CZECH TECHNICAL UNIVERSITY IN PRAGUE

FACULTY OF MECHANICAL ENGINEERING

Department of Production Machines and Equipment

Master's Thesis

Interferometer Positioning Control

2021 **Nikita Kuprin**

MASTER'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Kuprin Nikita** Personal ID number: **467918**

Faculty / Institute: **Faculty of Mechanical Engineering**

Department / Institute: **Department of Production Machines and Equipment**

Study program: **Industry 4.0**

Branch of study: **No Special Fields of Study**

II. Master's thesis details

Master's thesis title in English:

Interferometr positioning control

Master's thesis title in Czech:

Řízení polohování interferometru

Guidelines:

Description of the topic: Design of a positioning unit control for positioning a 6DOF interferometer for fast positioning and measuring the accuracy of movement in the workspace; Syllabus: 1) Research in the field of measurement using interferometers, 2) Design and creation of software for motion control of the positioning unit, 3) Design of algorithms for fast alignment of the interferometer, 4) Creation of software for reading data from the interferometer, 5) Testing and verification; Scope of the graphic part: control scheme, selected waveforms of the interferometer positioning; Range of text part: 60-80 pages;

Bibliography / sources:

Dokumentace Beckhoff [online]. Dostupné z: https://infosys.beckhoff.com/; Dokumentace Renishaw [online]. Dostupné z: https://www.renishaw.com/; STEJSKAL, V. a VALASEK, M. Kinematics and Dynamics of Machinery. New York: Dekker, 1996. ISBN-13: 978-0824797317;

Name and workplace of master's thesis supervisor:

Ing. Jiří Švéda, Ph.D., Department of Production Machines and Equipment, FME

Name and workplace of second master's thesis supervisor or consultant:

Ing. Štěpán Chládek, Ph.D., Department of Production Machines and Equipment, FME

Date of master's thesis assignment: **29.04.2021** Deadline for master's thesis submission: **25.07.2021**

Assignment valid until: **30.09.2021**

Ing. Jiří Švéda, Ph.D.	Ing. Matěj Sulitka, Ph.D.	prof. Ing. Michael Valášek, DrSc.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

. | Date of assignment receipt

Student's signature

Declaration of Authorship

I, Nikita Kuprin, declare that this thesis titled "Interferometer positioning control" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have the published work of others, this is always clearly attributed
- Where I have quoted from the work of others, the source is always given. Except for such quotations, this thesis is entirely my work.
- I have acknowledged all the main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Date: Signature:

Acknowledgement

First and foremost, I have to thank my supervisor Ing. Jiří Švéda, Ph.D. and my consultant Ing. Štěpán Chládek, Ph.D. Without their assistance and dedicated involvement in every step throughout the process, this thesis would have never been accomplished. I would like to thank you very much for your support and understanding.

I also place on record my sense of gratitude to one and all who directly or indirectly have helped this thesis.

Annotation

Author:	**Nikita Kuprin**
Title:	**Interferometer positioning control**
Extent:	70 pages
Academic Year:	2020/2021
Department:	Department of production machines and equipment
Tutor:	Ing. Jiří Švéda, Ph.D.
Submitter:	CTU FME, dept. 12135 with cooperation of TOS Varnsdorf
Application:	Measurement of volumetric accuracy
Keywords:	Beckhoff, PLC, interferometer, Renishaw XM-60, volumetric accuracy
Abstract:	The master's thesis deals with the design of control for a positioning mechanism of a 6DOF interferometer for fast alignment and measurement of the accuracy of movement in the workspace.

Anotace

Jméno autora: **Nikita Kuprin**

Název: **Řízení polohování interferometru**

Rozsah práce: 70 stran

Akad. rok vyhotovení: 2020/2021

Ústav: Ústav výrobních strojů a zařízení

Vedoucí: Ing. Jiří Švéda, Ph.D.

Zadavatel tématu: ČVUT FS, Ú12135 ve spolupráci s TOS Varnsdorf

Využití: Měření volumetrické přesnosti

Klíčová slova: Beckhoff, PLC, interferometr, Renishaw XM-60, volumetrická přesnost

Abstrakt: Předložená diplomová práce projednává o návrh řízení polohovací jednotky pro polohování 6DOF interferometru pro rychlé ustavení a měření přesnosti pohybu v pracovním prostoru.

Table of Contents

List of abbreviations

ADS	Automation Device Specification
API	Application Programming Interface
CAD	Computer Aided Design
CAM	Computer Aided Manufacturing
CMM	Coordinate-Measuring Machine
DOF	Degrees Of Freedom
GPS	Global Positioning System
HMI	Human Machine Interface
IDE	Integrated Development Environment
IO (I/O)	Inputs Outputs
IPC	Industrial PC
LED	Light-Emitting Diode
NC	Numerical Control
PLC	Programmable Logic Controller
POU	Program Organisation Unit
PSD	Position Sensitive Device
PTP	Point-To-Point
TCP	Tool Centre Point
TwinCAT	The Windows Control and Automation Technology
WPF	Windows Presentation Foundation
XAE	eXtended Automation Engineering
XAR	eXtended Automation Runtime
XM-60	Renishaw XM-60

1 Introduction

The work deals with the development of software for interferometer positioning control. The positioning is realised via the mechanism shown in Fig. 1. It was developed specifically for the Renishaw XM-60 multi-axis calibrator – a 6DOF interferometer. The mechanism consists of two units: a launch unit and a receiver unit. The launch unit positions the Renishaw XM-60 Launch. The launch unit is designed to be placed stationary in the machine's workspace, as shown in Fig. 2. The receiver unit positions the Renishaw XM-60 Launch. It is designed to be placed in the machine tool's spindle, as shown in Fig. 3.

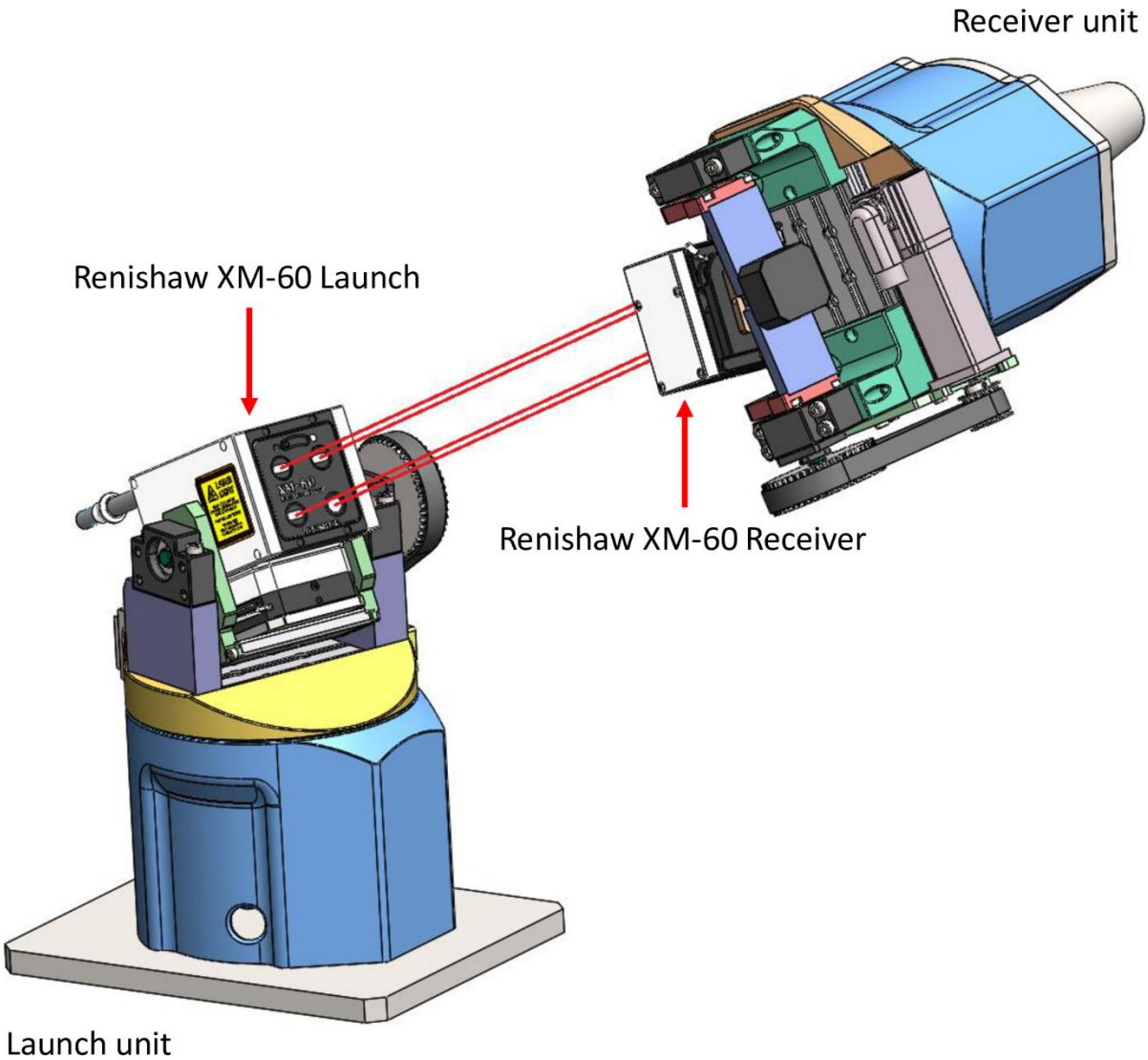

Fig. 1: Positioning mechanism

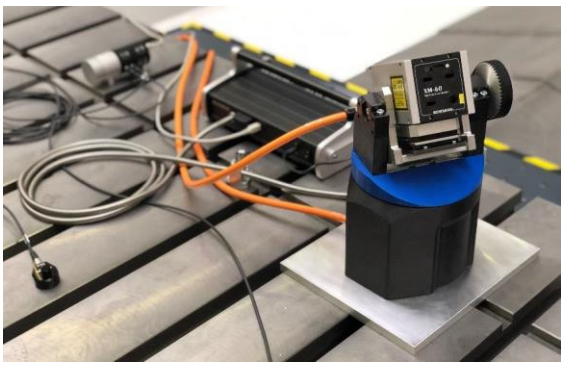

Fig. 2: Placement of the launch unit [1]

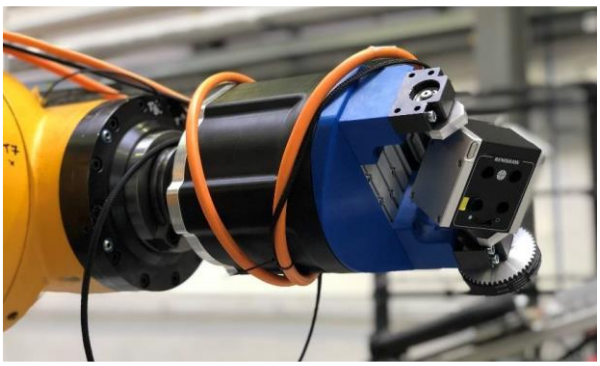

Fig. 3: Placement of the receiver unit [1]

The positioning mechanism is a part of a system for rapid identification of machine tool volumetric error using Renishaw XM-60 (Fig. 4). The system consists of:

- Automatic interferometer positioning
- Communication with the machine tool control system
- Kinematic machine model for the calculation of volumetric error over the whole operating area

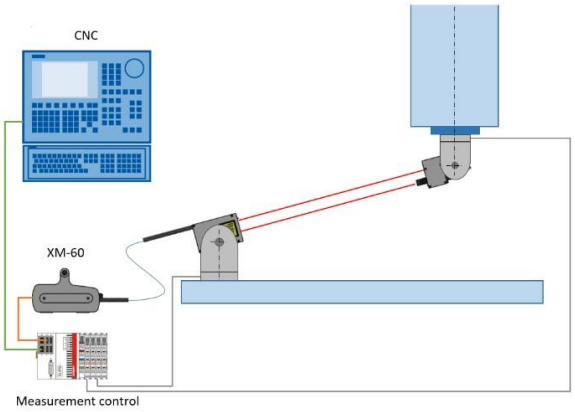

Fig. 4: System for identification of machine tool volumetric error [1]

The positioning mechanism is controlled by a PLC connected to the interferometer and the machine's control system for data acquisition. The essence of measurement lies in the automatic adjustment of the measurement path by both the machine and the interferometer to reduce the setup time. The measurement itself is performed with the interferometer position being locked to eliminate the error caused by the movement.

2 Goals

The goals for the work are as follows:

- Develop software for interferometer positioning control
- Develop software for communication with the interferometer
- Develop quick alignment algorithm

The software for interferometer positioning control should enable the manual operation of the mechanism.

The software for communication with the interferometer should enable reading the data from the interferometer and passing it to the mechanism's controller.

The quick alignment algorithm should enable quick alignment of the launch and the receiver units without knowing the exact distance between them.

3 Research

3.1 Accuracy of a machine tool

Accuracy is a critical parameter for a machine tool. Increasing accuracy is one of the prerequisites for the competitiveness of machine tool manufacturers. Furthermore, the demands on accuracy are ever-increasing. The traditional method of producing accurate machines was to increase mechanical accuracy. That method required substantial costs and efforts in production and maintenance. Thanks to the advancement in machine control systems, it is now possible to improve accuracy using software compensation. However, the measurement of errors is essential for software compensation to work. [2]

The accuracy of a machine tool can be divided into the following categories:

- Geometric accuracy
- Positioning accuracy
- Working accuracy
- Interpolation, control system and CAM accuracy
- Thermal dilation
- Error due to compliance

The errors mentioned above contribute to the volumetric accuracy – a vector map of error deviations in the workspace (Fig. 5). According to the ISO 230-1 standard, the volumetric accuracy for a 3-axis machining centre is defined as the maximum range of relative deviations between the actual and ideal position in X, Y, Z directions and the maximum range of orientation deviations for direction and motions of all axes in the defined workspace. The evaluation of volumetric accuracy is described in the CSN ISO 230-6 standard. [3]

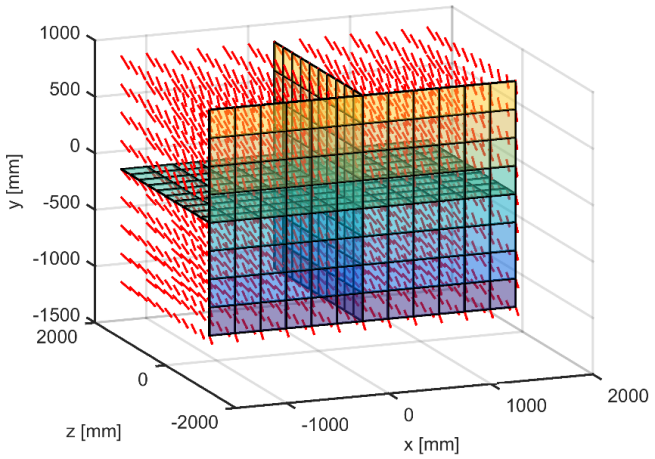

Fig. 5: Volumetric error map [1]

3.1.1 Geometric accuracy

Geometric accuracy represents the production quality of a machine tool. It is measured in the unloaded state. The measurement and evaluation are defined in the CSN ISO 230-1 standard. According to the standard, each axis has six geometric errors: positioning error, horizontal straightness, vertical straightness and three angular errors (roll, pitch, yaw). A typical three-axis machine tool contains 21 geometric errors – three other errors come from the relative squareness between the axes (Fig. 6). [4]

Geometric accuracy can be measured using various methods: straight edge with gauge, interferometer, laser tracker.

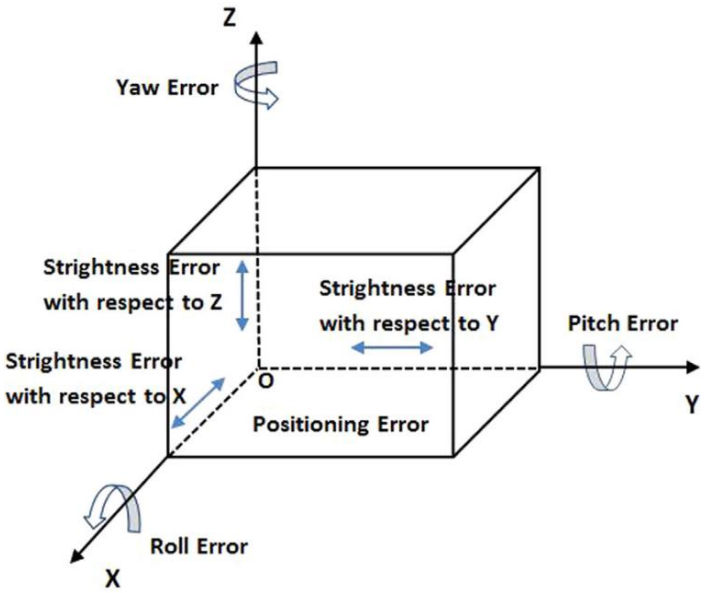

Fig. 6: Geometric errors [5]

3.1.2 Positioning accuracy

Positioning accuracy is defined as the distance between the programmed and the actual linear or rotary axis position. It is closely related to repeatability, defined as a deviation of actual positions when reaching the same programmed position during multiple attempts under the same conditions. The measurements are usually performed in an unloaded state. The preferred device for measurement is an interferometer. [3]

3.1.3 Interpolation, control system and CAM accuracy

The interpolation accuracy is defined as a deviation between the measured and programmed path. The measurement and evaluation are described in the CSN ISO 230-4 standard. This accuracy is a combination of errors caused by various systems. Firstly, there is an error in CAM software when the path is generated from the CAD model. Then, the control system optimises the NC code, using filters, for example, where the generated linear sections are transformed to splines, thus causing deviations. Finally, a control loop in the drive also contributes to the error. [3]

3.1.4 Thermal dilation

Machine tools are exposed to even and uneven temperature effects. Thus, deformations arise, which lead to a change in the position of the workpiece relative to the tool resulting in inaccuracies. This will be most prominent in the stability of a machined dimension. Heat sources can be found either in the machine itself (passive resistors in the motion axes or the cutting process) or outside. The thermal stability of machine tools is one of the most important factors for maintaining the specified tolerances on the workpiece.

Compensation of thermal errors can be divided into two categories:

- Direct compensation and
- Indirect (software) compensation.

