.0
23	TargetDecceleration	Real	14.0	0.0
24	TargetTorque	Real	18.0	0.0
25	reserve_real	Real	22.0	0.0
26	reserve_real_1	Real	26.0	0.0

Signal / Bit Index	Input Signal Name
1 / 0	OUT_Step_feeder_Exe
1 / 1	OUT_Step_feeder_Res
1 / 2	OUT_Step_feeder_Torque
1 / 3	OUT_Step_feeder_SpeedControl
1 / 4	OUT_Step_feeder_Enable
2	ControlBits1_dummy
3	OUT_Step_feeder_TargetPos
4	OUT_Step_feeder_TargetSpeed
5	OUT_Step_feeder_TargetAcc
6	OUT_Step_feeder_TargetDec
7	OUT_Step_feeder_TargetTorque
8	dummy
9	dummy2
10 / 0	OUT_Linear_loader_exe
10 / 1	OUT_Linear_loader_Res
10 / 2	OUT_Linear_loader_Torque
10 / 3	OUT_Linear_loader_SpeedControl
10 / 4	OUT_Linear_loader_Enable
11	ControlBits2_Dummy
12	OUT_Linear_loader_TargetPos

Name	OUT_Step_feeder_T
IO type	Analog
Data type	FLOAT
Signal value scaling:	
Input min	0
Input max	0
MIO min	0
MIO max	0

Figure 18. *Axis data structure on PLC side (1) and on PC side (2).*

To calculate the protocol stack efficiency, the payload size must be divided by the actual frame size and multiplied by 100%. The answer seems trivial, both, the frame sizes and payload sizes are shown in Figure 17. As mentioned earlier, there are parts of the Ethernet protocol frame that are not shown by the analysis tool. The CRC, preamble and SFD surely need to be added to the Ethernet frame size shown in Figure 17**.** Adding these components increase the frame size shown in Figure 17 by 12 bytes. Also, as each of the messages need 12-byte inter-frame gap (IFG) before sending another packet, the effect on efficiency of the interpacket gap should be also taken in consideration.

With the payload size and the actual packet size the protocol efficiency can be calculated by

$$\text{Protocol stack efficiency} = \frac{\text{Payload size}}{\text{Packet size}} * 100\% \qquad (1)$$

where the payload size is the actual data that is desired to be transmitted. In the 0 is calculated the relevant TCP/IP packet efficiencies using equation 1.

Table 3. *TCP/IP packet efficiencies*

	Payload size	**Frame size**	**Size with IFG**	**Effi-ciency**	**Effi-ciency with IFG**
Charger PLC to simulation	360	426	438	84,5 %	**82,2 %**
Simulation to charger PLC	134	200	212	67,0 %	**63,2 %**
Lathe PLC to simulation	390	454	468	85,9 %	**83,3 %**
Simulation to Lathe PLC	130	196	208	66,3 %	**62,5 %**
Maximum packet size	1460	1526	1536	95,7 %	**95,1 %**
Minimum packet size	6	72	84	8,3 %	**7,1%**

This way calculated efficiency is 63,2% at worst and 82,2% at best, not taking interpacket gap in calculation. Whereas the best possible efficiency for TCP/IP over Ethernet package with 1460-byte payload and 1526-byte packet size is 95,1%.

Even though the communication is not robust send / reply loop that sends only as response, the communication delay is an interesting parameter to be measured. Typically, the interesting parameter is so-called loopback duration, which measures the time from request to reply. As the sequence is not request-reply, but rather two hosts pushing each other information which is sent on certain interval, the most interesting variable for real-time control is the average and maximum time interval that it takes for control system to get information from simulation and maximum and average time that the control information reaches the simulation.

Wireshark is a tool that is used to capture and analyse the communication parameters. For analyse, only the messages with "push" flags are interesting, as only these contain data from simulation or control data to simulation. The analysis dataset contains tens of thousands of rows of data which is to be analysed, the relevant information of the dataset can be seen in Table 4.

