TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Maksim Moissejev 142667IASB

3D CRANE CONTROL USING ABB AC500 CONTROLLER

Bachelor`s thesis

Supervisor: Kristina Vassiljeva
PhD

Tallinn 2019

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Maksim Moissejev 142667IASB

3D KRAANA JUHTIMINE ABB AC500 KONTROLLERI ABIL

Bakalaureusetöö

Juhendaja: Kristina Vassiljeva
PhD

Tallinn 2019

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Maksim Moissejev

20.05.2019

Abstract

The main goal of this thesis is to develop a 3D Crane control system using ABB AC500 PLC. The sub-goals are to create a user interface and to realize an automatic control mode that moves the load to the desired set point and controls the load oscillations. The thesis explains hardware and software parts of the system, connections and program development process.

The end result of this work is a control program that allows operating the 3D Crane either manually or with the help of automation. The finished work can be used as a manual in setting up the ABB AC500 PLC.

This thesis is written in English and is 41 pages long, including 4 chapters, 37 figures and 6 tables.

Annotatsioon

3D kraana juhtimine ABB AC500 kontrolleri abil

Selle lõputöö põhieesmärgiks on luua 3D kraana juhtimissüsteem ABB AC500 kontrolleri abil. Alameesmärkideks on luua kasutajasõbralik kasutajaliides, mis võimaldaks kaugjuhtimist, ning luua automaatjuhtimisrežiimi, mis liigutaks koormuse soovitud kohale võnkumiseta. Antud töös on kirjeldatud 3D kraana – ABB AC500 PLC süsteemi riist- ja tarkvaraosasid ning ühendusi nende vahel, samuti juhtimisprogrammi ja selle arenguprotsessi.

Selle töö lõpptulemuseks on 3D kraana juhtimisprogramm, mis võimaldab juhtida kraanat nii käsitsi, kui ka automaatika abil. Lõputööd saab kasutada ABB AC500 kontrolleri seadistamise juhendina.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 41 leheküljel, 4 peatükki, 37 joonist, 6 tabelit.

List of abbreviations and terms

CODESYS	Controller Development System
CPU	Central Processing Unit
DC	Direct Current
FBP	FieldBusPlug
HMI	Human-Machine Interface
I/O	Input/Output
IDC	Insulation-Displacement Contact
IEC	International Electrotechnical Commission
PC	Personal Computer
PI	Proportional-Integral
PID	Proportional-Integral-Derivative
PLC	Programmable Logic Controller
PWM	Pulse-Width Modulation
RAM	Random Access Memory

Summary

The purpose of this work is to develop an application for controlling the 3D Crane using the ABB AC500 PLC system. Cranes are widely used in industry and the industrial environments are usually harsh and require higher reliability. For this reason, the author decided to use a PLC instead of a standard PC-based control system. This thesis describes hardware and software parts of the system, connections and control program development process.

As a result, a PLC-based control system was developed. It allows operating the crane either with mechanical controls or with a user interface. The user interface displays all the necessary information about the load position so the crane can be operated remotely. It ensures the safety of the crane operator when working in hazardous conditions. For remote control purposes, a web interface was also added. Moreover, an automatic control mode that moves the crane to the desired set point and controls the load oscillations was developed. For these needs, PID controllers were implemented and tuned.

However, this work does not cover the security issue. Cybersecurity is particularly relevant nowadays and understanding the importance of that, it will be the first problem to be solved in the future.

All in all, this work can be used as an ABB AC500 PLC setup manual.

Table of contents

List of figures

List of tables

1 Introduction

1.1 Background

Nowadays people can hardly imagine their lives without computers. Computer systems are everywhere and they do a lot of work. One of the most important applications of digital technology is the industry. As technology advances, more and more industrial processes become either fully automated or semi-automated and cranes are no exception.

Previously, the crane operator had to be an experienced, well-trained person because of the complexity of control and the number of factors required to minimize payload oscillations, saving the speed, safety, and accuracy. Today it becomes much easier to operate a crane, because control interfaces are more intuitive and user-friendly, while automation does most of the work.

The Control Systems Research Laboratory (A-lab) provides access to the real-time industrial overhead crane model (3D Crane), provided by Inteco, which acts as a control object. A-lab also provides access to the ABB AC500 PLC system, which can be used as a control device [1]. It allows the development of a PLC-based crane control system, which resulted in this thesis.

1.2 Objectives

The main goal of this thesis was to develop an application for controlling the 3D Crane using the ABB AC500 PLC system. The sub-goals of this thesis were to create an intuitive user interface that displays all the necessary information about the load position and to realize an automatic control mode that moves the load to the desired set point and controls the load oscillations.

2 System description

2.1 Inteco 3D Crane

The 3D Crane is a laboratory model of industrial overhead-gantry crane provided by Inteco. It comes equipped with a dedicated system of sensors, five incremental wheel encoders, and three DC motors. The model operates in real-time mode [2]. The 3D Crane setup is shown in Figure 1.

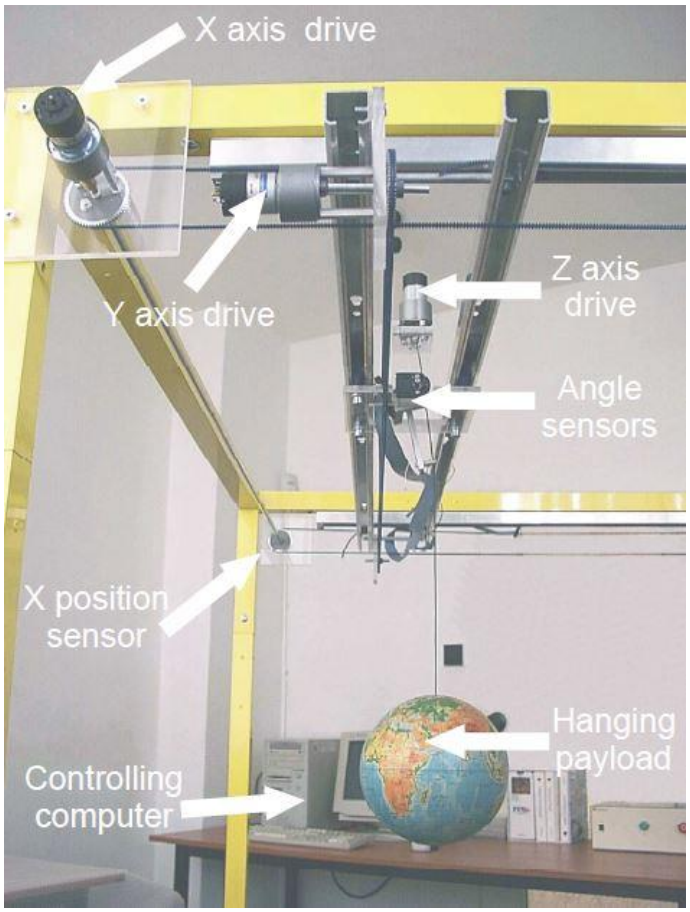


Figure 1. 3D Crane setup [3].

The 3D Crane consists of three moving parts: a payload, a rail, and a cart. The payload is hanging on a pendulum-like lift-line wound by a motor so it can be lifted and lowered in the z-direction. The motor is mounted on the cart. The cart can travel along the rail in the y-direction. The rail with the cart can move in the x-direction. Therefore, the payload

attached to the end of the lift-line is capable of three-dimensional motion. Three DC motors drive the 3D Crane: two are used for driving the cart in the x and y directions and the last one is used to lift and lower the load in the z-direction.

There are five incremental encoders measuring five state variables: the cart coordinates on XY-plane, the lift-line length and two deviation angles of the payload. Together with a specialized mechanical solution, they create a unique measurement unit. The encoders measure movements with a resolution equal to 4096 pulses per rotation and the deviation of the load with accuracy equal to 0.0015 rad. There are also three position limit switches used to determine a reference position.

The system has a power interface. The power interface amplifies the control signals transmitted from a controller to the DC motors and converts the encoder pulse signals to the digital 16-bit form to be read by the controller.

Originally, a PC equipped with an RT-DAC/PCI multipurpose digital I/O board is used as the controller. The whole logic necessary to read the encoder signals and to generate the PWM pulses to control the DC motors is configured in the Xilinx® chip of the RT-DAC/PCI board. The function of the board is managed by the 3DCrane Toolbox, which is integrated into the MATLAB®/Simulink® environment [3].

However, in this project it was decided to use a PLC instead of PC with RT-DAC/PCI board to control the 3D Crane. For these needs, Inteco offers a PLC to 3D Crane interface.

2.2 PLC to 3D Crane interface

PLC to 3D Crane interface is a device that enables a PLC to communicate with the power interface of the 3D Crane. It provides conversion between a digital I/O voltage standard accepted by PLC (+24V) and the voltage standard accepted by the power interface. The power interface connects to the rear panel of PLC to 3D Crane interface through a 40-pin ribbon cable compatible with the IDC standard. A front panel of PLC to 3D Crane interface is used to connect the PLC. The front panel consists of five plug-in terminal blocks designed for wire diameter range 0.5±1.5 mm^2. Signal pins are used to interact with the power interface [4]. Description of all used signals is shown in Table 1.

Table 1. Description of signals [4].

Signal	**Description**
EncA1_X	Incremental encoder for X-axis: wave A.
EncB1_X	Incremental encoder for X-axis: wave B.
EncA2_Y	Incremental encoder for Y-axis: wave A.
EncB2_Y	Incremental encoder for Y-axis: wave B.
EncA3_Z	Incremental encoder for Z-axis: wave A.
EncB3_Z	Incremental encoder for Z-axis: wave B.
EncA4_AX	Incremental encoder for AX-angle: wave A.
EncB4_AX	Incremental encoder for AX-angle: wave B.
EncA5_AY	Incremental encoder for AY-angle: wave A.
EncB5_AY	Incremental encoder for AY-angle: wave B.
Switch1_Z	Position limit switch of the Z-axis.
Switch2_Y	Position limit switch of the Y-axis.
Switch3_X	Position limit switch of the X-axis.
+24V	Power supply from PLC.
GND_IN	Ground level for the PLC digital inputs.
GND_OUT	Ground level for the PLC digital outputs.
PWM_Z	PWM control signal for the DC motor of the Z-axis.
Dir_Z	Signal changing the rotational direction of the Z-axis DC motor.
PWM_Y	PWM control signal for the DC motor of the Y-axis.
Dir_Y	Signal changing the rotational direction of the Y-axis DC motor.
PWM_X	PWM control signal for the DC motor of the X-axis.
Dir_X	Signal changing the rotational direction of the X-axis DC motor.

2.3 ABB AC500 PLC

The ABB AC500 is a complex state-of-the-art PLC system, so it is important to understand the basics of PLC operation to get a better understanding of the AC500 product family [5].

2.3.1 PLC

A PLC (Programmable Logic Controller) is an industrial computer adapted for automation processes and real-time control systems. The main difference from general-purpose computers is that PLCs are robust and can survive harsh conditions. They are highly reliable and designed to handle many inputs and outputs. Most of PLCs today are modular, so it allows using them in various setups.

Typical PLC system consists of a CPU (Central Processing Unit), a memory, an I/O (Input/Output) device, a power supply, and a rack. The CPU executes the program and operates the process. The memory stores user program and work data. The I/O device receives and sends signals to other devices (allows PLC to interact with external equipment). The power supply provides the PLC with the necessary voltage and current. The rack creates an electrical connection between all of the modules and holds them together [6], [7].

PLC works cyclically as long as the system is running. The cycle consists of three steps: scan inputs, execute logic and update outputs. The amount of time it takes is called scan cycle time. The scan cycle time depends on the device and the lengths of a program. It is crucial to know about its duration if timers and delays are used in the program. The scan cycle time is typically a few milliseconds [8].

2.3.2 ABB AC 500

As it was mentioned above, the ABB AC500 is a complex, state-of-the-art PLC system. The AC500 was designed to provide a reliable and powerful platform for creating scalable, cost-effective and flexible automation solutions. The whole AC500 product family consists of modules that can be scaled and combined to fit the desired requirements. The AC500 can be used both for simple tasks, such as environmental monitoring, traffic signals or crane control and for complex tasks, like robotics, building maintenance or complicated production line control [9].

The basic components of the AC500 system are a processor module (CPU), a communication module, and an S500 I/O module. The principal system structure is shown in Figure 2.

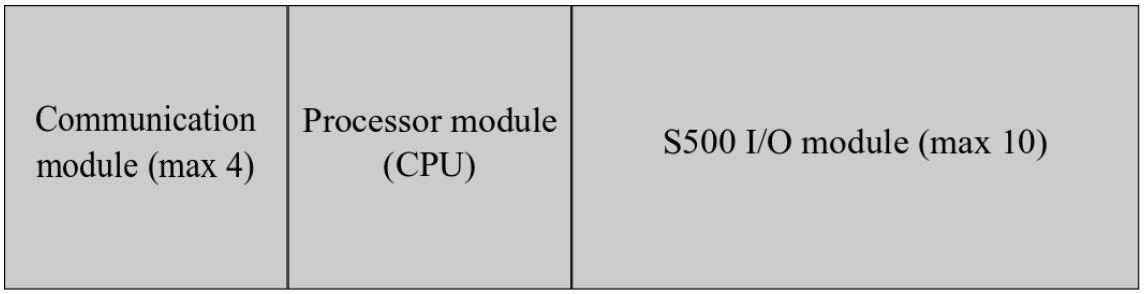


Figure 2. AC500 structure.

Similar to other computers, the core of the system is the processor module. Processor modules are available in various performance classes. Types differ in their speed, memory size, features, and networking capabilities. Each processor module has to be mounted on a terminal base and supplied with 24VDC. The terminal base type depends on the number of communication modules used together with a processor module and on the network interface type of a processor module. The maximum number of communication modules that can work with a single processor module is four.

Processor module functions are [5]:

- To receive user program and data entered
- To diagnose work faults of the power supply and PLC circuit as well as syntax error in programming
- To receive the state or data of the site via the input interface and save it into the shadow register or data register
- To read the user program in the memory one by one and execute it after interpretation
- To update the state of related flag bits and output shadow register contents according to execution results and realize output control by means of output unit.

Communication modules are mounted on the left side of the processor module on the same terminal base. They are required for a connection to standard field bus systems, for example, PROFIBUS®, CANopen®, DeviceNet™, PROFINET®, EtherCAT® and for integration into existing networks. Communication on different field buses is also possible. Communication between the processor module and the communication module is carried out through a communication module bus that is integrated into the terminal

base. The data interchange is realized by a dual port RAM. Communication modules are powered via the communication module bus, so a separate voltage source is not required.

The last components of the AC500 system are the S500 I/O modules. They act as messengers between the processor module and external I/O devices. They are designed to increase the number of channels for incoming and outgoing information. The S500 I/O modules are divided into analog and digital, however, there are modules that work simultaneously with both analog and digital signals. I/O modules are mounted on the right side of the processor module. They can be either simply plugged into a terminal unit (for central I/O expansion) or connected via an FBP interface module (as decentralized I/O expansion). The maximum number of I/O modules supported by a single processor module is ten [5].

In this project, the AC500 system (See Figure 3) consists of five modules presented in Table 2.