With direct compensation, the thermal deviations at the TCP are intermittently measured and superposed to the desired position value of the particular axis. The disadvantage of the direct approach is the interruption of the machining process to measure the displacement and the extra costs of the measuring. The principle of indirect (software) compensation is the readjustment of the positioning of the axes by the control system, which is based on corrections calculated by a mathematical model. [6]

3.1.5 Error due to compliance

The error is related to the design of a machine tool. There can be deformation in certain parts of the machine tool in an unloaded state due to compliance. This type of error is usually compensated using look-up tables. [4]

3.2 Measurement methods

As mentioned in the previous chapter, the measurement of errors is crucial when using software compensation for improving machine tool's accuracy. Measurement methods are constantly evolving to keep up with the increase in accuracy demand. The most common devices for measuring volumetric accuracy are: [3]

- Laser trackers
- Laser tracers
- Interferometers

3.2.1 Laser tracker

A laser tracker is a device that projects a laser beam onto an optical target that is in contact with a measured object and determines the three-dimensional position of the target (Fig. 7). The optical targets reflect the laser beam to its emitted origin point – typically a spherically mounted retroreflector (SMR). Dimension values, angles, geometrical tolerances are determined by the software that calculates the coordinates of the retroreflector. Advanced systems can measure all six geometric errors simultaneously. The biggest advantage of laser tracker is the ease of use and setup. The most significant disadvantage is accuracy – usually around $0.025\ mm$. [7]

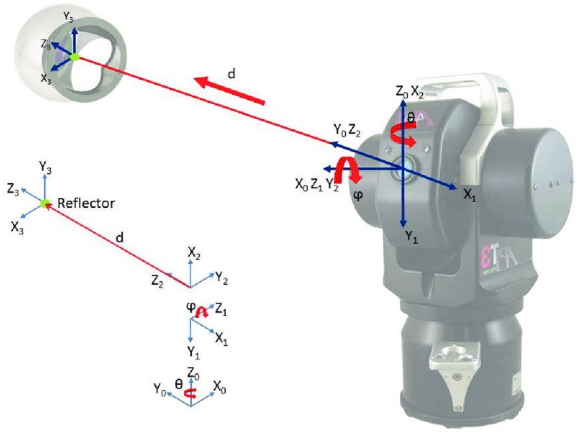

Fig. 7: Laser tracker [7]

3.2.2 Laser tracer

A laser tracer is a self-tracking laser interferometer that is usually used to measure geometric errors of CMMs and machine tools with the highest accuracy. This method is similar to the laser tracker. The biggest difference is the structure of the laser head – the reflected beam is focused on the sphere centre by a lens (Fig. 9). Also, multilateration can be used with a laser tracer – the principle is shown in Fig. 8. The method is presented as an analogy to the GPS. [8]

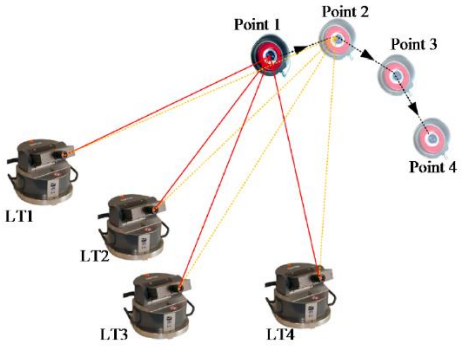

Fig. 8: LaserTRACER [8]

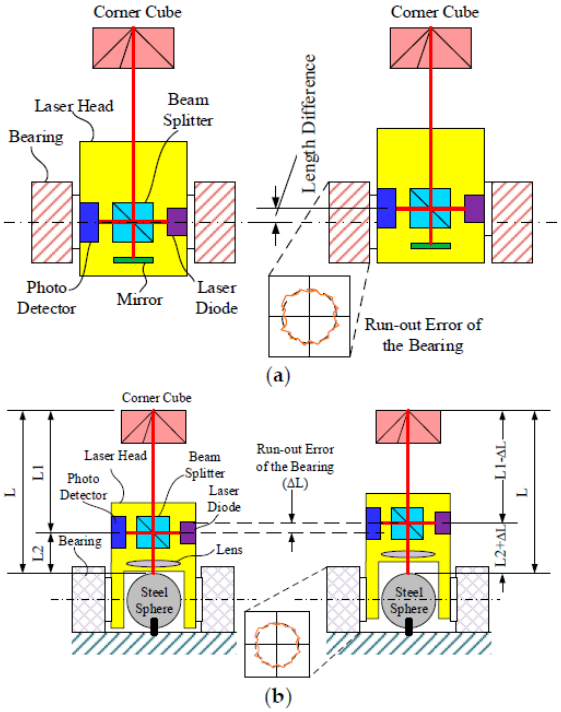

Fig. 9: Comparison between Laser Tracker (a) and LaserTRACER (b) [8]

3.2.3 Interferometer

The interferometer principle is based on the "interferometry" – measurement method using the phenomenon of interference of waves. The underlying principle of modern interferometers is based on a Michelson interferometer (Fig. 10). It consists of a beam splitter and two mirrors. When the light goes through the mirror/beam splitter, it is split into two beams with different optical paths: one going to the reference mirror and the other going to the measurement mirror. After being reflected at the mirrors, beams recombine again at the beam splitter before arriving at the detector. The path difference of these two beams causes a phase difference which creates an interference fringe pattern. The detector then analyses this pattern to evaluate the wave characteristics, material properties or the displacement of one of the mirrors. [9]

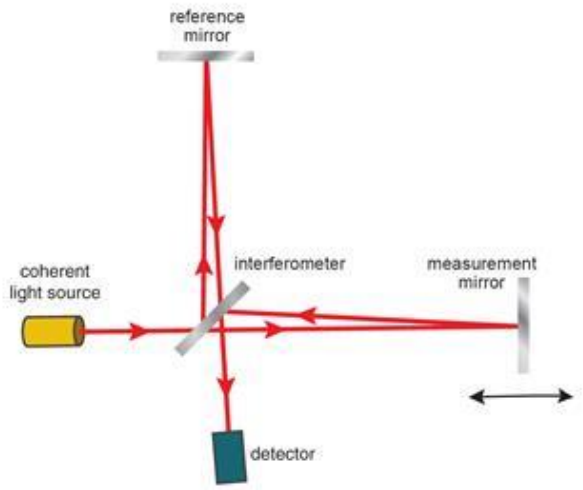

Fig. 10: Michelson interferometer [9]

Positioning accuracy is the most common form of measurement made with a laser interferometer. The laser system measures linear positioning accuracy and repeatability by comparing the programmed position of the machine to the actual position measured by the interferometer. Thus, this device can measure a single linear error simultaneously, and it is necessary to reconfigure the setup to measure the other errors. It is possible to combine multiple interferometers into a single unit forming a so-called multi-axis calibrator to measure all the errors simultaneously. [3]

3.3 Renishaw XM-60

XM-60 is a laser measurement system capable of measuring all six geometric errors in the axis from a single capture. The errors are as follows (numbering corresponds to Fig. 11): [10]

1. Linear positioning of the axis
2. Straightness in the horizontal plane
3. Straightness in the vertical plane
4. Roll
5. Yaw
6. Pitch

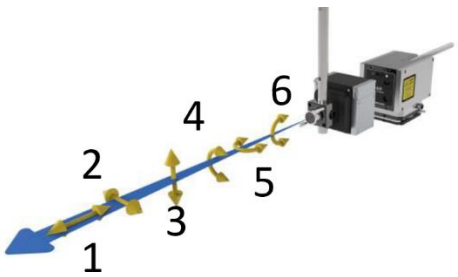

Fig. 11: Errors [10]

The system uses three laser beams (1,2, and 3) to measure the linear, pitch and yaw errors using interferometry. The light-emitting diode (LED) beam (4) is used for straightness and roll measurements. The 4th (diode source) beam is used to measure straightness and roll. Beams are illustrated in Fig. 12. [10]

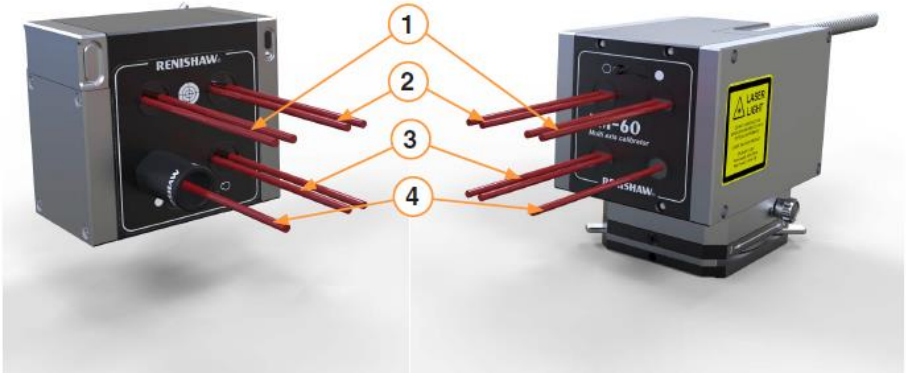

Fig. 12: XM-60 beams [10]

The specified accuracy of interferometric measurements is only valid for specific environmental conditions. Thus, XM-60 comes with the environmental compensator XC-80 (Fig. 13). It measures air temperature, pressure, and relative humidity to adjust the laser wavelength to environmental conditions. Also, there is a temperature sensor for a machine to compensate for thermal dilation. [10]

Fig. 13: XC-80 [10]

4 Hardware modifications

The testing on an actual device is a significant part of software development. It can serve as a method for the identification of areas for improvement of a hardware design. Such an approach was implemented in this work. The positioning mechanism was in the form of a prototype unit in the initial state, as shown in Fig. 14. Extensive testing has helped to identify the following areas for improvement:

- Arrangement of IO modules and IPC
- Safety features

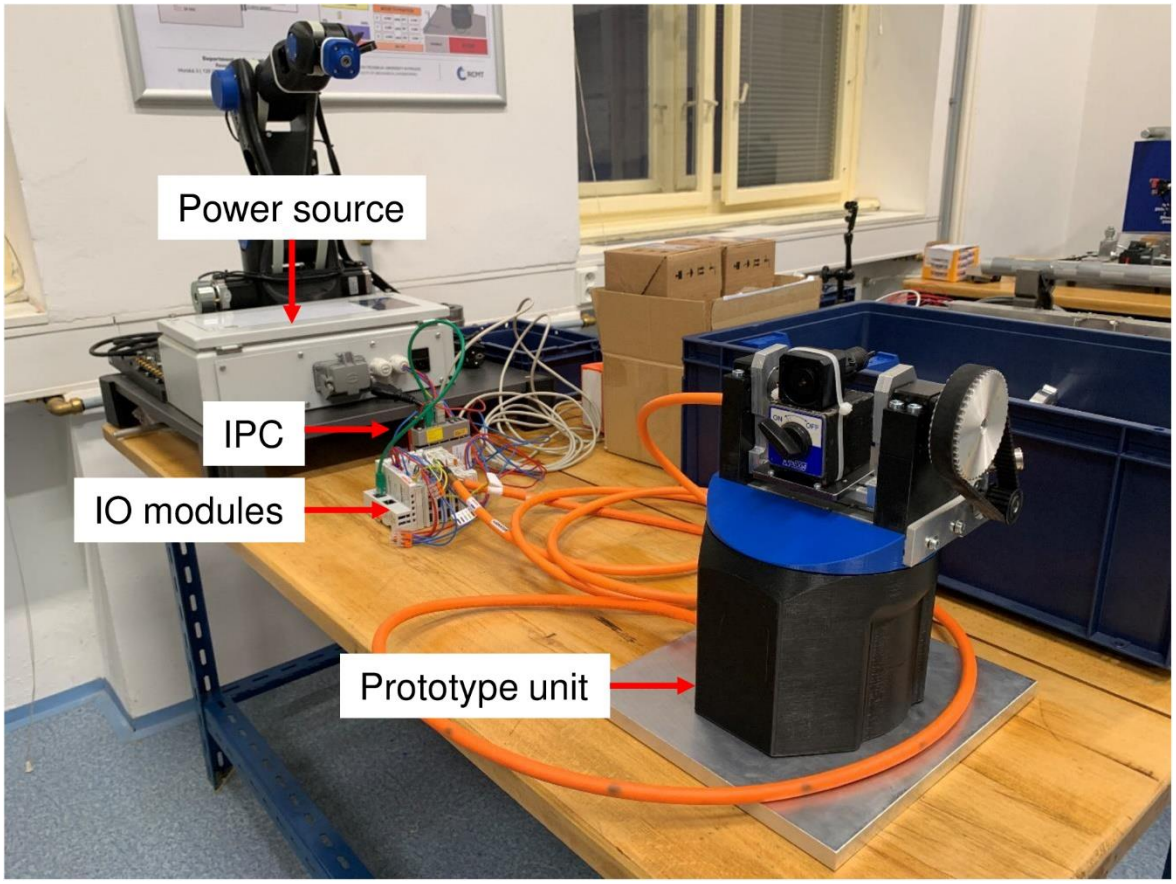

Fig. 14: Initial state

4.1 Arrangement of the IO modules and IPC

Initially, IO modules and IPC were placed in an enclosure and positioning mechanism units connected through motor cables. However, during the testing, it was identified that manipulation with long motor cables is problematic. Moreover, the cables must be fixed to the enclosure because they have connectors only on one side, making the enclosure bulky and unergonomic for transport. Based on this feedback, the mechanism was redesigned. Comparison between the initial and modified designs of the mechanism can be seen in Fig. 15 and Fig. 16. The result of modifications is shown in Fig. 19. In summary, the following modifications were introduced:

- Place IPC in a separate enclosure with power sources (Fig. 17)
- Add enclosures for I/O modules to each of the units (Fig. 18)
- Add HARTING Han connectors – power and data lines run together in one cable (Fig. 19)

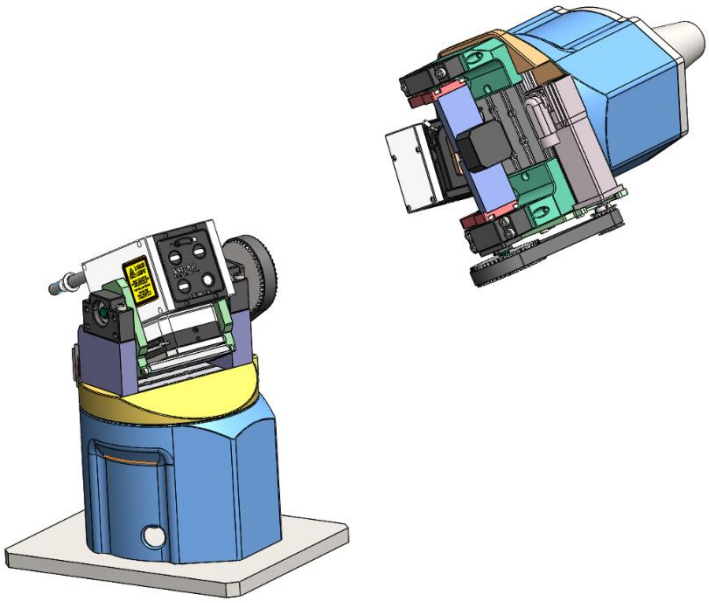

Fig. 15: Initial design

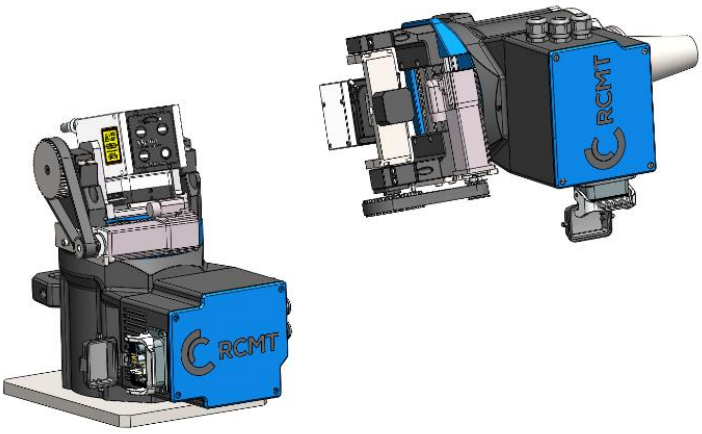

Fig. 16: Modified design

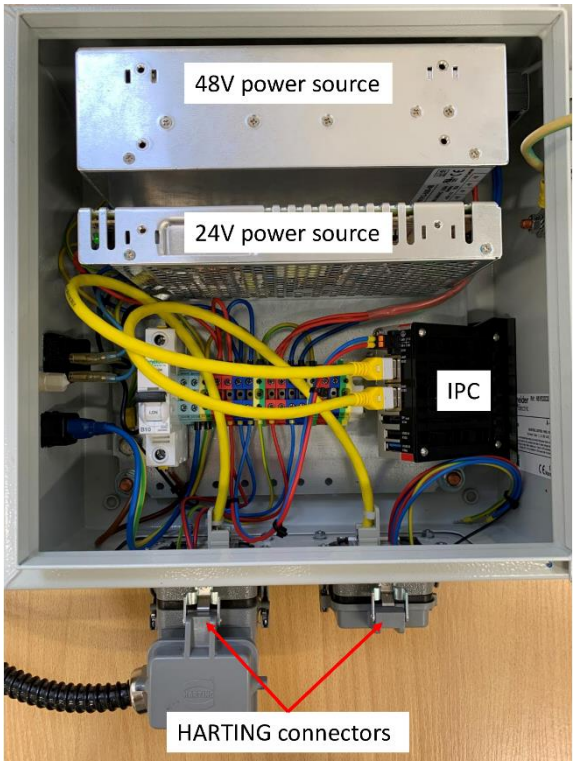

Fig. 17: Main enclosure

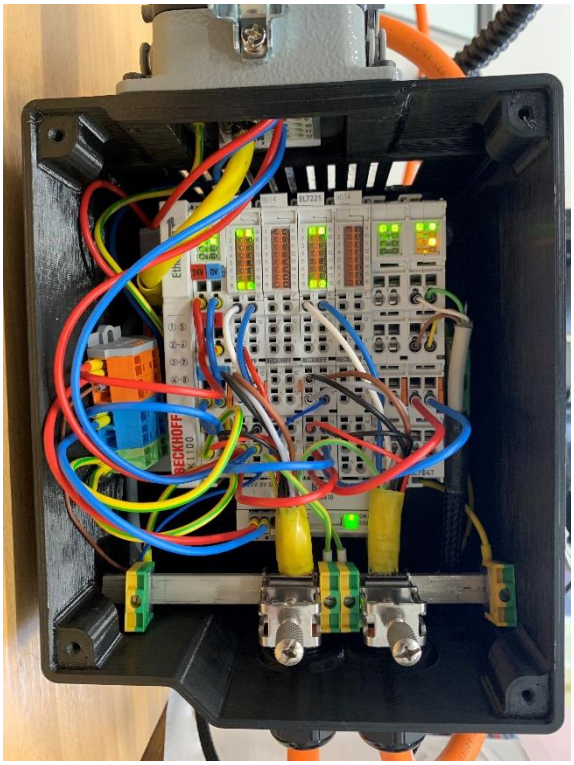

Fig. 18: Enclosure for I/O modules

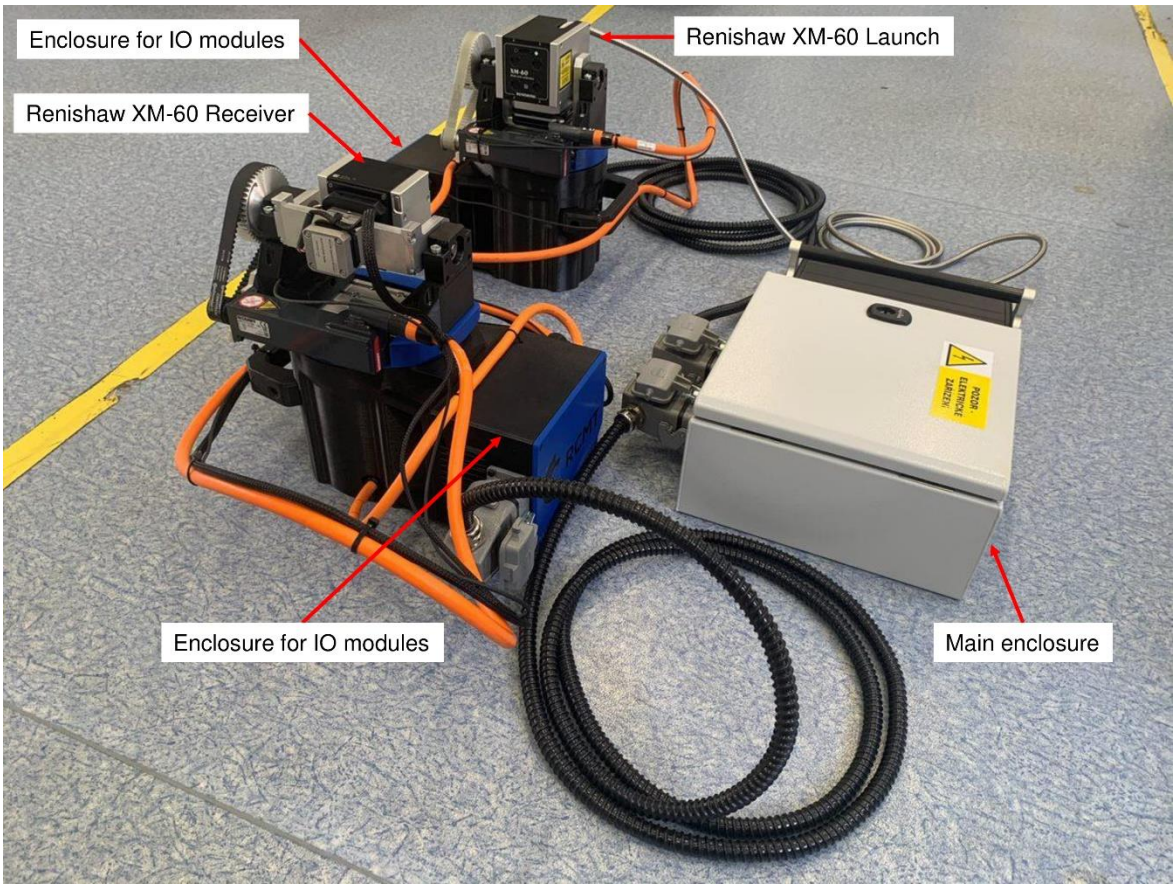

Fig. 19: State after modifications

4.2 Safety features

During the testing, several instances of strange behaviour from the Beckhoff controller led to uncontrolled motion. Due to this reason, it was decided to include autonomous safety measures. The servo drives have Safe Torque Off (STO) inputs that need to be energised with 24 volts to enable the motors. If STO input becomes de-energised during the motion, the axis comes to a controlled stop.

