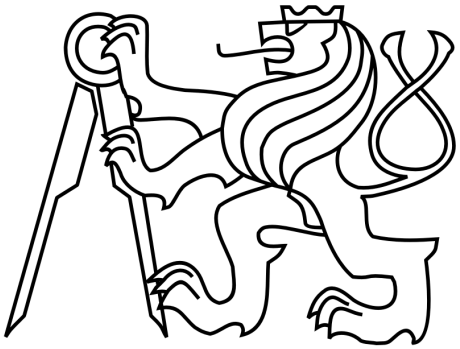
Czech Technical University in Prague
Faculty of Electrical Engineering

BACHELOR THESIS



Tomáš Froněk

Design of a physical model with modern control devices

Department of control engineering

Supervisor of the bachelor thesis: Ing. Pavel Burget Ph.D.
Study programme: Cybernetics and robotics
study branch: Control system engineering

Prague 2016

I declare that I carried out this bachelor thesis independently, and only with the cited sources, literature and other professional sources in accord with Methodical instructions about ethical principles for writing academic thesis.

In Prague, May 26, 2016 .....

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra řídicí techniky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Tomáš Froněk**

Studijní program: Kybernetika a robotika
Obor: Systémy a řízení

Název tématu: **Návrh fyzikálního modelu s moderními řídicími prostředky**

Pokyny pro vypracování:

1. Navrhněte fyzikální model pro polohové řízení v několika stupních volnosti.
2. Navrhněte strukturu řídicího systému s komunikací Profinet s připojením do systému LabLink. Řídicí systém musí být rozšiřitelný o další zařízení i modely.
3. V prostředí Matlab vytvořte model fyzikálního systému a jeho regulátor.
4. Implementujte regulátor do řídicího systému. Využijte automatické generování kódu pro PLC. Rovněž implementujte možnost řídit model v Matlabu z PLC bez nutnosti připojení fyzikálního modelu.

Seznam odborné literatury:

1. Siemens. System Manual for Step7 Professional (TIA Portal). 2014.
2. Siemens. Profinet with Step7 V13. Function manual. 2014.
3. Ellis, G. Control System Design Guide. Elsevier. 2004.

Vedoucí: Ing. Pavel Burget, Ph.D.

Platnost zadání: do konce letního semestru 2016/2017

L.S.

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 24. 2. 2016

Title: Design of a physical model with modern control devices

Author: Tomáš Froněk

Department: Department of control engineering

Supervisor: Ing. Pavel Burget Ph.D., Department of control engineering

Abstract: This thesis describes the production and programming of 3DOF (degree of freedom) RPS (rotational-prismatic-spherical joints) motion platform as school model for teaching courses with a focus on automation and industrial control. Feedback control is by means of industrial camera and the model is controlled by PLC. All parts of the model are standard industrial parts.

Abstrakt: Tato práce popisuje výrobu a programování 3DOF RPS platormy jako školního modelu pro výuku automatizace a průmylového řízení. Zpětnovazební smyčka je realizována pomocí průmyslové kamery a model je řízen standardním PLC. Všechny části modelu stojí na standardních průmyslových komponentách.

Keywords: 3DOF Platform Simatic IO-link Profinet Festo Siemens Industrial camera Linear actuators

I would like to thank my supervisor Ing. Pavel Burget Ph.D. for support and help with production of this thesis. Also I'd like to thank Ing. Josef Kvac, Ing. Ivo Zizka and Ing. Ondrej Rakusan from Siemens company for advices and technical consultations that save me lot of precious time. Big thanks for help with english, is to my friend Clara Cole-Hawthorne. Last but not least, I want to thank to my parents and my friend Pavla Blahova for unstoppable stream of good mood and support during writing this thesis and my studies.

Contents

Introduction

In recent years, industrial control systems have made a huge leap forward. Buzz words such as Industry 4.0 or IoT (Internet of Things) resonate the whole industry. New types of communication have appeared and systems have began to be much more integrated and connected together. During discussions with my supervisor, Mr. Burget in semester preceding my thesis, we set goals, to achieve modern interpretation of these thoughts. The model should:

- build on standard industrial hardware.
- build on hardware from more than one supplier.
- be easily expandable by means of communication and connection to school network and superior systems.
- be modular, meaning all parts of model should be useful in other applications.
- use several communication networks.
- be easy to use and configure.

We decided to make a simple mechanical model, build on modern control ideas and devices.
The model is a simplified Stewards platform, 3DOF manipulator. I was, in the beginning inspired by robotic maze that I have find on the web ref.[1]. I have decided to make model with future potential for solving maze and controlling path of ball traveling through it.
As I mentioned before, as there is demand for a modern system, I decided to use motors with higher communication possibilities and not standard PLC. PLC isa Computer/PLC hybrid system and is the first of it's kind in sales. This is described later in thesis see 2.5.
Thesis is written with respect for future development of model, as well as formal description of model production. In the first two chapters I am describing the physical model and production, in subsequent chapters I am talking about programming and connecting model into superior systems. Finally, I outline possible future development and students assignment on the model.

1. Physical model

Physical model is a basic planar manipulator, with a manipulated platform with placed ball. Control task is, to manipulate ball on plate, in the desired position or on the desired path. Thanks to cooperation with Festo company, we have been able to use electro mechanical drives for positioning platform and camera for feedback. Control hardware was supplied by Siemens company.

1.1 Moving platform

Main part of the whole model is 3-DOF (Degree of freedom) planar manipulated platform with standard configuration of three RPS (Rotational-prismatic-spherical) joints.
Axes, later referenced in thesis are defined as:

- X axis - is perpendicular to camera construction see 1.1.1 and positive in direction to technology on platform base
- Y axis - is parallel to camera construction see 1.1.1 and positive in direction to ET200SP or blue camera cable.

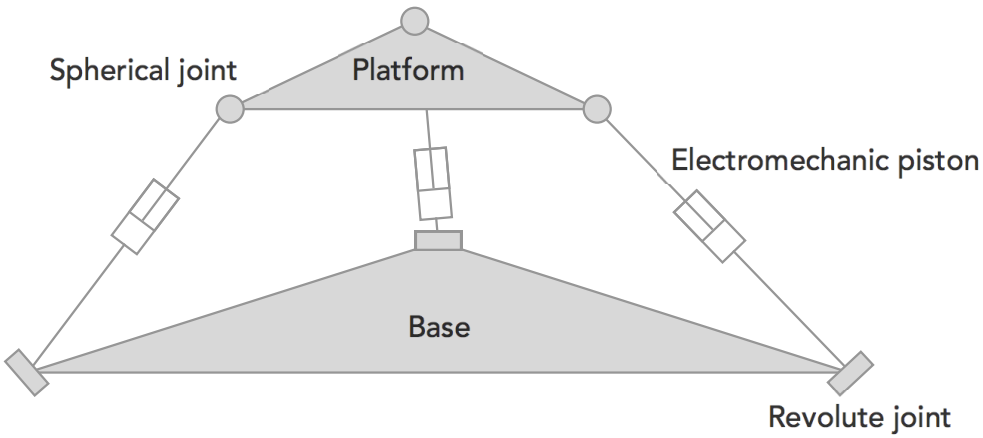


Figure 1.1: Graphical representation of 3-DOF RPS platform

1.1.1 Design

The model consists of several parts:

Model base is triangular shape made out of ITEM Profile 5, squared profile with $20mm$ sides. Mounting for motors in corners is made out of larch wood plank $25mm$ thick, because of larch strength and flexibility. On one side is DIN3 mounting for motor controllers and ET200SP communication unit. Motors are

connected with standard Festo mounting, allowing only revolute movement of motor. Model base is on Fig.1.2.

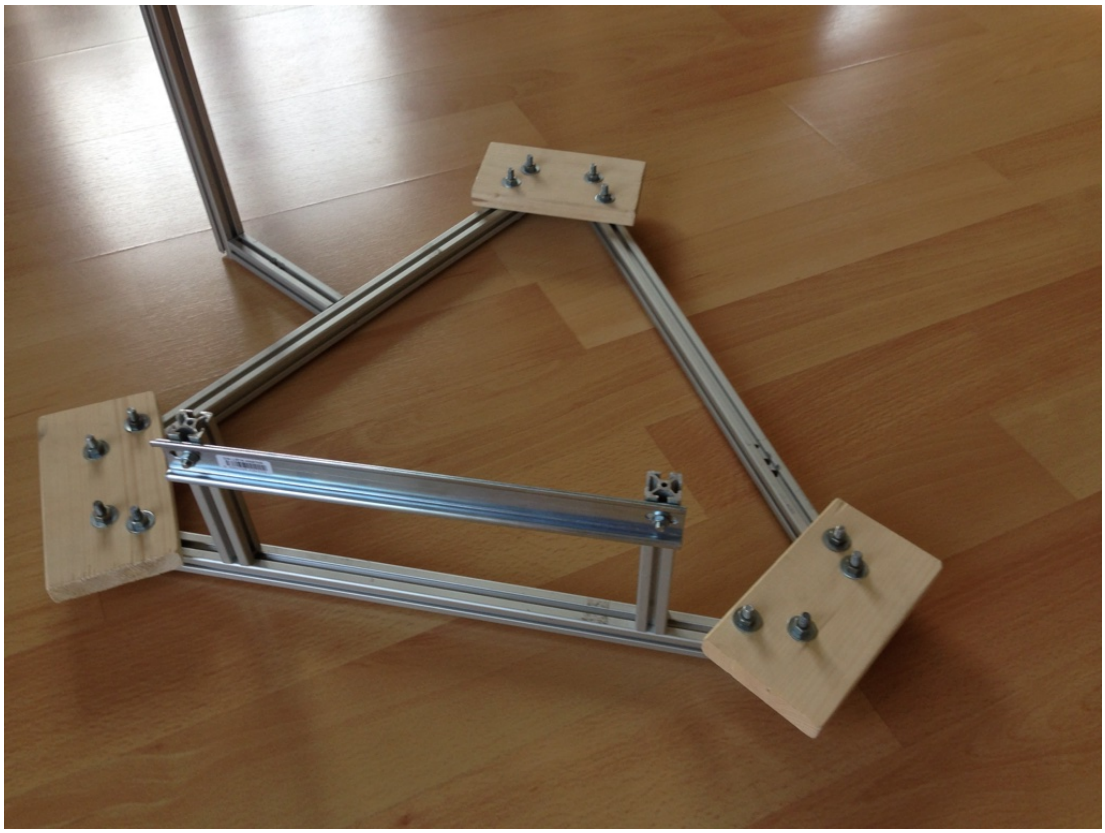


Figure 1.2: Model base before mounting technology.

Motors are linear electro-mechanical actuators EPCO-16 from Festo [2]. Communication between our controller and motors is, by means of IO-link connection. Each motor is connected into encoder unite CMMO-ST [3] that enable communication via IO-link to ET200SP IO-link Master distributed peripheral [4].

Controlled platform is a squared shape, plywood desk size $50cm$ to $50cm$ with cut $5mm$ deep $2,1mm$ wide grooves for variable shaped operating space. Platform is connected to EPCO actuators with spherical joints GFSM by Hennlich [5]. They allow unrestrained movement of the whole platform. Platform is on Fig.1.3.

Camera mounting was first attached to the positioning platform. First test of platform control, has showed that this solution is unsustainable during longer run of model, due to camera vibrations and the forces affecting aluminum profiles holding the camera. Later in production the camera mount was replaced and fit on base platform. Camera is, in platform highest possible point, no less than $110cm$ above platform, due to narrow view field of camera. It consists of "A" shape construction, where bottoms of legs are mounted to base and on top is our camera. The material chosen was aluminum U shape profile with $150mm$ x $150mm$ cross-section, because of flexibility and low weight.