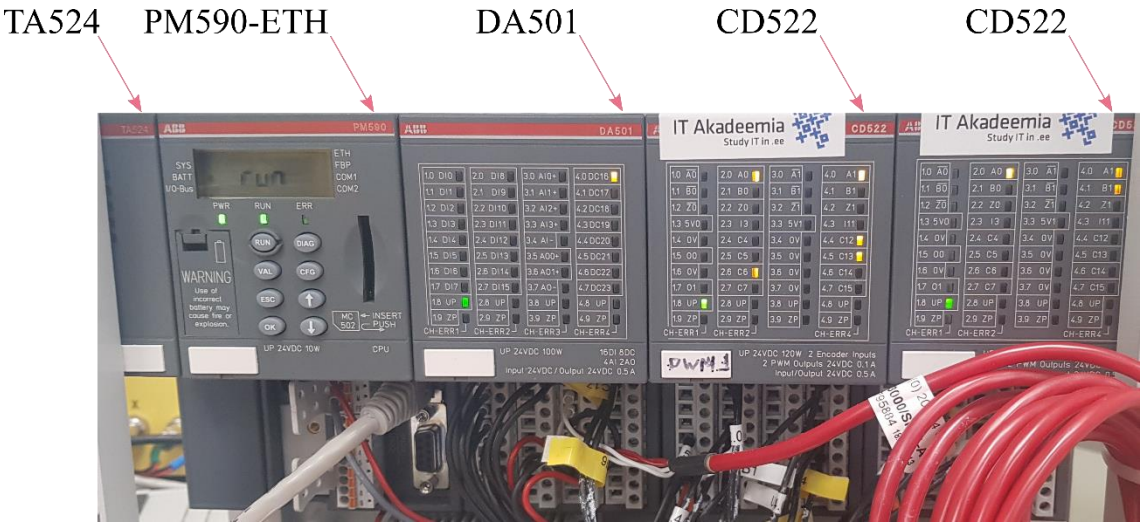


Figure 3. ABB AC500 PLC system.

Table 2. AC500 modules.

Module	Description
PM590-ETH	Processor module. 2 MB program memory, display, memory card slot, interfaces - 2 RS232/485, FBP, RJ45.
TA524	Dummy communication module. It is used to cover an unused communication module slot to protect it from dust and touch.
DA501	Digital/Analog I/O module. It contains 16 digital inputs, 8 configurable digital inputs/outputs, 4 analog inputs, 2 analog outputs, integrated fast counter.
CD522	Encoder and PWM expansion module. It contains 2 encoder inputs (frequency up to 300kHz), 2 PWM outputs (frequency up to 100kHz), 2 digital inputs, 8 configurable digital inputs/outputs.
CD522	

2.4 Connections

The module DA501 is used for connecting the mechanical crane controls. These controls include eight toggle switches (as digital inputs) and two potentiometers (as analog inputs) (See Figure 4). But only the first potentiometer (R1) is used in crane control because analog inputs in the DA501 module use a single ground plane. This is the reason why changes in the value of the first potentiometer affect the value of the second potentiometer. One of the encoders is also connected to this module because the 3D Crane uses encoders, that require frequency up to 15 kHz and the DA501 supports frequency up to 35 kHz (since firmware V2.0.6). The terminals to which they are connected and their meaning are shown in Table 3.

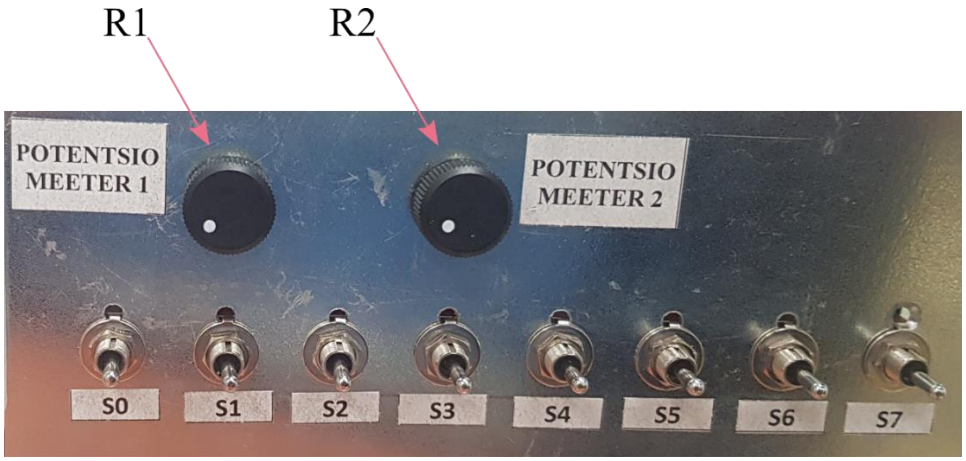


Figure 4. Mechanical crane controls.

Table 3. DA501 connections.

Component	Terminal	Signal	Meaning
S0	1.0	DI0	Activates/deactivates toggle-based control method.
S1	1.1	DI1	Moves the crane to the zero position, resets encoders.
S2	1.2	DI2	Moves the crane forward along the X-axis.
S3	1.3	DI3	Moves the crane backward along the X-axis.
S4	1.4	DI4	Moves the crane forward along the Y-axis.
S5	1.5	DI5	Moves the crane backward along the Y-axis.
S6	1.6	DI6	Lifts the load along the Z-axis.
S7	1.7	DI7	Lowers the load along the Z-axis.
R1	3.0	AI0	Controls the speed of the cart movement along X and Y axes.
R2	3.1	AI1	Unused potentiometer that should be turned to the right at the maximum.
Encoder A	4.0	DC16	Incremental encoder for Z-axis: wave A.
Encoder B	4.1	DC17	Incremental encoder for Z-axis: wave B.

The CD522 modules are used for reading encoders and controlling DC motors (speed and direction selection). These modules are used because of their high-speed PWM outputs that support frequencies up to 100 kHz. The 3D Crane uses DC motors that require frequencies up to 15 kHz. Axis limit switches are also connected to one of the CD522 modules because of their quick response to input changes. The terminals to which DC motors, encoders, and axis limit switches are connected are shown in Tables 4 and 5.

Table 4. CD522 (2) connections.

Terminal	Signal	Meaning
1.5	O0	PWM control signal for the DC motor of the Z-axis.
2.0	A0	Incremental encoder for AX-angle: wave A.
2.1	B0	Incremental encoder for AX-angle: wave B.
4.0	A1	Incremental encoder for AY-angle: wave A.
4.1	B1	Incremental encoder for AY-angle: wave B.

Table 5. CD522 (1) connections.

Terminal	Signal	Meaning
1.5	O0	PWM control signal for the DC motor of the X-axis.
1.7	O1	PWM control signal for the DC motor of the Y-axis.
2.0	A0	Incremental encoder for X-axis: wave A.
2.1	B0	Incremental encoder for X-axis: wave B.
2.4	C4	Position limit switch of the Z-axis..
2.5	C5	Position limit switch of the X-axis.
2.6	C6	Position limit switch of the Y-axis.
4.0	A1	Incremental encoder for Y-axis: wave A.
4.1	B1	Incremental encoder for Y-axis: wave B.
4.4	C12	Signal changing the rotational direction of the X-axis DC motor.
4.5	C13	Signal changing the rotational direction of the Y-axis DC motor.
4.6	C14	Signal changing the rotational direction of the Z-axis DC motor.

2.5 Software tools

2.5.1 ABB Automation Builder

Automation Builder is an integrated development tool made by ABB for PLC programming. It combines all the tools, required for programming, configuring, debugging, and maintaining automation projects [10]. It works together with CODESYS installed with Automation Builder.

2.5.2 CODESYS

CODESYS (Controller Development System) is an open source PLC programming environment, developed by 3S-Smart Software Solutions, a German software company. It enables the programming of PLC according to the international industrial standard IEC 611131-3. It is platform-independent and used by most PLC suppliers [11].

CODESYS uses five PLC programming languages defined in the IEC 611131-3 standard. These languages are:

- Instruction List (IL)
- Structured Text (ST)
- Ladder Diagram (LD)
- Function Block Diagram (FBD)
- Sequential Function Chart (SFC)

In addition to these five, a graphical editor called Continuous Function Chart (CFC) is available in CODESYS, which is not defined by IEC.

In this thesis, it was decided to use a Structured Text. Structured Text is a text-based PLC programming language. Syntactically it resembles high-level programming languages such as C, Pascal. Its main advantage is that program written in Structured Text takes up much less space, compared to other programming languages and the flow/logic is easier to read and understand [12].

3 Development

3.1 Configuration on Automation builder

The first step in the configuration on Automation Builder was to create a new project. It was done by clicking on *File -> New Project -> AC500 project* from a toolbar in the upper-left corner (See Figure 5).

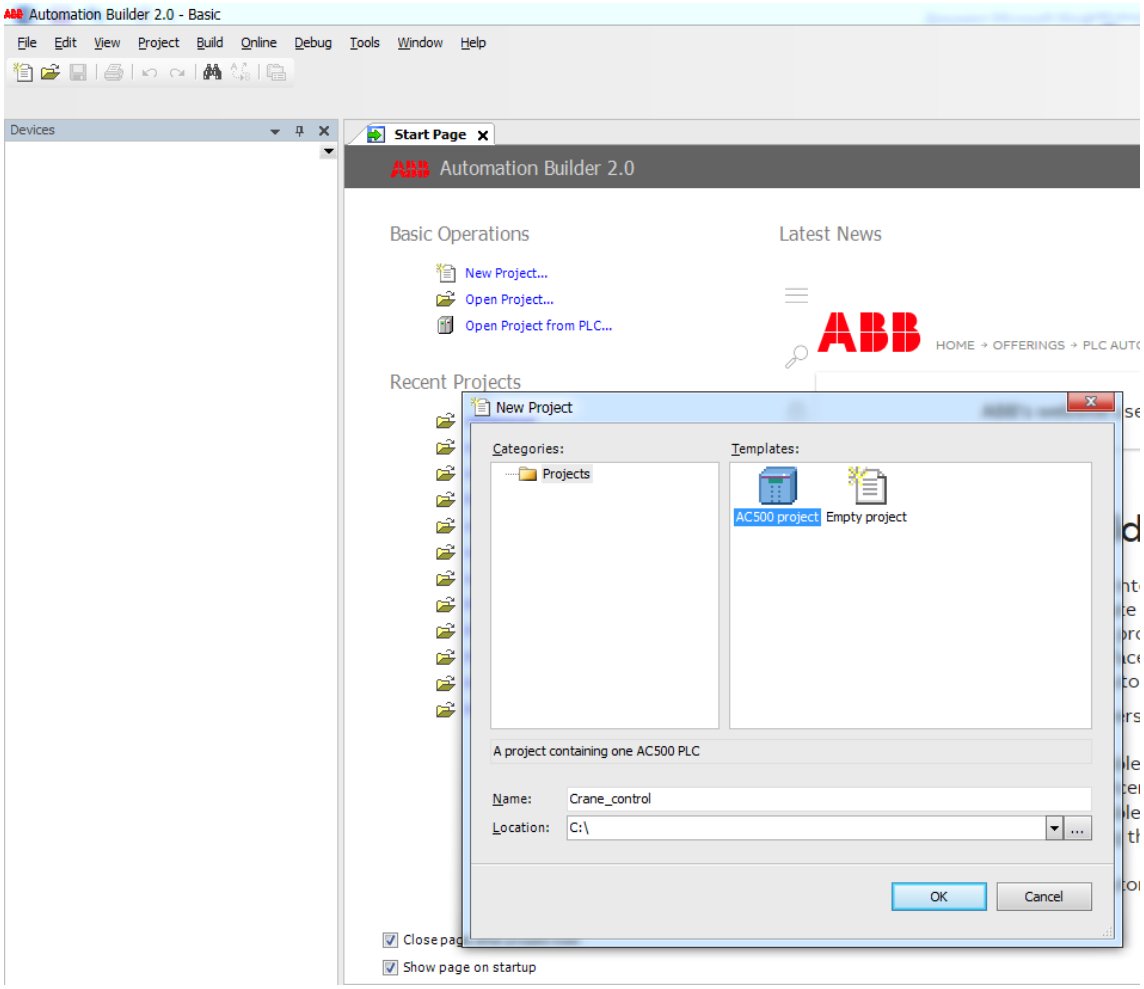


Figure 5. Creating a new project.

The processor module was then selected from a list in the pop-up window (See Figure 6).

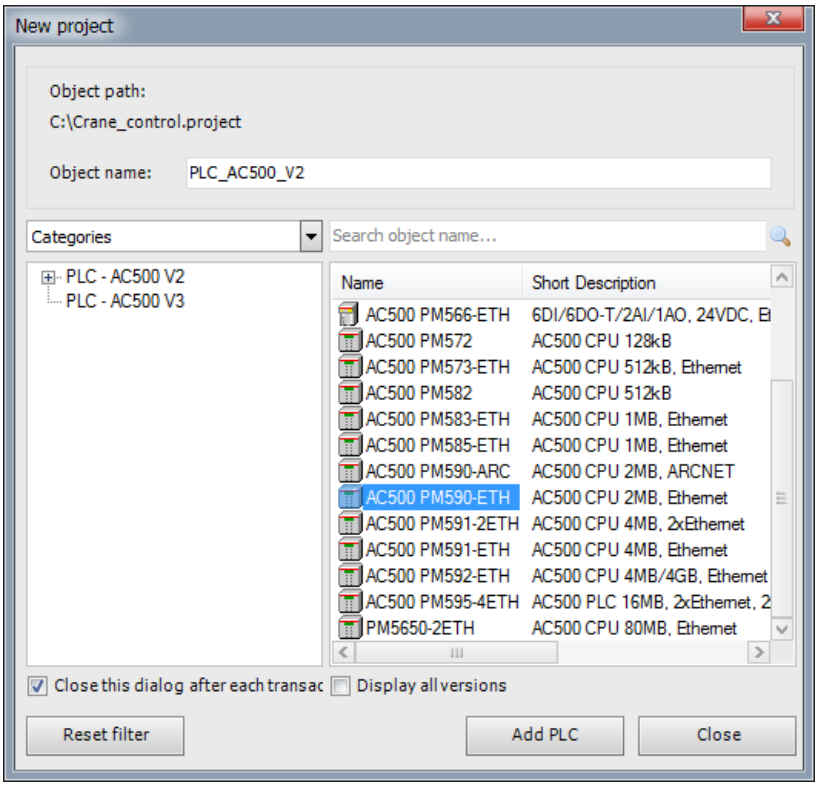


Figure 6. Selecting processor module.

The next step was to add the I/O modules on a bus. They were added one by one by right-clicking on *IO_Bus*, clicking on *Add object* and selecting the required I/O module from a list in the pop-up window (See Figure 7).

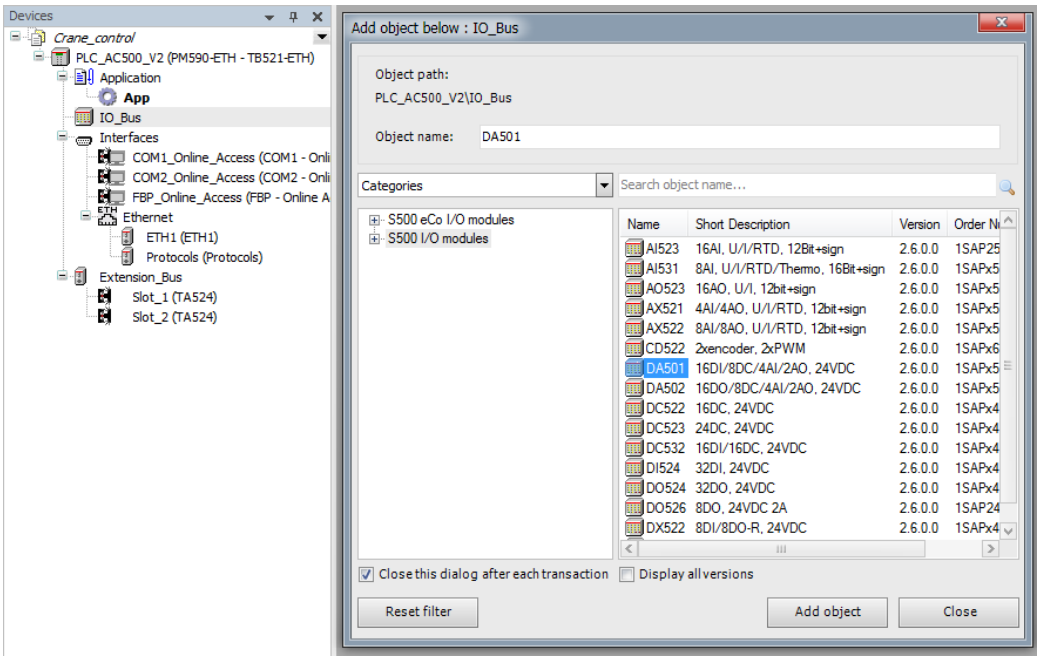


Figure 7. Adding I/O modules.