Table 4. *Table showing average TCP push intervals between various components*

	Dataset count	**Average delay (ms)**	**Max delay (ms)**	**Min delay (ms)**	**Variance (ms)**
PC to Charger	7907	15,61	35,34	1,39	0,01742
PC to Lathe	9993	15,61	38,85	2,00	0,01700
Charger to PC	12298	10,03	31,11	7,98	0,00094
Lathe to PC	15426	10,11	20,88	7,54	0,00154

Assuming the connection delay on wire to be 1ms, the status information at any given moment from simulation is 8,8ms old information on average, 40ms old on maximum and 1ms on minimum. Assuming the processing and sending of TCP/IP packet on PC side doesn't have any internal delay, meaning that when packet is sent from PC the information is up to date. Also, there is program cycle delay in the PLC end of the system, data is only read at the time the receiving function block is called. If the data reaches PLC right after the receiving function is called, it takes full program cycle before receiving function block is called again and data is registered to database.

The table shows that PLC to PC connections work much more robustly, having lower average time and having over tenfold lower variance. The difference in robustness is not a surprise, as the traditional computer isn't meant for real-time control. The PC did however, send ACK messages with no data to PLC, which are not accounted in this calculation. Also, the accuracy of the capturing software is unknown. To understand the oddities, such as minimum delay being less than 10ms in PC side socket client even though the supposed interval is 10ms, deeper understanding in processor core functioning and see the underlying socket client program code is needed.

The throughput is calculated from the use of the actual bandwidth. The throughput calculation can be interesting to see how much does the communication use of the given transmission medium. In the framework of this thesis the communication medium is Ethernet CAT 5e cable having full duplex features. The bottleneck of the communication is from switch to PC where one medium carries the communication of both PLCs. As the medium is full duplex, it is not interesting to sum all the communications together, only one directional communication together. The throughput can be calculated from formula

$$\text{Throughput}=\frac{\text{Packet size}}{\text{Send interval}} \tag{2}$$

where the packet size in this case is the total size of the package without interpacket gap. The ideal and actual interval is discussed in the following chapter.

The ACK packets also use the transmission medium, so the packets should be taken in consideration in the actual throughput calculation. The throughput of ACK packets is calculated from average time to send one of ACK messages to either of the PLCs. There is no need to separate these packets in calculation, as they are all the same size and share the same direction in this medium. ACK package has the size of minimum TCP/IP over Ethernet packet, which is 66 bytes. The relevant throughputs in the PC – switch channel is calculated in the Table 5 using equation 2.

Table 5. *Throughput of each of the connections and total throughput in the PC – switch channel*

Packet from / to	**Package size without IFG (bits)**	**Throughput with ideal 10ms interval (bits/s)**	**Throughput with actual interval (bits/s)**
Lathe PLC to PC	3632	363200	359250
Charger PLC to PC	3408	340800	339638
Total from PLCs to PC	**7040**	**704000**	**698888**
PC to Lathe PLC	1568	156800	100475
PC to Charger PLC	1600	160000	102528
ACK packages (to lathe and charger)	528	0	34887
Total from PC to PLCs	**3696**	**316800**	**237890**

The actual total throughput from PLCs to PC is 698,9Kbit/s (Table 5). Kilobit (Kbit) is 1000 bits and megabit (Mbit) is 1000 kilobits, not to be confused with kilobytes and megabytes which have 1024 conversion ratio. The channel usage $C\%_{IN}$ of the 100Mbit/s channel in percentage is:

$$C\%_{IN}=\frac{698{,}9/1000}{100}*100\%=0{,}699\% \qquad (3)$$

for incoming packets to PC. The total usage of the outgoing channel $C\%_{OUT}$ from PC can be calculated similarly:

$$C\%_{OUT}=\frac{237{,}9/1000}{100}*100\%=0{,}238\% \qquad (4)$$

With interframe gaps the percentages are slightly greater, but with these usage percentages and transfer intervals the transmission medium usage would still be relatively low.

4.2 Considerations on PLC program

The PLC program ends up being somewhat mixed composition of easily detachable parts of code and some changes inside the code. The program ends up working as intended since the control program cycle works similarly to the real system. However not all the aspects could be tested yet in the simulation system, the centring of the block to middle of the lathe is only done by perfectly cylindrical blocks since the simulation was yet only able to produce only perfectly cylindrical blocks. Captures of peeling machine simulation in operation can be seen in Figure 19, Figure 20 and Figure 21**.** In Figure 19 the block is in centring spindles (often called XY-spindles) being centred to be moved by the transfer arms to lathe spindles to be peeled. In Figure 20 the block is being peeled, so axes are on tracking mode and the back-up devices are supporting the peeling. In Figure 21 the out-coming veneer ribbon can be seen as the block is being peeled, the figure also shows the whole peeling machine.