No axis of the positioning mechanism uses the full range of motion (360°) – rotation around horizontal axes (Ax2 and Ax4 in Fig. 26) is restricted to 120°, and rotation around vertical axes (Ax1 and Ax3 in Fig. 26) is restricted to 350°. Software limit switches are used to enforce the restrictions. Motion beyond the allowed range indicates an error. Thus, hardware limit switches can be placed after software limit switches to detect the error.

The principle described in the above paragraph can be used to de-energise STO inputs in case of error. Instead of limit switches, it was decided to use normally-closed (NC) induction sensors. When the axis is within the allowed, range the STO input is energised. However, as soon as the axis leaves the allowed range, it is stopped because STO becomes de-energised. For limiting the rotation around the vertical axis, an NC induction sensor was placed on the stationary base, and a washer was placed on the underside of the rotary table, as shown in Fig. 20. The contact is open when the washer is on top of the sensor. The solution for limiting the rotation around the horizontal axis is shown in Fig. 21 – the contact becomes open when the sensor leaves the grove on the belt pulley.

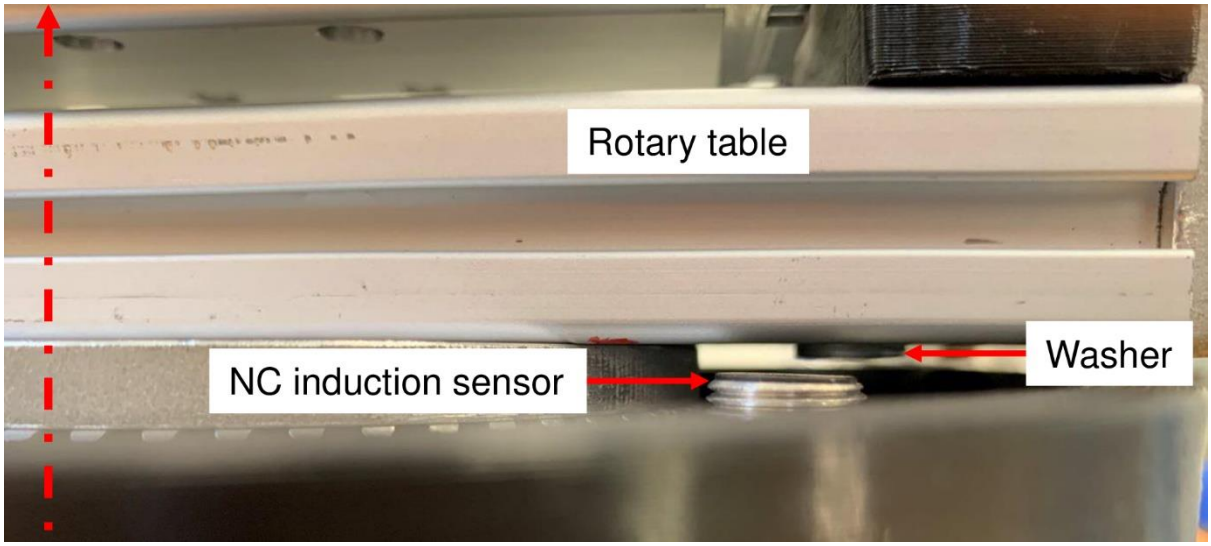

Fig. 20: Induction sensor – rotation around the vertical axis

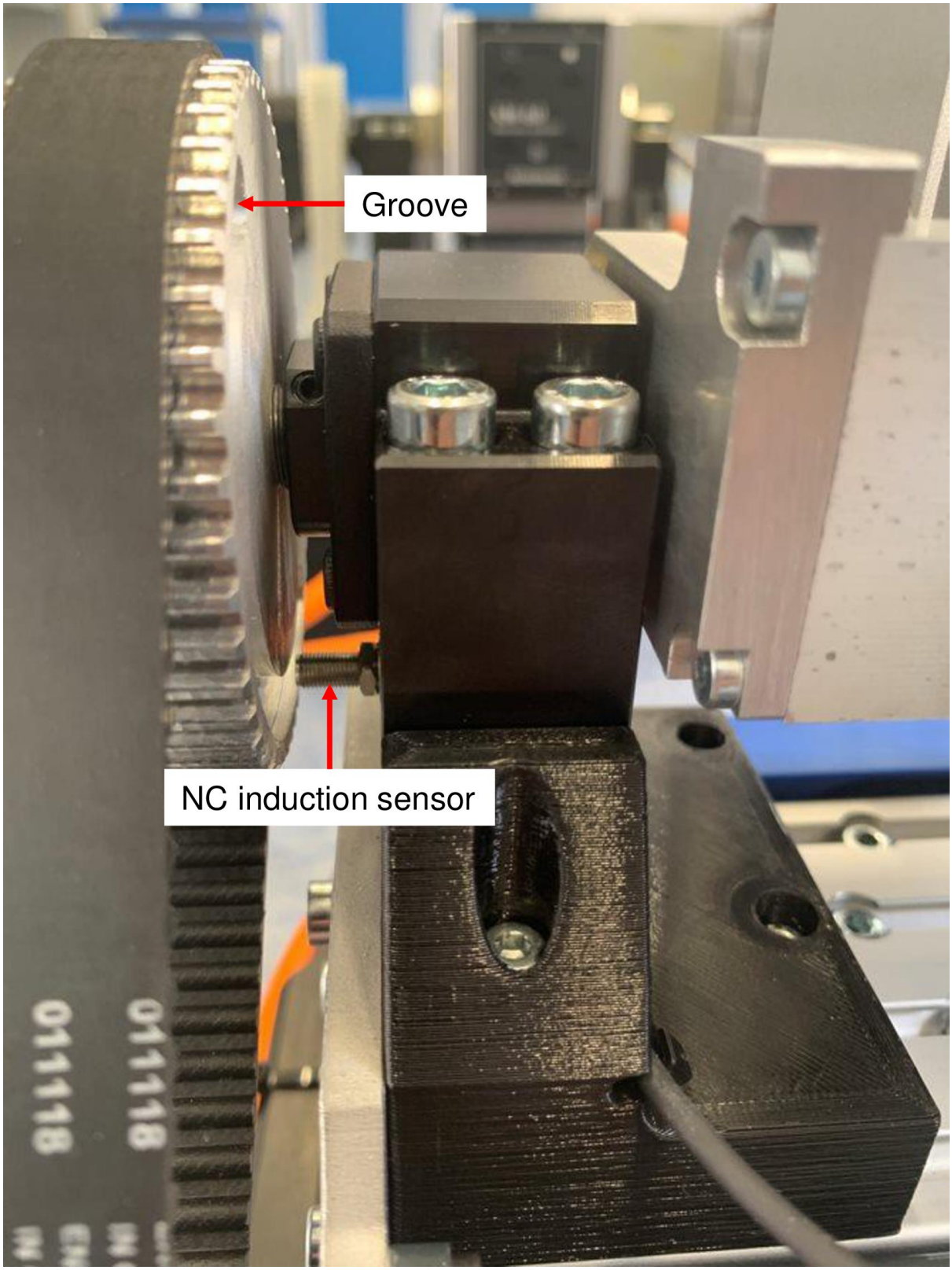

Fig. 21: Induction sensor – rotation around the horizontal axis

5 Development of control

The control aims to enable the manual operation of the mechanism. The development of control can be split into the following steps:

1. Commissioning of axes
2. Servo tuning
3. Creation of PLC project
4. Creation of HMI

However, firstly, it is essential to provide background information about the development environment used in this project. The control was developed using the TwinCAT 3 software from Beckhoff. TwinCAT (The Windows Control and Automation Technology) is software for developing and real-time control of Beckhoff PC-based systems. TwinCAT 3 is classified into two parts:

- eXtended Automation Engineering (XAE)
- eXtended Automation Runtime (XAR)

TwinCAT XAE is a software development tool. It is integrated into the Microsoft Visual Studio development environment as an extension (TwinCAT Integrated in Fig. 22). If the full version of the Microsoft Visual Studio is not installed, TwinCAT 3 setup automatically installs the necessary Visual Studio Shell (TwinCAT Standard in Fig. 22).

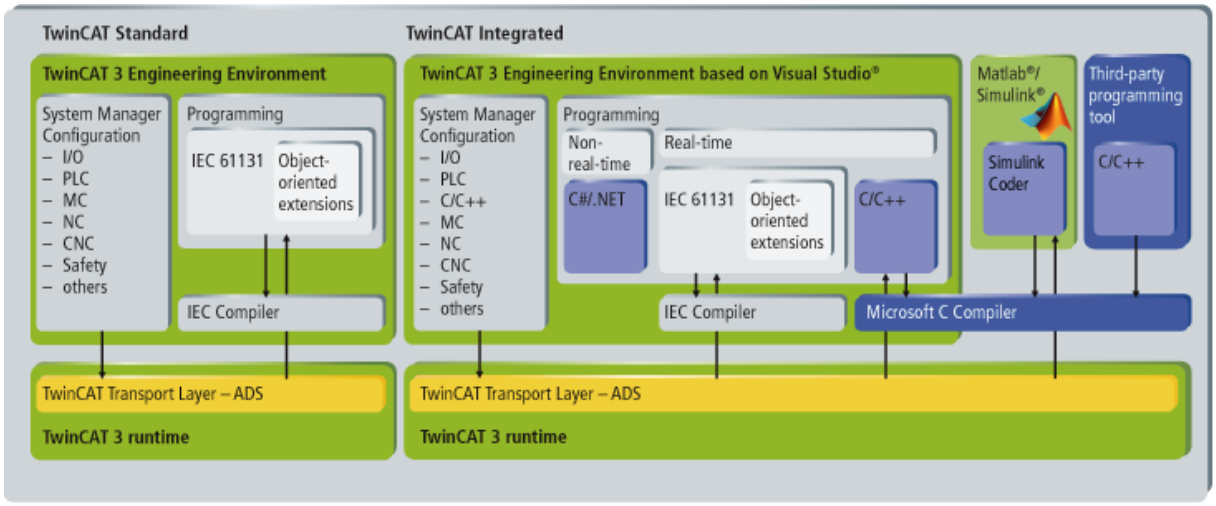

Fig. 22: TwinCAT XAE [11]

TwinCAT XAR is a real-time environment. It incorporates the modular structure shown in Fig. 23. The modules can be programmed independently in different languages and therefore compiled using different compilers. The generated modules are subsequently cyclically called during runtime using tasks. Individual tasks can be allocated to different cores of a CPU, as shown in Fig. 24. Thus, the performance of the multicore PCs can be used at maximum. [11]

TwinCAT 3 runtime consists of basic components that can be extended with functions, as shown in Fig. 25. For this project, the following components and functions are required:

- TC3 ADS – enables communication with TwinCAT systems
- TC3 IO – enables variable-oriented linkage of I/O devices to tasks
- TC3 PLC – enables running one or more soft PLCs on an industrial PC
- T3 NC PTP – enables motion control for point-to-point movements

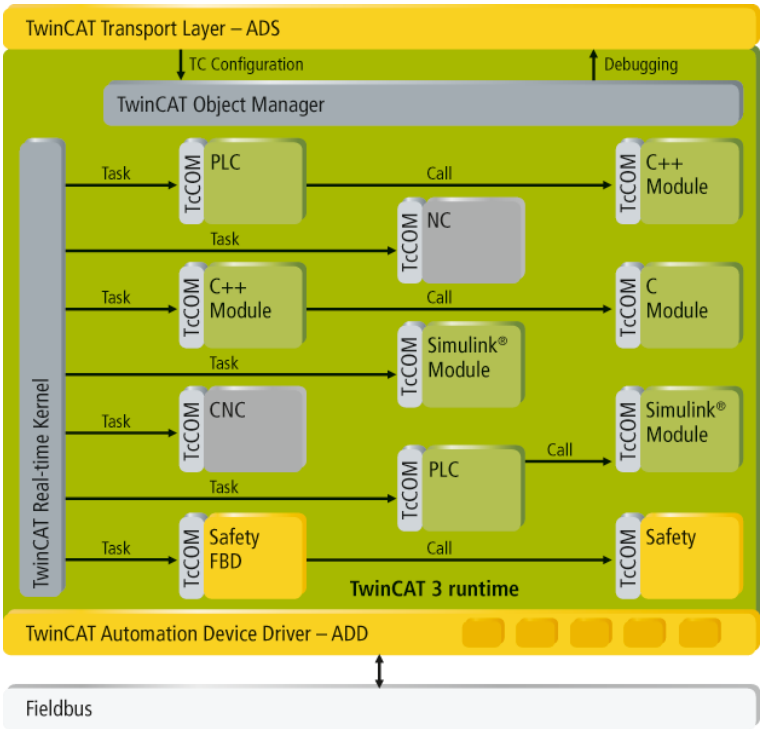

Fig. 23: TwinCAT XAR [11]

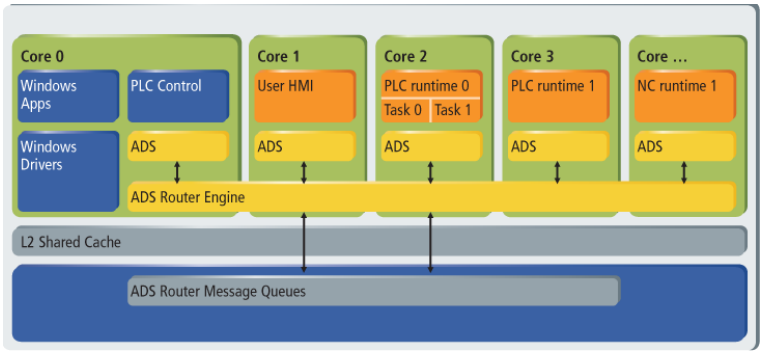

Fig. 24: Multicore functionality [11]

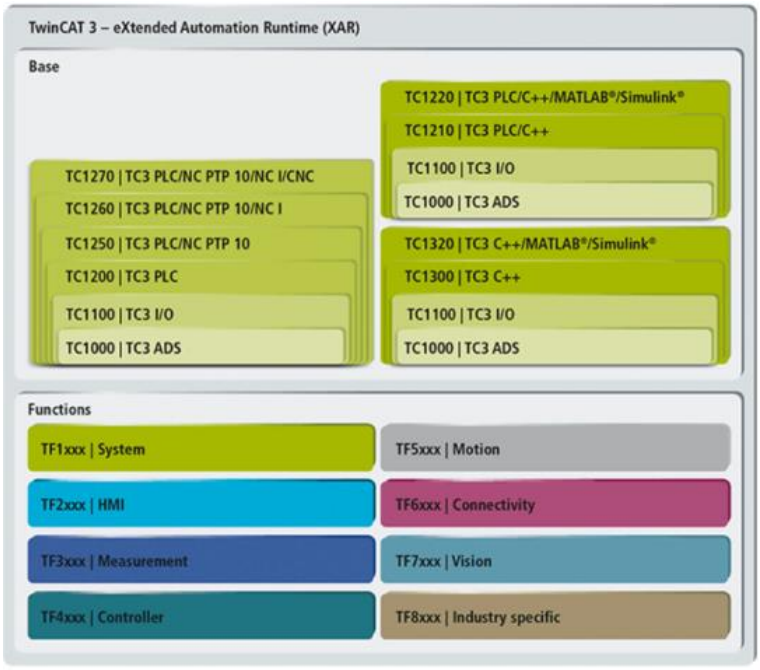

Fig. 25: TwinCAT 3 basic components and functions [11]

5.1 Commissioning of axes

The positioning mechanism consists of two units: a launch unit and a receiver unit. The arrangement of axes is shown in Fig. 26. The launch unit has two axes, all of which are driven by servomotors – Ax1 and Ax2. The receiver unit has three axes: Ax3 and Ax4 are driven by servomotors, Ax5 is driven by a stepper motor. The axes' motors and drives are listed in Table 1.

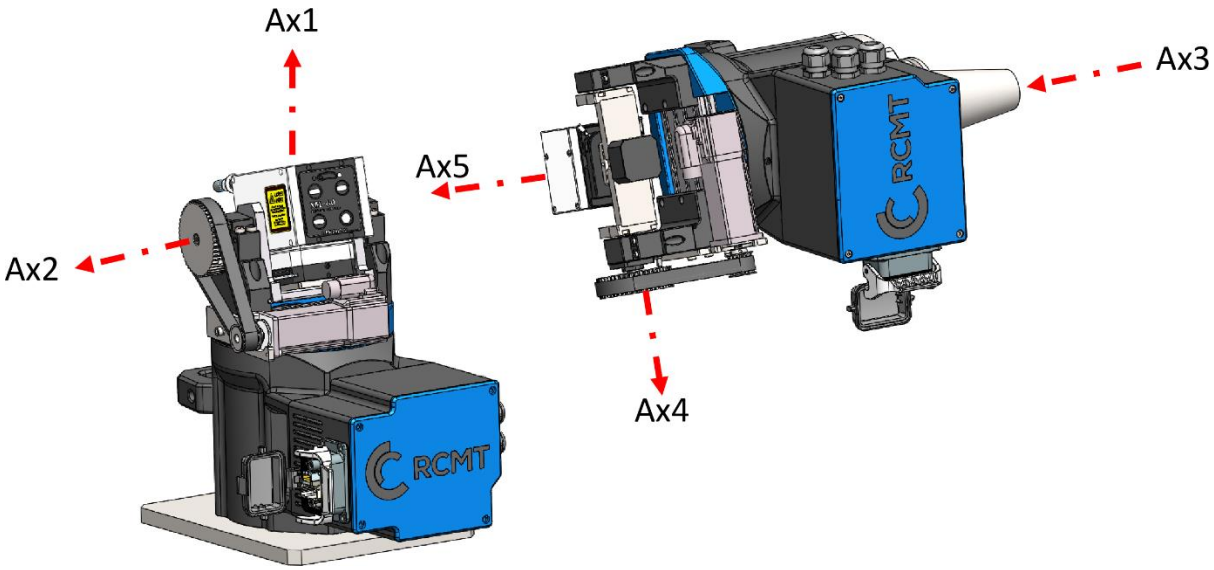

Fig. 26: Arrangement of axes

Table 1: Motor-drive combinations

Axis	Motor	Drive
1	Beckhoff AM8121-1F21-0000	Beckhoff EL7221-9014
2	Beckhoff AM8121-1F21-0000	Beckhoff EL7221-9014
3	Beckhoff AM8112-1F21-0000	Beckhoff EL7221-9014
4	Beckhoff AM8112-1F21-0000	Beckhoff EL7221-9014
5	MICROCON SX17-1003LQEF	Beckhoff EL7047

The positioning mechanism is controlled using a PC-based controller from Beckhoff, located in the main enclosure with power sources. It is connected to the EK1100 couplers of the launch and the receiver units through EtherCAT real-time bus. The EK1100 couplers are connected to the IO modules. The devices can be added automatically to the TwinCAT XAE project using the I/O section of the project tree using a scan option. The IO devices of the positioning mechanism are shown in Fig. 27. A device item indicates the network adapter where the EK1100 coupler is connected – the launch unit is Device 3, and the receiver unit is Device 1. The IO modules can be found under the Term … (EK1100) section with the component's name in the brackets. The description of IO modules is provided in Fig. 27: IO devices

Table 2.

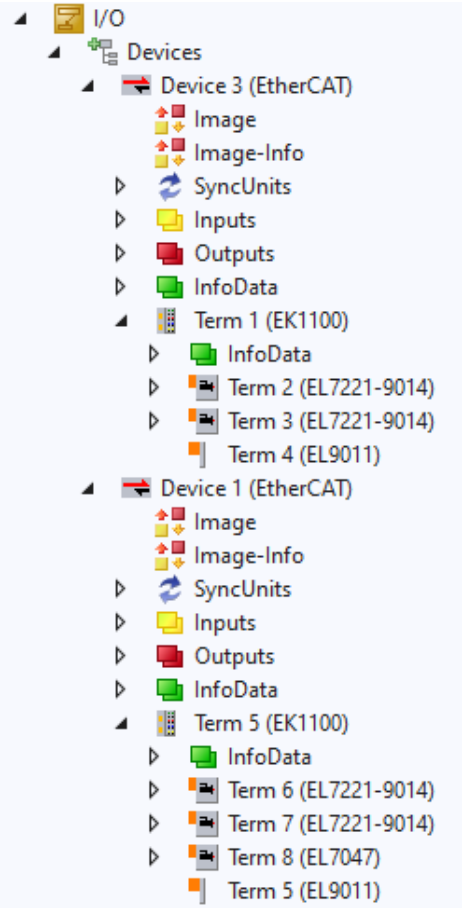

Fig. 27: IO devices

Table 2: Description of IO modules

Component	Description
EK1100	EtherCAT Coupler
EL7221-9014	Servomotor terminal
EL7047	Stepper motor terminal
EL9011	Bus end cover

In the next step of the commissioning, it is necessary to set the motor and feedback parameters. The parameters for the Beckhoff servomotors can be read automatically from the digital plates and can be conveniently accessed through a Drive Manager tool, as shown in Fig. 28. The parameters for the stepper motor must be set manually using the CAN application protocol over EtherCAT (CoE) parameter list.