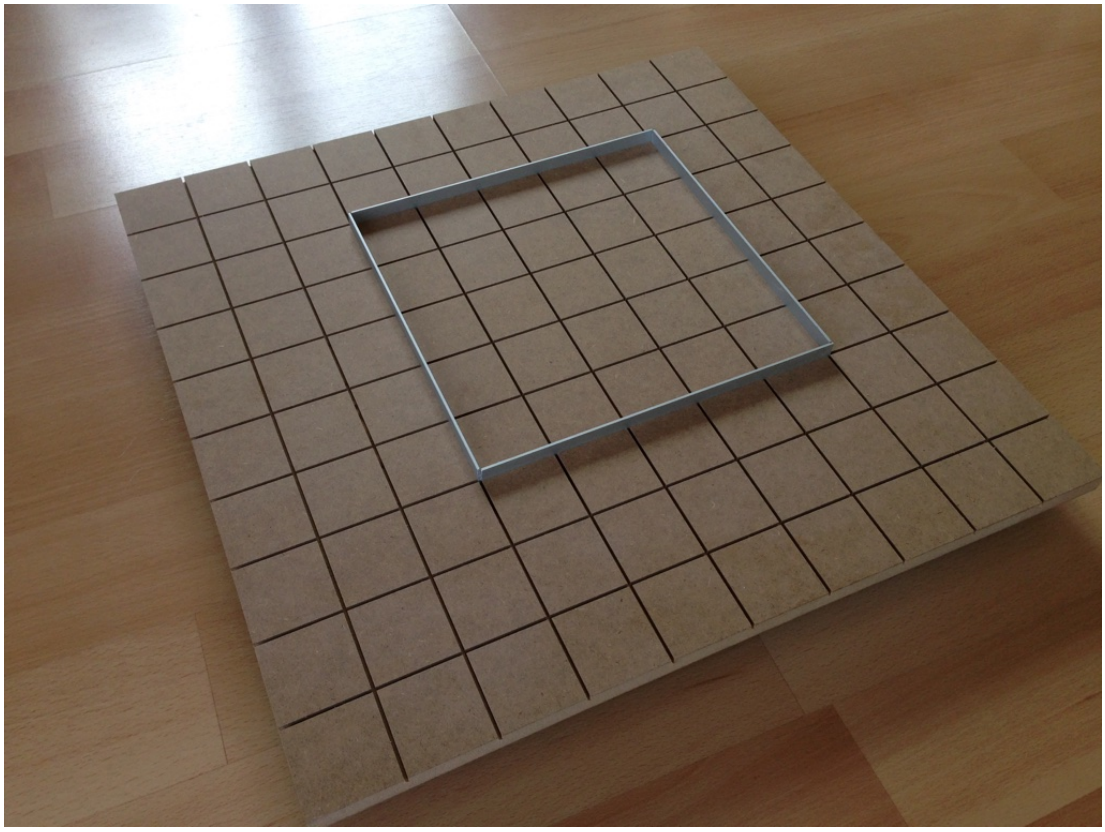


Figure 1.3: Controlled platform with small bounded playground.

1.1.2 Production

In model production, some issues occurred. The biggest problem was with the base of platform. This is made from ITEM aluminum profiles. Unfortunately, I was missing essential tools for manufacturing aluminum, so I was unable to precisely cut all the profiles, which has been found to be the main influence on precision of mathematical model and later during testing and programming where I had to approximate model several times. Some other obstacles were due to the type of material, I used for the positioning platform, It cannot be too heavy, because of maximal load of motors and sufficient dynamics of whole system. After a few tests, I have decided to use board from glued beech veneer. Problem with this type of material is low strength by means of layers, so if you need to screws something from side of platform, great caution and thin screw are necessary.
Platform board was cut on CNC machine, so there is no problem with precision. Another precision problem, was with camera mounting, I was again unable to bend aluminum belts in precise shapes. Fortunately, in flexibility of long aluminum profiles, these inaccuracies disappeared.

1.2 Control hardware stand

Except motor encoders, ET200SP distributed unite and vision sensor, there is need for placing actual PLC, power source, Profinet network switch and HMI panel. Because of thesis requirement for hight modularity, I have placed those parts on separate, easy-to-carry stands.

1.2.1 OpenControler stand

The stand for our main controller, the ET200SP CPU Open Controller, is designed from beginning, as a modular system with high portability. It contains, in current version, Open Controller itself, Scalance switch and Sitop power source. Connection to model is made by means of power cable with 24V voltage and Profinet cable for communication. The whole stand is made from ITEM 5 aluminum profiles, technology is mounted on DIN3 lines. It is possible thanks to low weight and modularity to use this stand for controlling other models or as presentation set for Profinet/ProfinetIRT or Siemens software PC technology.

1.2.2 HMI panel

Because of the requirement for controlling the model, HMI panel is needed to be placed close to the system. I was unaware, where the model is going to be placed later on, so I have decided for a simple mounting for HMI.The stand is just two simple bend $25mmx6mm$ aluminum belts, $50cm$ long each. Bent to 60 deg, HMI panel is connected to OC stand by Profinet cable and 24V power cable.

2. Control system

Control system consists of ET200SP CPU Open Controller [6], ET200SP distributed peripheral with IO-link Master communication card [4] with link to CMMO motor encoders [3], which control EPCO motors [2]. Closed loop control is enabled by vision sensor SBSI [7], which communicates data by ethernet to Open Controller.

2.1 Requirements on system

From the beginning, there is a request on possible future development and modularity. This is guaranteed by using industrial hardware, with long lasting support period and defined attributes.
By means of communication, the model has to be able to connect into the school model environment LabLink. I have prepared integration, by means of OPC server. For future development I would recommend to use VPN or router in front of Open Controller, it would prevent IP address collisions that I have encountered during testing.

2.2 Parts of system

You can see a graphical representation of systems communication structure on Fig.2.1. By means of communication, we can separate and describe several parts of the model.

2.3 Feedback - Vision sensor

For closed loop control, we use vision sensor for determining position of ball, on our moving platform. Used hardware is Festo SBSI-Q-R3B-F6-W, technical information can be found in [8].

2.3.1 Configuration

For camera configuration Festo "Vision sensor configuration studio" is used. Camera configuration file in .job format can be found in attachments and uploaded on open controller. I have set the camera for searching definite pattern, ball on platform. The ideal ball for these purposes is a standard squash ball, which has enough weight, not to get stuck in the drains for barriers of maze and black color is ideal contrast for the light brown of the platform. Camera detecting ball is shown on Fig.2.2. Camera is liable for bad detection in low light, because of a used detector. For speed purposes, I used grayscale pattern detector instead of contour detection. So in low light, the difference in grayscale drops and the camera is sending time to time fault signal. It is important to use job for dark conditions, if this happens.

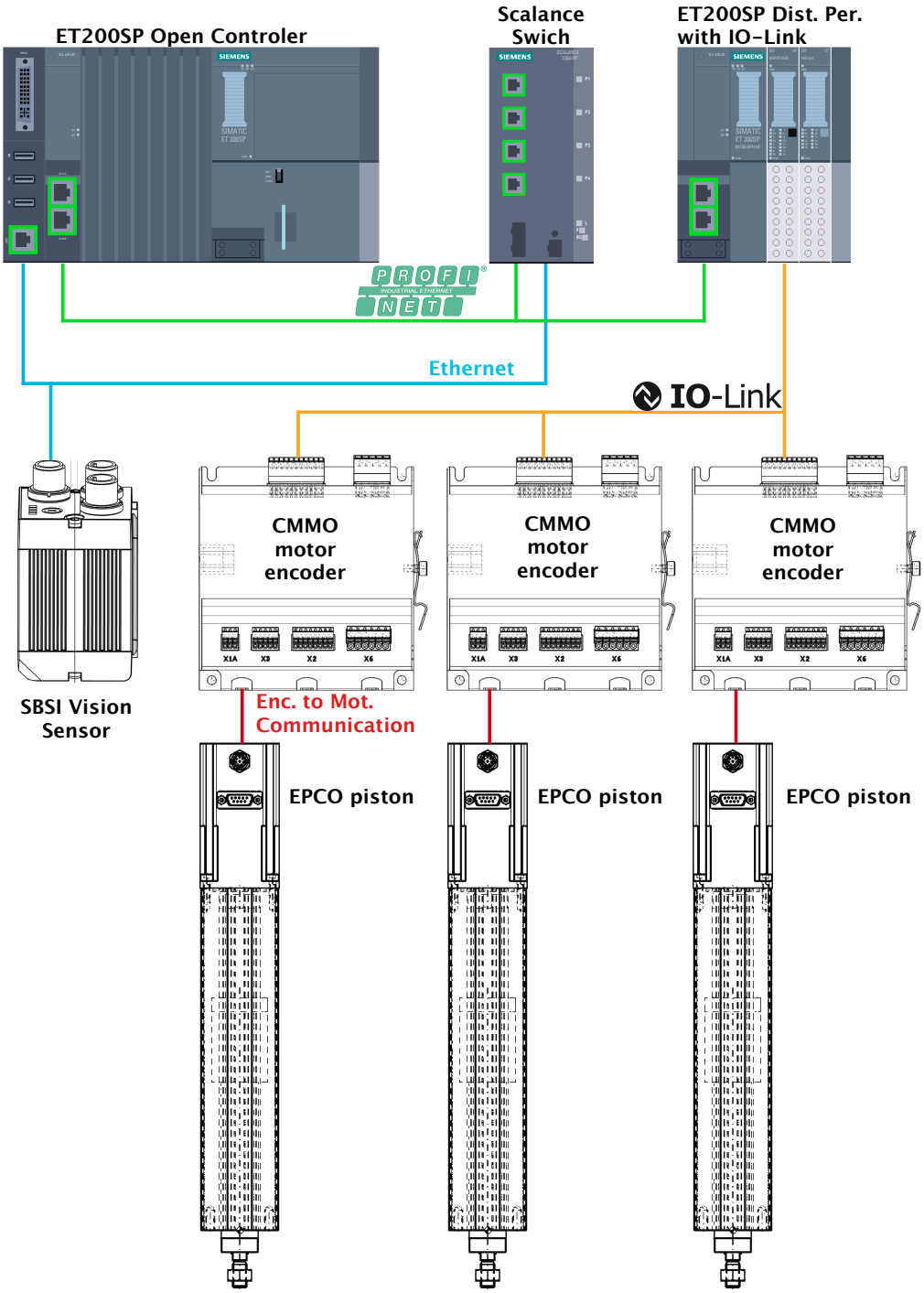


Figure 2.1: Communication structure of model

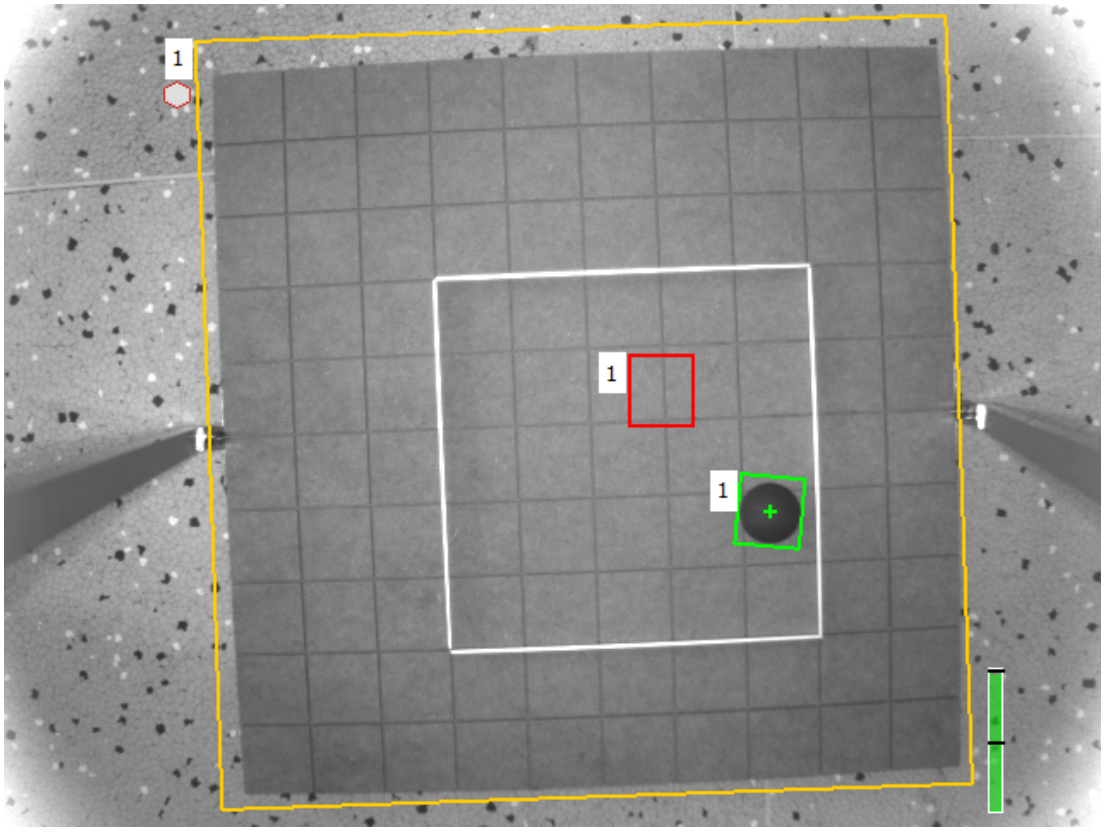


Figure 2.2: Camera detecting ball. Yellow frame is area where object is located, red is original position, green is actual position of object.