3.1.1 I/O modules configuration

After these actions were done, it is possible to configure the selected I/O modules. The configuration is divided into two parts: parameterization and I/O mapping. By setting parameters, the system can be configured for specific tasks. I/O signals that have been mapped are accessible as global variables.

Module parameterization is done by double-clicking on the required module from the project tree and by clicking on *Parameters*. (See Figure 8).

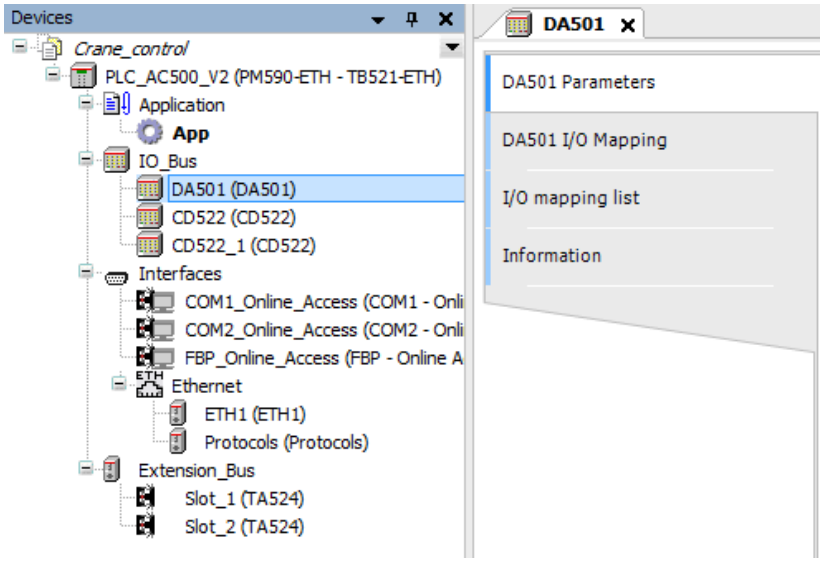


Figure 8. Parameterization.

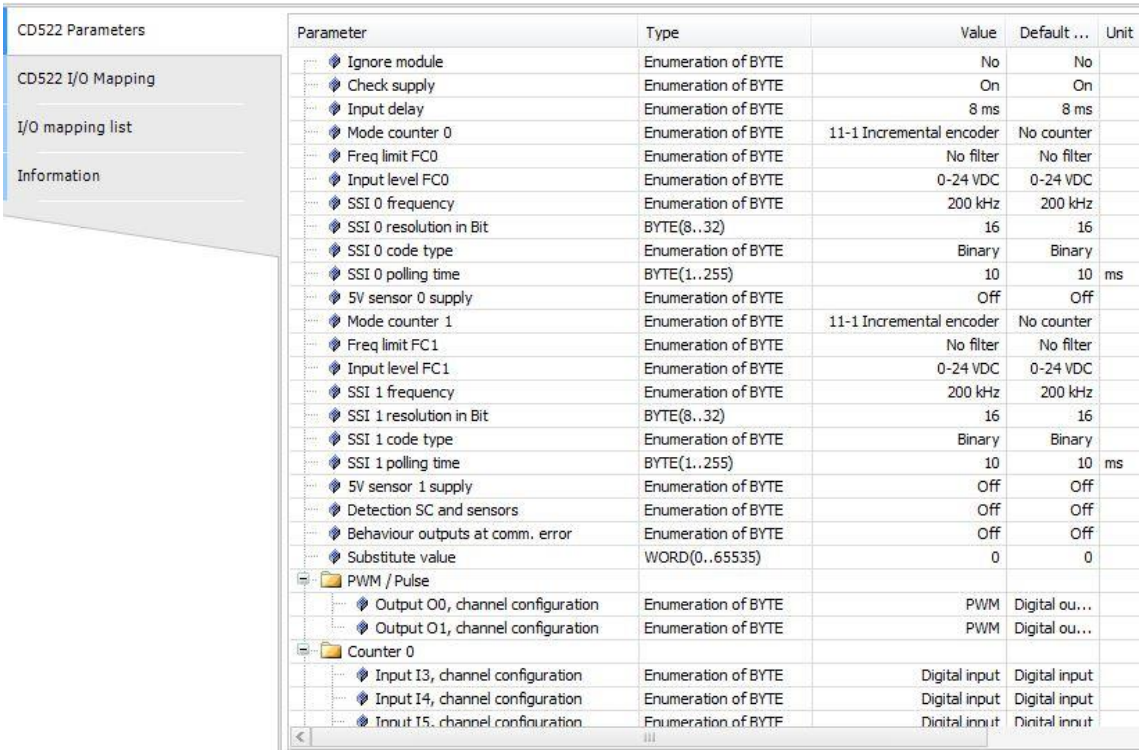
On the settings of the DA501 module, the following parameters were changed: *Fast counter* was set to *7-1 UpDown directional discriminator* and *Input 0, channel configuration* was set to *0…10 V* (See Figure 9).

DA501 Parameters
DA501 I/O Mapping
I/O mapping list
Information

Parameter	Type	Value	Default Value	Unit
Ignore module	Enumeration of BYTE	No	No	
Check supply	Enumeration of BYTE	On	On	
Input delay	Enumeration of BYTE	8 ms	8 ms	
Fast counter	Enumeration of BYTE	7-1 UpDown directional discr…	0-No counter	
Detect short circuit at outputs	Enumeration of BYTE	On	On	
Behaviour outputs at comm. error	Enumeration of BYTE	Off	Off	
Substitute value	BYTE(0..255)	0	0	
Input 0, channel configuration	Enumeration of BYTE	0…10 V	Not used	
Input 0, check channel	Enumeration of BYTE	Plausib, Cut wire, Short circuit	Plausib, Cut wire, Short circuit	
Input 1, channel configuration	Enumeration of BYTE	Not used	Not used	
Input 1, check channel	Enumeration of BYTE	Plausib, Cut wire, Short circuit	Plausib, Cut wire, Short circuit	
Input 2, channel configuration	Enumeration of BYTE	Not used	Not used	
Input 2, check channel	Enumeration of BYTE	Plausib, Cut wire, Short circuit	Plausib, Cut wire, Short circuit	
Input 3, channel configuration	Enumeration of BYTE	Not used	Not used	
Input 3, check channel	Enumeration of BYTE	Plausib, Cut wire, Short circuit	Plausib, Cut wire, Short circuit	
Output 0, channel configuration	Enumeration of BYTE	Not used	Not used	
Output 0, check channel	Enumeration of BYTE	Plausib, Cut wire, Short circuit	Plausib, Cut wire, Short circuit	
Output 0, substitute value	WORD(0..65535)	0	0	
Output 1, channel configuration	Enumeration of BYTE	Not used	Not used	
Output 1, check channel	Enumeration of BYTE	Plausib, Cut wire, Short circuit	Plausib, Cut wire, Short circuit	
Output 1, substitute value	WORD(0..65535)	0	0	

Figure 9. DA501 parameters.

The first CD522 module parameters were configured in the following way: *Mode counter 0* and *Mode counter 1* were set to *11-1 Incremental encoder*, *Output O0, channel configuration* and *Output O1, channel configuration* were set to *PWM* (See Figure 10).



Parameter	Type	Value	Default ...	Unit
Ignore module	Enumeration of BYTE	No	No	
Check supply	Enumeration of BYTE	On	On	
Input delay	Enumeration of BYTE	8 ms	8 ms	
Mode counter 0	Enumeration of BYTE	11-1 Incremental encoder	No counter	
Freq limit FC0	Enumeration of BYTE	No filter	No filter	
Input level FC0	Enumeration of BYTE	0-24 VDC	0-24 VDC	
SSI 0 frequency	Enumeration of BYTE	200 kHz	200 kHz	
SSI 0 resolution in Bit	BYTE(8..32)	16	16	
SSI 0 code type	Enumeration of BYTE	Binary	Binary	
SSI 0 polling time	BYTE(1..255)	10	10	ms
5V sensor 0 supply	Enumeration of BYTE	Off	Off	
Mode counter 1	Enumeration of BYTE	11-1 Incremental encoder	No counter	
Freq limit FC1	Enumeration of BYTE	No filter	No filter	
Input level FC1	Enumeration of BYTE	0-24 VDC	0-24 VDC	
SSI 1 frequency	Enumeration of BYTE	200 kHz	200 kHz	
SSI 1 resolution in Bit	BYTE(8..32)	16	16	
SSI 1 code type	Enumeration of BYTE	Binary	Binary	
SSI 1 polling time	BYTE(1..255)	10	10	ms
5V sensor 1 supply	Enumeration of BYTE	Off	Off	
Detection SC and sensors	Enumeration of BYTE	Off	Off	
Behaviour outputs at comm. error	Enumeration of BYTE	Off	Off	
Substitute value	WORD(0..65535)	0	0	
PWM / Pulse				
Output O0, channel configuration	Enumeration of BYTE	PWM	Digital ou...	
Output O1, channel configuration	Enumeration of BYTE	PWM	Digital ou...	
Counter 0				
Input I3, channel configuration	Enumeration of BYTE	Digital input	Digital input	
Input I4, channel configuration	Enumeration of BYTE	Digital input	Digital input	
Input I5, channel configuration	Enumeration of BYTE	Digital input	Digital input	

Figure 10. CD522 parameters.

The same parameters were applied to the second CD522 module.

Module I/O mapping is done by double-clicking on the required module from the project tree, by clicking on *I/O Mapping* and entering needed variables. The list of created variables, their types, and the definition of what they are responsible for is shown in Table 6.

Table 6. I/O variables.

Module	Variable	Type	Definition
DA501	DI_S0	BOOL	Responsible for toggle switch 0.
	DI_S1	BOOL	Responsible for toggle switch 1.
	DI_S2	BOOL	Responsible for toggle switch 2.
	DI_S3	BOOL	Responsible for toggle switch 3.
	DI_S4	BOOL	Responsible for toggle switch 4.
	DI_S5	BOOL	Responsible for toggle switch 5.
	DI_S6	BOOL	Responsible for toggle switch 6.
	DI_S7	BOOL	Responsible for toggle switch 7.
	SPEED_AXIS	INT	Responsible for potentiometer 1.
CD522 (1)	CD522_1_INPUT	WORD	Input index to adress the module.
	CD522_1_OUTPUT	WORD	Output index to adress the module.
	CD522_1_IN_BIT	WORD	Bits are responsible for inputs C4, C5, C6.
	CD522_1_OUT_BIT	WORD	Bits are responsible for outputs C12, C13, C14.
CD522 (2)	CD522_2_INPUT	WORD	Input index to adress the module.
	CD522_2_OUTPUT	WORD	Output index to adress the module.

3.1.2 IP configuration

In order to establish communication between PLC and PC with the Ethernet protocol, an IP address should be configured. The IP address of a connected device can be checked by selecting *Tools -> IP-Configuration* from the upper toolbar and clicking on *Scan*. This IP address should then be set by right-clicking on *programmable device* from project tree and clicking on *Communication Settings* (See Figure 11).

It is important to note, that PC and PLC should be in the same subnet for security reasons. After IP address has been set, the program can be downloaded to PLC and run by clicking on *Online -> Login* from the upper toolbar.

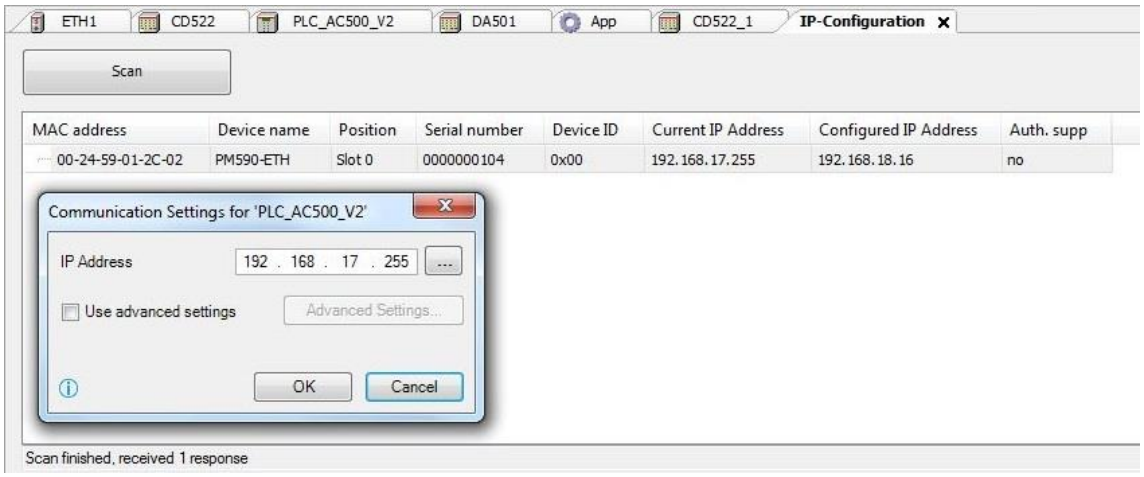


Figure 11. IP configuration.

3.1.3 Web visualization configuration

The next step was to enable an integrated web server on PLC. It allows making a visualization created in CODESYS available as web visualization that can be opened in a web browser of any computer. It may be useful for remote control purposes. The web server is activated by right-clicking on *Protocols* (under *Interfaces* -> *Ethernet* from the project tree) and clicking on *Add object* and selecting *Web Server* from a list in the pop-up window (See Figure 12).