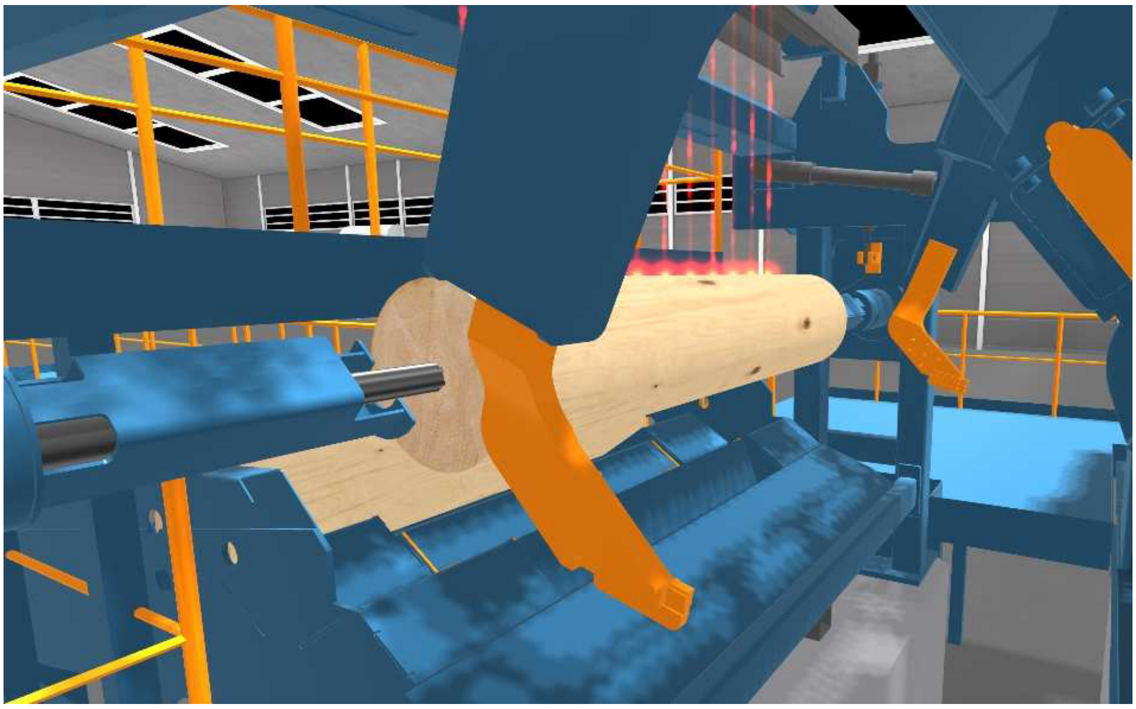


Figure 19. *Block in centring spindles*

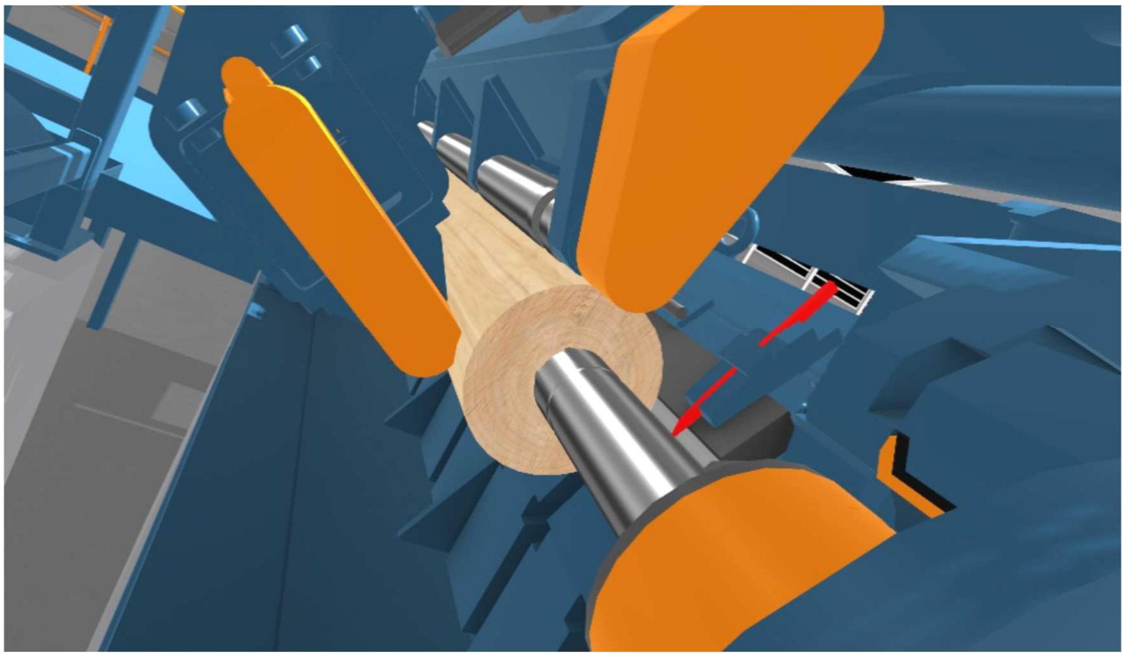


Figure 20. *Block being peeled*

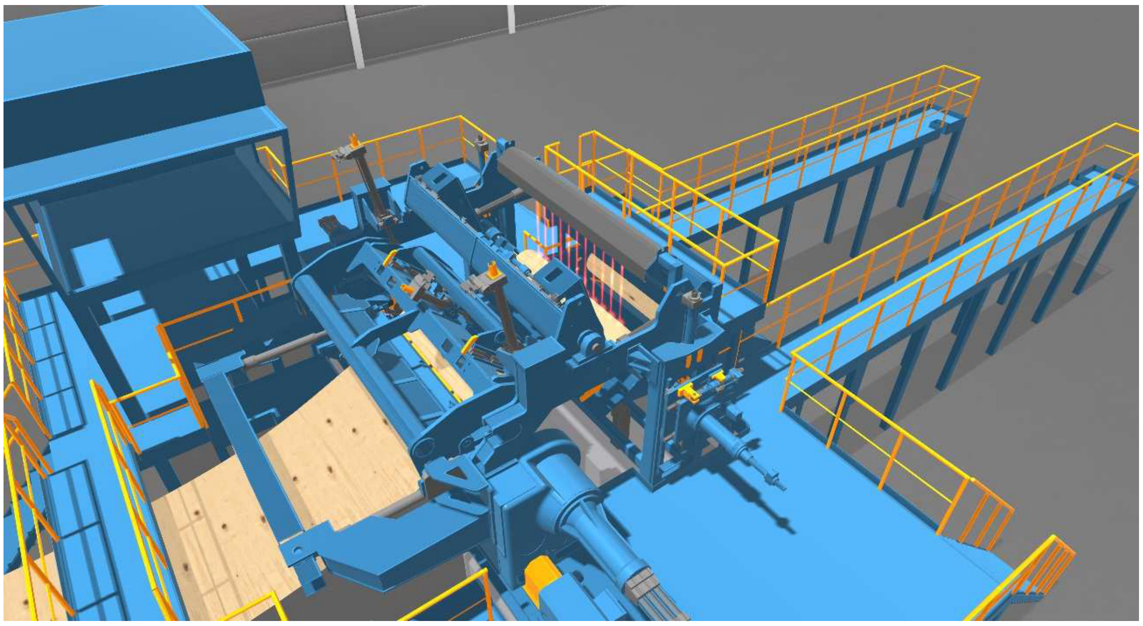


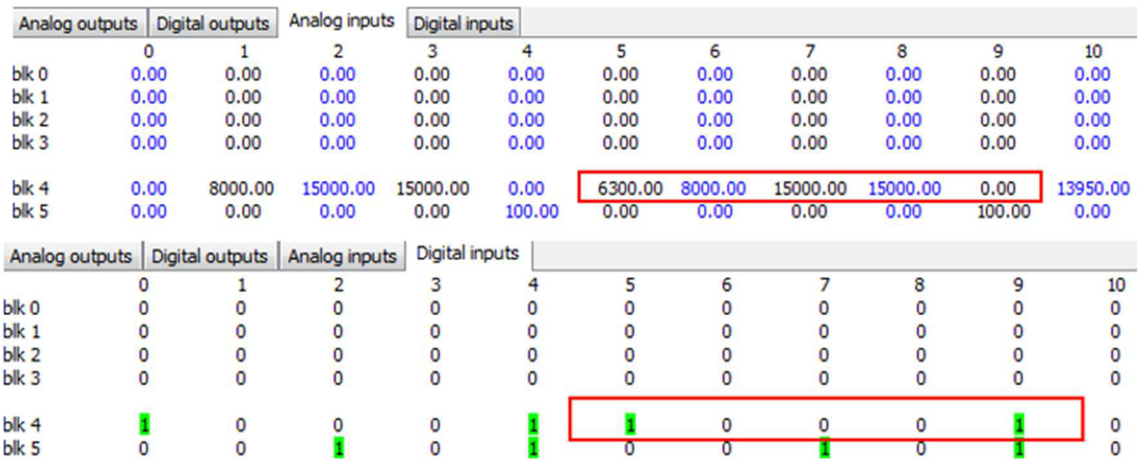
Figure 21. *Outline of peeling machine*

The HMI program works as intended on most parts, including parametrizing the offsets, peeling thickness and peeling speed. However, some alarm messages of the program leak to HMI screen. The alarms typically are about lubrication error or pump not running, which should not be a problem in simulated environment.

The PLC program can be studied by letting people use the system who have used the system before. The system was tested by the Raute staff who have designed peeling machine programs before. The testing resulted in comments such as "this axis movement should be faster", "this action is lacking" or "this movement is to wrong direction". This resulted with a program that imitate real system actions much more accurately. In the