Next, it is necessary to set a scaling factor to convert readings from the encoder to the actual units. It is set using two parameters:

- Numerator – distance travelled per motor revolution in actual units
- Denominator – increments per motor revolution

For the Beckhoff servomotors, it is only necessary to set the numerator using the Nc feed constant in the Scaling and NC parameters of the Drive Manager (Fig. 28). The denominator is read automatically from digital plates. The servo motors are connected to a belt drive with a 1:3 ratio. Hence, the Nc feed constant is 120°.

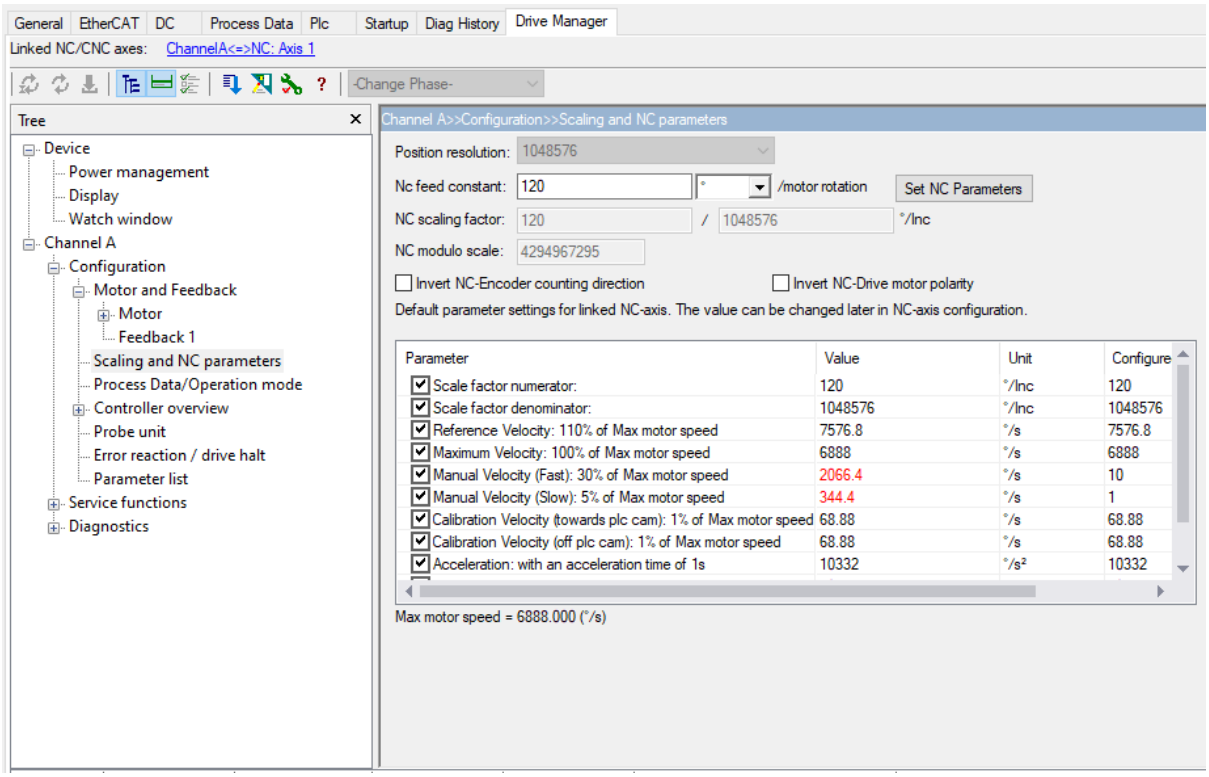

Fig. 28: Drive Manager – Scaling and NC parameters

The stepper motor does not have an encoder. Instead, it uses a step count for the determination of the position. There are 200 full steps divided into 64 micro-steps per motor revolution. Thus, the total number of steps per motor revolution is 12800. This value corresponds to the denominator. The numerator is 360° because there is no gearing mechanism.

Next, NC axes must be added to the project. Physical axes are represented by axis objects which provide a cyclic interface (e.g., for a PLC). Axis objects are located within the MOTION subtree (Fig. 29). They can be added automatically during I/O scanning or manually. The objects without the letter "e" at the end (e.g. Axis 1) were added automatically. The rest of the objects are extensions (e.g. Axis 1 e). The purpose of the extension is explained in chapter 6.

The axis object settings can be accessed by double-clicking the axis object item in the project tree and navigating to the Settings tab, as shown in Fig. 30. There, the servomotor terminal linked to the axis can be seen under Link To I/O field. During commissioning, the axis can be controlled using the Online tab (Fig. 31). Additional functions for the axis movement are available via the Functions tabs. The end-user will be controlling the mechanism via HMI rather than through IDE. Thus, an interface for accessing the axis object functionality is needed. Such an interface can be provided using PLC.

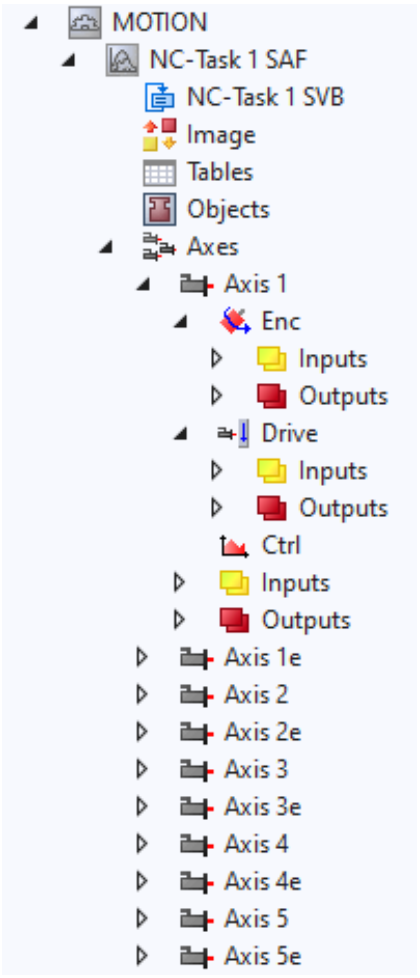

Fig. 29: Axis objects

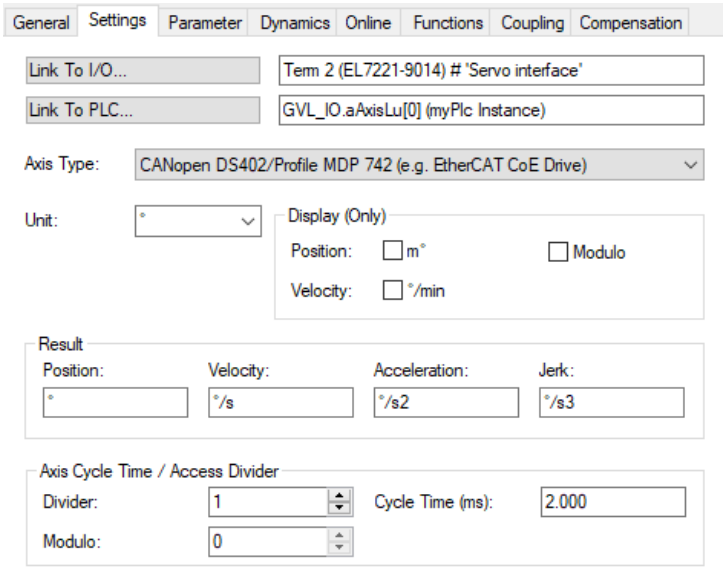

Fig. 30: Axis object settings tab

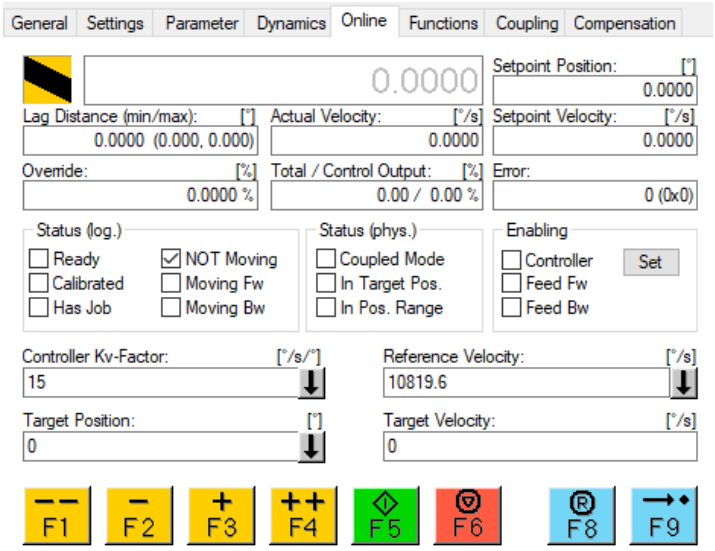

Fig. 31: Axis object online tab

5.2 Servo tuning

Servo tuning deals with the setting up of parameters of velocity and position controllers. Beckhoff's servo drives (EL7227-9014) use a cascade arrangement of controllers, as shown in Fig. 32. The parameters can be conveniently set using the Drive Manager – Controller overview (Fig. 32). The stepper motor driver (EL7047) has the same controller arrangement. However, it lacks the visual interface of the Drive Manager. Instead, the parameters must be set via the CoE list (CAN over EtherCAT).

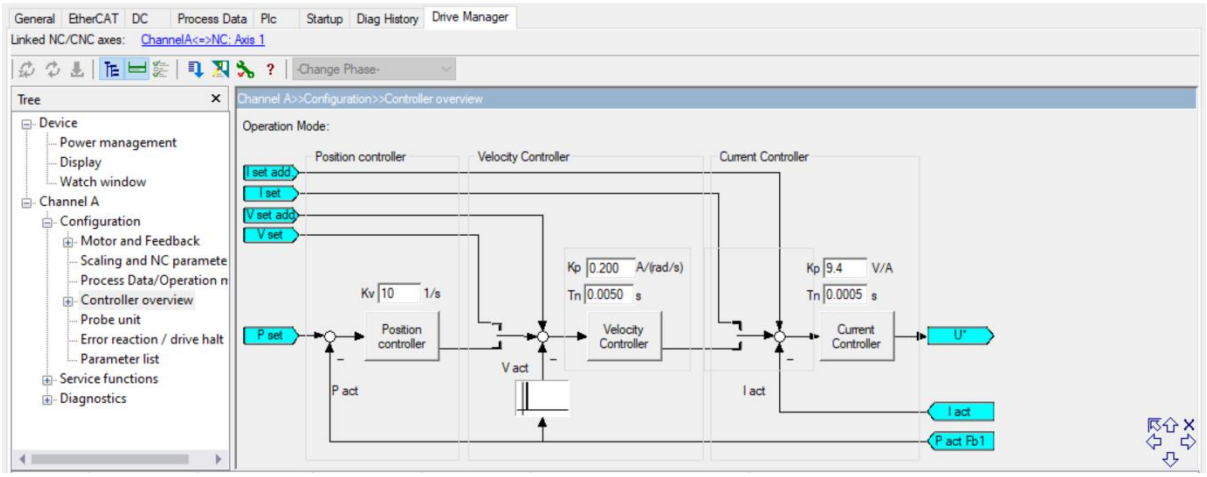

Fig. 32: Drive Manager – Controller overview

The tuning starts with velocity controller with eliminated position controller (Kv=1) and disabled feedforward. Next, the position controller is tuned. Finally, the feedforward weight is adjusted.

5.2.1 Velocity controller

The default parameters of the velocity controller enable an axis to be operated without a load. For proper functioning with load proportional gain Kp and integral action time, Tn must be adjusted to find a good compromise between responsiveness and noise generation. Beckhoff recommends the following tunings procedure [11]:

1. Eliminate the influence of position controller and integral part of a velocity controller
2. Set up a scope to monitor actual and feedback values of velocity and acceleration
3. Set up "Velo Step Sequence" function from NC functions
4. Increase Kp in small steps until stability limit is reached – the axis starts to oscillate as shown in Fig. 27
5. Calculate correct proportional gain using the formula $Kp = 0.45 \times Kp_{crit}$
6. Adjust Tn according to application. In many cases, the default value Tn = 8 ms is sufficient

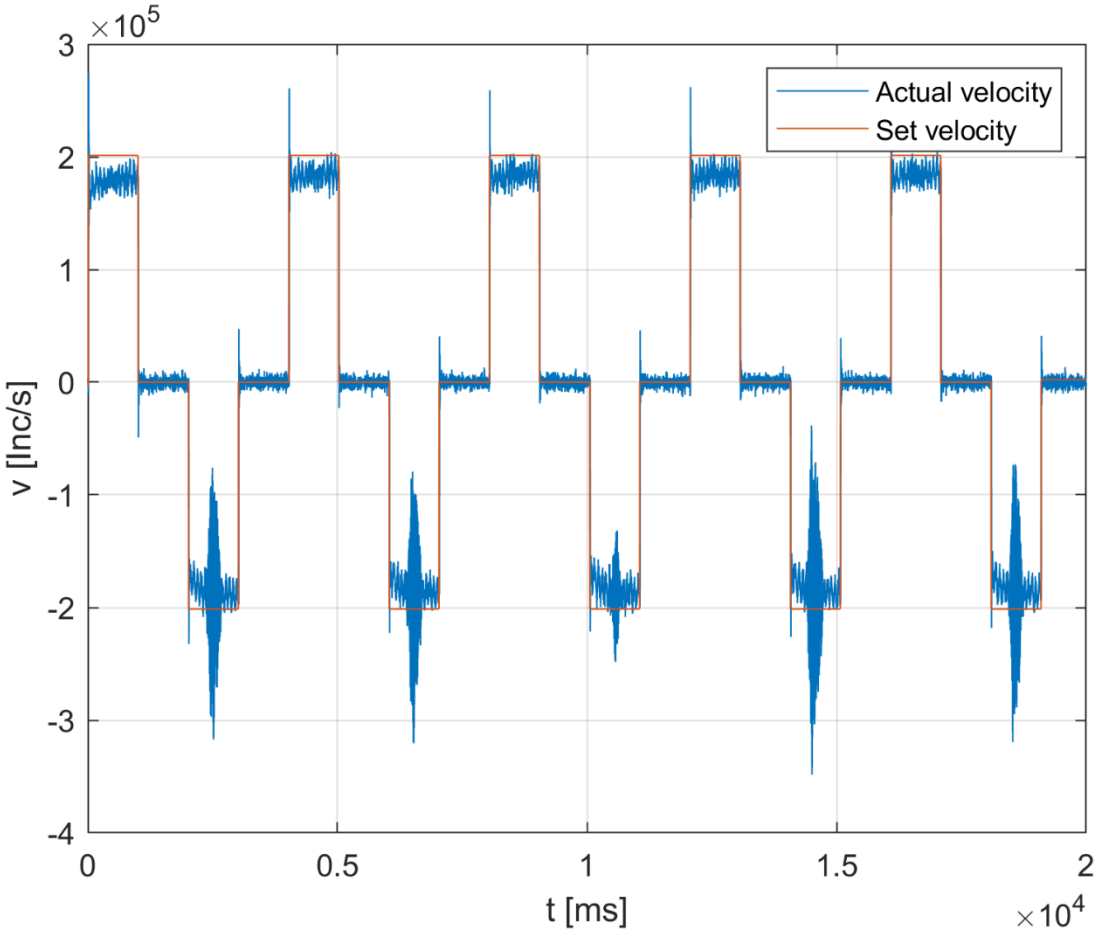

Fig. 33: Oscillation for Kp_{crit}

The following parameters were obtained after tuning according to the procedure above (Table 3):

Table 3: Velocity controller parameters

Axis	Kp [A/(rad/s)]	Tn [s]
1	0.2	0.005
2	0.2	0.005
3	0.2	0.005
4	0.2	0.007
5	0.5	0.01

Table 4: Position controller parameters

Axis	Kv [°/s/°]	Feedforward weight [-]
1	15	0.5
2	15	0.5
3	15	0.5
4	15	0.5
5	15	0.5

5.2.2 Position controller

The position controller is tuned by increasing the proportional gain Kv in small steps with feedforward disabled. The performance is evaluated by checking position lag during the motion of the type position ramp. The under-tuned controller is shown in Fig. 34. The ideally-tuned controller is shown in Fig. 35. The final step is to set the feedforward gain such that there is minimal overshooting, as shown in Fig. 36. The obtained parameters are provided in Table 4.