2.3.2 Communication

By means of standard TCP/IP server, it is receiving data on port 2005 and sending them on port 2006. I am therefore able to control camera jobs as well as receiving data on Open Controller, running as TCP client on this connection.

2.3.3 Operation

I have created two different camera jobs, that are uploaded into the camera, and you can switch between them by HMI panel.
The job switch is done by sending to camera IP address on port 2006 telegram "`CJB001`" rep. "`CJB002`". The job CJB001 is for light conditions, when internal illumination of camera is switched off, thanks to that, it has shorter cycle time about $50ms$, also, it is shortest cycle time I was able to reach. The job CJB002 is created for dark conditions, internal illumination is switched on, that prolong cycle time, because of need to charge capacitor for LED flash. Cycle time for this job is about $90ms$.
In a standard run, camera is sending, in previously mentioned cycle time telegrams on port 2005 in format "`positionx;positiony;a`". They are received and parsed in Open Controller CPU.
In chapter 4.2 are my recommendations for camera. This cycle time can be definitely better, I would recommend to consult this topic with expert from Festo.

2.4 Platform control - motor encoders

On platform, I have ET200SP communication unit with IO-link Master card for communication and CMMO-ST motor encoders. For information about programming ET200SP go to part 2.5. List of used parts is in *Attachment 1.*

2.4.1 Motor encoder

Encoders are using IO-link communication, to exchange data with open controller. Due to power consumption of EPCO motors, they have two separate power inlets. One for motor supply, with 24V, and a maximum of 1,3A per motor. This connection is connected straight to Sitop power source on Open Controller stand ref.1.2.1. The second 24V connection is for logical high value, it is connected to ET200SP communication card.

2.4.2 Configuration

First configuration of encoder is done by web server, running on each encoder. I had to upload motor configuration and set values for movement. For first testing I have set values for acceleration and deceleration to $1500mm \cdot s^{-2}$. Later during testing, it turns out that this value of acceleration causing oscillation of the platform, due to clearance of rotational joints between motors and base. I have been forced to reduce the value, low to $1300mm \cdot s^{-2}$ when oscillation didn't occurre during operation. Maximal speed of motors is set to $180mm \cdot s^{-1}$. This turns out to be sufficient during testing.

2.4.3 Communication

As mentioned before, communication between encoders and PLC is by means of IO-link connection. IO-link is world standardize IO technology based on IEC 61131-9. Using a 3 wire serial connection with one data wire it can reach a maximum speed of 230 kBaud. Master and slave IO-link devices, both communicate with this speed. Communication is a master-slave, where slaves are my motors encoders.

Communication configuration

To configure IO-link communication, it is necessary to use S7-PCT(Port Configuration) tool. Launch of S7-PCT is done from TIA portal. Every IO-link slave device need to have IODD (IO device description) file, which is the description of communication between master and slave. Those files are available usually at manufactures support web pages. You can upload them to IO-link master device with S7-PCT tool, and check if the connection between slaves and master is correctly configured.
Second part of configuration is in TIA portal, where it was necessary to setup addresses for communication. This is because, IO-link Master device behaves as standard Profinet IO device, I and Q address in device view in TIA portal can be found, see2.3. In communication blocks for encoders, we have to setup beginning of address space, in my case for I and Q both 0, where we want to write or read

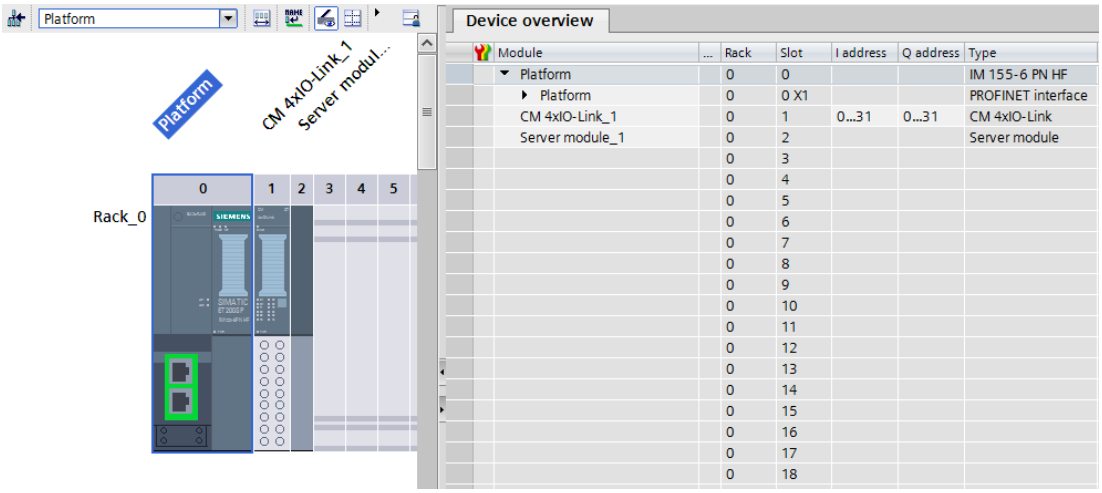


Figure 2.3: On right side are addresses of I and Q to IO-Link Master

motors data. Later in 2.4.4 I describe how I used motor mode using 8 Byte of data, so every motor has address increased by 8 from previous address. Used ET200SP IO-link Master has the possibility to use up to 128 Byte address space. Currently, I'm using only 24 Byte, IO-link Master has just 32 Byte reserved data space for inputs and output, this is enough.

2.4.4 Operation

Here I touched enveloped of possibility of the system by means of its dynamics. During the testing, it turns out, that the time delay of motors is simply to big for system to be controllable.
First I reduced exchanged data between encoders and IO-link Master. Motors can communicate in two modes:

- FHPP (Festo handling and positioning profile) - Communication takes place of 8 byte, cyclicly exchanging only process and error data.
- FHPP + FPC (Festo parameter channel) - In addition to 8 byte of FHPP, FCP part takes 8 byte more. It allows to change all motors parameters via IO-link (acceleration, max. speed,...).

During first testing I have been using extended mode FHPP + FPC, where you can configure motor with IO-link, you can set motor settings and read extended diagnostic, this feature was particularly useful during commissioning. However an issue with this was, that FHPP + FPC communication consuming 16 Byte of data, in contrast to 8 Byte for standard FHPP. Because of that, I was forced to hard set values of motors with integrated web server and use shorter communication protocol. Those values are mentioned in 2.4.2.
Unfortunately, this only helped partially, another problem cause is that you cannot dynamically change set point for motors. When you assign final position for motor, you can change it only after reaching final position. This radically reduced reaction speed of system. Therefore, I had to treat the motor differently. Every time, I received a new final position from the controller, I stop the motor, delete the present position and write down new and start task again see2.5.4.

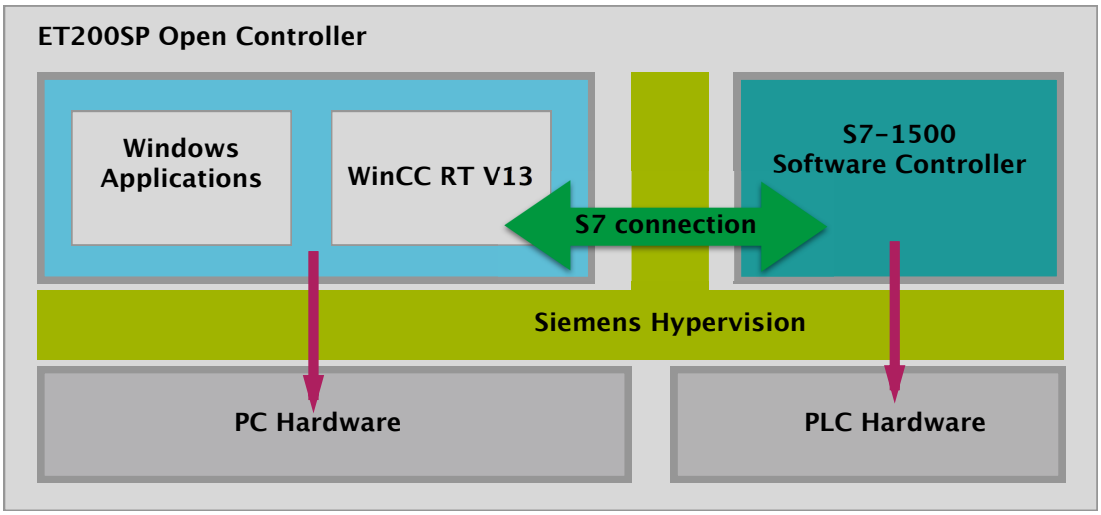


Figure 2.4: Internal structure of ET200SP Open Controller

To avoid uneven movements causing instability and mechanical tension, motors stops without the use of brakes. Brakes are used only in manual controlled mode see 3, when it is stabilized in desired position.

2.5 ET200SP Open Controller

The center point of the whole model is ET200SP CPU, so called Open Controller. In this section, I will be talking particularly about on board installed S7-1515SP PC. The structure of Open Controller is described in the next section. Whole control program is running on this CPU, there are no IOs on this rack. All IOs are connected by means of Profinet IO-device on platform.

2.5.1 Description of Open Controller system

The system which I refer as "Open Controller" is IPC running Windows 7 embedded with installed S7-1515SP PC on board. Big difference from standard IPC is, that controller is running separately from Windows, it is not affected by Windows crash or restarts. It exchanges data with internal S7 connection, it is possible to create user application running on Windows, that will exchanging data with controller. Structure of controller is on fig. 2.4.
On Windows an SW tool is preinstalled for configuring Festo camera jobs in case of big changes of condition to be able to run the model anyway. Also there I have stored documentation and archived TIA project, for everyone to be able to restore the model if anything happen to S7-CPU. Visualization for model control is running on WinCC RT V13. This means, that you can remotely operate whole model just by connecting to remote desktop of open controller.

2.5.2 Network structure

The whole model is connected by means of Profinet network. I'm using two TCP connections in network, TCP1 is communicating between open controller and camera on port 2005. It is used for getting the position data from the

camera. TCP2 communicating also between open controller and camera, but on port 2006 and it is used to configuration and error handling of camera. There is also S7 connection between CPU and my OPC server, which later on can be used for connecting model into faculty network. The structure of network with IP addresses is shown on Fig.2.5.