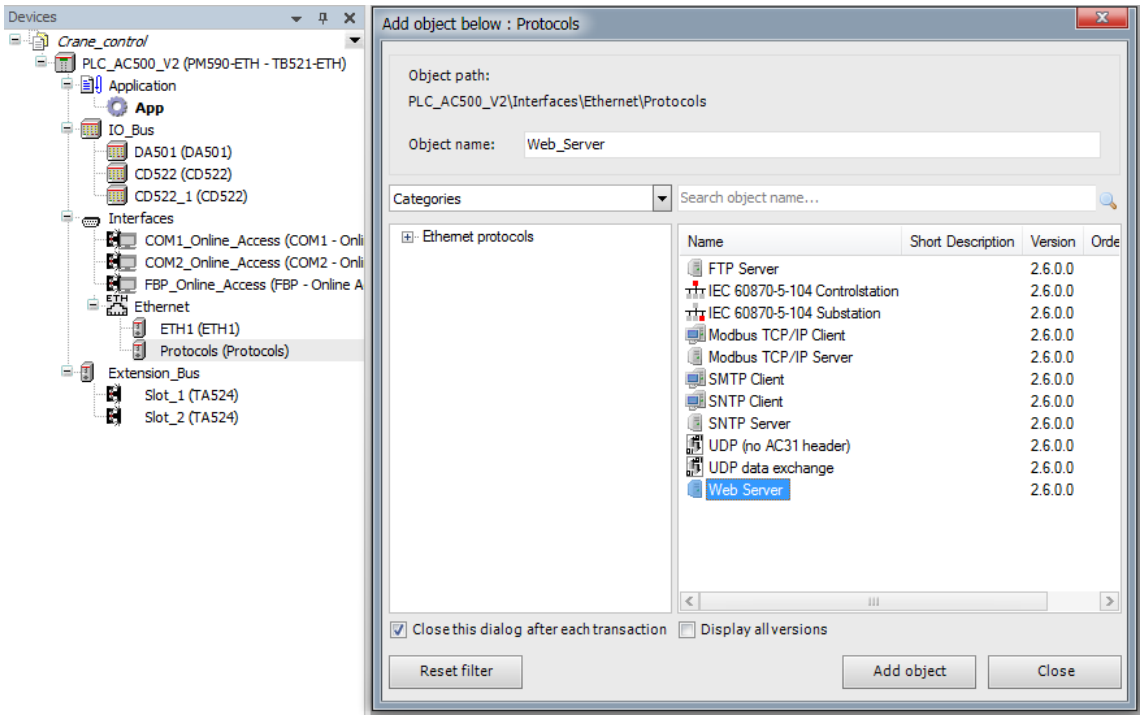


Figure 12. Enabling web server.

After activating the web server, its settings should be checked. It is done by double-clicking on *Web Server*. In the opened tab *Port* should be set to *80* and *Connections* set to *2*.

The next step is to allow web visualization. This is done in the CODESYS window. Under the project tree on *Resources* tab double-click on *Target Settings*. In the pop-up window select *Visualization* tab and enable *Web visualization*. Also, make sure that *Inhibit download of visualization files* is disabled (See Figure 13).

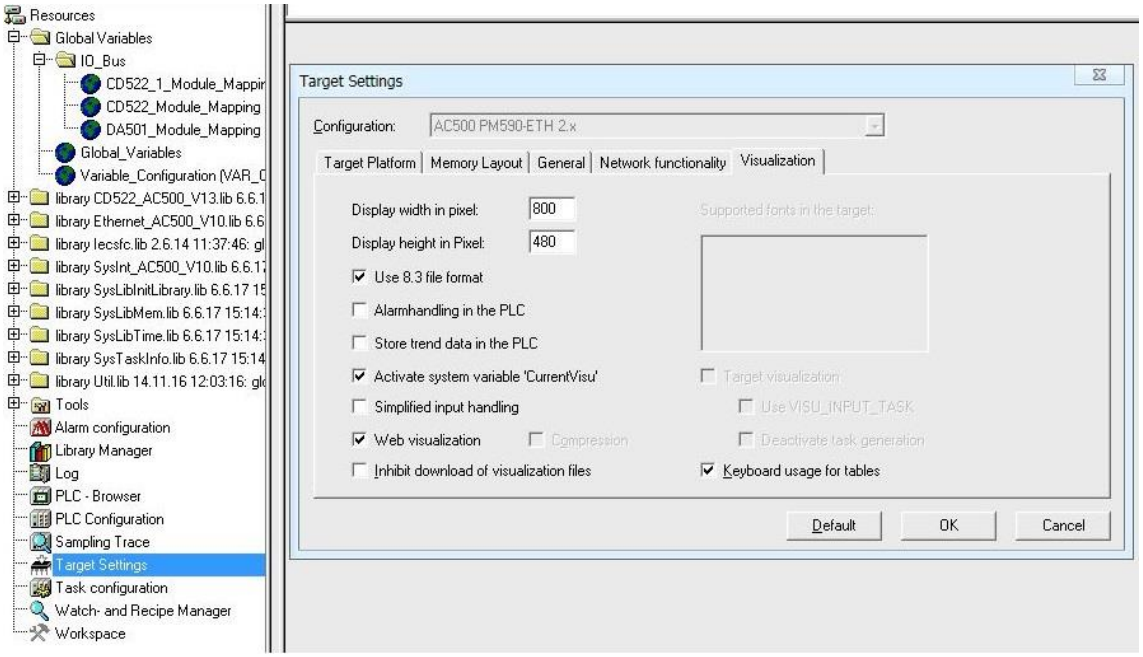


Figure 13. Allowing web visualization.

Once the web visualization is allowed, ensure that visualization created in CODESYS is marked for web visualization. Under the project tree on *Visualizations* tab right-click on created visualization and click on *Object Properties*. In the pop-up window select *Visualization* tab and enable the *Web-Visualization* (See Figure 14).

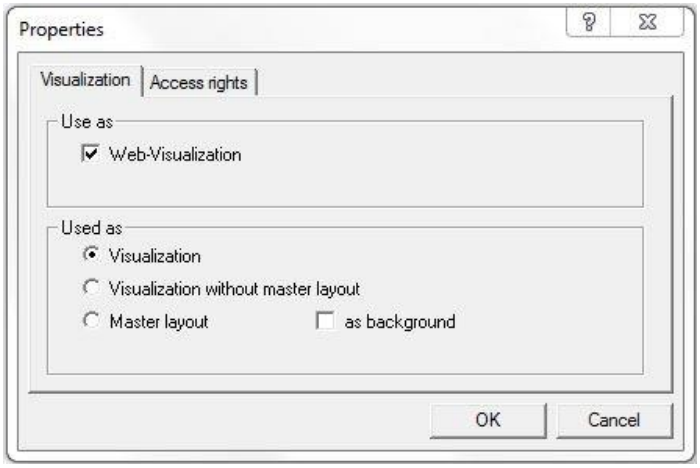


Figure 14. Enabling web visualization.

After all these steps were done, the HMI is available in the web browser by typing http://<IP address of the PLC/webvisu.htm> into the address bar. For example: http://192.168.17.255/webvisu.htm (See Figure 15). It is important to note that the web browser should be with the latest Java version installed. But it works correctly only in Internet Explorer.

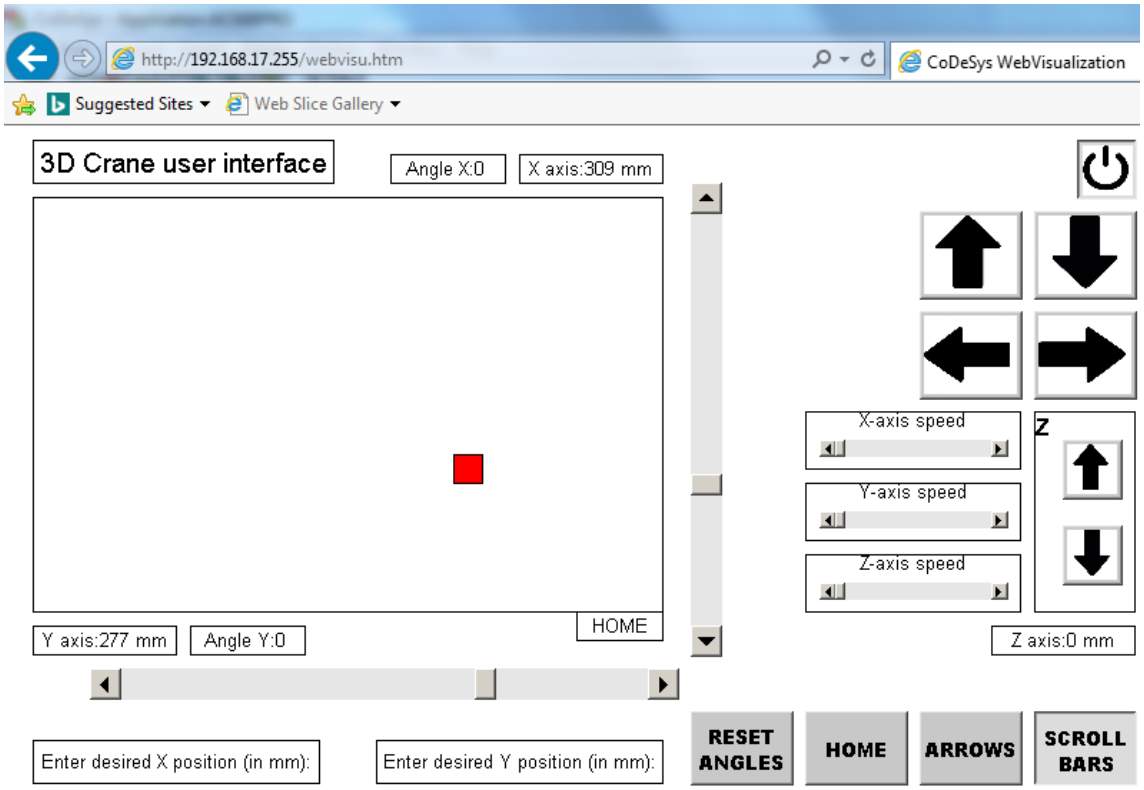


Figure 15. HMI in web browser.

3.2 Control program development

The development of a control program is done in an integrated CODESYS environment. CODESYS window opens by double-clicking on *Application* from the project tree. The opened window is divided into four sections: the project tree is on the left side, the right-top window is for variables declaration, the right-middle window is for program code, the right-bottom window is for system messages.

3.2.1 CD522 function blocks

The function of the CD522 module is managed by special function blocks. These function blocks are contained in the CD522_Library, which is automatically included into the project after the CD522 module has been added into the project tree.

The CD522_32BIT_ENCODER function block is used to read encoder signals connected to CD522 modules (See Figure 16). As it was mentioned before, encoders are used to measure the cart coordinates on XY-plane, the lift-line length, and the deviation angles of the payload.

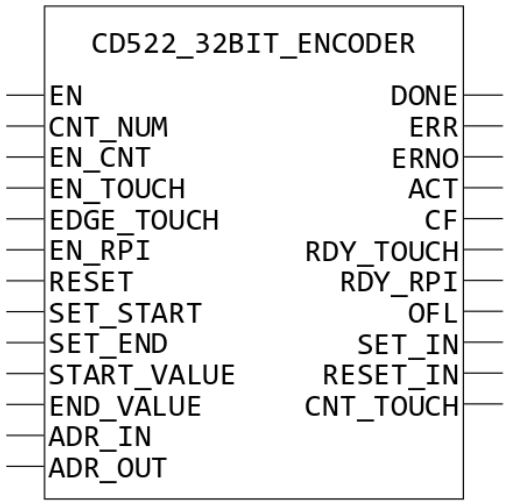


Figure 16. Encoder function block.

EN is a Boolean variable that enables/disables the function block processing. *CNT_NUM* defines which encoder to read. If *0*, the encoder connected to A0, B0 is read, if *1* – the encoder connected to A1, B1. *EN_CNT* is a Boolean variable that enables/disables pulse counting. *RESET* is a Boolean variable that resets *ACT* value as long as it is set to *TRUE*. *ADR_IN* and *ADR_OUT* are the addresses of the first input and output indexes from the structure of the CD522 module. *DONE* is a Boolean output variable that indicates the

processing of the function block. *ERR* is a Boolean output variable that indicates an error. *ACT* is a current counter value.

The function block CD522_PWM_OUT is used to control DC motors (See Figure 17). As it was said, DC motors are used to drive the cart in the x and y directions and to lift and lower the load in the z-direction.

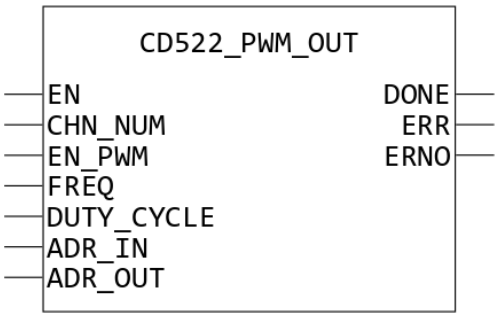


Figure 17. PWM function block.

EN is a Boolean variable that enables/disables the function block processing. *CHN_NUM* defines which DC motor is managed. If it is set to *0*, DC motor connected to output O0 is managed, if *1* – DC motor connected to output O1. *EN_PWM* is a Boolean variable that enables/disables the PWM. *FREQ* specifies the frequency of PWM. *DUTY_CYCLE* specifies the percentage of time when the signal is high. *ADR_IN* and *ADR_OUT* are the addresses of the first input and output indexes from the structure of the CD522 module. *DONE* is a Boolean output variable that indicates the processing of the function block. *ERR* is a Boolean output variable that indicates an error.

3.2.2 CNT_IO function block

The CNT_IO function block is used to read the encoder signals connected as Fast Counter of the DA501 module (See Figure 19). This function block is contained in the Counter_AC500_V20 library, which is not included into the project by default. In order to add this library, in the CODESYS window, under the project tree on *Resources* tab double-click on *Library Manager*. The opened window is divided into four sections. The upper-left is for a list of linked to the project libraries. Right-click on this area and click on *Additional Library*. In the pop-up window select a required library (See Figure 18). Usually, library files are located at:
C:\Program Files (x86)\Common Files\CAA-Targets\ABB_AC500\AC500_V12\library

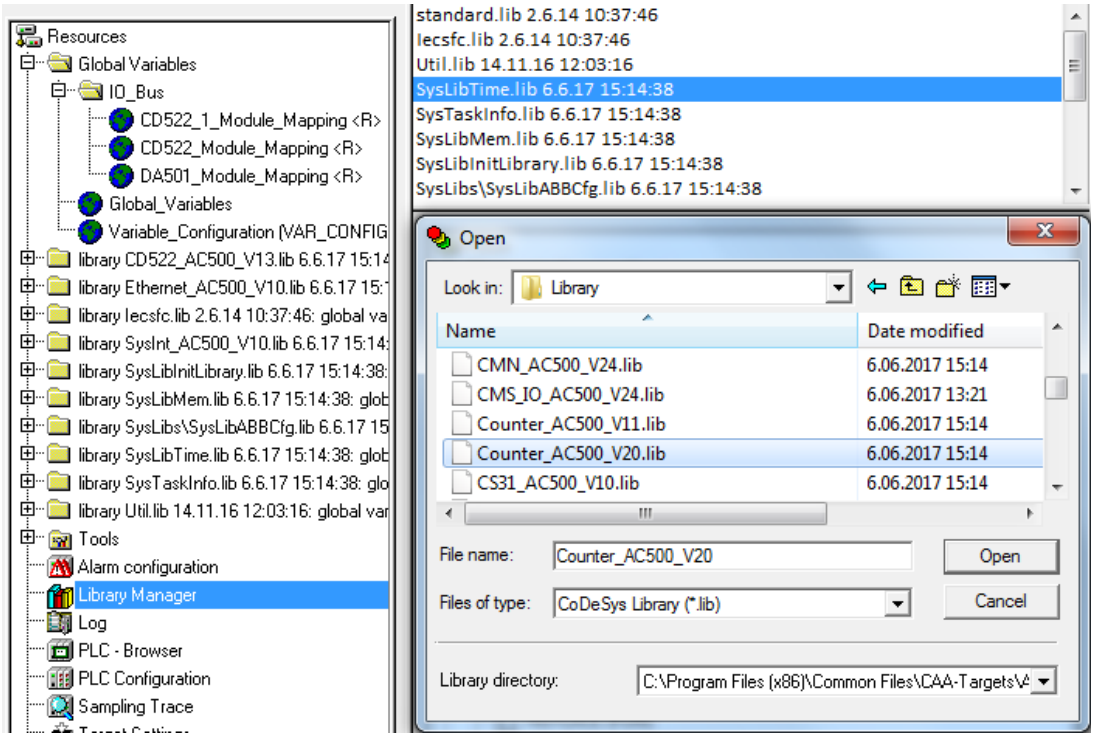


Figure 18. Adding library into project.