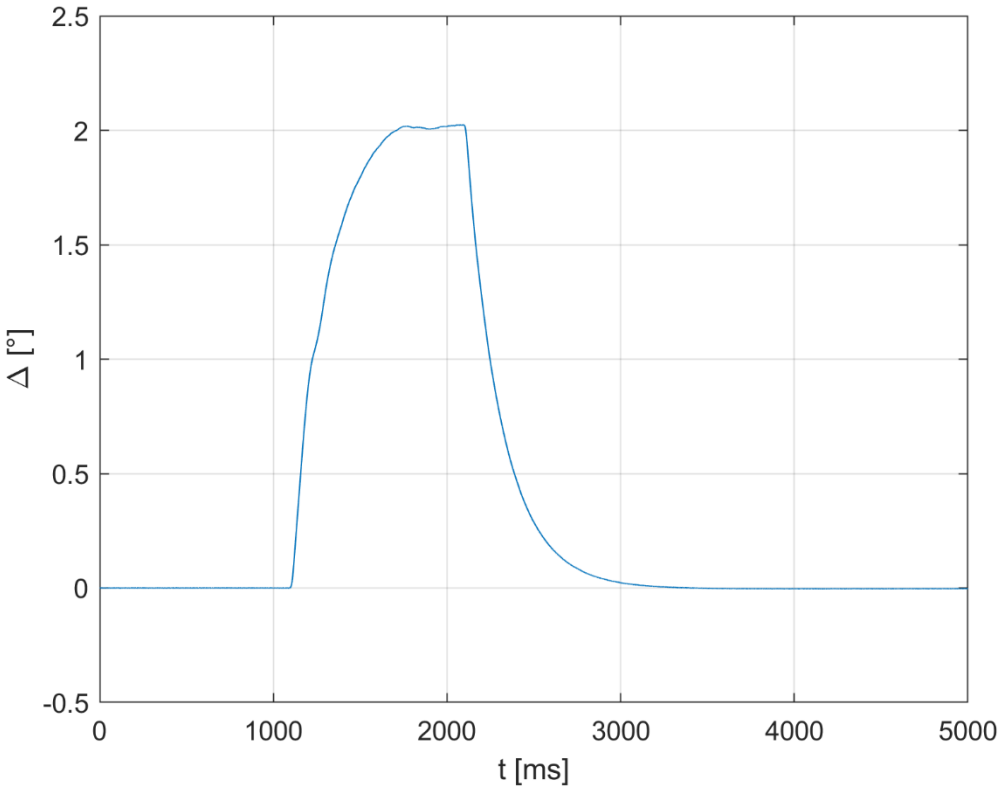

Fig. 34: Position lag of under-tuned controller

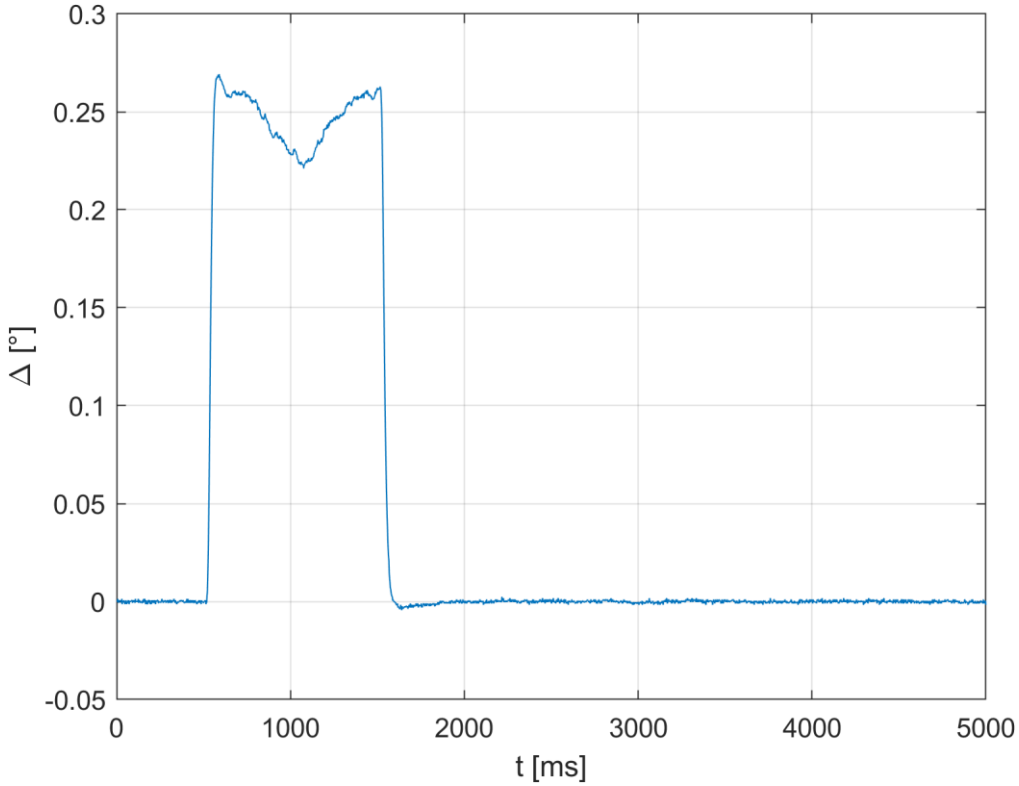

Fig. 35: Position lag of ideally-tuned controller without feedforward

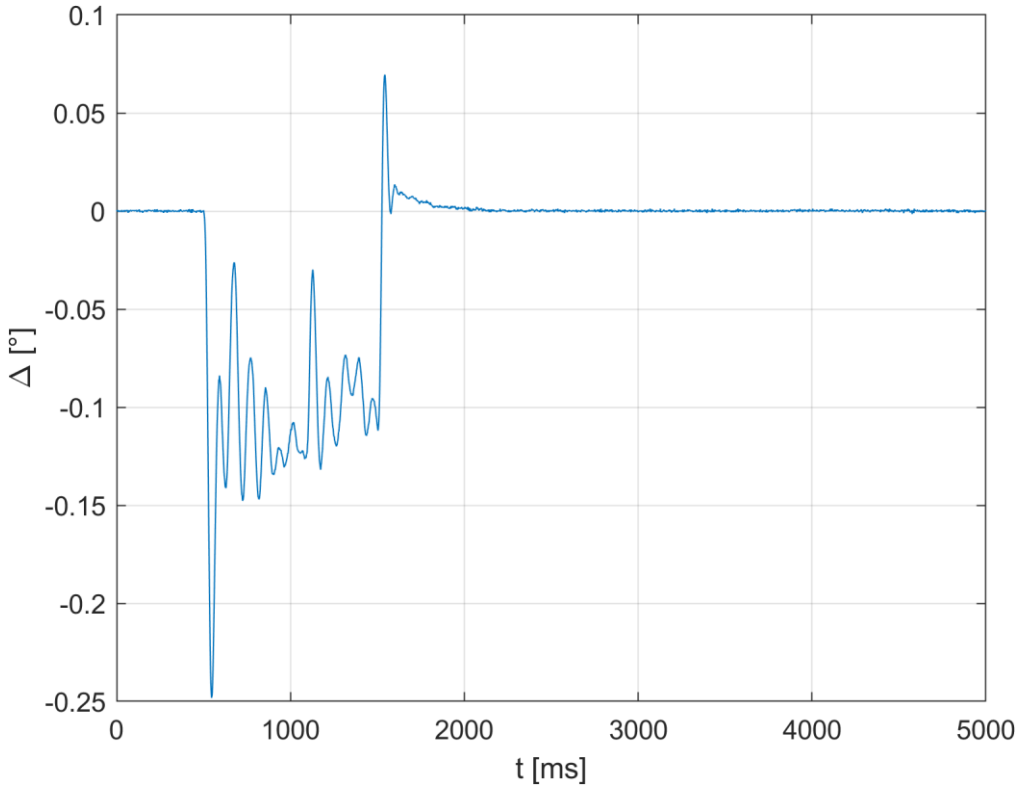

Fig. 36: Position lag of ideally-tuned controller with feedforward

5.3 PLC project

The purpose of the PLC project is to provide an interface for controlling the axis objects created in the previous chapter and the servomotor terminals for controlling the brakes. Ultimately, it should enable manual operation of the positioning mechanism through HMI. Communication between the PLC and the HMI is enabled via the ADS communication protocol. The following functions should be implemented in the PLC:

- Enabling axes
- Releasing brakes
- Point-to-point motion

The functionality can be achieved using function blocks from the Motion Library (Table 7). However, it is necessary first to introduce the structure of the TwinCAT 3 PLC project.

The source code for the PLC project can be theoretically written in a single MAIN file. However, such an approach is not practical for large projects. Instead, the source code should be split into the program organisation units (POUs). The following types of POUs are available:

- Function (prefix FC)
- Function block (prefix FB)
- Program (prefix PRG)

In addition, POUs can be extended with the following objects:

- Action
- Transition
- Method
- Property

The variables can be declared either directly in the declaration part of the POU or global variable lists (GVLs).

The source code for the positioning mechanism's PLC project was split into the POUs as listed in Table 5. The GVLs are listed in Table 6.

Table 5: POUs

POU	Description
FB_Axis	Includes instances of functions blocks for axis control (Table 7)
FB_Unit	Includes instances of FB_Axis
MAIN	The main entry point; includes instances of FB_Unit

Table 6: GVLs

GVL	Description
GCL	Contains constant variables
GVL_HMI	Contains variables that provide an interface for HMI
GVL_IO	Contains I/O variables and axis objects

Table 7: Function blocks from the Motion Library

Function block	Description
MC_Power	Activates software enable for an axis
MC_Reset	Resets the NC axis
MC_MoveRelative	Starts a relative positioning procedure
FB_EcCoESdoWrite	Permits an object from the object directory of an EtherCAT slave to be written

5.4 HMI

The HMI is a part of the application for rapid identification of machine tool volumetric error using Renishaw XM-60 – VOLECOM. The application is written in C# language using .NET Framework 4.7.2. It implements Model-View-ViewModel (MVVM) that enables the separation of user interface (View) from the logic behind (ViewModel). The project is arranged according to the pattern, as shown in Fig. 37.

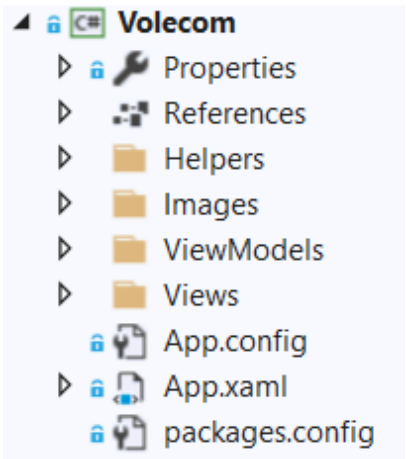

Fig. 37: MVVM project structure

The user interface (UI) of HMI was developed as a view for the VOLECOM application. The communication with the PLC was implemented in the corresponding ViewModel. The UI of the application is shown in Fig. 38. The ribbon contains buttons for enabling axes and for releasing breaks. The mechanism can be positioned using plus/minus buttons next to the actual position display. The positioning can be done either as continuous jogging or as incremental and is set up via buttons in the Jog Mode region.

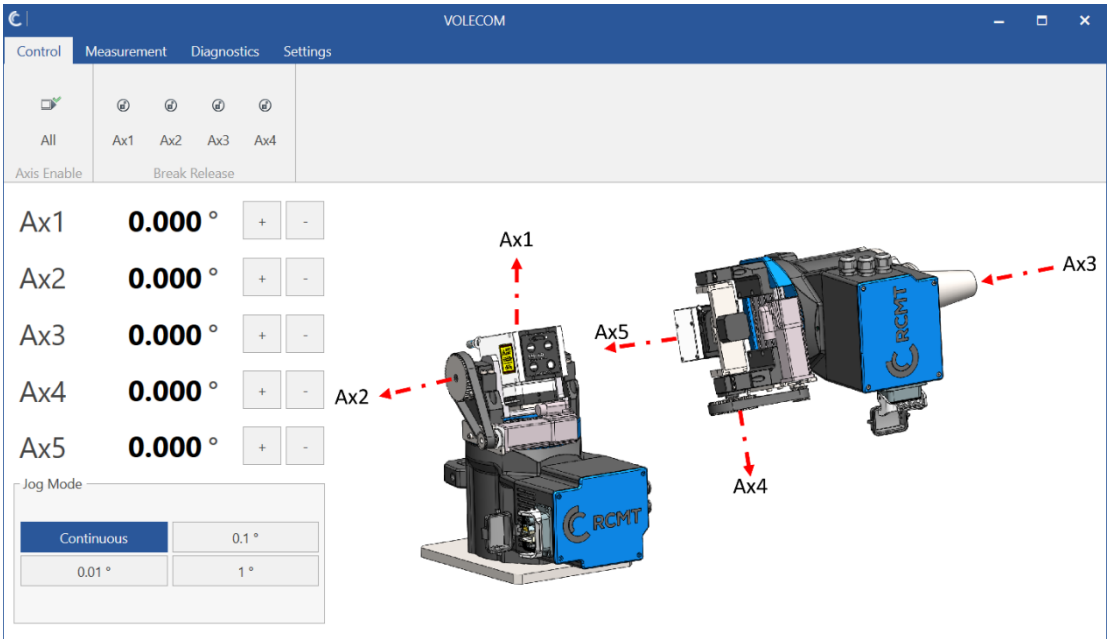

Fig. 38: HMI

6 Development of quick alignment algorithm

The quick alignment algorithm aims to replace a tedious manual alignment procedure with an automatic one. The algorithm will then be used as a part of the identification process – determination of combinations of the positioning mechanism configuration and machine axis path for measurement such that the positioning mechanism remains stationary and within range.

6.1 Manual alignment

The recommended manual alignment sequence is as follows [10]:

- Visual axis alignment
- Fine axis alignment
- Receiver alignment

Visual axis alignment includes the following steps [10]:

- Set the pitch and yaw adjusters in the middle of their travel
- Move the launch and the receiver as close to each other as possible
- Visually check the launch and the receiver are parallel to each other
- Open the shutter on the launch, and the receiver
- Translate the machine so that one of the beams is on the receiver target
- Continue the process below (Fig. 39) until the beam stays on the target during the full machine movement. Use the machine to perform translations and pitch/yaw adjusters for rotational alignment

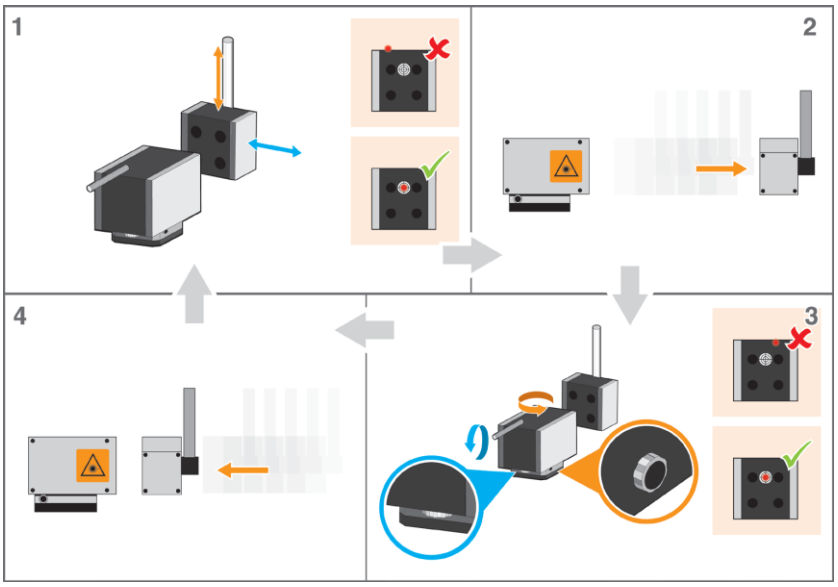

Fig. 39: Visual axis alignment [10]

Fine axis alignment includes the following steps [10]:

- Start CARTO program and run Capture
- Select New or Open
- Select Align
- Move the machine to align the laser beams into the receiver apertures. Adjust position until the straightness beam appears in CARTO
- Continue the process in Fig. 40 until the beam stays on the target in CARTO software during the machine movement
- Adjust the roll lever to the centre of the roll display, as shown in Fig. 41

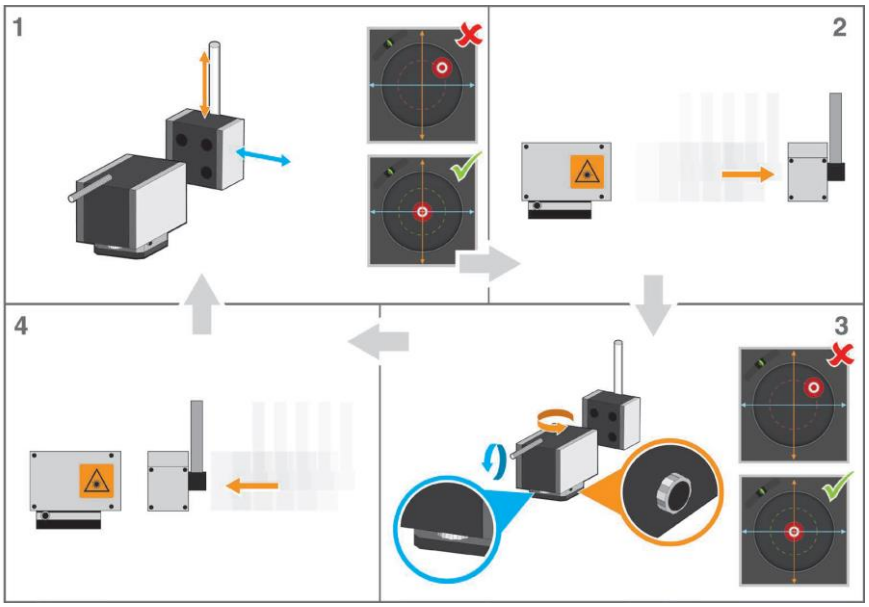

Fig. 40: Fine axis alignment [10]

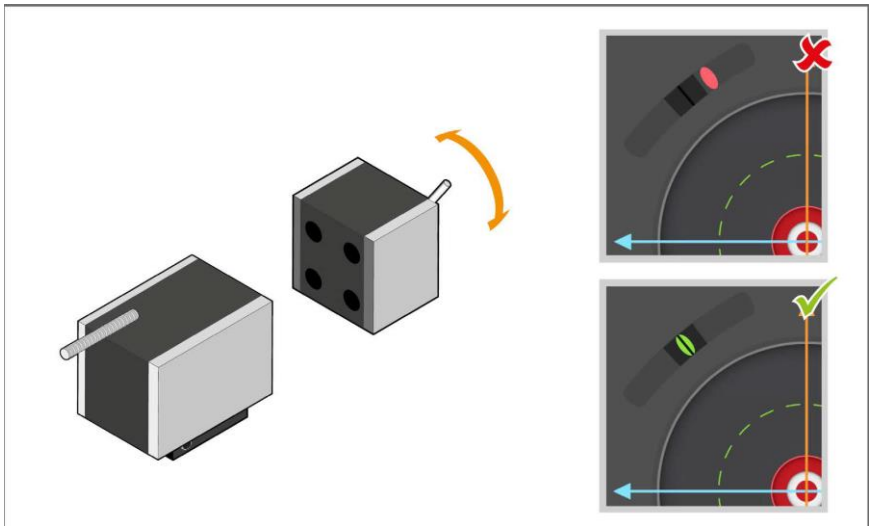

Fig. 41: Roll adjustment [10]

6.2 Quick alignment

The whole process described above is intended to be replaced by the following:

1. Align the laser beams manually such that all laser beams are in the range
2. Perform automatic fine alignment
3. Position machine to the next point
4. Repeat steps 1-3 until the required number of points for identification have been acquired

As mentioned at the beginning of the chapter, this work aims to develop a quick alignment algorithm for automatic fine alignment in step 2. Renishaw XM-60 allows measuring of 6 degrees of freedom (viz chapter 3.3). Positioning error is relative and is irrelevant for alignment. Thus, there are 5 degrees of freedom: two translations and three rotations. The positioning mechanism was designed in such a way that it also has 5 degrees of freedom. It was decided to split degrees of freedom in the following way:

- Launch unit – 2 translational degrees of freedom
- Receiver unit – 3 rotational degrees of freedom

It follows that each axis is responsible for an individual error term. Thus, alignment can be realised by positioning the individual axis such that the respective error term is zero. The challenge, however, is how to determine the position. All axes, except for the one with stepper motor, have feedback from an encoder and the angular position of the trunnion table on which the launch or receiver is mounted is obtained using belt drive ratio. Hence, it is necessary to calculate the angles based on the errors. Such a calculation is possible only when the distance between the launch and the receiver is known. However, since positioning error is relative, it is not possible to obtain the exact distance. The solution is to use feedback directly from the interferometer. Essentially, this allows controlling positioning in the interferometer's coordinate system shown in Fig. 43. However, there should also be an ability to switch back to the feedback from the encoder for manual operation. Thus, the controller must have either two feedback systems, as shown in Fig. 42 (XM-60 designates interferometer) or an additional position controller that uses feedback from the interferometer. The advantage of the latter approach is that position controllers can be tuned precisely for a specific type of feedback. Thus, the usage of the second position controller was chosen in this work.