Ligna fair, the system was tested by corporative CEOs, engineers, children and random people who just happened to pass by. The control program state stayed in designed limits of the real peeling machine, even with tens of people testing the simulator. So it is safe to say that the program controls function reasonably well and the control state machines and sequences work as intended.

Example of a movement that is executed by the position control command can be seen in the figures below. The Figure 22 shows a capture from Mevea IO pool, where the control script reads the control parameters. In the Figure 22 target values can be seen in the order that is explained in the chapter 3.1.2, target speed being the second value of the frame, and thus the second value of the values that are squared. The digital inputs in the Figure 22 lower part showing the control bit frame for linear feeder axis movement. At the time of the capture the linear feeder axis control bits are commanding the linear feeder for position control movement since only enable and execute bit are “1”. Figure 23 shows the output control voltage to 4/3 valve controlling the movement of the “linear feeder” axis, this control voltage is controlled in the scripts defining the real-time control. Figure 24 showing the linear feeder axis velocity. As can be seen that the speed control reaches the 8000 as maximum speed which is the target speed of the control value frame shown in Figure 22.



	0	1	2	3	4	5	6	7	8	9	10
blk 0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
blk 1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
blk 2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
blk 3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
blk 4	0.00	8000.00	15000.00	15000.00	0.00	6300.00	8000.00	15000.00	15000.00	0.00	13950.00
blk 5	0.00	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00	100.00	0.00

	0	1	2	3	4	5	6	7	8	9	10
blk 0	0	0	0	0	0	0	0	0	0	0	0
blk 1	0	0	0	0	0	0	0	0	0	0	0
blk 2	0	0	0	0	0	0	0	0	0	0	0
blk 3	0	0	0	0	0	0	0	0	0	0	0
blk 4	1	0	0	0	1	1	0	0	0	1	0
blk 5	0	0	1	0	1	0	0	1	0	1	0