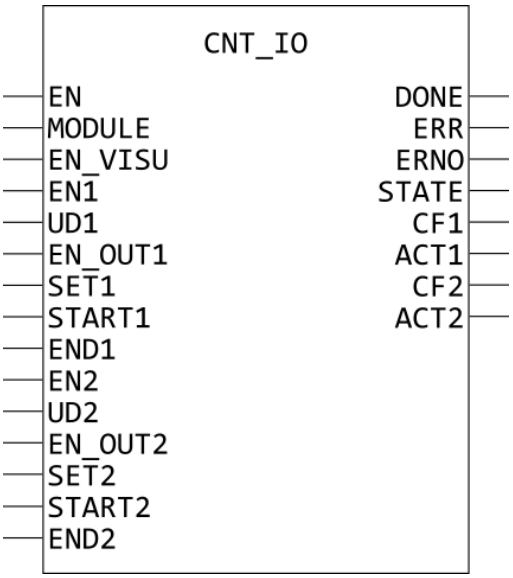


Figure 19. CNT_IO function block.

EN is a Boolean variable that enables/disables the function block processing. *MODULE* defines the S500 I/O module number on the I/O bus. The first module that is directly right to the CPU is with number 1. *EN1* is a Boolean variable that enables/disables pulse counting of the encoder connected to inputs DC16, DC17. *SET1* is a Boolean variable. When it is set to *TRUE*, *ACT1* is always overwritten by the *START1* value. *DONE* is a Boolean output variable that indicates the processing of the function block. *ERR* is a

Boolean output variable that indicates an error. *ACT1* is a current counter value of the encoder connected to inputs DC16, DC17.

3.2.3 LIN_TRAFO function block

The function block LIN_TRAFO is used to rescale a number from one range to another (See Figure 20). It allows displaying the information about the position of a cart and the payload lift-line length not in the encoder units, but in the units used in everyday life. It also allows converting the raw value from the potentiometer into the percentage of PWM duty cycle.

Figure 20. LIN_TRAFO function block.

IN is a number to rescale. *IN_MIN* and *IN_MAX* are the lower and upper limits of the input number range. *OUT_MIN* and *OUT_MAX* are the lower and upper limits of the output number range. *OUT* is a rescaled number.

3.2.4 PID function block

The control system has an automatic control mode that moves the cart minimizing the load oscillations. For these purposes, the PID function block is used (See Figure 21). It allows to improve the positioning accuracy, to control the speed of a cart movement, and to decrease the load oscillations. In this subsection you can find functionality of the current block, but in the following sections you will see how it affects the control loop.

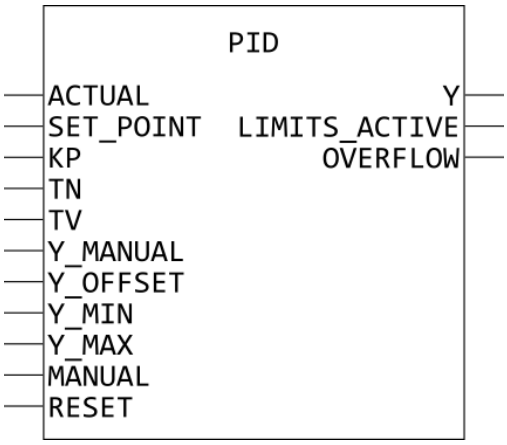


Figure 21. PID function block.

ACTUAL is a present value of the process variable. *SET_POINT* is a desired value of the process variable. *KP* is a proportional gain (P-component), *TN* is a reset time (I-component), and *TV* is a derivative action time (D-component). *Y_MIN* and *Y_MAX* are the lower and upper limits of the output value. *RESET* is a Boolean variable that resets the output value. *Y* is the output value, calculated by the function block.

3.2.5 Program structure

The crane control program is implemented as a state machine. It simplifies maintenance and provides a potential for modification for future development. The control program starts with the initialization phase. In this step, the encoder and PWM function blocks described above are called for the first time in order to check the validity and plausibility of the inputs connected to encoders and DC motors. If an error occurs, the program goes to „Error state“ and stops. It means that connections and module configuration should be checked. To start the program again PLC should be reset. If there is no problem – a rail, a cart, and a load go to the zero position that is determined by input bits C4, C5 and C6, in other words, the crane „Goes home“. At this position, encoders are reset. This must be done because the 3D Crane uses incremental encoders that require determination of a reference position. After these actions are carried out, the user can operate the crane. The crane can be operated either with mechanical toggle switches and a potentiometer in „Manual control“ mode or by using buttons and scrollbars in „HMI control“ mode (See Figure 22).

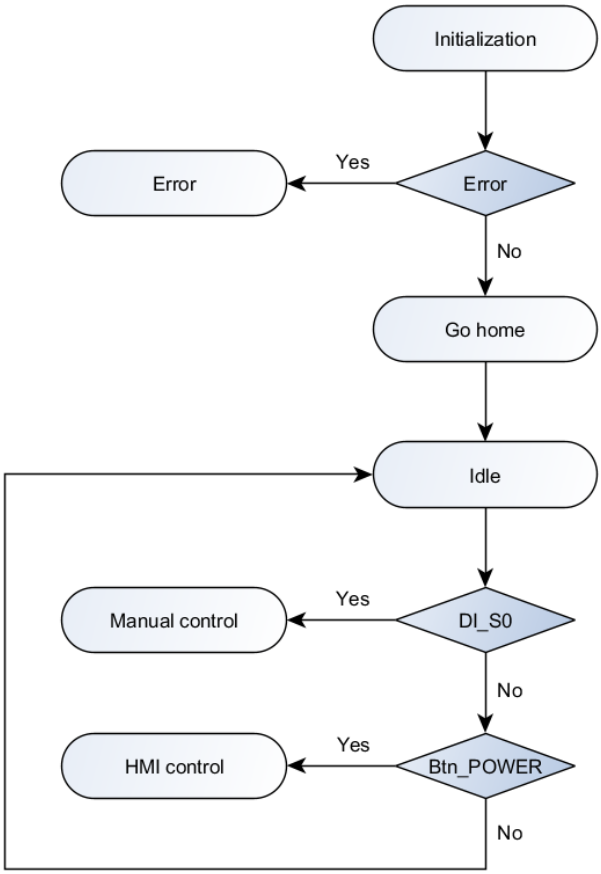


Figure 22. Initialization.

Toggle switch S0 activates the „Manual control“ mode. S0 should be activated continuously until the user wants to log out of the „Manual control“ mode. Toggle switch S1 causes the crane to „Go home“ and resets encoders at zero position. Toggle switches S2 and S3 move the crane along the x-axis – forward and backward, respectively. If both toggle switches are active, it causes an interlocking and movement along x-axis stops. The range of movement is limited by the limit switch, connected to input C5 from one side and by the maximum encoder value (measured by trial method) from the other side. In the same way work toggle switches S4 and S5 that move the crane along the y-axis – forward and backward and toggle switches S6 and S7 that lift and lower the load along the z-axis. Potentiometer 1 controls the speed of a cart movement along x and y axes. Turning it to the left decreases the speed and turning it to the right increases the speed. Potentiometer 2 should be turned to the right at the maximum for the correct operation of the speed control. The reason, as it was mentioned above, is that potentiometers use a single ground plane and changes in the value of one potentiometer affect the value of another potentiometer (See Figures 23 and 24).

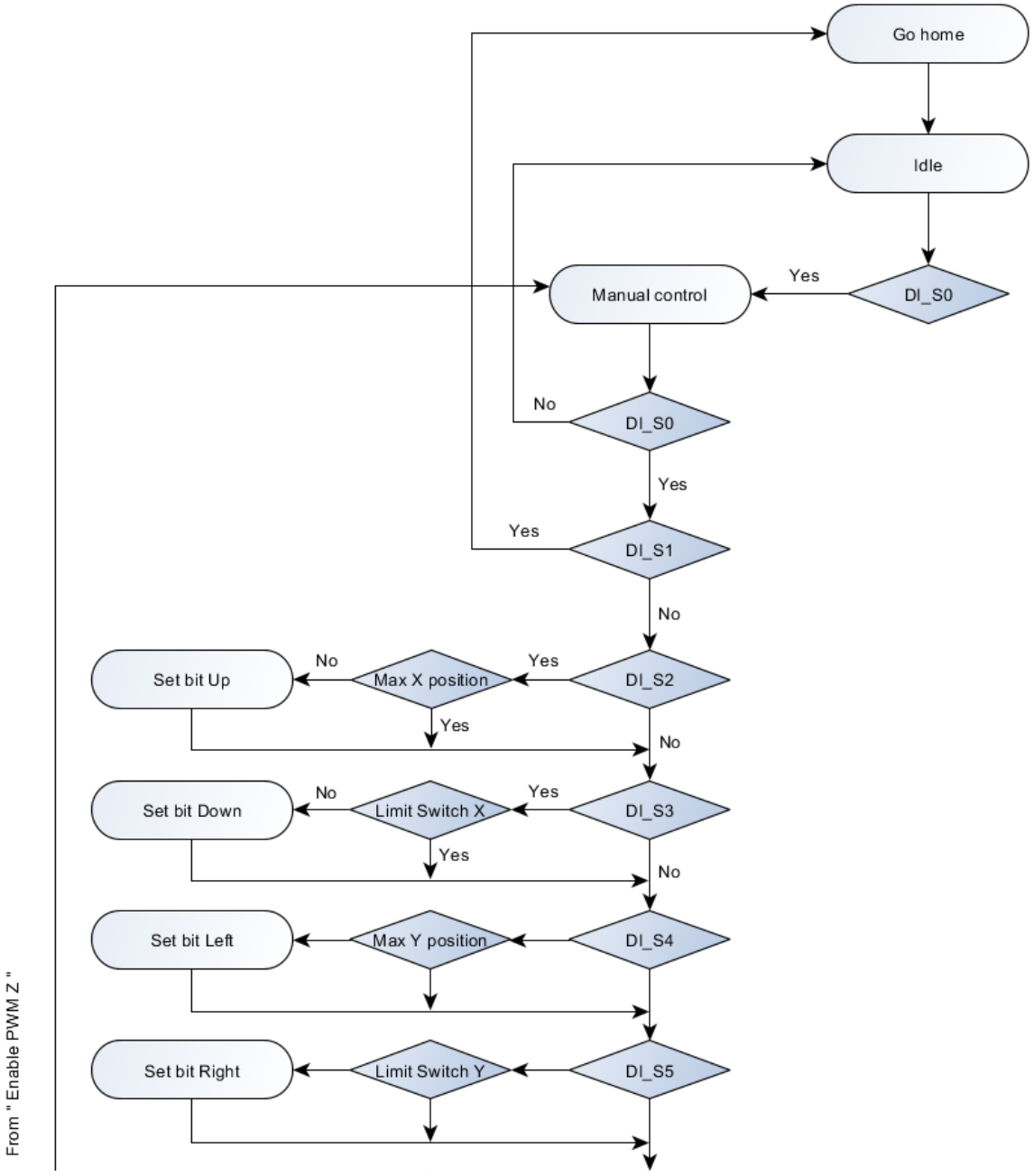


Figure 23. Manual control (1).

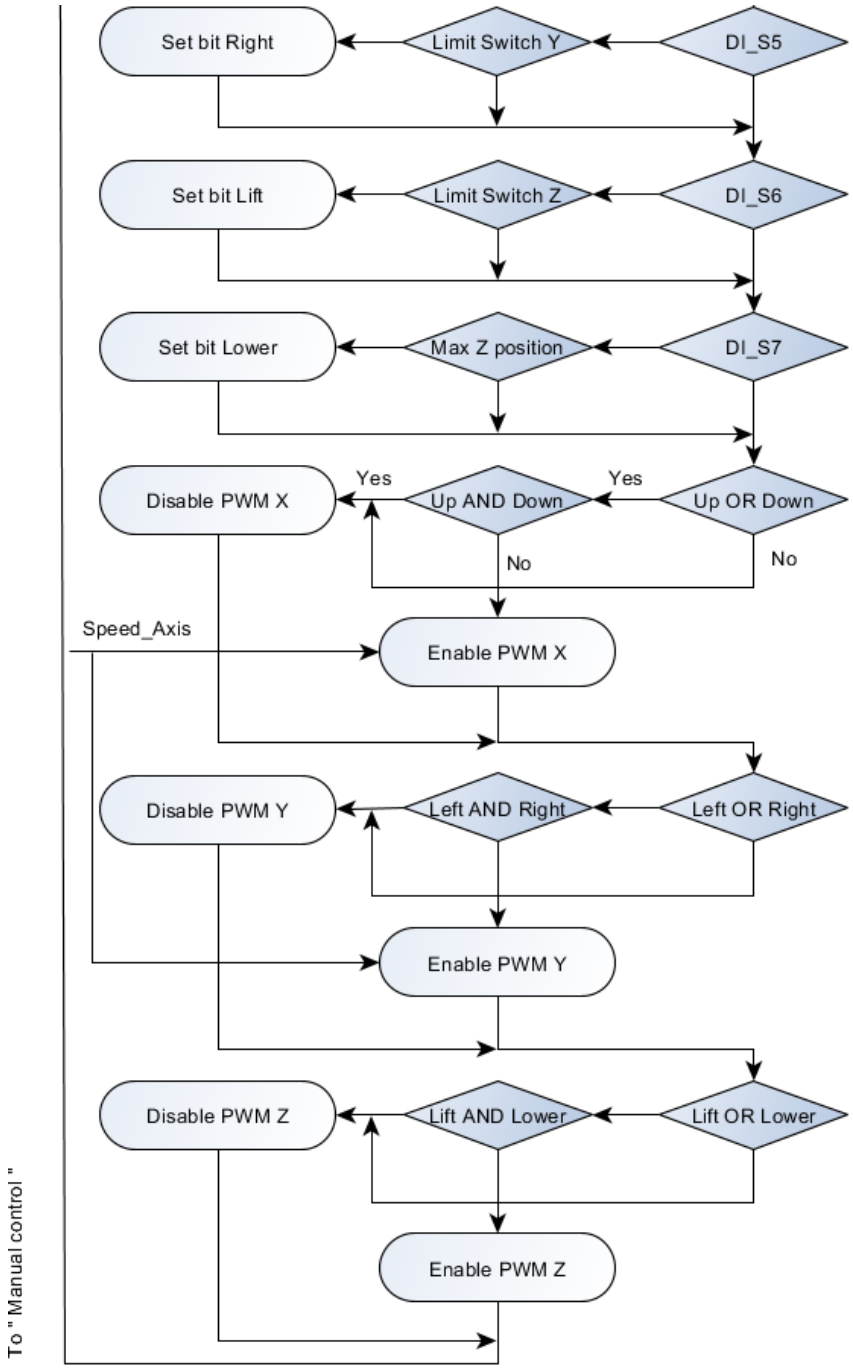


Figure 24. Manual control (2).

Since people today prefer to use graphical user interfaces, the HMI was developed. It makes the system more user-friendly and allows adding some extra features. It also allows controlling the crane remotely. The „HMI control“ mode is activated with the power symbol button from the user interface. The power-button should be activated continuously until the user wants to log out of the „HMI control“ mode. A control method should then be selected: either „Arrows control” or “Scrollbars control”. Switching between these control methods is carried out by clicking on *ARROWS* or *SCROLLBARS* buttons, which are located on the down-right side of the user interface (See Figure 25).