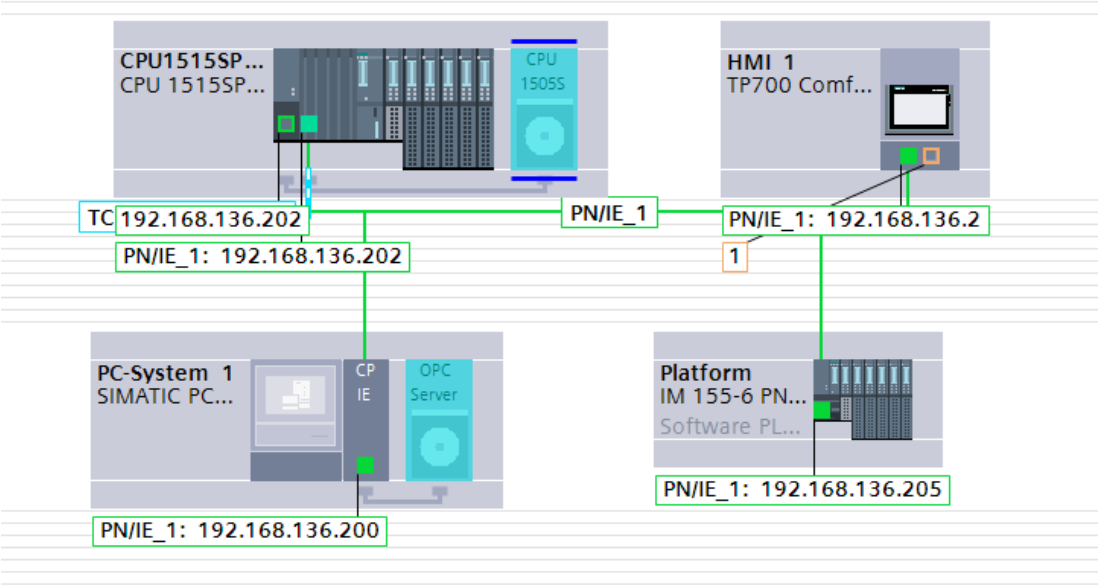


Figure 2.5: Model network structure

2.5.3 Control program structure

The whole program is created in TIA Portal V13 SP1. The mathematical description is created in Matlab 2016a exported to TIA Portal with PLC Code toolbox. The program is separated into several structures, each one is responsible for different process. On Fig.2.6 you can see the structure. All functions and function block are programmed with consideration of creating global function library for working with this model.

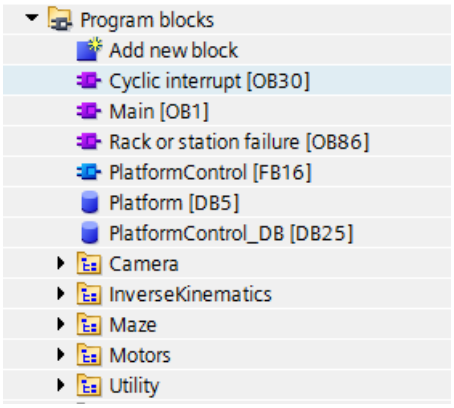


Figure 2.6: Program logical structure

2.5.4 Main control structure

In the main part of the program structure, you can see three organization blocks (OB), two data block (DB) and one function block(FB). Now I will describe function of each.

OB1 - Main

This is the main function block, in here, I'm calling FB1-CameraCom block, FB17 - Maze Run and Visualization Check. This block proceeds cyclicly with cycle time of CPU.

OB30 - Cyclic Interrupt

This block is set to be called cyclicly every $500\mu s$. This particular time, is set, because of stability of system. As I mentioned in 2.4.4, motors cannot receive final position continuously. So when I have new position for motors, I stop motor and write a new one. This is done by interrupting this block, every $500\mu s$ is motor for time of processing OB30 without power and slows down. If it receives new position, it continues to new position. This time is as short as possible, so motors cannot speed up much, so they wont be oscillating between high and low speed. This thought is backed up by testing with different cycle times. When OB1 is processed and OB30 call for interruption, CPU create a break point, execute OB30 and than continue. On fig.2.7 is structure of program run.

In this block I have two PID controllers for controlling X and Y motion of

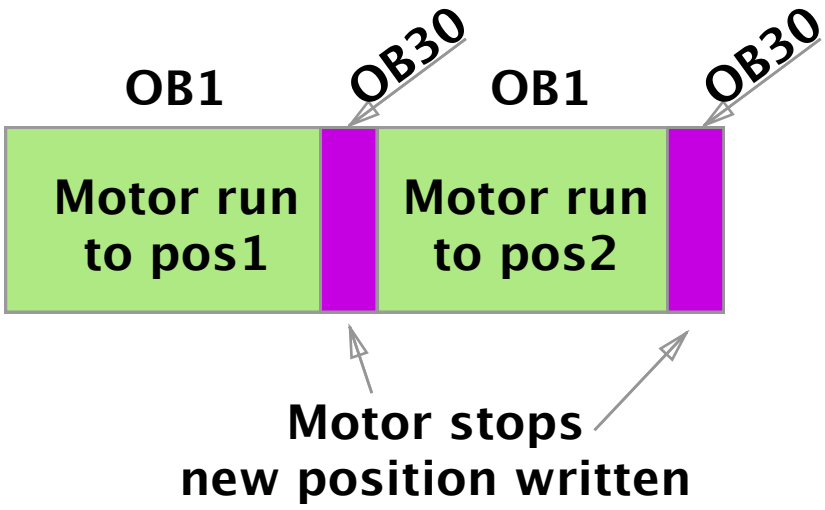


Figure 2.7: Program run, OB1 is executed multiple times before calling OB30 every $500\mu s$.

ball. Controllers are tuned with auto-tune option in TIA portal and adjusted manually by testing. During testing, I have tried various ball on platform. Each one is good for different purpose. PIDs are tuned to compromise between them, so you can use different balls without need of adjusting controllers or camera feedback. Blocks FB16-PlatformControl and FB4-Motors used in this block are

described later. The reason why I have used this block without the camera, is simple. Camera cycle time is $50ms$ at best and with this speed there is no point of parsing data from camera that often, executing PID is critical, and therefore, is done here.

OB86 - Rack or Station failure

This block is executed only, when an error happens on Profinet station, for me, on Platform. From power loss to loss just one IO-link channel. In the beginning of the production, I have used this block just for testing, because, I haven't got all the hardware connected. Now, when this block is executed, whole platform stops its movement where it is, to prevent damage motor by dragging.
This approach is according to modern trend of central error handling. Also, finding error is much easier, because they are written in diagnostic buffer of CPU. To enable this feature, it is necessary to check the error handling box in the device description of IO-link Master card on Platform.

FB16 - PlatformControl

This block has three inputs and three outputs, values that coming in are Rotation on X, Rotation on Y and Height of platform. This block contains block FB8-Subsystem, described later and transfers calculated values from inverse kinematics into values for motors. PlatformControlDB (DB25) is just data space for this function block.

DB5 - Platform

In this data block are stored all values for inverse kinematics as well as all runtime values from block above. I have also placed HMI tags for user control here. To enable control of whole model, you need access just into this data block. This means that the rest of the program can be protected by password and follow trend Security Integrated to prevent unauthorized data handling and access. This block could be considered as somewhat API into model.

2.5.5 Camera

This organization folder contains all the necessary blocks for communication with Festo Camera. As mentioned in 2.2, the camera is using standard TCP connection as server, block FB1 - CameraCom is used to serve this.

FB1 - CameraCom

This block contains TCP communication blocks. First couple of TSEND_C and TRCV_C with TCP1 connection is used for receiving data from the camera, second couple of TSEND_C block with TCP2 connection is used for changing job on the camera.
In Network2 of this block, I have block structure to parsing rough data from the camera into two LInt numbers PosX and PosY contains position data from the camera.

Second block structure is used to acknowledge if camera is detecting ball. Because, the camera cannot send separate telegram with information that the ball is not detected, I used the first two letters of rough data. If there is string "0;" then the ball is not detected and motion of platform is stopped.

Block for receive ball position - TCP1 Block TSEND_C and TRCV_C with connection TCP1 - connection to camera on port 2005 are used to get data from camera. In the beginning of testing, I have been using block TSEND_C to request data from the camera. When the camera receives telegram "TRG" it sends the position data back. But as mentioned, my control block is running every $500\mu s$ and camera cycle time is $50ms$ - this caused camera over load and actually increased cycle time up to 5 seconds. In present configuration, the camera is in free run, which means that with highest frequency it can send position data to PLC.
They are received by block TRCV_C in a 15 character long telegram into array of char, I have to used block in AdHoc mode, which means that length of telegram may vary. Simply because when the camera doesn't find the ball telegram is just 4 character long in format `0;0a`. When the camera found the ball, I did not know where to start parsing the telegram. AdHoc mode prevented this, by starting to write data every time from beginning of array.

Block for changing camera job - TCP2 As mentioned in 2.2 there is a need for changing job of camera. It is done by sending specific telegram on camera IP on port 2006. It is done, by means of those two blocks. For sending just one telegram, I'm using function to scanning rising edge of HMI button and than I send telegram during only one PLC cycle.

Camera data blocks Data blocks in this folder are used just to store internal values. In block CameraData, you can find variables PosX and PosY, which contains camera position data.

2.5.6 Inverse kinematics

In this block, there is an SCL code block used for calculation platform position on desired roll, pitch and height. For this block I used kinematics prepared in Matlab and toolbox PLC code, avoiding unpleasant work of matrix calculations in PLC.

Matlab mathematical model

Mathematical model is represented by MBS (Multi body system), the rigid bodies connected by joints. We see the system only from kinematic point of view so we could neglect weights of parts. Important however is to implement physical restrictions of system e.g. maximal length of actuators or deviation of joints. Model of system is based on the representation on figure 2.8.

Geometry Figure 2.8 describes spatial representation of my 3-DOF RPS platform. Triangle with label A represents base for model, triangle labeled B is the

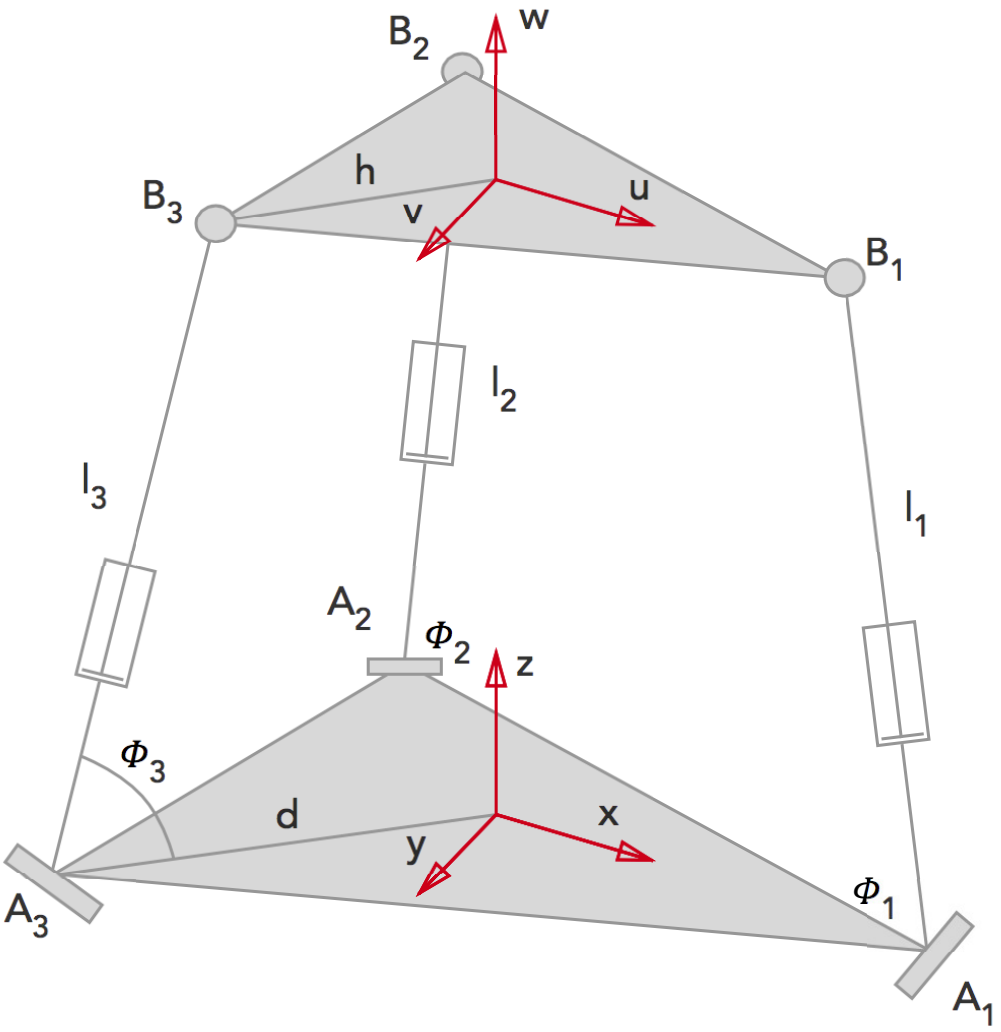


Figure 2.8: RPS platform geometry

moving platform. Joints $\mathsf{A}_{1,2,3}$ represent revolute joints and $\mathsf{B}_{1,2,3}$ represent spherical joints. Base and platform are connected with linear actuator, prismatic joints represent by $\mathsf{l}_{1,2,3}$. Following equations were determined based on articles [9], [10], [11], [12] and [13].
So now we can easily validate degree of freedom of our model. We can calculate that we have three revolute, prismatic and spherical joints. When we use *Grubler-Kuzbach* formula:

$$m = 6(l - n - 1) + \sum_{i=1}^{n} d_i = 3 \tag{2.1}$$

Where m in number of degrees of freedom, l is number of rigid bodies of robot, n is number of joints and d_i in degree of freedom of joint i.
Now we easily see, that platform has 3 degrees of freedom, two rotational in axis u, v and one translatory in ax w.