Figure 22. *Capture of Mevea IO pool containing control information*

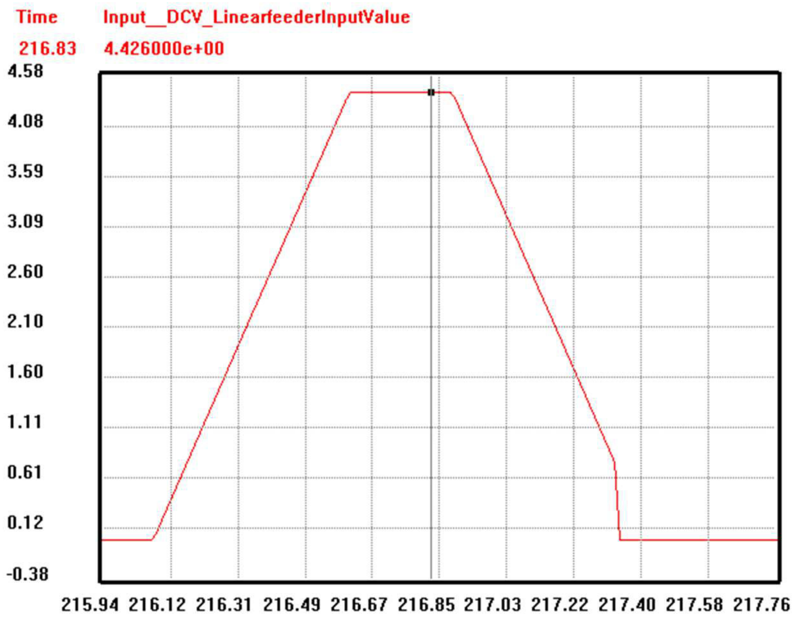


Figure 23. *Control voltage to simulated valve of linear feeder*

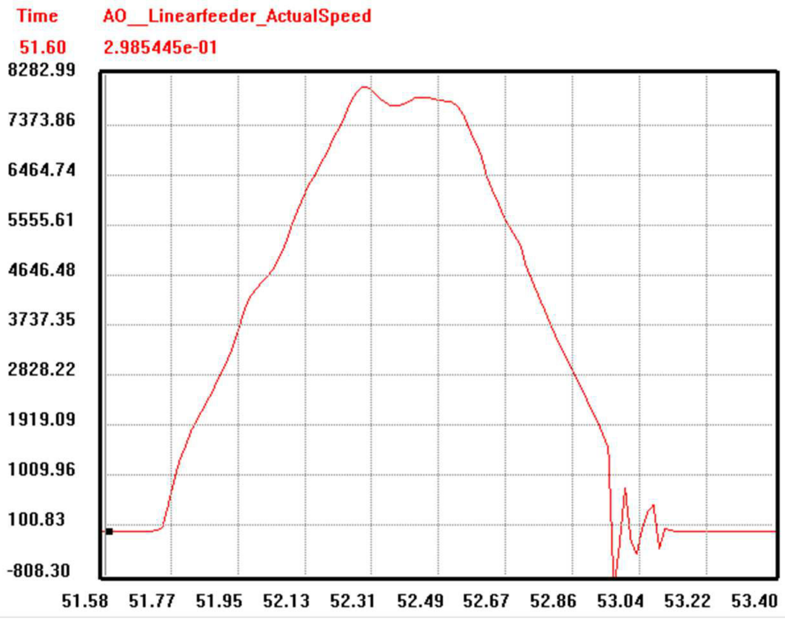


Figure 24. *Plot of actual speed of linear feeder cylinder*

5. CONCLUSION AND FURTHER WORK

The resulting control works as is desired to work, since the actions of the automatic peeling cycle as well as the joystick movements work similar as they would in real machine. The communication interface could work faster, in more robust fashion and could have less data transmitted, but for this application these imperfections are acceptable since it is not used for real-time control.

The objective of the thesis was to create functioning interface from PLC to simulation to present a peeling machine simulator in Ligna fair. The TCP/IP socket interface is used as there is no other options available for the used PLCs. The interface resulted in two interfaces, one for each PLC. The interfaces have one interface to send and one to receive data. Further, each of the interfaces are split to similar interfaces among each axis. The resulting interface has some excess information as not all the axes need all the information that is exchanged, but each interface is similar from one to another and the motion control functions are easier to program to simulator scripts. Also, the adding of a new axis with new motion control script is easier using interface with similar interfaces.

Another objective was to modify the control program to work with the simulation, meaning that the inputs come from simulation and outputs control the axes in the simulation. The changes needed to the actual program included mostly forcing some input bits and bypassing the input data to come from simulation. For control the actual program didn't need to be changed, only functions which changed the interface control data as the command triggered in the actual program.

For reusability, PLC program for simulation communication is done to be easily detachable from the actual control program and easily attachable to new program, still requiring some expertise to detach and attach the right parts. Also, some overwriting and removal of some code manually is still necessary, but the improving code reusability could be point of improvement. Naturally the axes would need to be redefined for another project.

The goal to get everything working smoothly and to look nice so that the simulator can be presented in Ligna fair was a success. The schedule for testing was tight as the system could only be tested with the physical controls for only two days before sending it off to the fair. After the system was presented, the development was continued to other aspects.