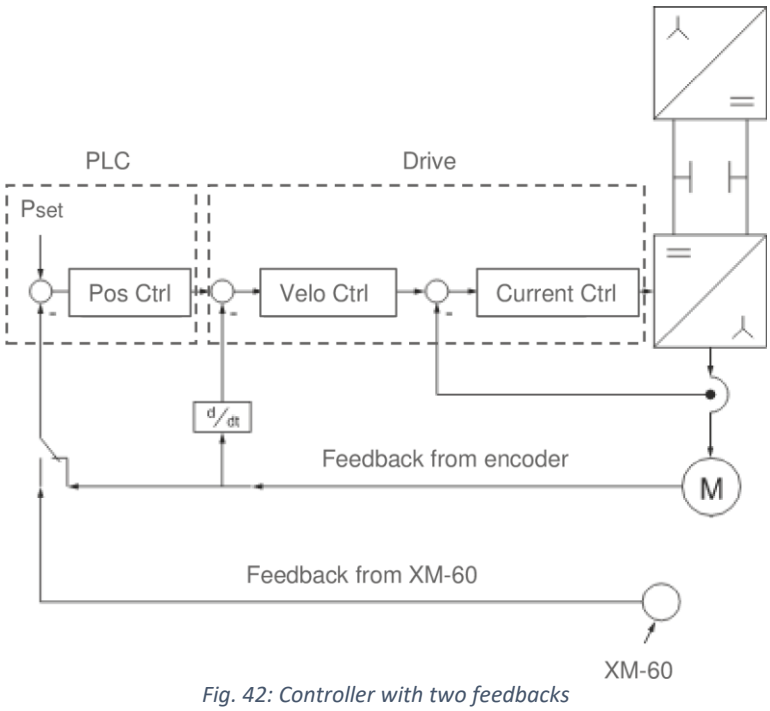

Fig. 42: Controller with two feedbacks

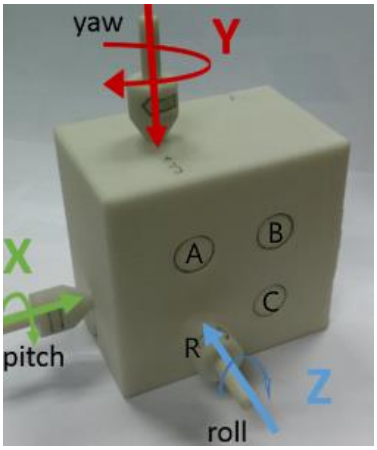

Fig. 43: XM-60 coordinate system [10]

6.3 Identification

The purpose of identification is to prepare a complete measurement program – positions of the mechanism and the machine tool that allows all the measurement of geometric errors to be performed with the positioning mechanism being stationary without going out of range. It can be described in the following steps:

1. Choosing an axis for identification (X, Y, Z)
2. Choosing measurement range – 2 points
3. Positioning the receiver manually in the first point from the step above
4. Using the quick alignment algorithm to determine the configuration of the mechanism
5. Repeating steps 3, 4 for the second point from step 2
6. Calculating measurement program based on the positions from the steps 3-5

The positions can be calculated by a solving system of equations constructed using the basic principles of trigonometry. For example, it is necessary to perform identification for Y-axis. Steps 3-5 can be visualised in the XY plane as shown in Fig. 44, where **P_1** and P2 are two points for positioning the receiver (min and max of the measurement range) and **L** is the launch of the XM-60. From Fig. 44, it follows that the launch needs to rotate by angle **α** to stay in range. However, by moving the line **P_1 P_2** by distance **d_x** along the X-axis, a new line **P_{1m} P_{2m}** can be obtained. The points **P_{1m}** and **P_{2m}** can be measured with the positioning mechanism being stationary. The same principle applies in the XZ plane – moving **P_1 P_2** by the distance **d_z** along the Z-axis. With the help of the cosine rule, a system of two equations (1) can be constructed. Thus, the coordinates of the launch can be obtained by solving the system. The mechanism's configuration is obtained by positioning the machine tool in the new starting points and performing the quick alignment.

$$\begin{cases} (P_1P_2)_{xy}^2 = (LP_1)_{xy}^2 + (LP_2)_{xy}^2 - 2(LP_1)_{xy}(LP_2)_{xy}\cos\alpha \\ (P_1P_2)_{xz}^2 = (LP_1)_{xz}^2 + (LP_2)_{xz}^2 - 2(LP_1)_{xz}(LP_2)_{xz}\cos\beta \end{cases} \quad (1)$$

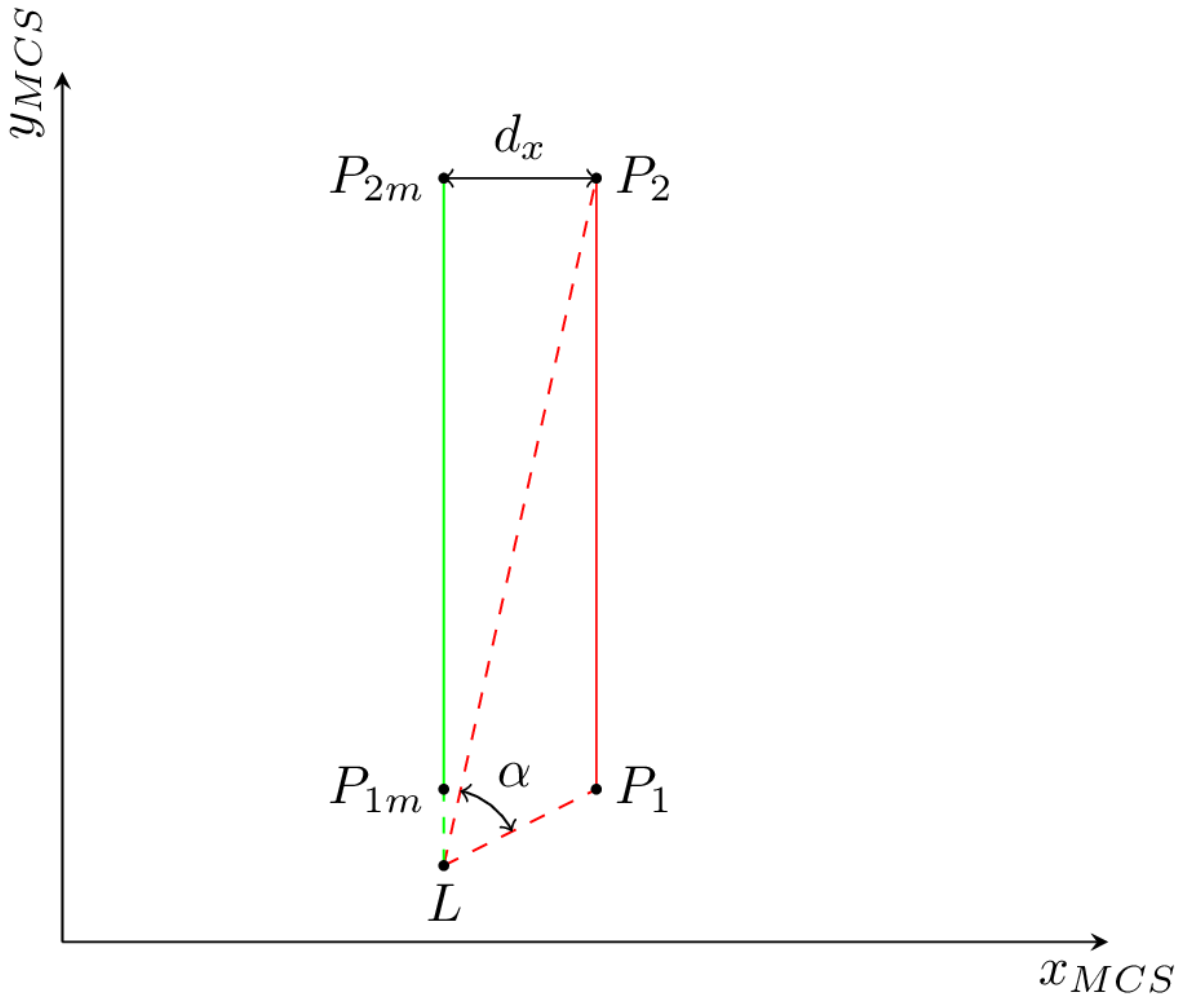

Fig. 44: Identification principle

6.4 Implementation

As described in chapter 5, axes are controlled by Beckhoff drives. They can be operated in the following modes:

- Cyclic synchronous position mode (CSP)
- Cyclic synchronous velocity mode (CSV)
- Cyclic synchronous torque mode (CST)
- Cyclic synchronous torque mode with commutation angle (CSTCA)

The two most common modes are CSP and CSV. In CSP mode, the position controller is in the drive, and the NC task generates commanded position. In CSV mode, the position controller is in the NC task, and the drive receives commanded velocity via the DRV Target velocity variable. The overview of the complete controller is shown in Fig. 44. Drives used in this project do not have built-in functionality for additional feedback. Thus, the drive's position controller cannot be used. Instead, it is necessary to use CSV mode – only velocity and current controller are used from the drive, and position controller is in PLC, as illustrated in Fig. 42. Communication between the PLC and the drive is realised through the EtherCAT vis ADS interface using the following process data objects:

- DRV Controlword – enables drive output stage. The values must be entered in a specified order, as illustrated in Fig. 45
- DRV Statusword – outputs status messages are output
- FB Position – provides feedback from the encoder
- DRV Target velocity – sets command velocity

All these variables are linked to the axis object's PDOs and are controlled by them. The linking of variables is done automatically during the hardware configuration phase of commissioning (viz chapter 5.1). However, as mentioned previously, it is necessary to set command velocity also using an output from the additional encoder. Thus, the implementation can be split into the following steps:

1. Switching between the controllers
2. Design of the controller
3. Changes to the PLC project

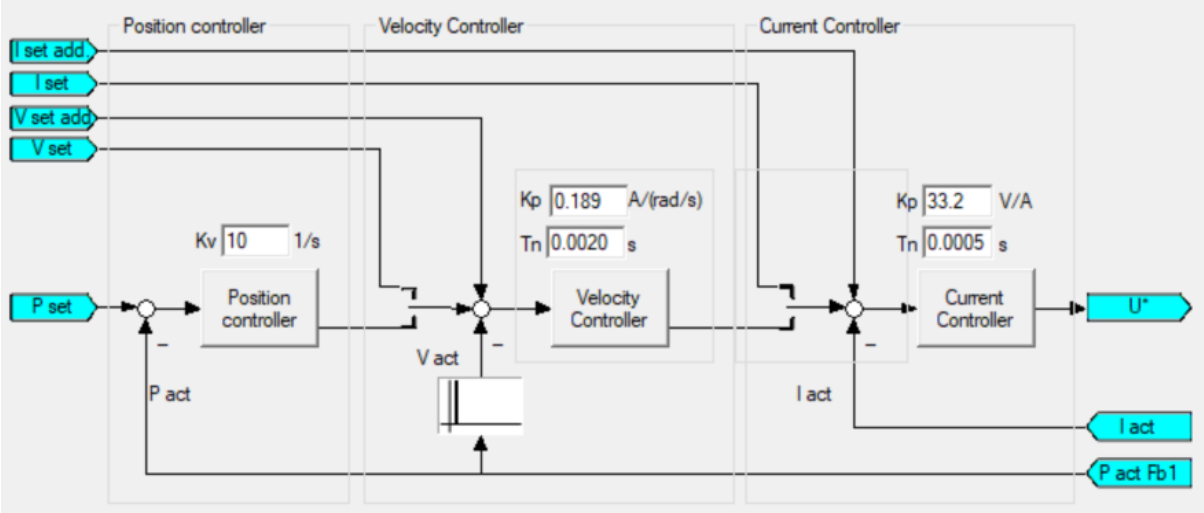

Fig. 45: Controller overview

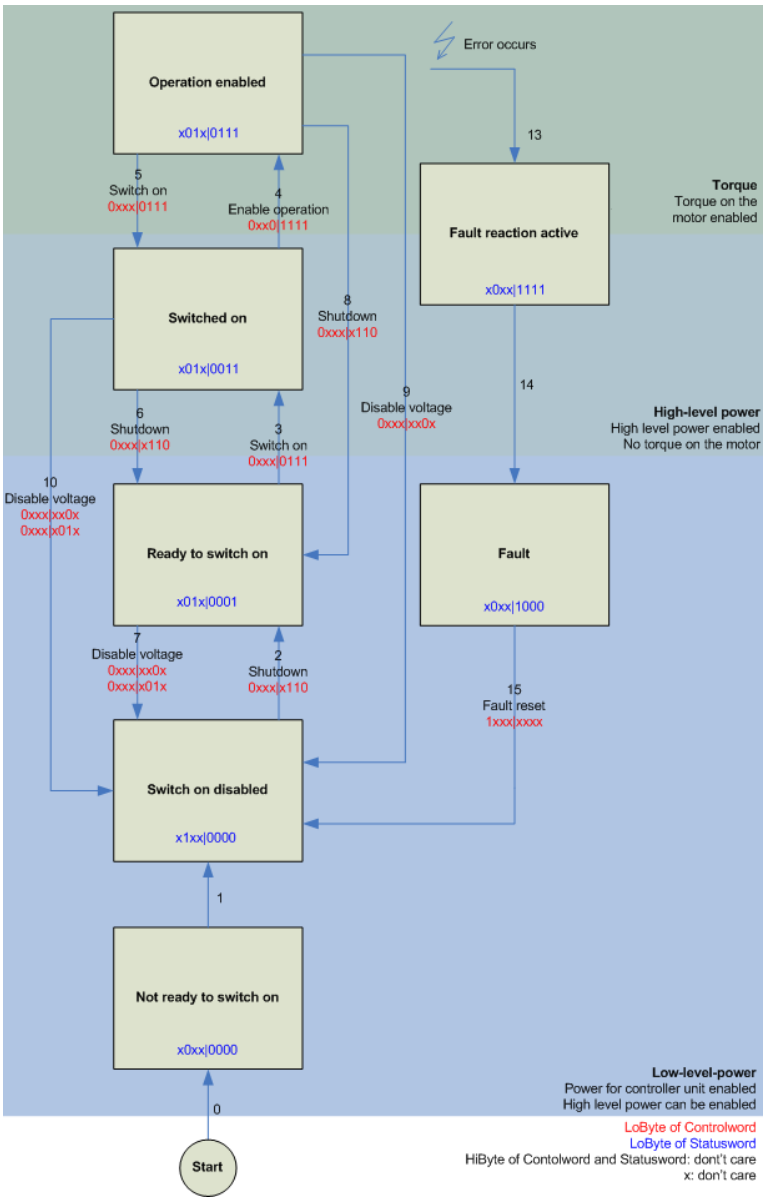

Fig. 46: DS402 State Machine [11]

6.4.1 Switching between the controllers

A controller sets the drive's command velocity based on the encoder's feedback or output from the interferometer. Each drive's input can be linked only to a single variable. As mentioned in the previous chapter, the inputs are automatically linked to the NC task. However, it was experimentally determined that the *DRV Target velocity* variable could be linked to an arbitrary variable of the appropriate type without impacting neither drive's nor the axis object's functionality. Thus, switching can be realised by linking the *DRV Target velocity* variable to a variable from PLC. Then, this variable can be assigned a value from either of the controllers' outputs. However, it is important to note that the variable assignment needs to be done in a faster cycle time than one of PLC's main tasks. The reason being is that NC has two tasks, as illustrated in Fig. 46:

- SVB – responsible for planning the requested traversing path
- SAF – responsible for the execution of motion

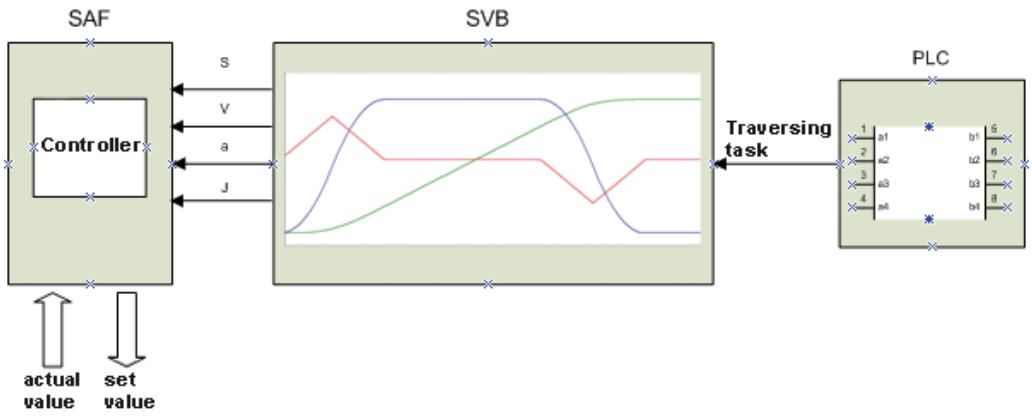

Fig. 47: Structure of NC PTP [11]

It follows that the command velocity is set with the SAF task's cycle time of 2 ms. The PLC's cycle time is 10 ms. Even though the cycle time can be changed, there is a better approach – a program POU can be called using the SAF task. This can be done using the following procedure:

1. Right-click on the plc project's sub-node
2. Click Add → Referenced Task
3. Choose SAF task from Available Tasks
4. Click Open
5. To assign a program to the task, drag and drop it to the newly created sub-node

Returning to the switching between the controllers, the whole principle behind the switching is nothing more than changing which variable is assigned to the drive's command velocity input. Of course, there must be implemented safety measures, but this is the topic of chapter 6.3.3.

6.4.2 Design of the controller

The second position controller can be implemented by adding an additional controller and additional encoder to the existing axis object or adding a new axis object. The latter approach gives better flexibility in terms of settings and is used in this work. Thus, a single real axis has two axis objects for controlling (Fig. 48). The specific axis object's PDO's must be linked to the drive's one for synchronisation. The linking can be performed by manually linking all the variables in the project tree or by editing the mappings file as explained in the next paragraph.

Automatically created variable links can be exported as an XML file from the Mappings section of the project tree (Fig. 47). The contents of the XML file for mappings of one axis are shown in Fig. 49. To copy the links to the second axis object (Axis 1e), contents of the XML file between tags <OwnerA …> and </OwnerA> can be simply copied with the following changes:

- Change the name to "Axis 1e"
- Delete links to the drive inputs (because inputs cannot have more than one link)
- Specify the variable for the feedback under <Link VarA=" Enc^Inputs^In^nDataIn1" VarB= "FeedbackVar"/>, where FeedbackVar should be replaced with the appropriate variable

The XML can then be imported back into the project using the same menu as shown in Fig. 47. As a result, there are two axis objects linked to one real axis. The switching between them is realised via changing which command velocity output is linked to the driver's input. The logic for controlling the axis objects is explained in the next chapter.

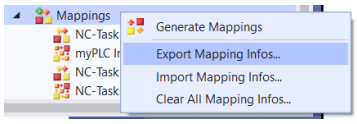

Fig. 48: Mappings

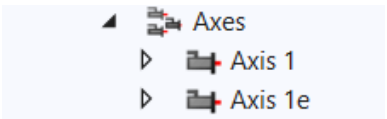

Fig. 49: Axis objects for one real axis

```
<?xml version="1.0"?>
<VarLinks>
    <OwnerA Name="TINC^NC-Task 1 SAF^Axes^Axis 1">
        <OwnerB Name="TIID^Device 3 (EtherCAT)^Term 1 (EK1100)^Term 2 (EL7221-9014)">
            <Link VarA="Drive^Inputs^In^nDataIn3[0]" VarB="DRV Torque actual value^Torque actual value"/>
            <Link VarA="Drive^Inputs^In^nDcOutputTime" VarB="InfoData^DcOutputShift"/>
            <Link VarA="Drive^Inputs^In^nState1" VarB="DRV Statusword^Statusword"/>
            <Link VarA="Drive^Inputs^In^nState2" VarB="DRV Statusword^Statusword" Size="8" OffsB="8"/>
            <Link VarA="Drive^Inputs^In^nState4" VarB="WcState^InputToggle" Size="1" OffsA="1"/>
            <Link VarA="Drive^Inputs^In^nState4" VarB="WcState^WcState"/>
            <Link VarA="Drive^Outputs^Out^nCtrl1" VarB="DRV Controlword^Controlword"/>
            <Link VarA="Drive^Outputs^Out^nCtrl2" VarB="DRV Controlword^Controlword" Size="8" OffsB="8"/>
            <Link VarA="Drive^Outputs^Out^nDataOut1" VarB="DRV Target position^Target position"/>
            <Link VarA="Enc^Inputs^In^nDataIn1" VarB="FB Position^Position"/>
            <Link VarA="Enc^Inputs^In^nDcInputTime" VarB="InfoData^DcInputShift"/>
            <Link VarA="Enc^Inputs^In^nState4" VarB="WcState^InputToggle" Size="1" OffsA="1"/>
            <Link VarA="Enc^Inputs^In^nState4" VarB="WcState^WcState"/>
        </OwnerB>
    </OwnerA>
</VarLinks>
```

Fig. 50: Mappings of one axis

```
<?xml version="1.0"?>
<VarLinks>
    <OwnerA Name="TINC^NC-Task 1 SAF^Axes^Axis 1">
        <OwnerB Name="TIID^Device 3 (EtherCAT)^Term 1 (EK1100)^Term 2 (EL7221-9014)">
            <Link VarA="Drive^Inputs^In^nDataIn3[0]" VarB="DRV Torque actual value^Torque actual value"/>
            <Link VarA="Drive^Inputs^In^nDcOutputTime" VarB="InfoData^DcOutputShift"/>
            <Link VarA="Drive^Inputs^In^nState1" VarB="DRV Statusword^Statusword"/>
            <Link VarA="Drive^Inputs^In^nState2" VarB="DRV Statusword^Statusword" Size="8" OffsB="8"/>
            <Link VarA="Drive^Inputs^In^nState4" VarB="WcState^InputToggle" Size="1" OffsA="1"/>
            <Link VarA="Drive^Inputs^In^nState4" VarB="WcState^WcState"/>
            <Link VarA="Drive^Outputs^Out^nCtrl1" VarB="DRV Controlword^Controlword"/>
            <Link VarA="Drive^Outputs^Out^nCtrl2" VarB="DRV Controlword^Controlword" Size="8" OffsB="8"/>
            <Link VarA="Drive^Outputs^Out^nDataOut1" VarB="DRV Target position^Target position"/>
            <Link VarA="Enc^Inputs^In^nDataIn1" VarB="FB Position^Position"/>
            <Link VarA="Enc^Inputs^In^nDcInputTime" VarB="InfoData^DcInputShift"/>
            <Link VarA="Enc^Inputs^In^nState4" VarB="WcState^InputToggle" Size="1" OffsA="1"/>
            <Link VarA="Enc^Inputs^In^nState4" VarB="WcState^WcState"/>
        </OwnerB>
    </OwnerA>
    <OwnerA Name="TINC^NC-Task 1 SAF^Axes^Axis 1e">
        <OwnerB Name="TIID^Device 3 (EtherCAT)^Term 1 (EK1100)^Term 2 (EL7221-9014)">
            <Link VarA="Drive^Inputs^In^nDataIn3[0]" VarB="DRV Torque actual value^Torque actual value"/>
            <Link VarA="Drive^Inputs^In^nDcOutputTime" VarB="InfoData^DcOutputShift"/>
            <Link VarA="Drive^Inputs^In^nState1" VarB="DRV Statusword^Statusword"/>
            <Link VarA="Drive^Inputs^In^nState2" VarB="DRV Statusword^Statusword" Size="8" OffsB="8"/>
            <Link VarA="Drive^Inputs^In^nState4" VarB="WcState^InputToggle" Size="1" OffsA="1"/>
            <Link VarA="Drive^Inputs^In^nState4" VarB="WcState^WcState"/>
            <Link VarA="Enc^Inputs^In^nDcInputTime" VarB="InfoData^DcInputShift"/>
            <Link VarA="Enc^Inputs^In^nState4" VarB="WcState^InputToggle" Size="1" OffsA="1"/>
            <Link VarA="Enc^Inputs^In^nState4" VarB="WcState^WcState"/>
        </OwnerB>
    </OwnerA>
</VarLinks>
```

Fig. 51: Edited mappings

6.4.3 Changes to the PLC project

The PLC project introduced in chapter 5.3 needs to be adopted for the implementation of additional controllers. The structure of the modified PLC project is shown in Fig. 51. The source code for the full PLC project is provided as an electronic attachment.