Coordinates of triangle peaks of base and platform are:

$$\mathsf{A_1} = [d, 0, 0], \mathsf{A_2} = [-\frac{1}{2}d, \frac{\sqrt{3}}{2}d, 0], \mathsf{A_3} = [-\frac{1}{2}d, -\frac{\sqrt{3}}{2}d, 0]. \tag{2.2}$$

$$\mathsf{B_1} = [h, 0, 0], \mathsf{A_2} = [-\frac{1}{2}h, \frac{\sqrt{3}}{2}h, 0], \mathsf{A_3} = [-\frac{1}{2}h, -\frac{\sqrt{3}}{2}h, 0] \tag{2.3}$$

Forward kinematic represent position of platform with respect to base based on known length of electromechanical joints $l_{1,2,3}$.
First we need to define position of joints $\mathsf{B_{1,2,3}}$ with respect to center of base, we will describe it as vectors $b_{1,2,3}$:

$$b_1 = \begin{bmatrix} d - l_1\cos(\phi_1) \\ 0 \\ l_1\sin(\phi_1) \end{bmatrix}. \tag{2.4a}$$

$$b_2 = \begin{bmatrix} -\frac{1}{2}(d - l_2\cos(\phi_2)) \\ \frac{\sqrt{3}}{2}(d - l_2\cos(\phi_2)) \\ l_2\sin(\phi_2) \end{bmatrix}. \tag{2.4b}$$

$$b_3 = \begin{bmatrix} -\frac{1}{2}(d - l_3\cos(\phi_3)) \\ -\frac{\sqrt{3}}{2}(d - l_3\cos(\phi_3)) \\ l_3\sin(\phi_3) \end{bmatrix}. \tag{2.4c}$$

Second, we know that distance of joint is constant and equal to:

$$|A_iA_{i+1}| = \sqrt{3}d, |B_iB_{i+1}| = \sqrt{3}h \tag{2.5}$$

When we substitute Eqs2.4 into Eq2.5 we get three nonlinear equations:

$$f_1(\phi_1, \phi_2, \phi_3) = 0 \tag{2.6a}$$

$$f_2(\phi_1, \phi_2, \phi_3) = 0 \tag{2.6b}$$

$$f_3(\phi_1, \phi_2, \phi_3) = 0 \tag{2.6c}$$

By substituting ϕ_1, ϕ_2, ϕ_3 back into Eqs2.4 we get our position vectors and we are able to calculate position of center of platform.

$$p = \frac{1}{3}(b_1 + b_2 + b_3) \tag{2.7}$$

We can describe position vectors b_1, b_2, b_3 as:

$$b_i = p +^A R_B \cdot {}^B b_i; i = 1, 2, 3 \tag{2.8}$$

Where p is position vector of platform center, ${}^A R_B$ is rotational matrix of B with respect to A and ${}^B b_i$ is position vector of platform peaks with respect to moving platform. Desired rotational matrix is than described as:

$${}^A R_B = \begin{pmatrix} u_x & v_x & w_x \\ u_y & v_y & w_y \\ u_z & v_z & w_z \end{pmatrix} \tag{2.9}$$

Vectors u, v, w are unit vectors of platform B with respect to base A and the comply:

$$\begin{aligned} u_x^2 + u_y^2 + u_z^2 &= 1 \\ v_x^2 + v_y^2 + v_z^2 &= 1 \\ w_x^2 + w_y^2 + w_z^2 &= 1 \\ u_x v_x + u_y v_y + u_z v_z &= 0 \\ u_x w_x + u_y w_y + u_z w_z &= 0 \\ v_x w_x + v_y w_y + v_z w_z &= 0 \end{aligned}$$

and position vectors with respect to platform:

$$b_1 = \begin{bmatrix} p_x + hu_x \\ p_y + hu_y \\ p_z + hu_z \end{bmatrix} \tag{2.11a}$$

$$b_2 = \begin{bmatrix} p_x - \frac{1}{2}hu_x + \frac{\sqrt{3}}{2}hv_x \\ p_y - \frac{1}{2}hu_y + \frac{\sqrt{3}}{2}hv_y \\ p_z - \frac{1}{2}hu_z + \frac{\sqrt{3}}{2}hv_z \end{bmatrix} \tag{2.11b}$$

$$b_3 = \begin{bmatrix} p_x - \frac{1}{2}hu_x - \frac{\sqrt{3}}{2}hv_x \\ p_y - \frac{1}{2}hu_y - \frac{\sqrt{3}}{2}hv_y \\ p_z - \frac{1}{2}hu_z - \frac{\sqrt{3}}{2}hv_z \end{bmatrix} \tag{2.11c}$$

Inverse kinematics means calculating length of planar joints e.g. our electromechanical actuators based on desired position and rotation of platform. Because, platform position is determine exactly by those lengths as we don't need any sensor on platform.
We have to calculate unknown length of l_1, l_2, l_3 based on known or desired position of platform center ${}^{B}P_{A}$ means center in system B with respect to A.
From [11] we know that inverse kinematic algorithm for 3DOF is:

$$l_i = \Big| A_i B_i \Big| = \sqrt{b_i^2 - \mathsf{A}_i^2}, \quad i = 1, 2, 3. \tag{2.12}$$

When we now substitute Eqs.2.2 and Eqs.2.11 into Eq.2.12 we get desired position equations:

$$l_1^2 = p_x^2 + p_y^2 + p_z^2 + 2h(p_x u_x + p_y u_y + p_z u_z) - 2dp_x - 2dhu_x + d^2 + h^2, \tag{2.13a}$$

$$\begin{aligned} l_2^2 = {} & p_x^2 + p_y^2 + p_z^2 - h(p_x u_x + p_y u_y + p_z u_z) + \sqrt{3}(p_x v_x + p_y v_y + p_z v_z) + \\ & d(p_x - \sqrt{3}p_y) - \frac{1}{2}dh(u_x - \sqrt{3}u_y) + \frac{1}{2}dh(\sqrt{3}v_x - 3v_y) + d^2 + h^2, \end{aligned} \tag{2.13b}$$

$$\begin{aligned} l_3^2 = {} & p_x^2 + p_y^2 + p_z^2 - h(p_x u_x + p_y u_y + p_z u_z) - \sqrt{3}(p_x v_x + p_y v_y + p_z v_z) + \\ & d(p_x - \sqrt{3}p_y) - \frac{1}{2}dh(u_x + \sqrt{3}u_y) - \frac{1}{2}dh(\sqrt{3}v_x + 3v_y) + d^2 + h^2. \end{aligned} \tag{2.13c}$$

Matlab model implementation

In the beginning we have to define our rotational matrix. According to [9]. Due to space restrictions, in following matrix is, c-cosine and s-sine.

$$ {}^{a}R_{b} = \begin{bmatrix} c\alpha c\beta & c\alpha c\beta s\gamma - s\alpha c\gamma & c\alpha s\beta c\gamma + s\alpha s\gamma \\ s\alpha c\beta & s\alpha s\beta s\gamma + c\alpha c\gamma & s\alpha s\beta c\gamma - c\alpha s\gamma \\ -s\beta & c\beta s\gamma & c\beta c\gamma \end{bmatrix} \tag{2.14} $$

And than calculate constrain equations for platform.

$$ \alpha = \arctan \frac{s\beta s\gamma}{c\beta + c\gamma} \tag{2.15a} $$

$$ x_c = \frac{h(c\alpha c\beta - s\alpha s\beta s\gamma - c\alpha c\gamma)}{2} \tag{2.15b} $$

$$ y_c = -hs\alpha c\beta \tag{2.15c} $$

I have created Simulink model of those equations, structure of model is on Fig.2.9

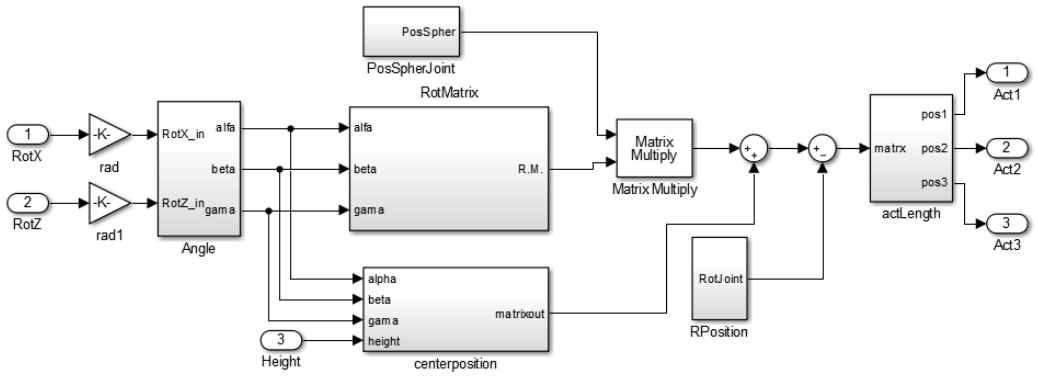


Figure 2.9: Simulink equations structure

Generating SCL code

Thanks to PLC code toolbox in Matlab, I was able to generate from inverse kinematic subsystem desired SCL code. Code generation had one fault, the squared operation was generated as $**2$ but SCL compilator in TIA Portal needed to be defined as `SQRT()`, so I had to manually re-write those parts of the code.
To use externally generated SCL code, I had to import it into project as external source file and let TIA portal to generate function block. That is, where **FB8 -Subsystem** was generated.

2.5.7 Maze

FC8 - VisualizationCheck

This function, is used to show alarms on visualization for starting PID control and maze run. If the ball is not placed within 15 pixel radius to point where stabilization or maze run should start. Red warning label appears next to button to start process.

FC7 - MazeCount

When mate run mode is initiated, this function is responsible to check, if the ball reached waypoint on run through maze. The ball has to be in 15 pixel radius from waypoint and stay there for at least $50ms$. Than next waypoint is initiated.

FB17 - Maze Run

This block is activated, when you start mode "Maze Run". In block is the function MazeCount, checking if block reached targeted position on maze.
When you start maze run, first, the platform is stabilized to zero pitch and roll and lifted $5cm$ above to have no restrain movement. Then from block DB31-Maze are loaded first waypoint values and as setpoint loaded into PID controllers for X and Y axes. When the ball reaches waypoint, it is confirmed by FC7-MazeCount and the next waypoint is loaded.
When the ball reaches the end of the maze, it is stabilized here.

DB37 - Maze

In this data block are stored waypoints, which the ball has to reach during the maze run. If you want a different path, you can change it here.

2.5.8 Motors

The organization folder Motors contains all communication via IO-link to encoders resp. motors. Inside, there is one more folder IO-link, containing all data blocks and functions for IO-Link communication.

FB4 - Motors

In this block, I have all the connections by IO-link to motors. Every motor has its own instance of three blocks, one for sending data, one for receiving and one for telegram handling. Each motor has own network with those three blocks. Structure of communication is on fig.2.10 In this block logic is also implemented deleting motor job, when receiving the new position for motor.

Other blocks in this folder are just for serving motors and IO-link communication. As you can see in folder, IO-link is standardized however slow and quite complex communication, connecting motors with IO-link is than quite limiting.

2.5.9 Matlab to PLC connection

As you can see on fig.2.5, connection to superior systems is by means of OPC server. I have followed manual Ref.[14].

OPC stands for "OLE (Object Linking and Embedding) for Process Control". It is standardize communication protocol designed for single interface between control hardware and software. For us it means, that the connection between Matlab and model is on a straight forward platform with sufficient speed.