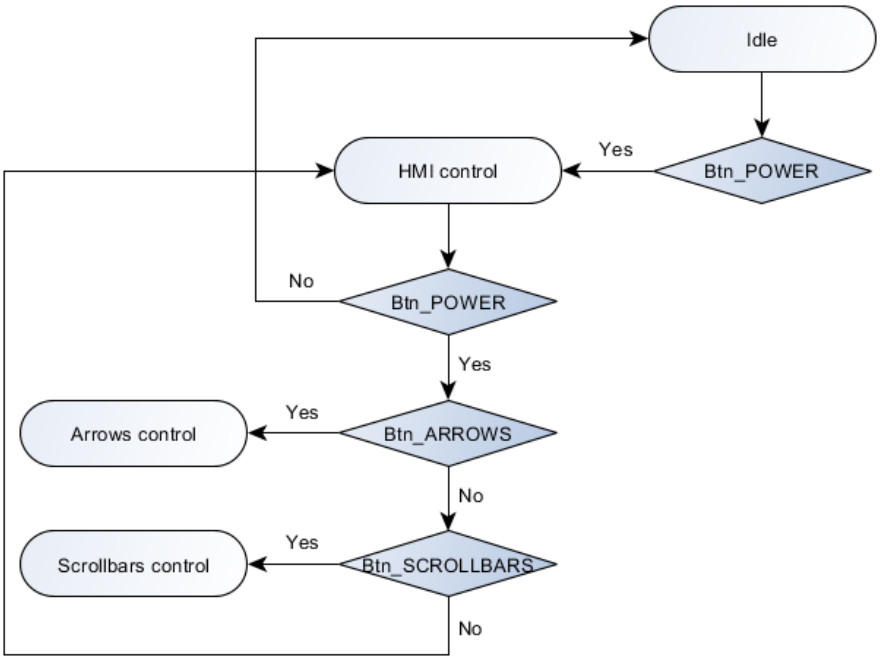


Figure 25. HMI selecting control method.

Selecting the arrows control method, the crane is operated using buttons with the arrow symbol. There are four big arrow-buttons that move the rail with a cart on the XY-plane. Under these four buttons, there are up-arrow and down-arrow buttons in a frame, that lift and lower the load along the z-axis. These six buttons work similarly to toggle switches in the „Manual control“ mode. On the left of z-axis control buttons, there are three scrollbars that control the speed on x, y, and z axes. Moving the slider to the left reduces the speed, moving it to the right increases the speed on the corresponding axis (See Figure 26).

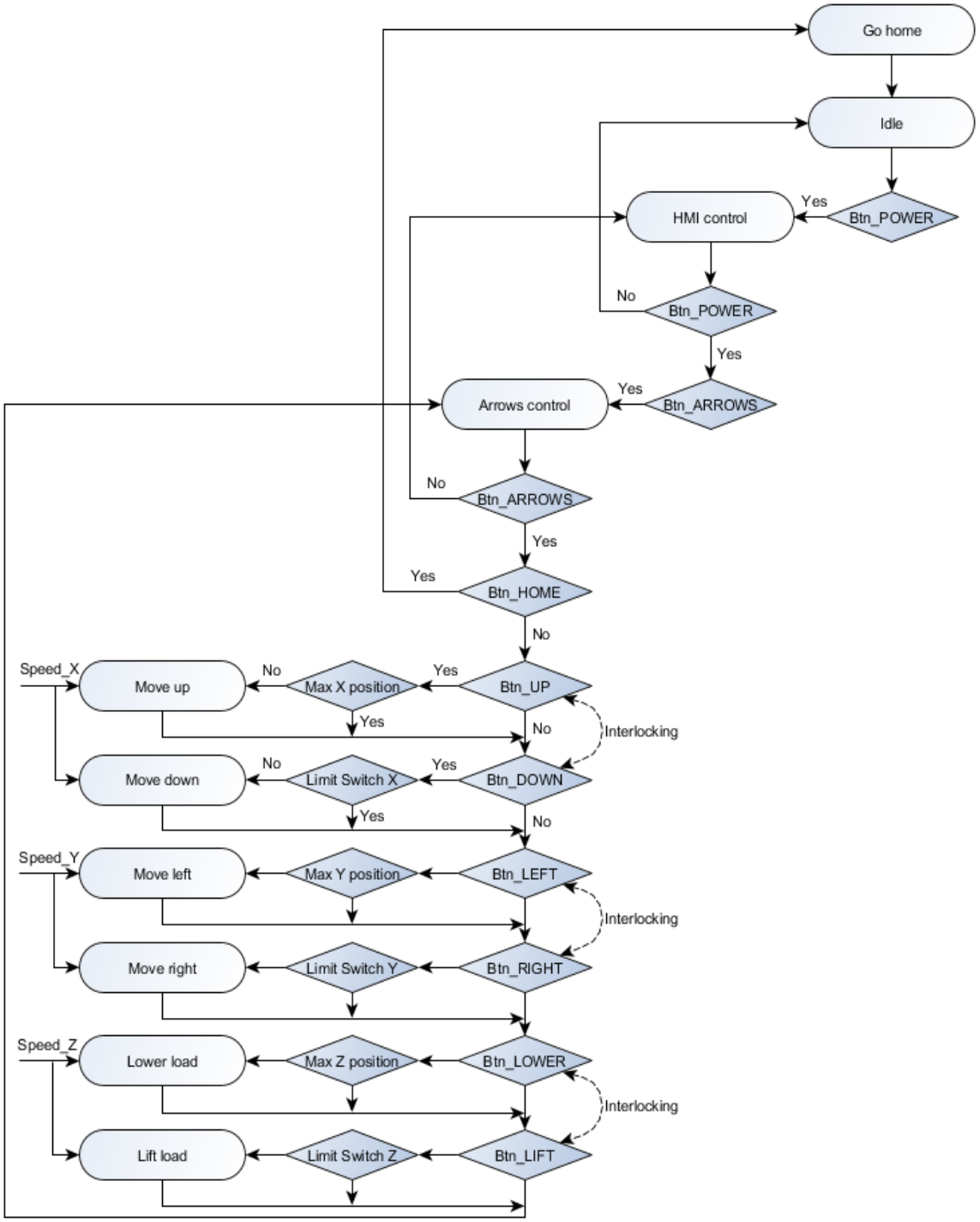


Figure 26. HMI arrows control.

Selecting the scrollbars control method, the position of a cart on XY-plane is set either with scrollbars (located on the right and under the graphical representation of a cart position) or by entering a number in millimeters into textboxes (located under the horizontal scrollbar). The load oscillation control is a distinctive feature of this control method. The only limitation is that the load must be in the zero position. An exact movement to the desired set point and the load oscillation control is carried out by implementing PI controllers. At the first activation of the scrollbars control method, the

system asks to set the load motionless. In this step, a dialog window opens and there is a color indicator that turns to green if the load is motionless and turns to red if the load is hanging (See Figure 27). This must be done to reset angle encoders. Occasionally, the angle encoders unsettle during the crane operation. That is why the *RESET ANGLES* button was created. It opens the dialog window and allows to reset the angle encoders again.

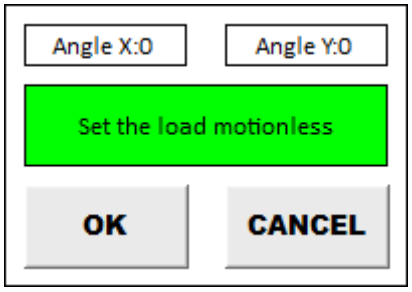


Figure 27. Dialog window.

The vertical scrollbar that is on the right of graphical representation – controls the position of a cart on the x-axis. The upper side corresponds to the maximum encoder value and the downside corresponds to the zero position. In other words, moving the slider up, the cart moves forward and moving the slider down, the cart moves backward along the x-axis. The horizontal scrollbar that is located under the graphical representation – controls the position of a cart on the y-axis. The left side corresponds to the maximum encoder value and the right side corresponds to the zero position, so moving the slider to the left, the cart moves forward and moving the slider to the right, the cart moves backward along the y-axis (See Figures 28 and 29). There is also a *HOME* button that moves the crane to the zero position and resets the encoders, but it works only with the arrows control method. The user interface is shown in Figure 30.

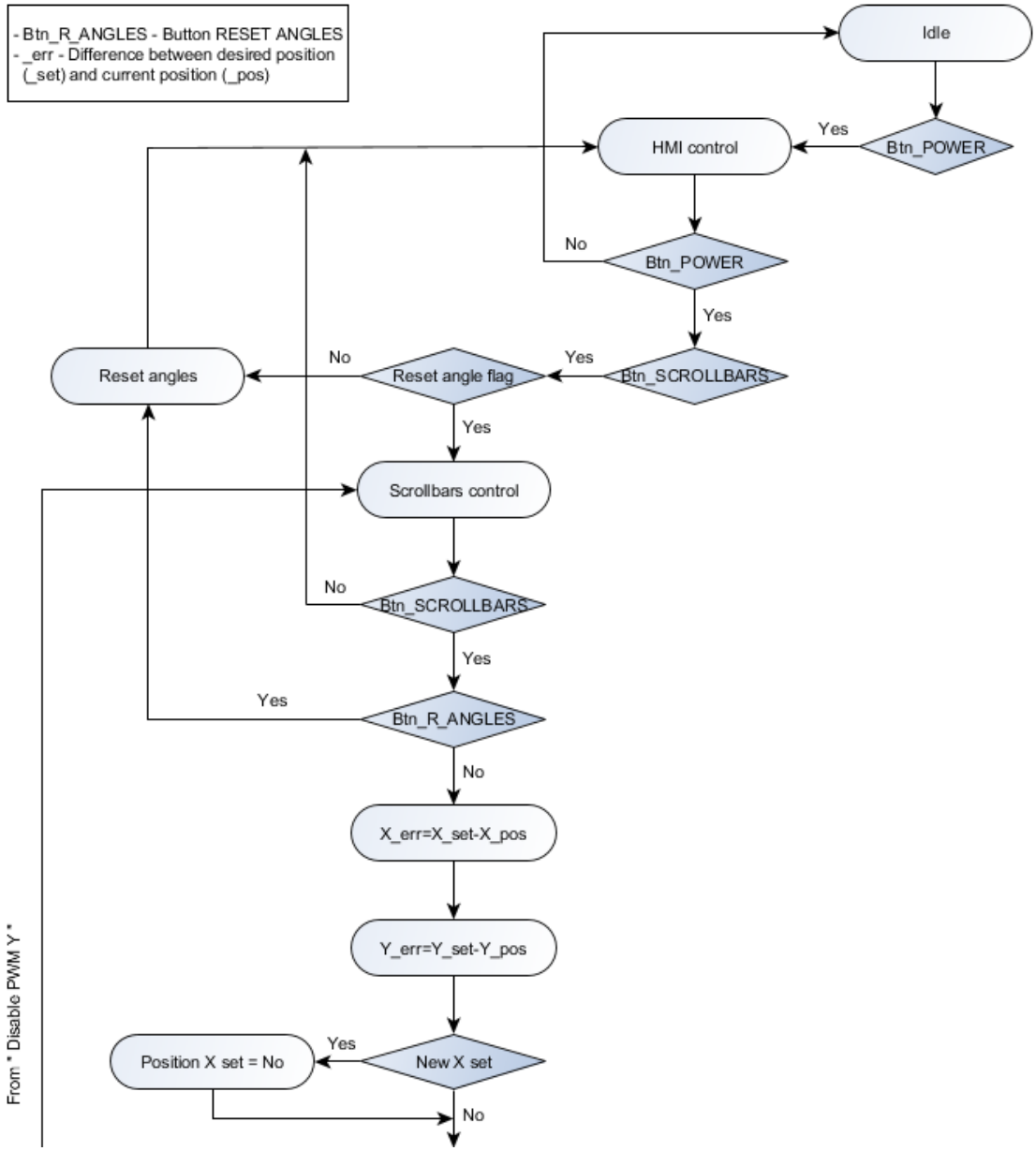


Figure 28. HMI scrollbars control (1).

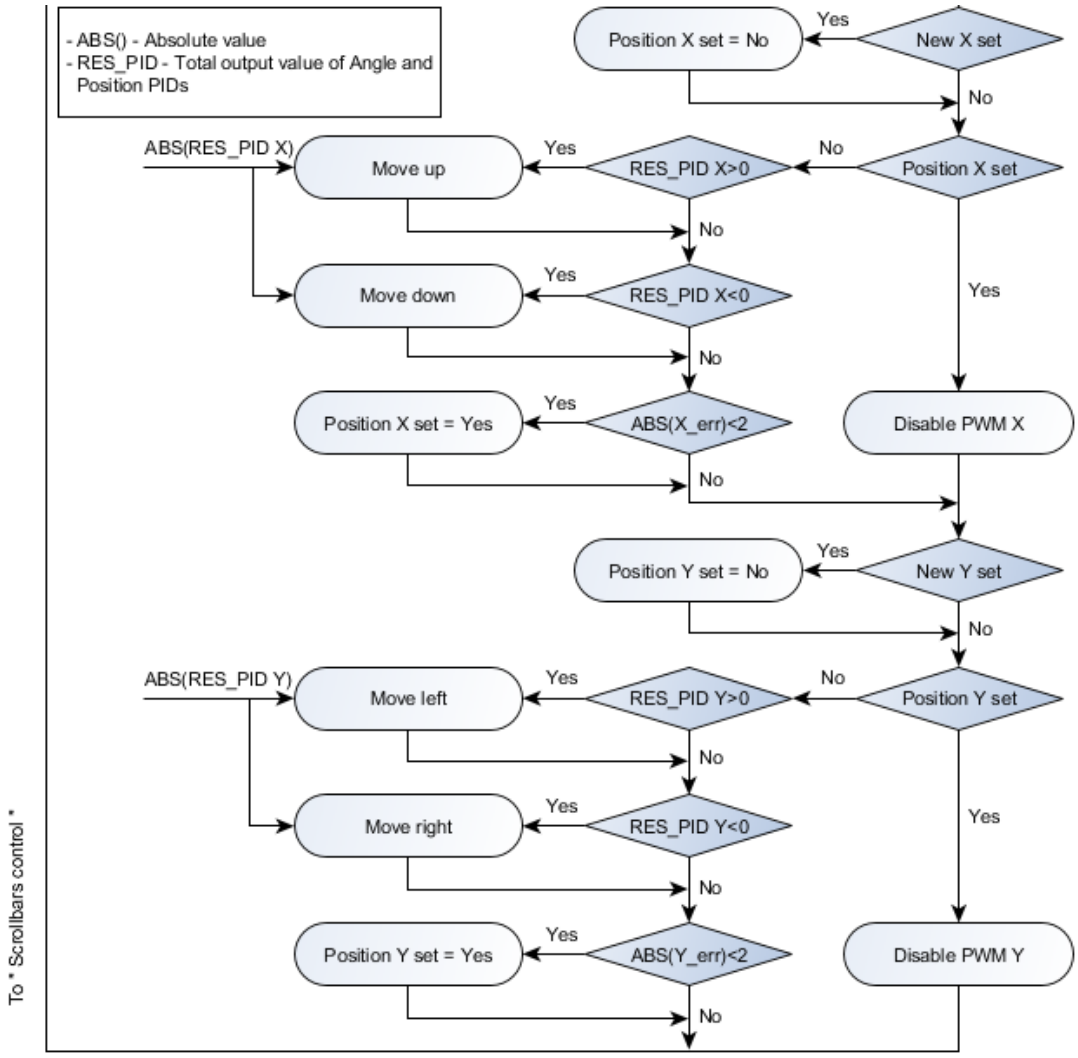


Figure 29. HMI scrollbars control (2).