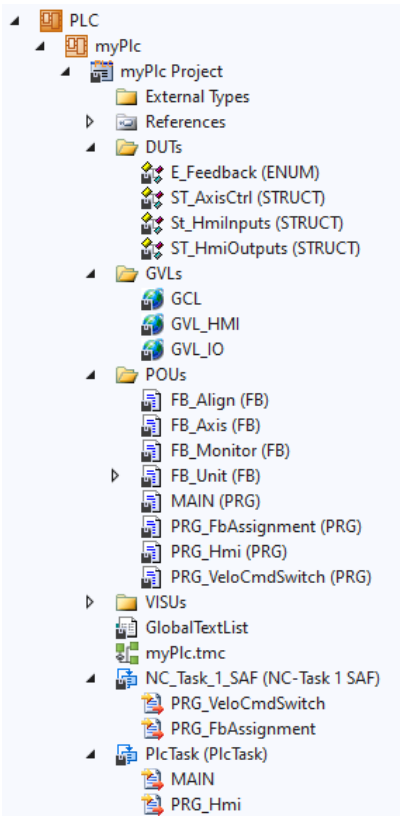

Fig. 52: Structure of modified PLC project

Firstly, it is necessary to add variables for feedback switching and the axis object's interface to GVL_IO. Next, it is necessary to create a program POU where the assignment of velocity command value is performed – PRG_VeloCmdSwitch. The program must be assigned to the SAF task of the NC (viz chapter 6.3.1).

Besides the program for velocity command assignment, two more POUs were added:

- FB_Align
- FB_Monitor

FB_Align serves for the absolute positioning of an axis to the centre of the measuring range (position feedback value of zero). Essentially, it encapsulates a piece of code that is repeated many times throughout the project. FB_Monitor serves for monitoring the actual movement of the axis using the encoder when it uses feedback from the interferometer. In case of failure of communication, the value of the feedback can become "frozen". Therefore, the controller will be positioning the axis incorrectly. The expected range of motion is not more than 0.1° when the axis is in the keep aligned mode (uses the second controller with the feedback from the interferometer). Thus, if the range is exceeded, it can be concluded that there is a problem, and the axis must be put into the error state.

Next, it is necessary to implement the logic for switching between the controllers. The logic needs to be added to the FB_Unit because the instances of FB_Axis are controlled there. There are two operating modes of the positioning mechanism:

- Manual – allows manual positioning using HMI
- Keep aligned – brings all the axis to the centre of the measuring range and maintains a setpoint position of zero

In manual mode, the main position controller with the feedback from the encoder is used, while in keeping aligned, the position controller is switched to the second one with the feedback from the interferometer. As explained in chapter 6.3.2, position controllers come from the different axis objects. The main axis object controls the drive, and the additional axis object only generates command velocity. Thus, the software enable is set via the main axis object. Even when the velocity command is set via the second axis object, the drive's master control word is controlled by the main drive object. The complete logic for switching is best demonstrated using the state machine in Fig. 52.

```
CASE nState OF
  0:  // Wait for axis enable
    IF bEnableReq THEN
      bError := FALSE;
      bEnableReq := FALSE;
      nState := nState + 10;
    END_IF
  10: // Clear old velo cmd and switch feedback to encoder
    ResetVeloCmd1();
    bEnableState := TRUE;
    eFeedback := E_Feedback.eEncoder;
    nState := nState + 10;
  20: // Manual mode
    IF bDisableReq THEN
      bDisableReq := FALSE;
      nState := 100;
    ELSIF bAlignStart THEN
      bAlignStart := FALSE;
      nState := nState + 10;
    END_IF;
    aAxisCtrl := aAxisCtrlHmi; // Read cmds from HMI
  30: // Switch to feedback from  XM-60
    ResetVeloCmd2();
    // Enable axis extension
    bEnableE := TRUE;
    // Start monitoring actual motion using encoder
    // necessary in case of a problem with communication
    bMonitor := TRUE;
    eFeedback := E_Feedback.eXm60;
    nState := nState + 10;
  40: // Keep aligned mode
    bAlignCmd := TRUE;
    IF bDisableReq THEN
      bDisableReq := FALSE;
      nState := 100;
    ELSIF bAlignStop THEN
      bAlignStop := FALSE;
      nState := nState + 10;
    END_IF
  50: // Disable keep aligned mode
    bAlignCmd := FALSE;
    bEnableE := FALSE;
    nState := nState + 10;
  60: // Switch main controller off to reset setpoint pos
    bEnableState := FALSE;
    IF DisableDone() THEN
      nState := 10;
    END_IF;
  70: // Reset main controller
    bReset := TRUE;
    IF ResetDone() THEN
      nState := nState + 10;
    END_IF
  80: // Return to manual mode
    bReset := FALSE;
    nState := 10;
  100: // Disable all axes
    bEnableState := FALSE;
    bEnableE := FALSE;
    nState := 0;
END_CASE
```

Fig. 53: Switch velocity command state machine

7 Development of communication software

The purpose of communication software is to provide PLC with measurements from the XM-60. The measurements are used for the quick alignment algorithm. The software is written as a class library in C# language using .NET Standard 2.0 specification. A class library compiles into *.dll file and defines types, methods that are called by an application. It can then be distributed as a single file.

The reasoning behind using a class library is that the software from this work will be integrated into the VOLECOM application (viz chapter 5.4). There will be various modules for communication with different devices, as shown in Fig. 53. The modules are communicating with the devices through specific interfaces:

- Communication with XM-60 – through XMApi
- Communication with PLC – through ADS
- Communication with CNC control system – through an interface for the specific control system

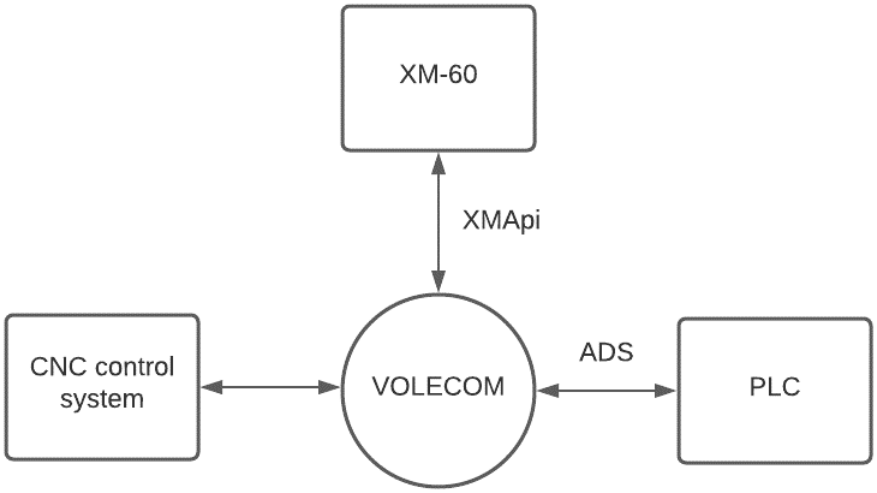

Fig. 54: Communication scheme

7.1 Communication with XM-60

For communication with XM-60, Renishaw provides an application programming interface (API) to implement communication into any .NET Framework application. The API is distributed using NuGet packages:

- Renishaw.XMApi – communication with XM-60
- Renishaw.LaserApi – communication with XC-80 environmental compensator

The first package contains an XMApi class with all the methods, events, and properties for communication with XM-60. An instance of the class must be added to the application. The communication is established using the Open method. It returns true if the connection was successful. The Datum task must be called before the test begins to compensate for the dead path error – the distance pitch and yaw measurements will be zeroed with the help of the task. Also, the roll sensor must be calibrated before the test using the CalibrateRollSensorAsync task. The SetAirCompensation method must be used to set the environmental compensation. Values of air temperature, air pressure and air humidity are entered as inputs to the method and can be obtained from the XC-80 station. Material compensation is set using the SetMaterialCompensation method with a temperature of the material (can also be obtained using the XC-80 station) as an input. The measurements can be captured using the TriggerAsync task. The measurements are available from the LatestReading property. The property is a member of the XMReading class, which contains the following properties:

- Distance – Z displacement in meters
- StraightnessX, StraightnessY – X, Y displacement in meters
- Roll, Pitch, Yaw – angular measurements in radians
- AbsolutePitch, AbsoluteYaw – an estimate of the absolute pitch and yaw between receiver and launch units in radians
- BatteryCharge – battery level of the receiver in per cent
- ReceiverTemperature and LaunchTemperature – temperatures of the launch and receiver units in degrees Celsius
- SignalLevelA, SignalLevelB, SignalLevelC, SignalLevelR – signal strength for each of the interferometric beam and roll beam
- LaunchStatus – enumerator that indicates the status of the launch unit
- ReceiverStatus – enumerator that indicates the status of the receiver unit
- IsSynchronized – indicates whether the launch and receiver readings are synchronised

The update of the LatestReading property is signalled via the ReadingUpdated event. It is raised about every 50 ms.

The methods and properties for the XC-80 station are in the LocalXCHost class. The connection is established using the Connect method. The information about the environment can be obtained using the GetEnvironmentalRecord method, and the information about the material temperature can be obtained using the GetMaterialTemperatureRecord method.

7.2 Communication with PLC

For communication with PLC, Beckhoff also provides an application programming interface (API) distributed using the Beckhoff.TwinCAT.Ads NuGet package. It includes everything to develop a custom application to communicate with TwinCAT devices. The full description of the API is available from [11]. For the scope of this work, the following methods from the AdsClient class are sufficient:

- Connect – connects to the ADS device
- Close – closes AdsClient
- CreateVariableHandle – determines the symbol handle
- DeleteVariableHandle – release variable handle
- WriteAny – writes an object to the ADS device

7.3 Implementation

As mentioned at the beginning of the chapter, the application is written as .NET Standard 2.0 class library. As IDE, Microsoft Visual Studio 2019 Community Edition was used. Complete source code is available from the electronic attachments.

The project (Xm60Feedback.cs) contains two classes:

- PlcVar – helper class for PLC variables (**Error! Reference source not found.**)
- Xm60Feedback – main class

PlcVar class serves as a container for a PLC variable – it contains all the necessary information (properties) for reading/writing variables through ADS:

- Path – the symbolic name of the variable (e.g. "GVL_IO.nVar1")
- Value – the value of the variable of generic type T
- Handle – variable handle for communication

Xm60Feedback contains functionality for providing PLC with readings from the XM-60. It was developed to be used as a module in the main application containing only single instances of XMApi and AdsClient. They are passed to the XmFeedback class in the constructor. The class contains a subscription to the ReadingUpdated event (signals when XM-60 readings are refreshed). The readings are sent to the PLC when the event is triggered. For efficiency reasons, the readings are transmitted as an array. The readings are scaled and converted to integers because it is the type that the axis object uses as feedback.

The example of usage of the library is shown in Fig. 54.

```
using (var plc = new AdsClient())
using (var xm60 = new XMApi())
{
    var fb = new Feedback(plc, xm60);

    // try connecting to PLC
    try
    {
        plc.Connect(AmsNetId.LocalHost, 851);
    }
    catch
    {
        // exit if connection failed
        Console.WriteLine("\nCannot connect to PLC!\nPress
    any key to quit...");
        Console.ReadKey(true);
        return;
    }

    // try connecting to XM-60
    if (!xm60.Open())
    {
        Console.WriteLine("\nCannot connect to XM-60!\nPress
    any key to quit...");
        Console.ReadKey(true);
        return;
    }

    // start communication with PLC
    fb.Start();

    // run until ESC is pressed
    Console.WriteLine("\nFeedback is running. Press ESC to
    stop...");
    do
    {
        while (!Console.KeyAvailable)
        { }
    } while (Console.ReadKey(true).Key != ConsoleKey.Escape);

    // stop communication and disconnect from PLC and XM-60
    fb.Stop();
    plc.Close();
    xm60.Close();
}
```

Fig. 55: Usage of Xm60.Feedback

8 Testing and verification

8.1 Preliminary testing of positioning resolution

The test aimed to determine the positioning resolution of the mechanism in the initial state. The straightness measuring range of the Renishaw XM-60 interferometer is a circle with a radius of 250 µm. The mechanism must position a laser beam within this circle when the launch and the receiver are at 4 m apart (maximum value of measuring range of positioning error).

The reason for the test was to check whether there are any problems with the construction of the mechanism. The servo motors used in the mechanism have encoders with 18-bit resolution and accuracy of ± 120 ” (approximately 0.003°). Thus, theoretically, the mechanism should achieve a resolution of 100 µm when the launch and the receiver are at 4 m apart.

8.1.1 Equipment

Name	Manufacture	Type	Description
AlignMeter	Duma Optronics Ltd.	100	Electro-Optics Simultaneous measurements of position and angular displacements. Technical specifications can be found in [12].
LaserOn	Duma Optronics Ltd.	PS	Compact visible laser diode (635nm), with built-in screws for fine beam direction adjustment. Technical specifications can be found in [15].

Table 8: Equipment for preliminary testing of positioning resolution

Fig. 56: AlignMeter 100 [12]

Fig. 57: LaserOn [13]

8.1.2 Procedure

The SpotOn laser source was mounted on the mechanism and placed at 4 m from the AlignMeter, as shown in Fig. 57. The mechanism was then commanded positioning in such a way that a laser would "draw" a square on AlignMeter's PSD, where the position was continuously recorded. The size of the square was gradually decreased until the point when it became indistinguishable from noise.

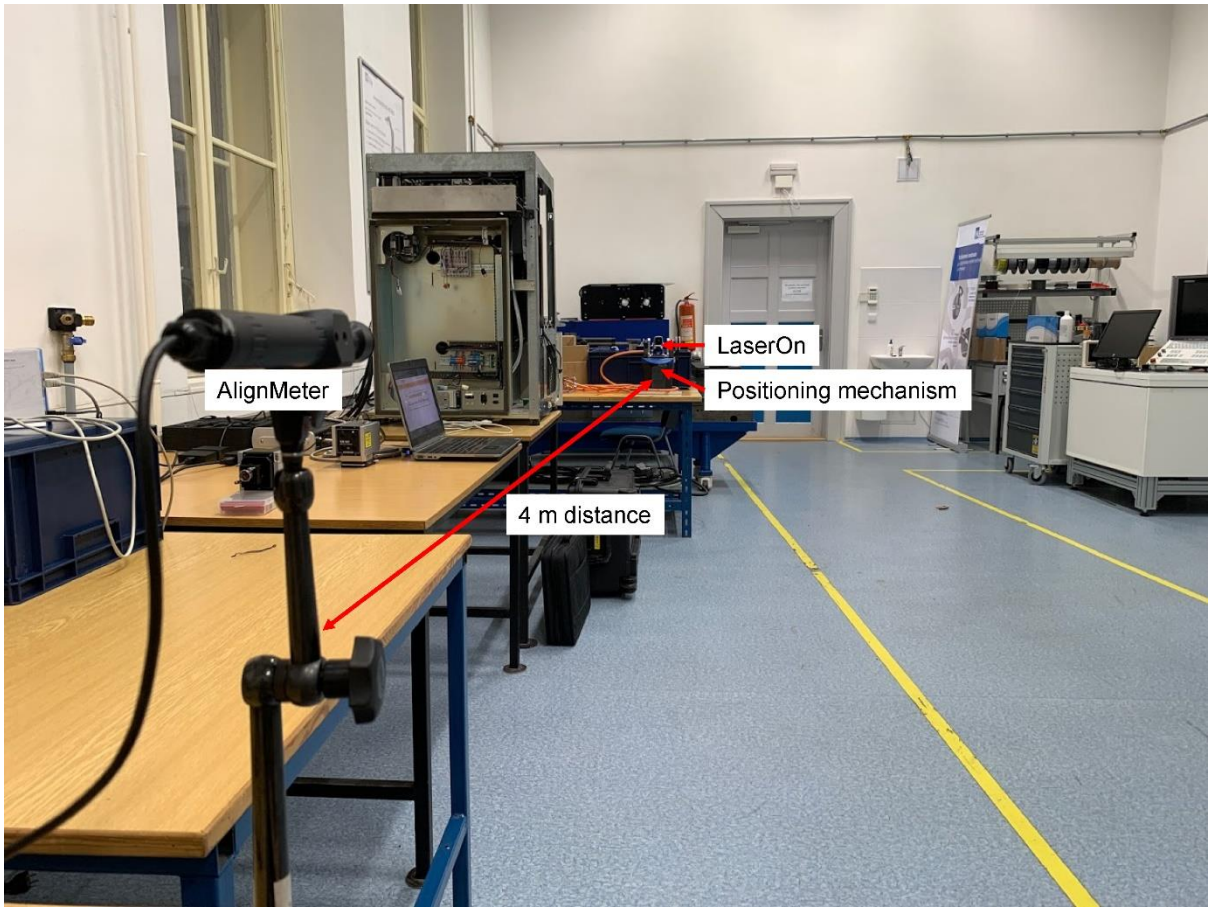

Fig. 58: Setup for preliminary testing of positioning resolution

8.1.3 Evaluation

The smallest achieved square is shown in Fig. 58. From the picture, it is clear that the profile is not exactly square. The reason being is that AlignMeter's software has a limited number of points that can be recorded. Considering that only the last points are present on the graph, it can be concluded that the smallest size of the side of the square is 500 µm. Thus, the achieved resolution is 500 µm and is five times lower than the theoretically predicted ones. The most probable causes for the discrepancy are belt pulleys and poorly tuned controllers. The problem was addressed by modifying pulleys drives (using finer toothing) and by retuning the controllers.

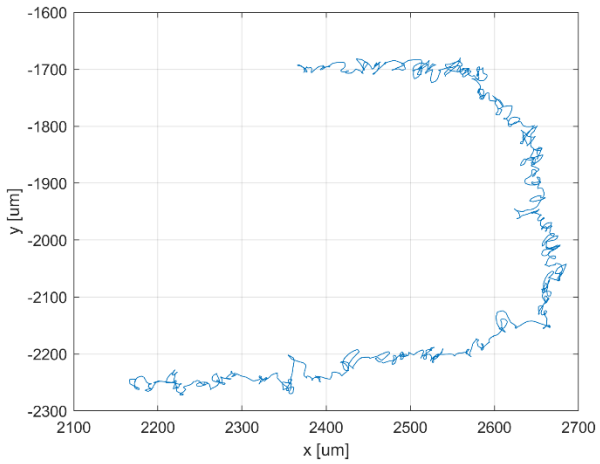

Fig. 59: Smallest square

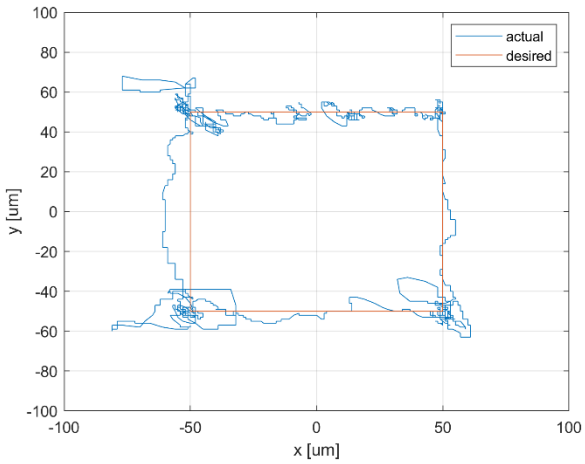

Fig. 60: Smallest square with direct measurement

8.2 Testing of direct measurement

The aim is to test the controller described in chapter 6.3.2. The equipment and the procedure are the same as in the previous test. The only difference lies in the position controller, which in this case uses direct measurement from the AlignMeter.