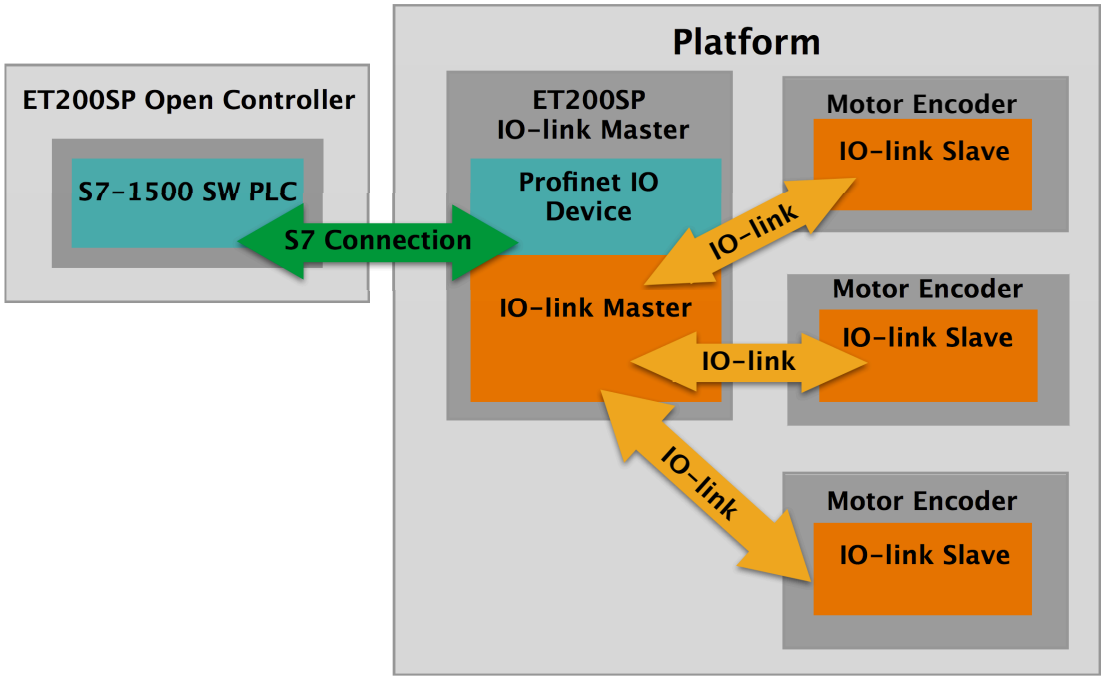


Figure 2.10: IO-link communication structure from platform to open controller

Simatic NET OPC Server in V13 is my OPC UA (unified architecture) server for testing purposes, I have installed it on my PC. Later in model commissioning, it is necessary to install it on the server, where students will be able to control model. It is also possible to use OPC connection for communication between application made in different language such as C or Java.
When installed, there was a problem with connecting to the OPC UA server The system could not automatically detect it. It turns out, that because of my PC network configuration OPC UA server is running on very high port, in my case 55105. It was then necessary to connect to server manually. Port for server connection can be found in SW Communication Settings in OPC protocol selection as you can see on fig.2.11. For connecting PLC to OPC, all data block used for data transfer need to have optimized data access. On fig. is screen shot from SW OPC Scout V10 used for testing connection of OPC to Simatic. On fig.2.12 is screenshot of proper setting and connection to model.

Matlab OPC connection

Because of use OPC UA, I had to use Matlab functions ratter than Simulink block for OPC connection. Blocks for Simulink are not ready for use yet. According to MathWorks, they will be available in the end of summer 2016. I have used OPC communication toolbox and created three .m files to be used for communication.

opcuainit.m is file for initialization connection, here you can specify what values you want to transfer to Matlab. Set same variables as on fig.2.13 to get proper results with uneddited code.

opcuaread.m is file to read and display values from model. With parameter "run" you specify how many samples you get. On fig.2.14 is graphical result of

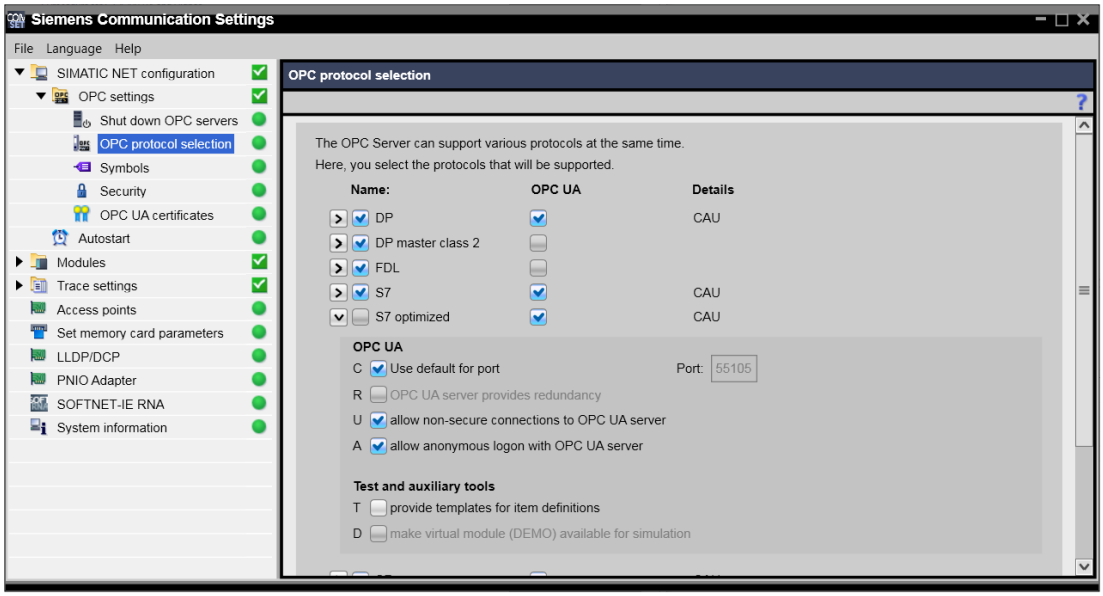


Figure 2.11: OPC Communication Settings, here is port where OPC communicate

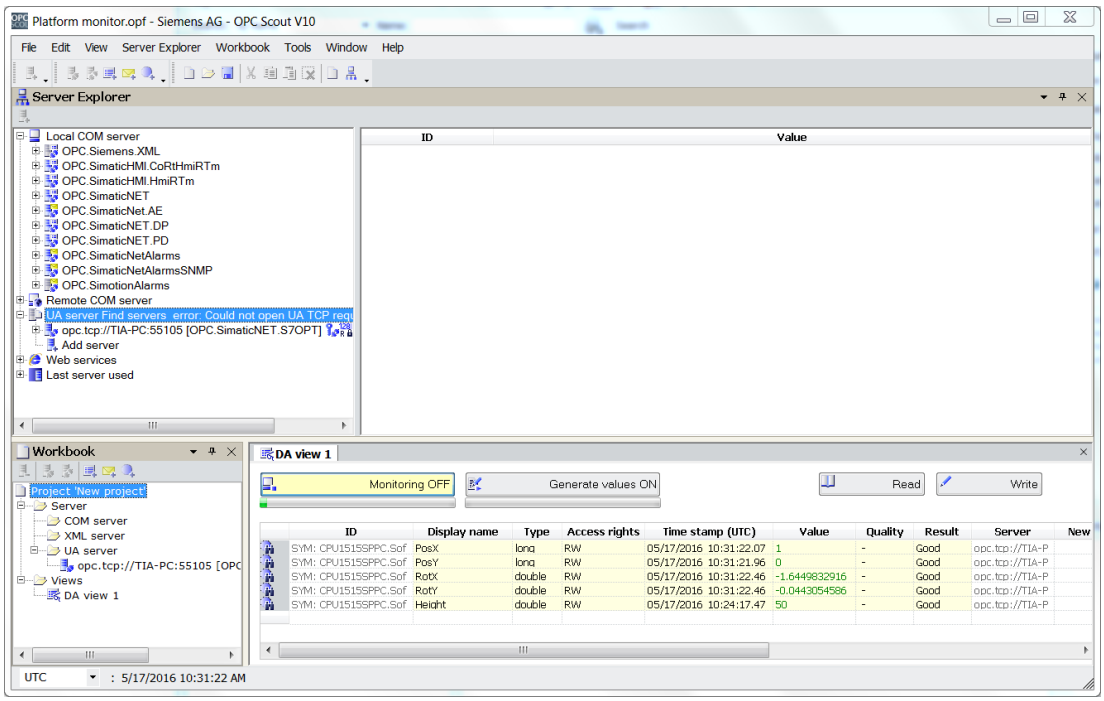


Figure 2.12: OPC Scout V10 with connection to model

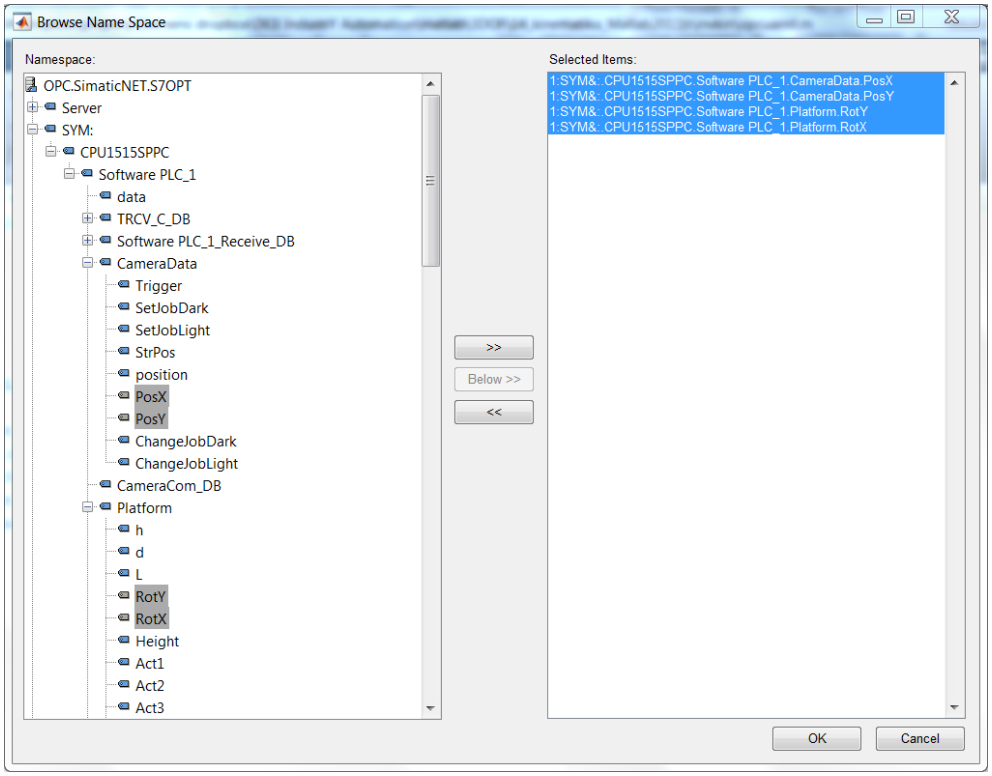


Figure 2.13: For proper functionality of read and write files, use these variables.

the ball running through the maze.

opcwrite.m is file to be used, to write values to model. Variables "RotX", "RotY" and "Height" will be written to model and platform will be positione in those coordinates.

Before using m-files, be sure to properly set OPC server connection and run opcuainit.m .