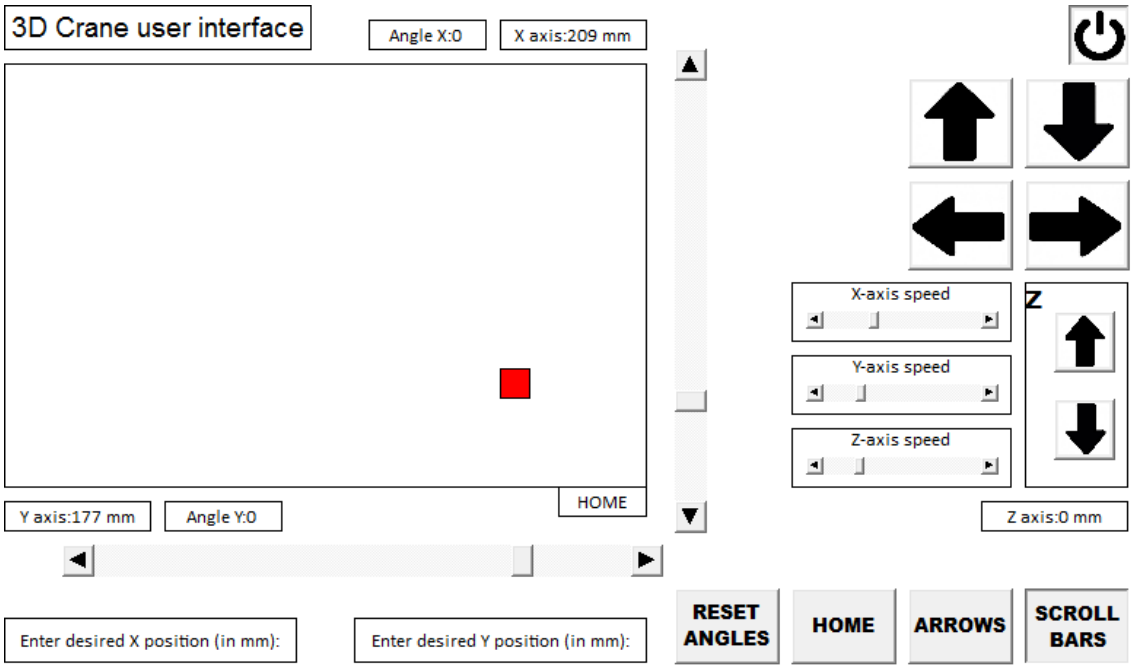


Figure 30. User interface.

3.2.6 PI controllers tuning

As it was mentioned before, the scrollbars control method uses PI controllers. PI components should be tuned to provide the necessary quality of a control loop. Therefore, it is important to specify the requirements. In the case of crane control, the requirements are:

- An exact movement to the desired set point (in this project the positioning accuracy is 1.66 mm (0.18%) on the x-axis and 1.28 mm (0.14%) on the y-axis. Such positioning accuracy corresponds to four encoder units).
- The speed of movement should be maximized.
- The overshoots in positioning should be avoided (it can destabilize the system at end points).
- The load oscillations should be minimized.

The problem is that these requirements are interconnected and there is always a trade-off between them. Furthermore, the setting is complicated because two PI controllers are used simultaneously: the first is responsible for moving to the desired set point (position PI controller), the second is responsible for load oscillation control (angle PI controller). The

quality of the control loop was evaluated using graphs. Each experiment is supported by two graphs. On the first, the red line is the desired set point and the black line is the current position, measured with encoder. The second graph shows the load oscillations.

PI tuning was started from the x-axis by setting a P-component. It was increased until the system became unstable. The results of the system behavior without an I-component can be seen in Figures 31 and 32.

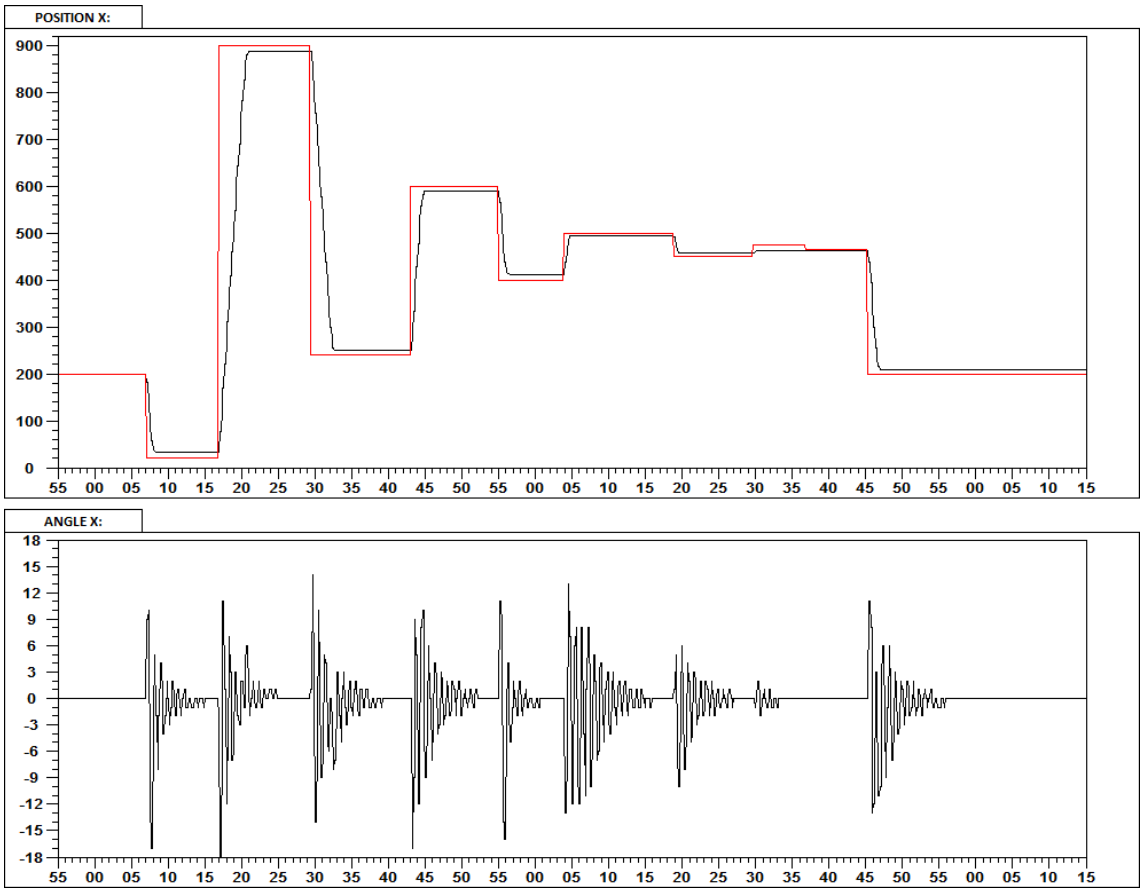


Figure 31. PI tuning, X axis [Position PI: P=1, I=0 | Angle PI: P=1, I=0].

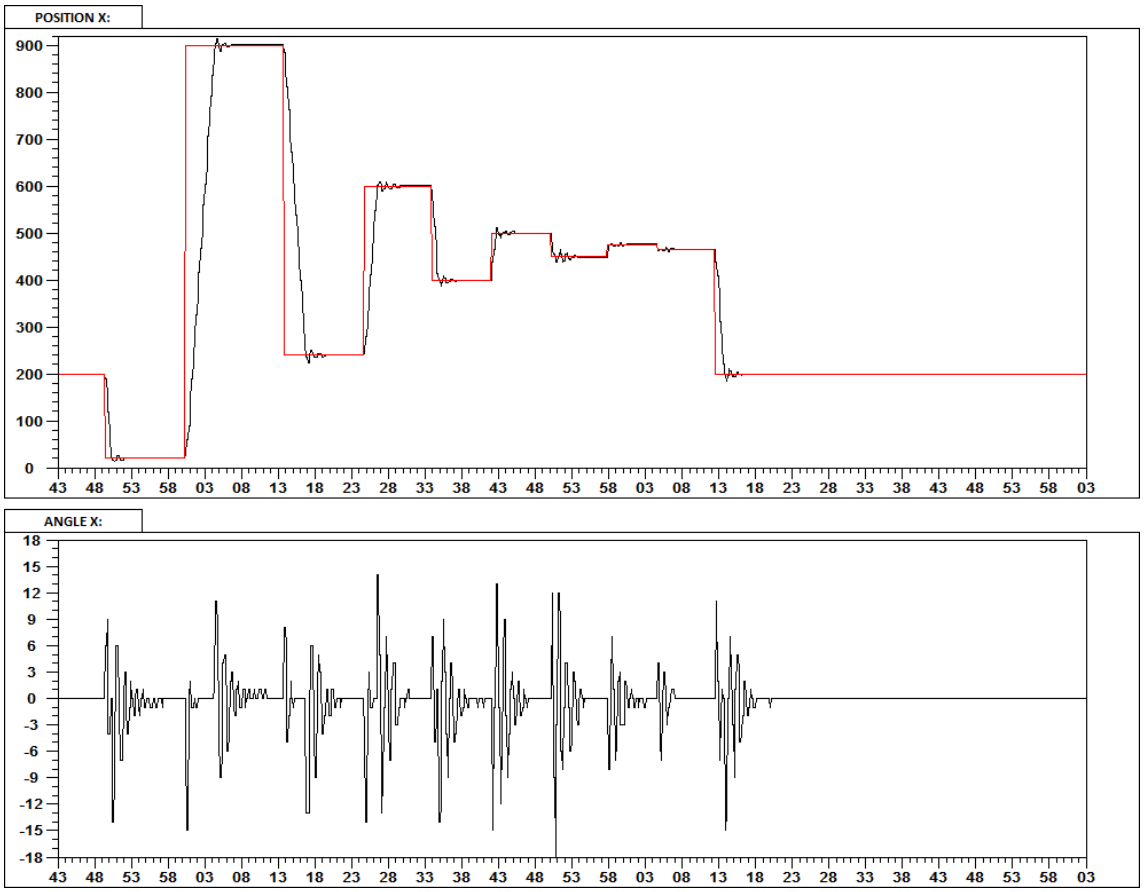


Figure 32. PI tuning, X axis [Position PI: P=6, I=0 | Angle PI: P=10, I=0].

Comparing these two results it can be found that increasing the P-components increases the speed and positioning accuracy (an effect of position PI controller) and decreases the load oscillations (an effect of angle PI controller). The negative effect is that increasing the P-components causes overshoots (a result of both PI controllers). It is important to note, that using only the P-component of position PI controller does not guarantee entry into the positioning accuracy range, especially at small distances. Further increase of P-components makes the system unstable.

Next, the I-component of position PI controller was added, the result can be seen in Figure 33.

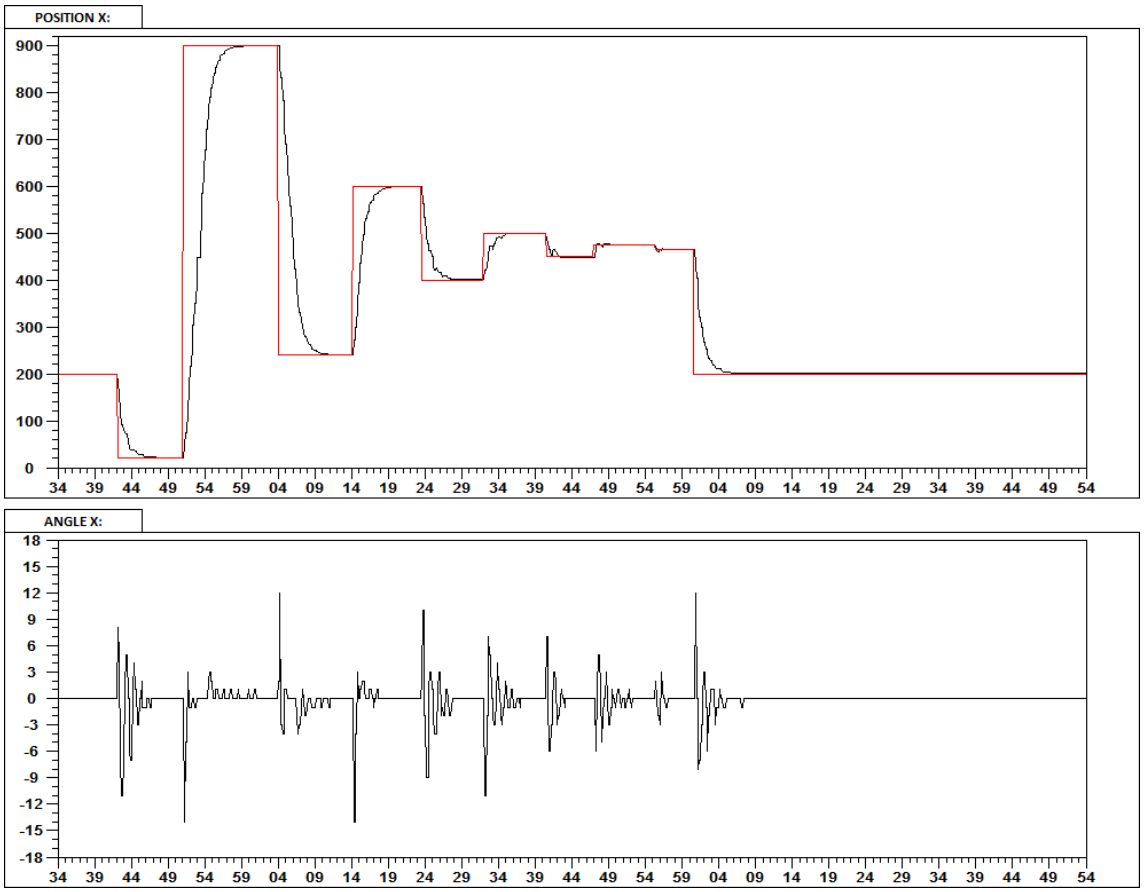


Figure 33. PI tuning, X axis [Position PI: P=5, I=1 | Angle PI: P=10, I=0].

As you can see, the I-component made the movement smoother, as a result, the load oscillations decreased. The overshoots also decreased but still appear at small distances. The cart always enters into the positioning accuracy range. The speed of movement decreased, however the whole loop time (until the cart is in positioning accuracy range and there are no load oscillations) is also decreased.

Next, the I-component of angle PI controller was added, because an instant reaction of the system to load oscillations makes the movement sharp at middle distances (as an effect the load oscillations grow). The result can be seen in Figure 34.