Used TCP/IP connection provided reliable connection with high baud rate and sufficiently small connection delay. The send rate on both ends could be faster, and would be needed to be faster on some other applications, but for this application the send rate is sufficient. The interface has some downsides, such as it could be downsized and optimized, but the fact that the interface is the same for all the axes eases the use of the interface.

One goal was to study if the created simulator system is applicable to be used as anything else but fair presentation. From the study it seems that there are more applications for simulator than for what it is used currently. The use of the simulation as program or product development tool takes lots of bravery and trust in new technology from the company.

5.1 Points of development and further work

Even now as the communication interface is ready and working there are some areas that could be developed further or tested. The goal is to improve the reliability, reusability and add some new features to interface.

Even as the interface on PLC side is already easily detachable to be used on another application, the interface should be developed so that the interface is a neat package which could be detached to a new machine control program where only standardized reconfiguration is needed. This reconfiguration would include, the number of axes to be controlled, the end positions of axes, the control type of each axis and which signal is used for each command. All the parameters to be reconfigured should be configured as easily as possible. The package could have some documented instructions how to attach the communication interface to a project and how to reconfigure the parameters.

Another development is in the communication between the simulator and PLC. The communication between the two seems excessive, as all the data is sent every cycle even though no change has happened. The interface could be changed to send values only when parameters of one axis has changed, or the interface could only send the command number which would then be read on the PC side and interpret the command values from the command number. These changes would reduce the amount of data transmission between the two and make the configuring of the axis movements much easier. Another thing is to reduce the hold time of execute bit, this could be done by some response bit telling that the rising edge of execute bit has been noted and the execute bit could be set back to zero afterwards. The reduction in execute bit hold time could increase the amount of commands that can be sent to one axis in certain time period.

The interface could carry more diagnostic information from the simulation. These diagnostics could be the status or error information of the given device which is normally read from the device interface. If the status and error information is received from the system as it is from the real system, the program response to the information could be studied. These statuses could be used for previously mentioned model implemented fault injection to observe the response to fault.

As further work the virtual machine applications should be applied to machine. Currently the simulator is used as fair presentation machine to promote sales. The simulator could be used to debug the peeling machine logic program and to improve the core program without the risk of collisions or delay in delivery time.

REFERENCES

Alani, M.M., 2014. Guide to OSI and TCP/IP models, Springer-briefs in computer science. Springer, Cham Heidelberg.

Ameri, F., Dutta, D., 2005. Product Lifecycle Management: Closing the Knowledge Loops. Comput.-Aided Des. Appl. 2, 577–590.

Auweraer, H.V. der, Donders, S., Mas, P., Janssens, K., 2008. Breakthrough Technologies for Virtual Prototyping of Automotive and Aerospace Structures, in: Product Engineering. Springer, Dordrecht, pp. 397–418. https://doi.org/10.1007/978-1-4020-8200-9_20

Bartak, J., Chaumes, P., Gissinger, S., Houard, J., Houte, U. van, 2000. Operator training tools for the competitive market. IEEE Comput. Appl. Power 13, 25–31. https://doi.org/10.1109/67.849022

Delamer, I.M., Martínez Lastra, J.L., 2007. Factory information systems in electronic production, 1st ed. Tampere University of Technology, Tampere, Finland.

Donahoo, M.J., Calvert, K.L., 2009. TCP/IP sockets in C: practical guide for programmers, 2. ed. ed, The Morgan Kaufmann practical guides series. Morgan Kaufmann, Amsterdam.

Dubey, A., 2011. Evaluating software engineering methods in the context of automation applications. IEEE, pp. 585–590. https://doi.org/10.1109/INDIN.2011.6034944

Edwards, J., Bramante, R., 2009. Networking self-teaching guide: OSI, TCP/IP, LANs, MANs, WANs, implementation, management, and maintenance. Wiley, Indianapolis, Ind.

Gerlach, I., Tholin, S., Hass, V.C., Mandenius, C.-F., 2016. Operator Training Simulator for an Industrial Bioethanol Plant. Process. Basel 4, 34. http://dx.doi.org.libproxy.tut.fi/10.3390/pr4040034

Henning, C., 2016. SWITCHED ETHERNET NETWORKS FOR PROFINET DETERMINISM.