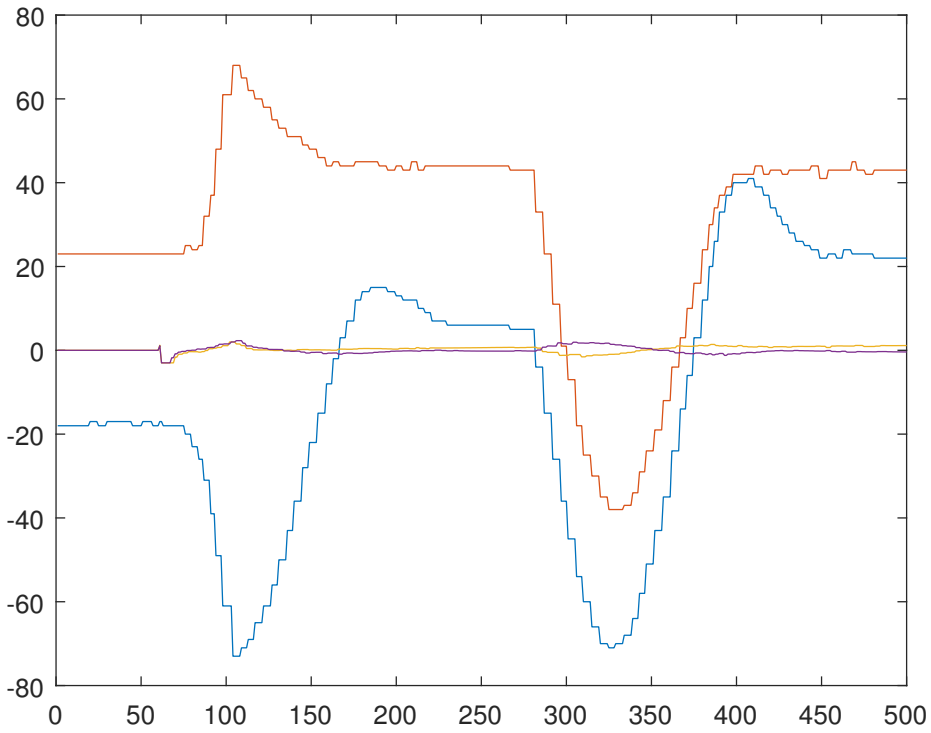


Figure 2.14: Figure with values of rotation of platform (yellow, purple) and position of ball (blue, orange), on X axis is number of sample

3. Operations

This chapter is considered as handbook for operations with model, if you want to start programming go to *Attachment 1 - How to start programming*. Before starting model, please read these pages.
Switching between screens is possible by slide in menu from right hand side of panel. From there you can get into two screens Start - this screen appears in power on of model and User - screen for operating model. Before using HMI wait for start of CPU. Start is indicated by green LED on Open Controller PLC indicator.

3.1 Start screen

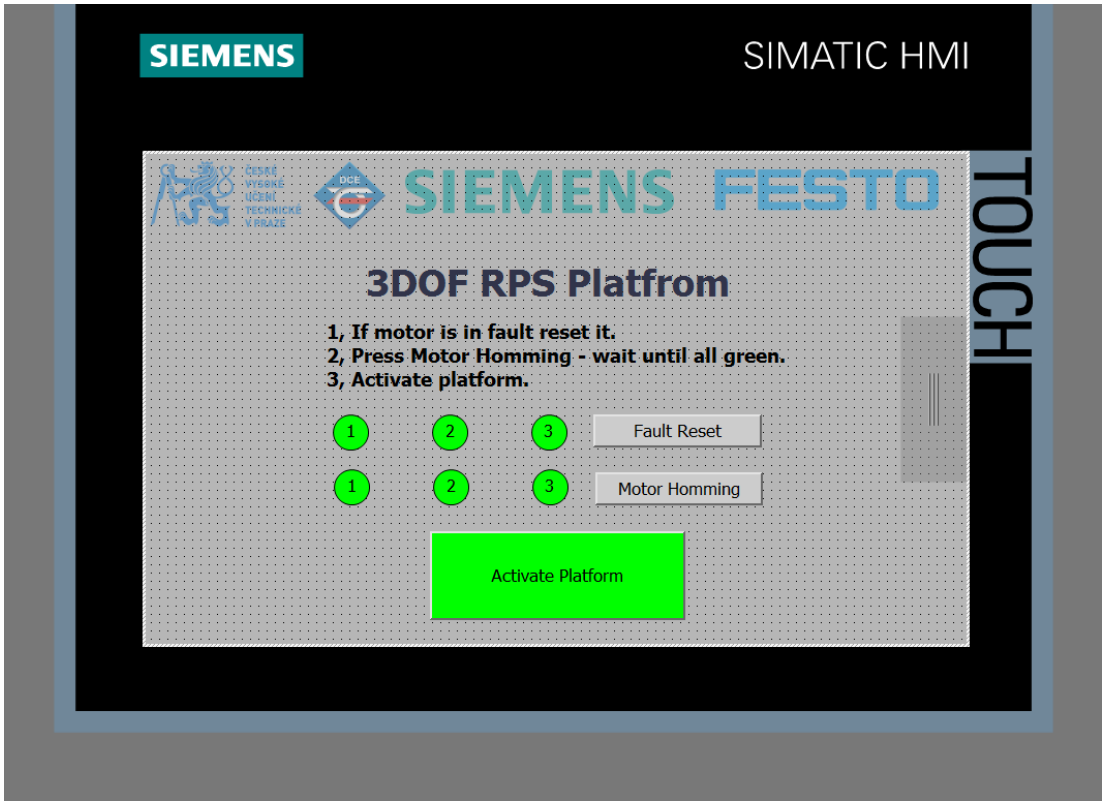


Figure 3.1: Start screen

This screen fig.3.1, appears after power on of system. First you need to homming motors, they need to get reference point to measure length. After all homming indicators are green you can activate platform. Red color of control buttons means, technology is not activated.

3.2 User screen

After startup phase, you can start control the system, screen on fig.3.2 appears.

Left side of screen with bar graphs is indicating position of each of three linear motors. Green target indicates whether they are in fault, if some of them is red, press "Fault Reset" button.

Central part is occupied by three windows, RotX, RotY and Height. This it for direct control of model. Height of platform is from 0 to 90 and it is in mm. Rotation is in range -15 to $+15$ degrees, to get proper angle, lift platform to $50mm$ before rotating it. Above, there are two buttons for starting automatic modes. On right side of buttons are indicators of modes, if there is red indicator follow the instructions written on it to be able to correctly start of mode. If you start mode with red indicator on, system would not be able to reach desired position.

Position data is part with camera data, position of the ball on X and Y axis. Buttons on bottom are for switching camera modes, if model is running in dark, you should press dark button which activate LED light from camera and slightly adjust detector.

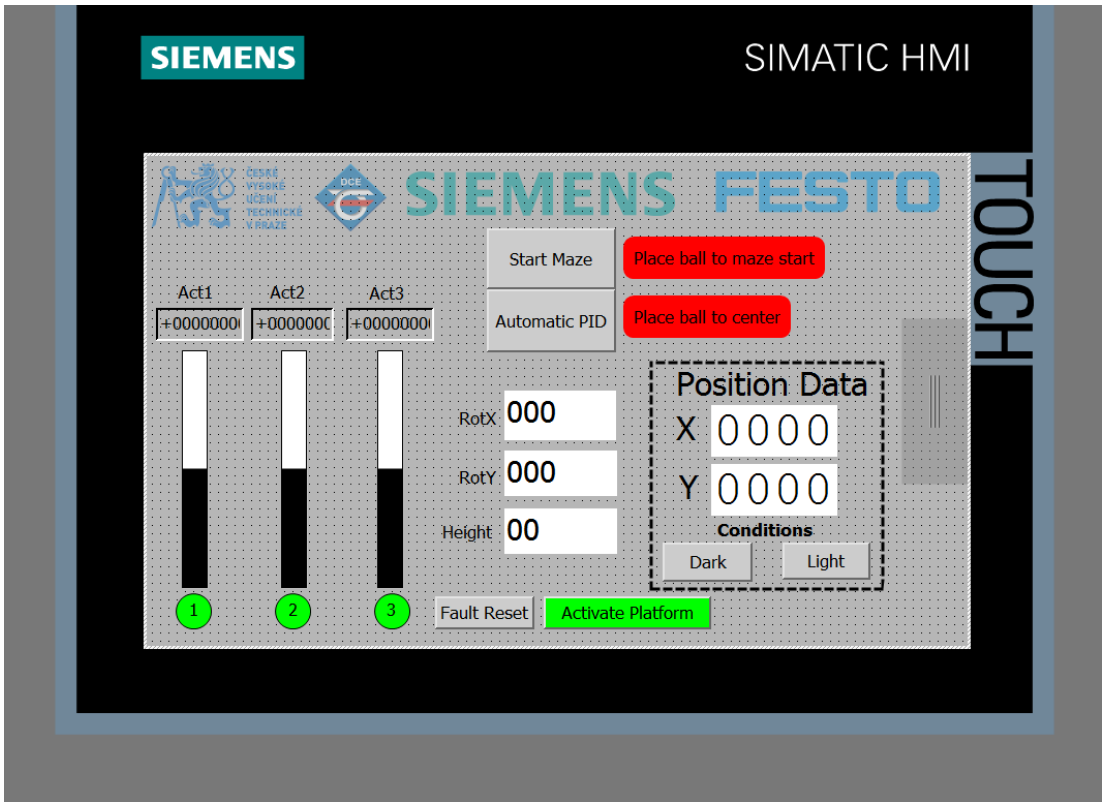


Figure 3.2: User screen

3.3 PID control screen

If you start automatic mode, this screen fig.3.3 appears. On screen is safety button "Manual/STOP" which immediately stops platform. On left, there are similar indicators as on screen 3.2 indicating motor positions. Main part of screen

is trend view. Blue line is setpoint for X and red is X actual position. Black line is setpoint for Y and green is actual position.

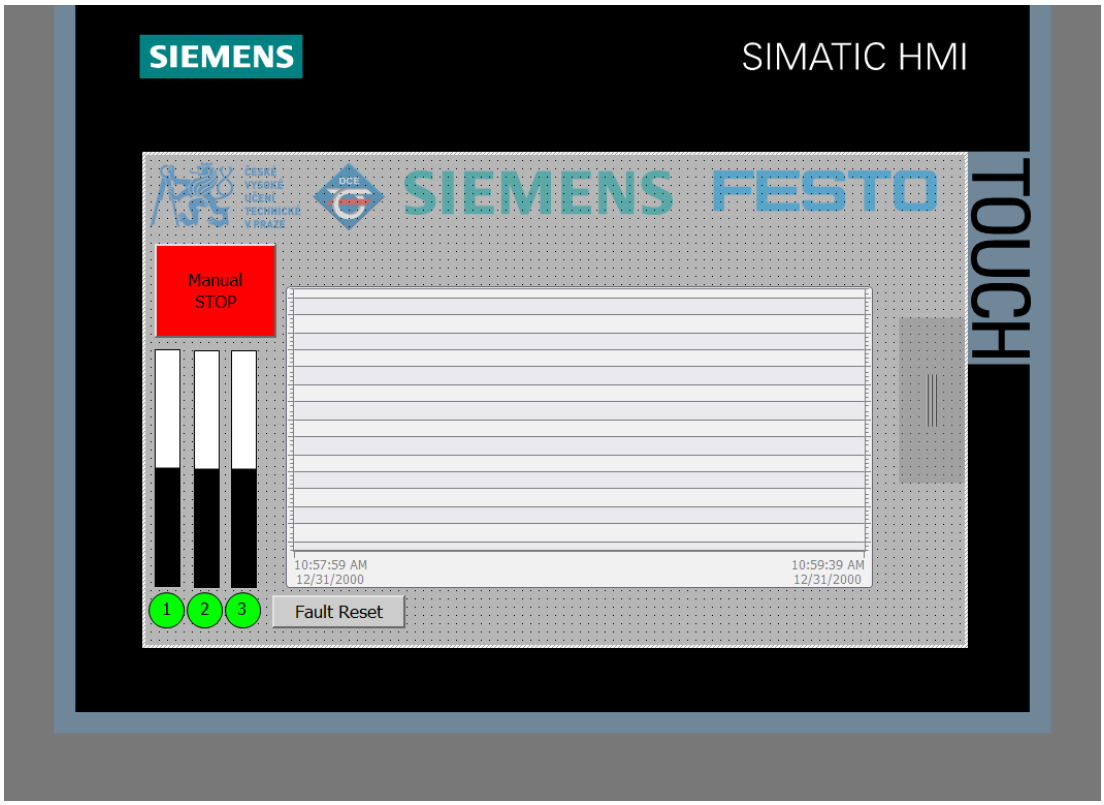


Figure 3.3: PID screen

3.4 Modes of operations

You can switch to different modes of operations on HMI as described above. Default mode is direct control.