8.2.1 Equipment

Same as in 8.1.1.

8.2.2 Procedure

Same as in 8.1.1.

8.2.3 Evaluation

The result of the test is shown in Fig. 59. From the graph, it follows that the achieved resolution is 100 µm. Overshooting in the left part of the graph was due to the problems with communication – the values were not updated consistently in each cycle. Nevertheless, it can be concluded that the test successfully verified the functionality of the controller.

8.3 Testing of quick alignment algorithm

The test was aimed at checking the quick alignment algorithm.

8.3.1 Equipment

Same as in 8.1.1.

8.3.2 Procedure

The mechanism was first manually positioned so that the laser was in the range of AlignMeter's PSD and the linear readings were non-zero. Then, the mechanism was commanded to execute the quick alignment. The readings were continuously logged during the alignment procedure.

8.3.3 Evaluation

The results of the test can be seen in Fig. 60. The alignment starts at a time of 0.33 seconds, and at 0.56 seconds, the mechanism is aligned. Thus, the test proved the functionality of the quick alignment algorithm.

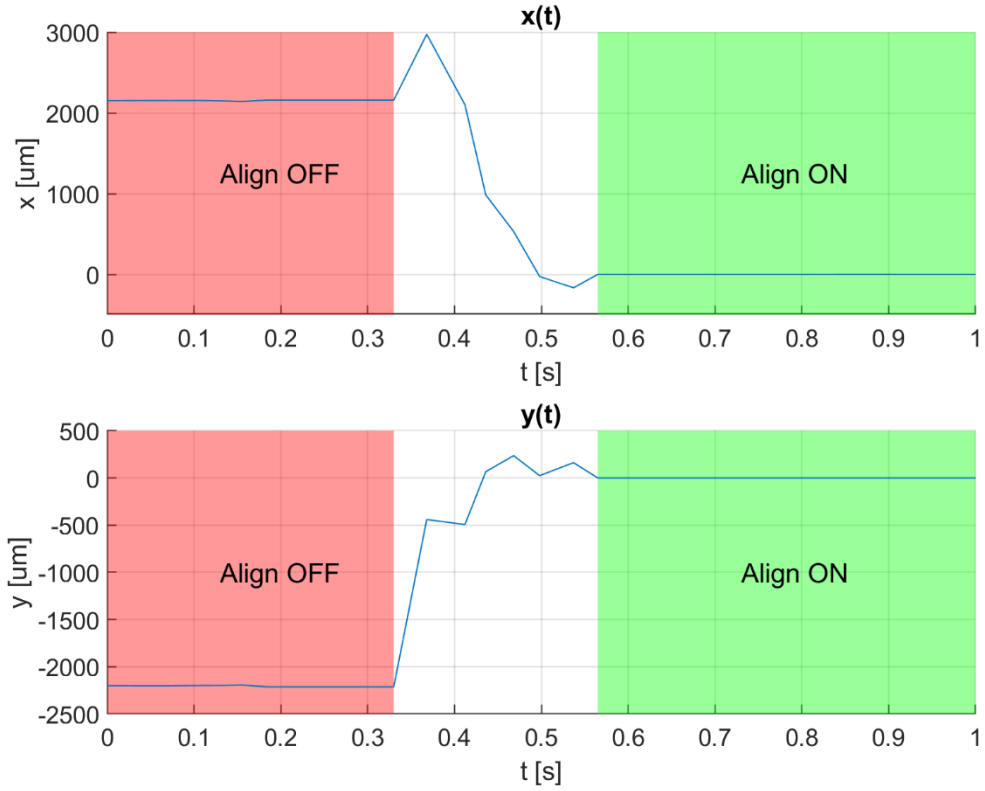

Fig. 61: Quick alignment algorithm test results

8.4 Verification

Verification is aimed at testing the functionality of all the software as well as the quick alignment algorithm.

8.4.1 Equipment

Name	Manufacture	Serial Number	Description
XM-60	Renishaw	22UJ75	Multi-axis calibrator

8.4.2 Procedure

The setup is shown in Fig. 61. It was arranged as close as possible to the one planned to be used during the actual measurements.

The testing procedure was as follows:

- Position the mechanism's units in manual mode such that all the readings are in the range
- Turn on alignment and log the readings continuously during the alignment process

The Renishaw CARTO Capture software was used for determining the readings during the manual positioning. An example of readings before the alignment is shown in Fig. 62, and the corresponding readings after alignment are shown in Fig. 63.

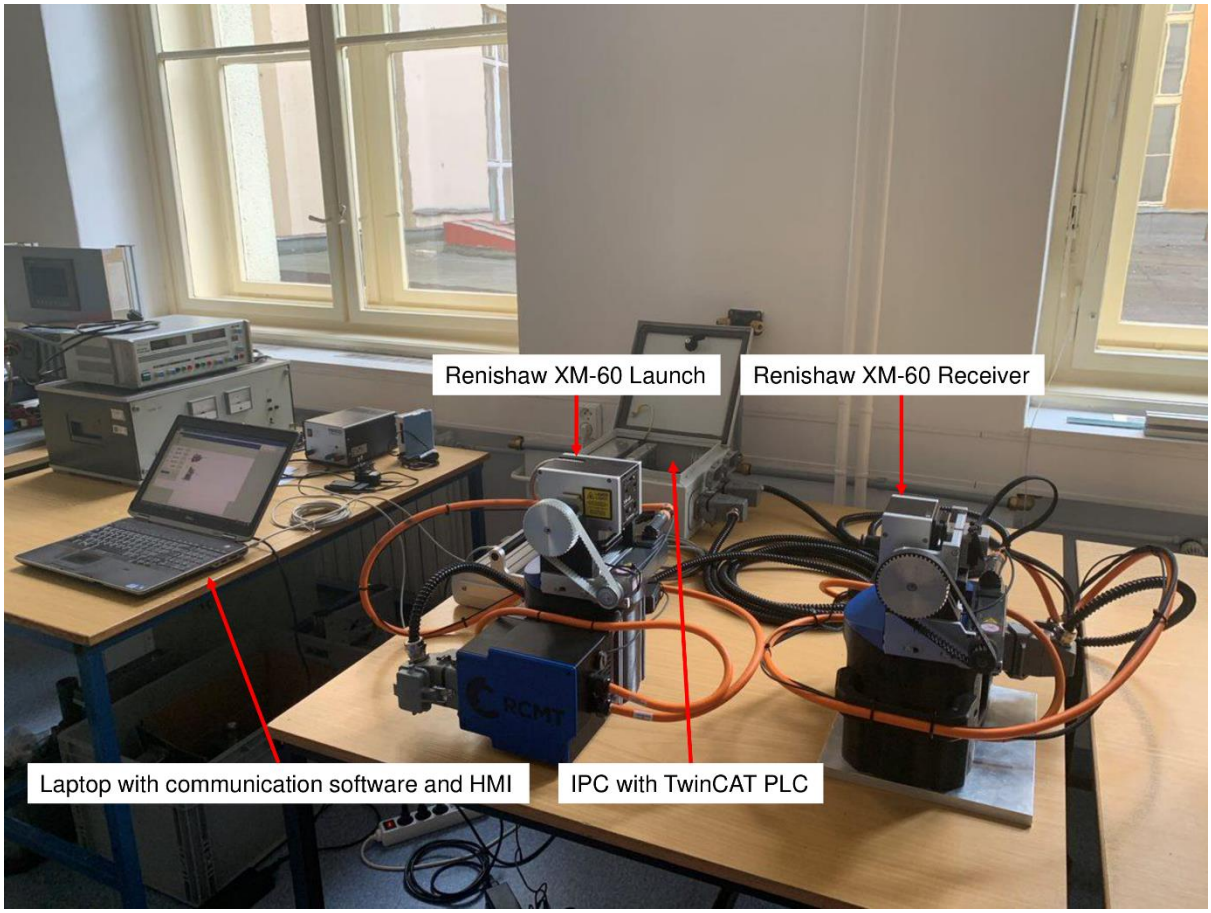

Fig. 62: Verification setup

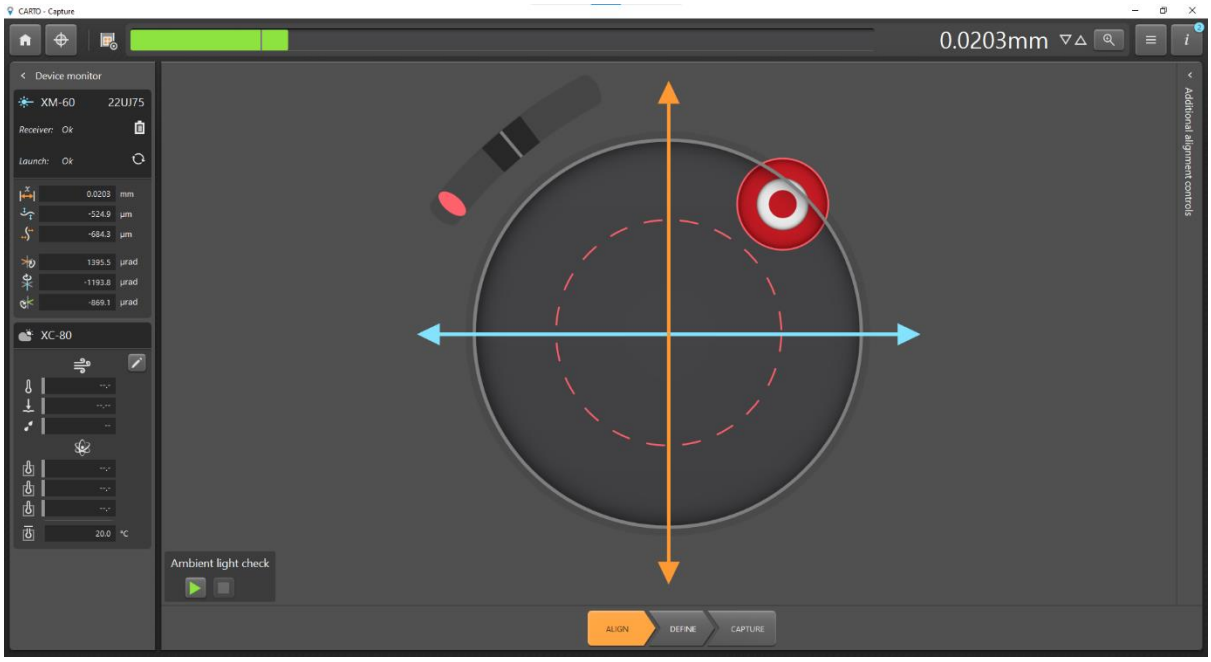

Fig. 63: Readings before alignment

Fig. 64: Readings after alignment

8.4.3 Evaluation

The results of the test can be seen in Fig. 64. From the graph, it follows that the mechanism was able to successfully position the axes such that all the readings were very close to zero. Thus, the XM-60 can be considered aligned.

The reason for omitting readings of the yaw readings from the graph in Fig. 64 is since there was a significant level of noise. Yaw readings can be seen separately in Fig. 65. The trend of the readings can be observed with the help of a moving average filter (Fig. 65). It clearly shows how the readings were approaching zero as the mechanism was being aligned. Therefore, the alignment algorithm was working even for a noisy signal.

It was not possible to determine the source of the noise for the yaw readings. Moreover, the noise was also present in other readings. However, its level was lower. The noise was observed consistently during repeated measurements. Hence, this can be an area for further investigation.

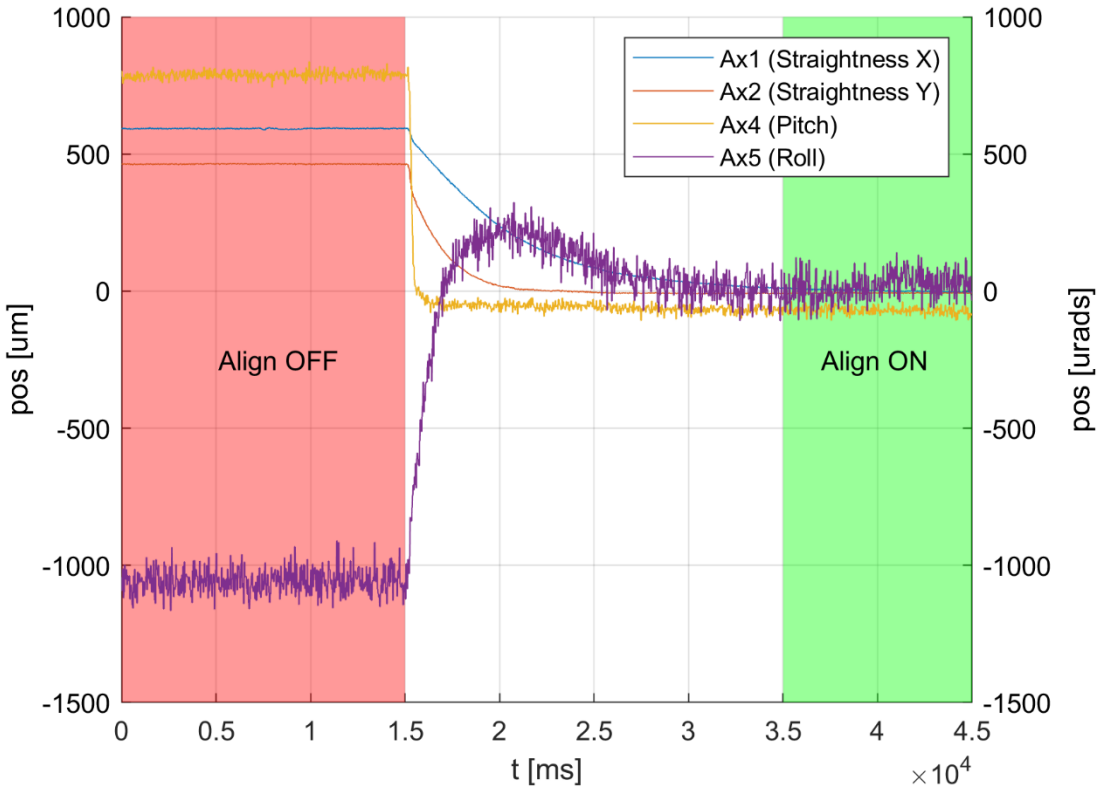

Fig. 65: Verification results

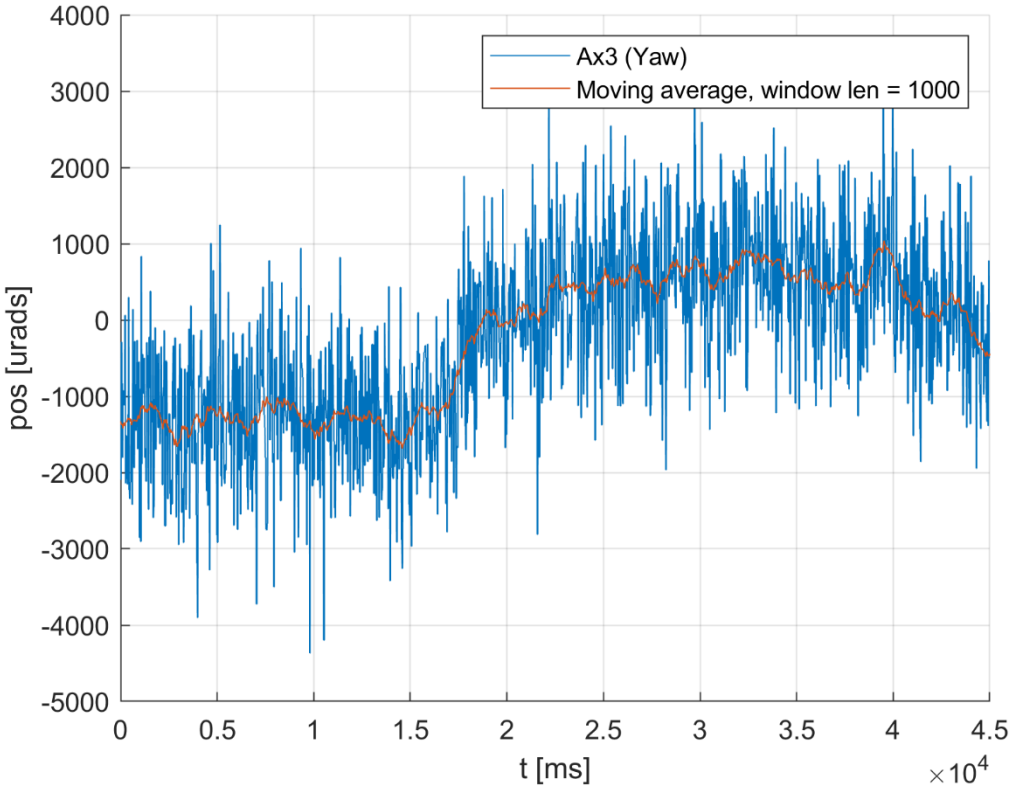

Fig. 66: Noisy signal from yaw readings

9 Conclusion

The work aimed to develop the software for the positioning mechanism – the device for positioning of 6DOF interferometer (Renishaw XM-60). The motivation behind the mechanism was to shorten and automate the alignment procedure.

Firstly, the software for interferometer positioning was developed. The control software was written in the form of a PLC project to enable manual control of the mechanism.

Then, the algorithm for quick alignment was designed and tested. The algorithm allows aligning the launch and the receiver of the interferometer automatically.

Finally, communication was developed and implemented. The communication software enables accessing the readings of the interferometer measurements from the PLC.

The software was successfully tested with the actual mechanism, and the functionality was verified. The quick alignment algorithm was also successfully tested.

Thus, the work assignment was fulfilled. The positioning mechanism with the developed software and quick alignment algorithm can be used to measure the machine's axes geometric errors: positioning error, horizontal straightness, vertical straightness and three angular errors (roll, pitch, yaw).

In future, the software will be extended with the functionality for rapid identification of volumetric error of a large machine tool.

Bibliography

1. ŠVÉDA, J. et al. *WP09 - Výsledek na stroji - TOS Varnsdorf-2019 Měření volumetrické přesnosti.* Praha : ČVUT v Praze, Fakulta strojní, RCMT, 2019. Výzkumná zpráva V-19-061.

2. Krulewich, Debra A. *Rapid Mapping of Volumetric Machine Errors Using Distance Measurements.* Lawrence Livermore National Laboratory. San Sebastian : s.n., 1998.

3. Máčala, Daniel. *VOLUMETRICKÁ PŘESNOST OBRÁBĚCÍCH.* VUT BRNO. 2019.

4. Morávek, Martin. *Výzkum přesnosti pětiosých frézovacích center.* FS ČVUT. Praha : s.n., 2018. Disertační práce.

5. Virtual machining considering dimensional, geometrical and tool deflection errors in three-axis CNC milling machines. *ResearchGate.* [Online] https://www.researchgate.net/figure/Geometric-error-parameters-of-3-axis-milling-machine_fig1_262690721.

6. *Thermal error compensation of a 5-axis machine tool using indigenous temperature sensors and CNC integrated Python code validated with a machined test piece.* Mareš, Martin, Horejš, Otakar and Havlík, Lukáš. s.l. : Elsevier Inc., 2020, Precision Engineering, Vol. 66, pp. 21-30.

7. *A New Methodology for Kinematic Parameter Identification in Laser Trackers.* Majarena, Anna Christina, et al. s.l. : IntechOpen, 2017.

8. *Relationship between ISO 230-2/-6 Test Results and Positioning Accuracy of Machine Tools Using LaserTRACER.* Lee, Hau-Wei, et al. 2016, applied science.

9. Interferometry explained. *Renishaw.* [Online] https://www.renishaw.com/en/interferometry-explained--7854.

10. XM-60 and XM-600 multi-axis calibrator. *Renishaw.* [Online] https://www.renishaw.com/en/xm-60-and-xm-600-multi-axis-calibrator--39258.

11. *Beckhoff Information System.* [Online] https://infosys.beckhoff.com/.

12. AlignMeter USB. *Duma Optronics Ltd.* [Online] https://www.dumaoptronics.com/alignmeter-usb.

13. LaserOn. *Duma Optronics Ltd.* [Online] https://www.dumaoptronics.com/laseron.

14. Stejskal, V. and Valasek, M. *Kinematics and Dynamics of Machinery.* New York : Dekker, 1996. ISBN-13: 978-0824797317.

15. New compensation system improves machine tool accuracy. *design products & applications.* [Online] 2012.

16. Renishaw. *Documentation for XMApi.*

List of figures

List of tables

Electronic attachments

1. Source code for PLC project: PosMechCtrl.tzip
2. Source code for communication software: Xm60Feedback.zip