Hloska, J., Kubín, M., 2014. Virtual Commissioning of Mechatronic Systems with the Use of Simulation, in: Mechatronics 2013. Springer, Cham, pp. 33–40. https://doi.org/10.1007/978-3-319-02294-9_5

Ko, M., Park, S.C., 2014. Template-based modeling methodology of a virtual plant for virtual commissioning. Concurr. Eng. 22, 197–205. https://doi.org/10.1177/1063293X14531423

Koponen, H., 1998. Puulevytuotanto. Gummerrus Kirjapaino Oy.

Leino, S.-P., 2015. Reframing the value of virtual prototyping. Intermediary virtual prototyping - the evolving approach of virtual environments based virtual prototyping in the context of new product development and low volume production. Tampere University of Technology.

Markovič, J., Popovič, R., Trebuňa, P., Pekarčíková, M., Kliment, M., 2015. Virtual Commissioning as a Part of Mechatronical System. Appl. Mech. Mater. 816, 521–525. https://doi.org/10.4028/www.scientific.net/AMM.816.521

Mehta, B.R., Reddy, Y.J. (Eds.), 2015. Industrial Process Automation Systems: Design and Implementation. Butterworth-Heinemann, Oxford. https://doi.org/10.1016/B978-0-12-800939-0.00030-9

Miller, P., Cummins, M., 2000. LAN technologies explained. Digital Press, Boston, Mass.

Oppelt, M., Urbas, L., 2014. Integrated virtual commissioning an essential activity in the automation engineering process: From virtual commissioning to simulation supported engineering. IEEE, pp. 2564–2570. https://doi.org/10.1109/IECON.2014.7048867

Pigan, R., Metter, M., 2008. Automating with PROFINET: industrial communication based on industrial Ethernet, 2., rev. and extended ed. ed. Publicis Publ, Erlangen.

PROFINET System Description, System manual, 2018.

Reinhart, G., Wünsch, G., 2007. Economic application of virtual commissioning to mechatronic production systems. Prod. Eng. 1, 371–379. https://doi.org/10.1007/s11740-007-0066-0

RMC70/150 Motion Controllers And RMCTools Software User Manual, 2017.

Rösch, S., Vogel-Heuser, B., 2017. A Light-Weight Fault Injection Approach to Test Automated Production System PLC Software in Industrial Practice. Control Eng. Pract. 58, 12–23. https://doi.org/10.1016/j.conengprac.2016.09.012

Shahim, N., Moller, C., 2016. Economic justification of Virtual Commissioning in automation industry. IEEE, pp. 2430–2441. https://doi.org/10.1109/WSC.2016.7822282

Shi, S., Walker, J.C.F., 2006. Wood-based composites: plywood and veneer-based products, in: Primary Wood Processing. Springer, Dordrecht, pp. 391–426. https://doi.org/10.1007/1-4020-4393-7_11

Stone, R., 2001. Virtual reality for interactive training: an industrial practitioner's viewpoint. Int. J. Hum.-Comput. Stud. 55, 699–711. https://doi.org/10.1006/ijhc.2001.0497

Varis, R., 2017. Puulevyteollisuus. Suomen Puuteollisuusinsinöörien Yhdistys Ry.

Vergnano, A., Berselli, G., Pellicciari, M., 2017. Interactive simulation-based-training tools for manufacturing systems operators: an industrial case study. Int. J. Interact. Des. Manuf. IJIDeM 1–13. https://doi.org/10.1007/s12008-016-0367-7

Vieira, M.F.Q., Neto, J.A.N., Scaico, A., Santoni, C., Mercantini, J.-M., 2010. A Real-time Interface Simulator for Operator Training: A Proposed Architecture. SIMULATION 86, 53–63. https://doi.org/10.1177/0037549709346281

Wilamowski, B.M., Irwin, J.D. (Eds.), 2011. Industrial communication systems, 2. ed. ed, The industrial electronics handbook. CRC Press, Boca Raton, Fla.

Zucker, G., Dietrich, D., 2011. ISO/OSI Model, in: Industrial Communication Systems, Electrical Engineering Handbook. CRC Press, pp. 1–9. https://doi.org/10.1201/b10603-3

Zurawski, R. (Ed.), 2015. Industrial communication technology handbook, Second ed. ed, Industrial information technology series. CRC Press, Boca Raton.