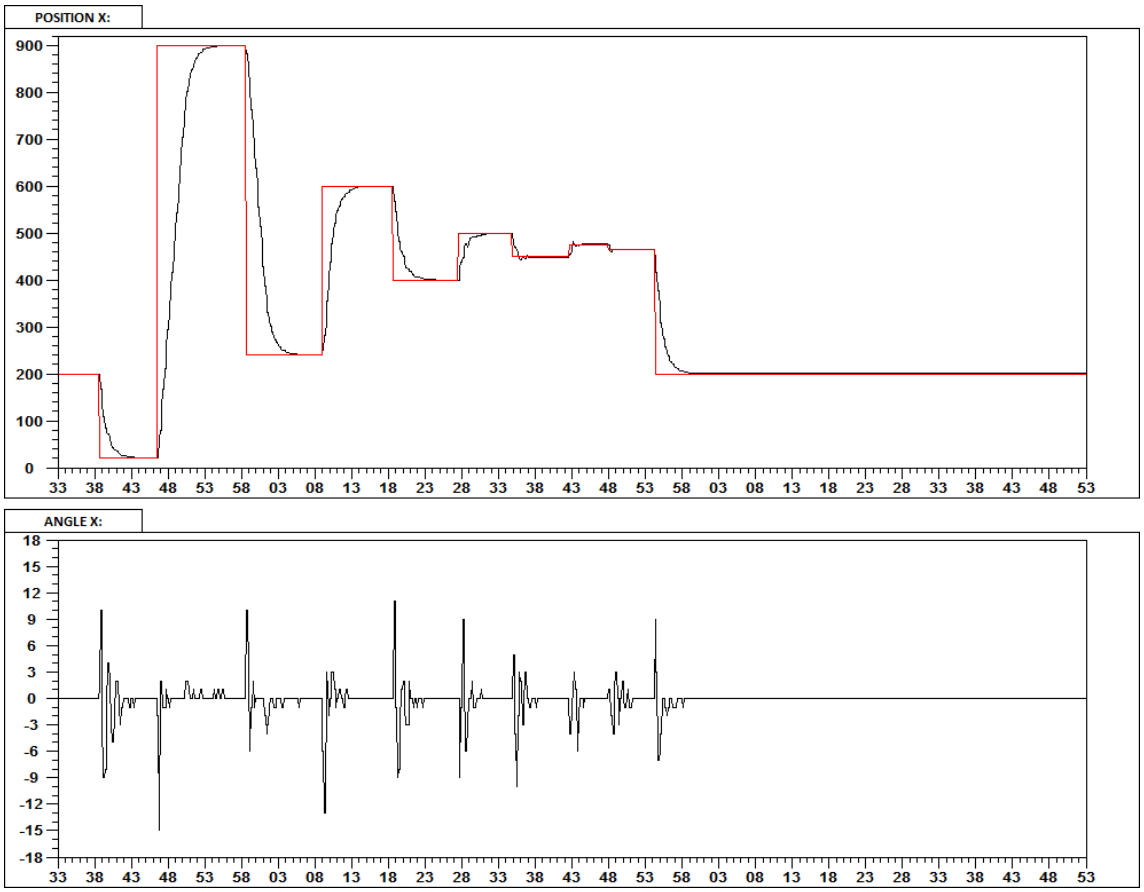


Figure 34. PI tuning, X axis [Position PI: P=5, I=1 | Angle PI: P=10, I=1].

Comparing with the previous result the movement at middle distances became smoother and the corresponding load oscillations decreased, but it is still sharp. The negative effect is that the I-component of angle PI controller increased the overshoots at small distances.

Taking into consideration the results and having a representation of how each parameter of both PI controllers affects the control loop, it was decided to set the following parameters: position PI controller [P = 3.4, I = 1], angle PI controller [P=5.5, I=0.45]. The P-components of both PI controllers were decreased to minimize the overshoots at small distances. Decreasing the P-component of angle PI controller makes the system less sensitive to load oscillations, so it was decided to decrease the I-component. The I-component of position PI controller was left unchanged because decreasing it causes overshoots while increasing it makes the movement slower. The result can be seen in Figure 35.

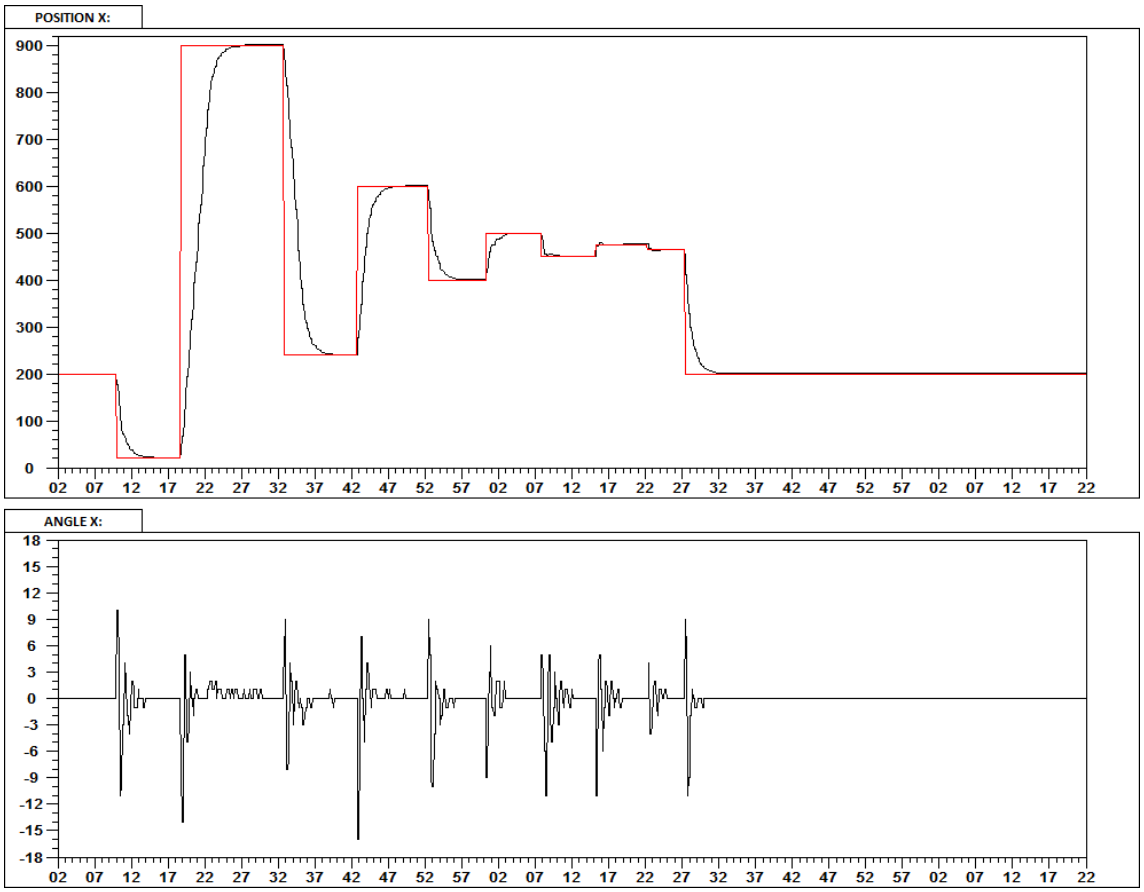


Figure 35. PI tuning, X axis [Position PI: P=3.4, I=1 | Angle PI: P= 5.5, I=0.45].

As you can see, by setting such parameters, there are no losses in movement speed. The movement is smooth at all distances. There are almost no overshoots at small distances. The load oscillations slightly increased but remain insignificant.

Similarly, the y-axis PI controllers were tuned. The initial and final results can be seen in Figures 36 and 37.

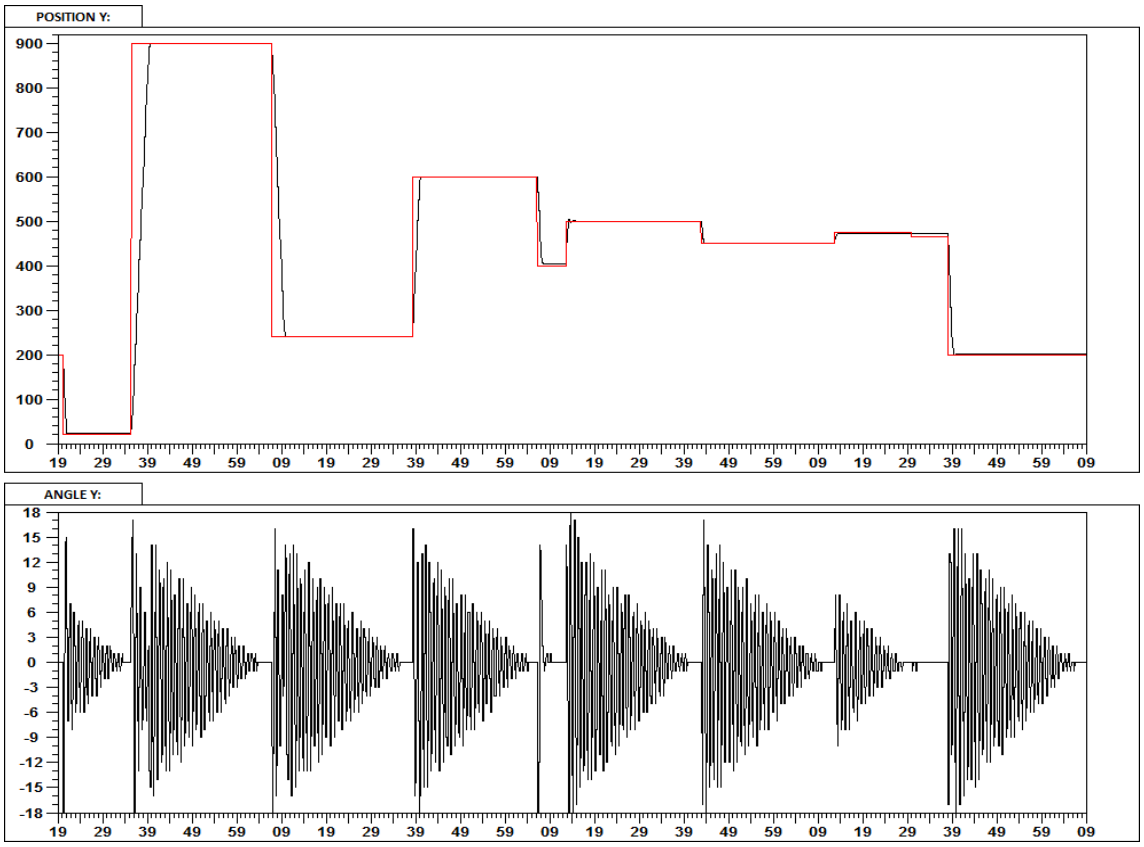


Figure 36. PI tuning, Y axis [Position PI: P=1, I=0 | Angle PI: P=1, I=0].

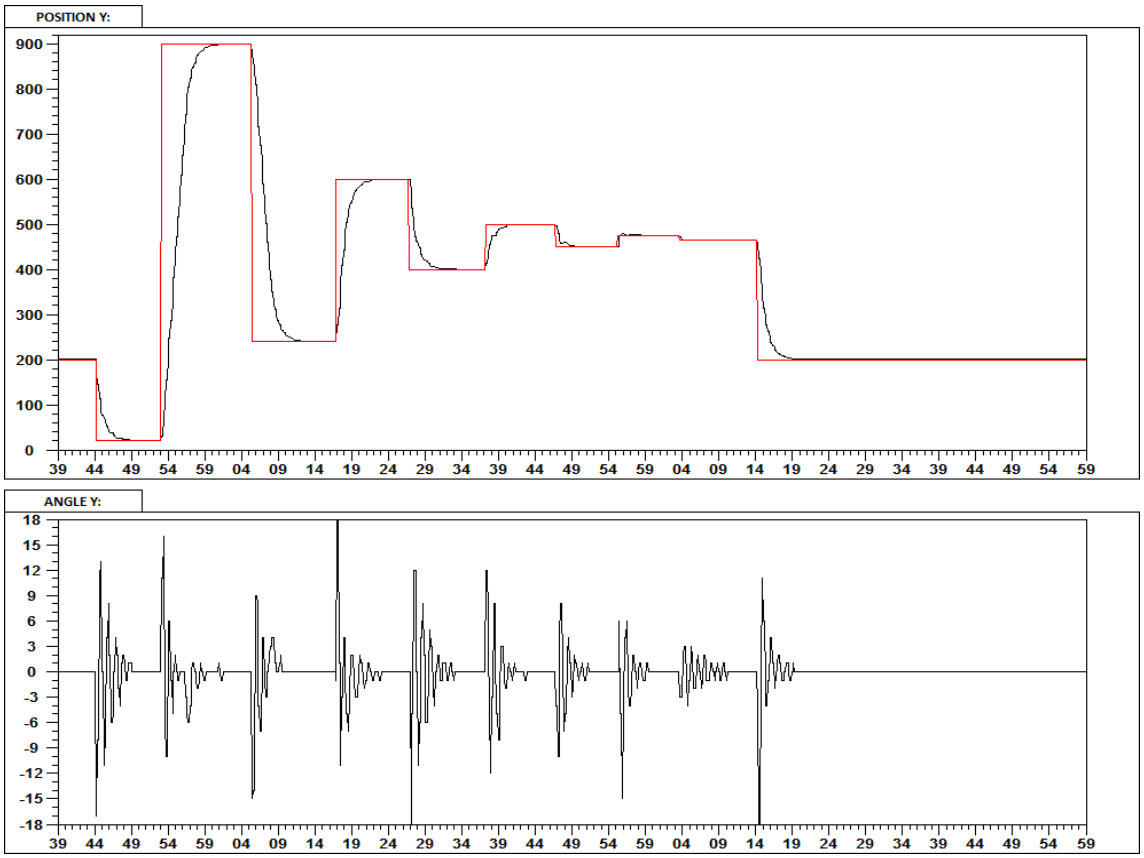


Figure 37. PI tuning, Y axis [Position PI: P= 2.55, I=1 | Angle PI: P=4, I=0.45].

Evaluating the final result, it can be seen that the cart movement is fast and smooth enough at all distances. There are almost no overshoots at small distances. The load oscillations are insignificant and quickly declines.

4 Conclusion

The PLC-based 3D Crane control system was developed. It allows operating the crane either with mechanical controls or with the user interface. The user interface displays all the necessary information about the load position so the crane can be operated remotely. An integrated web server was also configured for remote control purposes. Automatic control mode that moves the load to the desired set point and controls the load oscillations was realized. Therefore, the main and sub-goals of this thesis were successfully achieved. The whole development process is described so this work can be used as a manual.

However, the system has a room for future development. For example, to improve the automatic control system so that it works without a limitation that load must be in zero position. Another thing to do is to implement security because everyone who knows IP address of the PLC can have control over the current system.

References

[1] Alpha Control Laboratory, "Alpha Control Laboratory," [Online]. Available: https://a-lab.ee. [Accessed 19 02 2019].

[2] Inteco, "Inteco 3D Crane," [Online]. Available: http://www.inteco.com.pl/products/3d-crane/. [Accessed 19 02 2019].

[3] Inteco, "3DCrane User's Manual," 2012.

[4] Inteco, "PLC to 3D Crane interface".

[5] ABB Group, "Automation Builder online help system".

[6] S. S. Michael, "What Is a PLC? An Introduction to Programmable Logic Controllers," 2018. [Online]. Available: https://www.allaboutcircuits.com/technical-articles/what-is-a-plc-introduction-to-programmable-logic-controllers/. [Accessed 19 02 2019].

[7] Myodesie, "Programmable Logic Controller (PLC)," [Online]. Available: https://www.myodesie.com/wiki/index/returnEntry/id/2962. [Accessed 19 02 2019].

[8] PLC Academy, "Scan time of the PLC program," 2015. [Online]. Available: https://www.plcacademy.com/scan-time-of-the-plc-program/. [Accessed 19 02 2019].

[9] ABB Group, "ABB AC500," [Online]. Available: https://new.abb.com/plc/programmable-logic-controllers-plcs/ac500. [Accessed 19 02 2019].

[10] ABB Group, "ABB Automation Builder," [Online]. Available: https://new.abb.com/plc/automationbuilder. [Accessed 19 02 2019].

[11] M. Budimir, "FAQ: What is CoDeSys software," 2017. [Online]. Available: https://www.motioncontroltips.com/faq-what-is-codesys-software/. [Accessed 19 02 2019].

[12] PLC Academy, "Structured Text Tutorial to Expand Your PLC Programming Skills," 2015. [Online]. Available: https://www.plcacademy.com/structured-text-tutorial/. [Accessed 19 02 2019].