3.4.1 Direct control

In this mode, you insert values to platform directly on HMI. It is recommended to lift platform to $50mm$ above launch point to get proper angles. For rotation around X and Y axis do not exceed values of $\pm 15°$. If so, platform angle will not be as desired.

3.4.2 Automatic control

If you launch automatic control, system starts to stabilize ball in the middle of the platform. Camera position $[0, 0]$. Stabilization is done by PID controller for each axis. Make sure, that the ball can reach position at the center of platform, that it doesn't get stuck behind some maze obstacle.

3.4.3 Maze run

Maze run is mode, when system is controlling ball thru maze. Before initiating this mode, place ball in the beginning of the maze. The run is done by moving setpoint for PID controllers along the path thru maze. Because of this, you need to specify for each maze shape different path in PLC program in DB "Maze". After it reaches final position, ball is stabilized there. On Fig3.4 is the structure of the maze, before starting maze mode, place obstacles in same shape as on photo. You can also change path in TIA portal project.

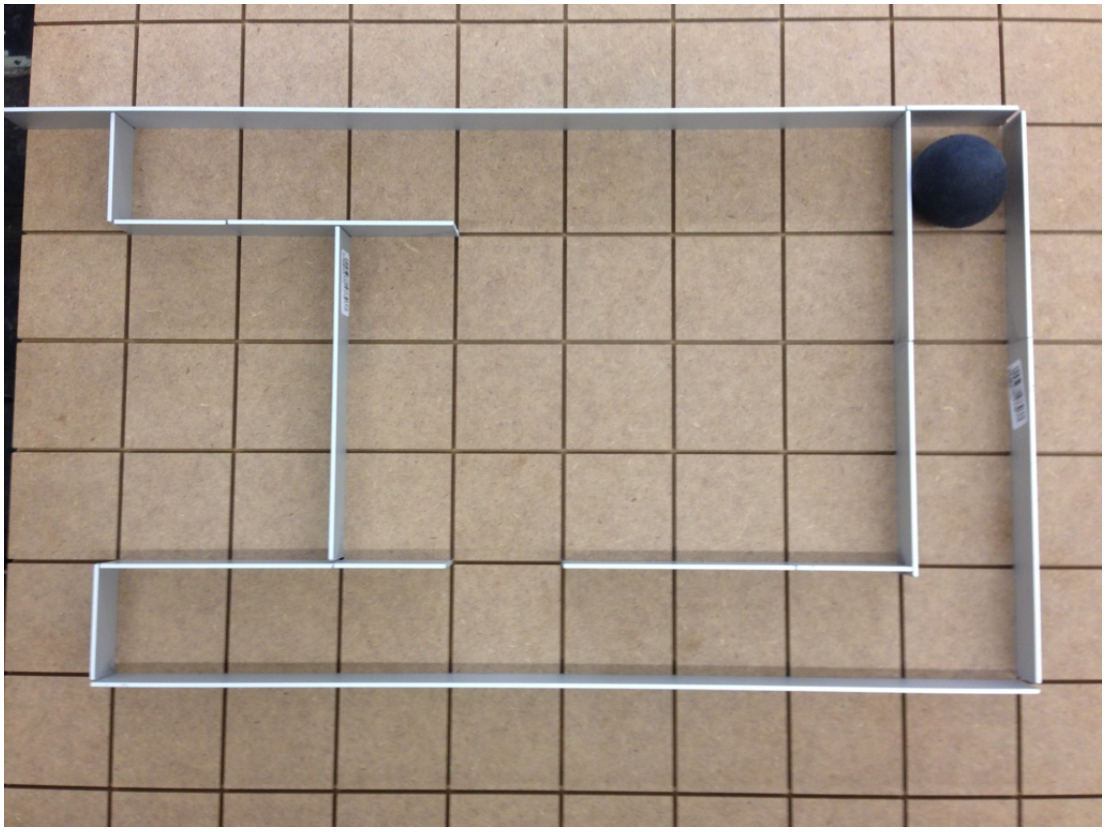


Figure 3.4: Structure of maze with ball on maze start point.

3.5 Remote desktop connection

Because Open Controller is small IPC you can actually install your applications on it and use them in cooperation with software PLC. Thanks to that, I have installed "Festo SBSI vision sensor device manager". Purpose of this is simple, if light conditions changes, during model deployment, you can adjust camera to be able to work in those conditions. Connecting to remote desktop is done by any Windows remote desktop client, system is reachable on following IP:

- IP:192.168.136.215

3.5.1 Visualization on remote desktop

You can use visualization on remote desktop, ratter than on HMI. To do so, WinCC RT V13 is installed. WinCC is launch with start of windows. Control of

model is following same principles as on HMI, screenshot from home screen on Open Controller is on Fig.3.5.

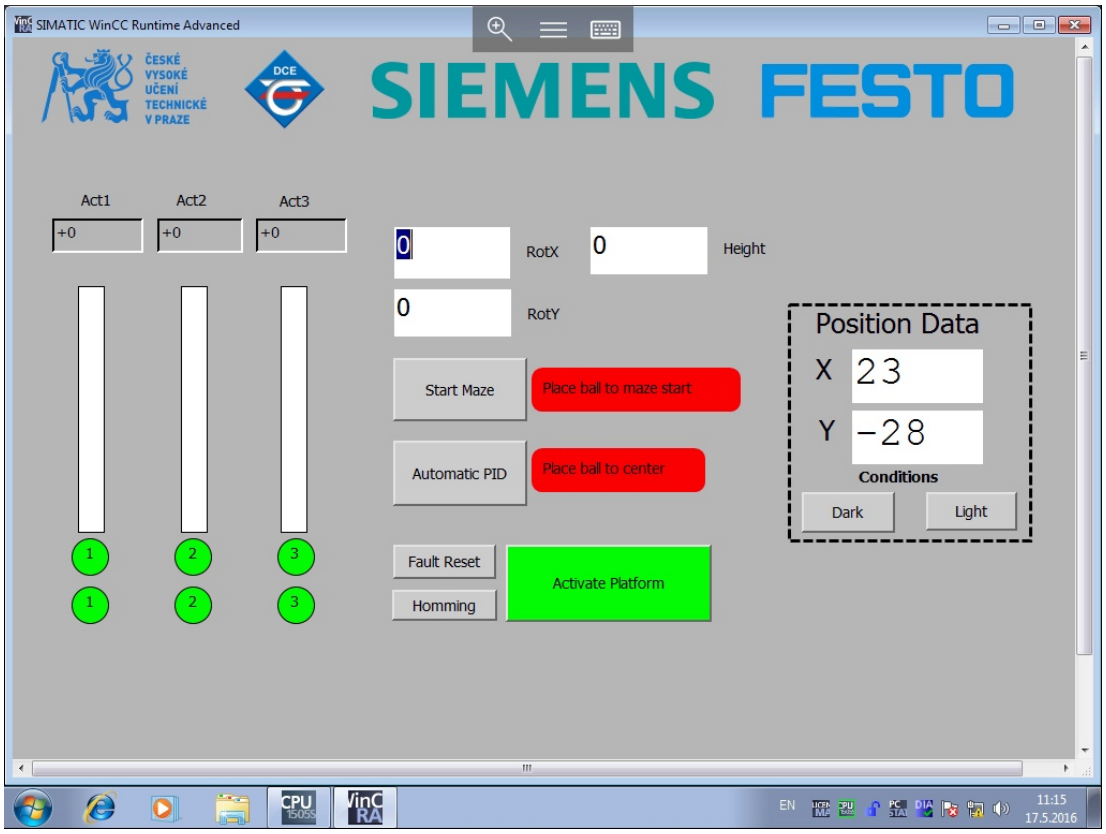


Figure 3.5: Home screen on Open Controller

3.5.2 Error and diagnostic of system

In remote desktop connection, you can also access diagnostic on SW PLC. If the red LED is flashing on front facing side of Open Controller, simply open SW PLC application on remote desktop and you can diagnose the faults.

Platform is moving with little respect to ball position

First, try to switch the light mode on HMI panel section camera, if this doesn't help. Open desktop on Open Controller and launch SBSI Vision Sensor app and check if camera receiving clear image of the ball with plate. You can adjust gain and shutter of camera if necessary.

One or more actuators are not moving

Check if they are all online (symbols on HMI are green), if there is some red one, click on "Fault reset" button. Than press "Activate platform" button.

Model is not responding to HMI commands

Check if PLC is running, if not, switch it on by switch on front of Open Controller. If it is in run mode, put it in Stop mode and Run again, than click on "Activate

platform" button on HMI.

Other problems

If there is a red fault LED flashing, connect to remote desktop and try to find it in diagnostic buffer of SW PLC. Last choice is restarting the system, don't worry about force power off to Windows system, they are running in NVRAM memory.

4. Possible use of model

Model is, as it is required, highly modifiable. Possible use is vast, due to multiple interfaces. Especially into Matlab or any other system using OPC technology.

4.1 Students assignments

On the model, it is possible, to teach students programming routines of PLC technology, thanks to visualization and possible connection of classical IO on distributed peripherals.
Modeling assignment for courses teaching system control and modeling is possible, thanks to connection to Matlab. You can model system and check results within one environment, also, you can tune PID controller and let it control model from Matlab. Model can be simplified just for one axis to standard "ball on beam" assignment.
Final but not last use is, for kinematics assignments, to specify planar manipulator system of three linear motors. Also a finding way through a maze is possible assignment. You can get raw image data from camera to Matlab by OPC server by simple modification of TCP telegrams from camera.

4.2 Future development

4.2.1 Model control

As I found out, my tuned PIDs for stabilization, are not perfect. Due to time delay between camera, PLC and motors, the whole system has big reaction times. One of the possible solutions for this, is to configure predictor and model into PLC. It could definitely improve system behavior.

4.2.2 Communication

During configuration, I used my own Wi-Fi router for communication with model, it is definitely more convenient instead of wired connection to computer. I recommend to use Industrial Wi-Fi router with model, especially for presentation purposes. It definitely looks better, present it on iPad than on standard computer. Also connection to school network through router is better than assigning multiple IP addresses on all devices.
For speedup model reactions, I'm recommending to switch from IO-link communication to standard Modbus TCP protocol, motor encoders can use Modbus TCP as well. Dynamics of whole system should be much better.

4.2.3 Maze

From camera, it is possible to get raw picture data, for smoother maze run, it would be good to program maze pathfinder into PLC and let PLC to do the job instead of replacing obstacles on platform.

4.2.4 Platform

A big problem with the model is with light. Internal light on camera prolongs cycle time, so external light on top of platform could solve this problem.
In mode of ball stabilization, there is a problem with drains for maze obstacles, the ball get stuck on them. So for this assignment it could be better to place smooth surface on platform. Plastic or metal cover could do the job.

4.2.5 Camera

Problem with camera cycle time is mentioned before, I believe that after consultation with expert in this field from Festo co., cycle time of camera can be significantly reduced. Also use of external light for camera is good way for model cycle time reduction.

4.2.6 TIA portal V14

In summer of 2016, a new version of TIA portal V14 going to be released. I highly recommend to recompile and upload software with V14. For HMI panel this will be possible through a remote connection by Sm@rt server and with Simatic.NET V14, it will be possible to install OPC UA server on open controller. All necessary parts for model control will be than integrated in one hardware.

Conclusion

I have created a 3DOF RPS motion platform, as a physical model for teaching purposes of automation and simulation courses. The whole model is implemented on standard industrial hardware and programed with respect to programming guidelines of given systems.
During the testing phase of the model, I found out that the dynamics characteristic of the model is quite high for used hardware. The biggest obstacle is communication between motors and PLC on IO-link bus. I proposed in part 4.2 some adaptations of the model, that can improve whole behavior.
Connection to other systems especially Matlab is also implemented. It depends on faculty staff where the model will be connected to network. Unfortunately Simulink block for OPC UA communication are not yet available, so connection is prepared just for standard Matlab environment.
I have placed all necessary manuals, source codes etc. on an attached CD and I have uploaded them on an open controller connected to model. All materials are therefore simply reachable for future development and use of model. In Appendix 1 is short intro how to start programming model.

Bibliography

[1] Pid algorithm implementation using computer vision. http://electronics.stackexchange.com/questions/16138/pid-algorithm-implementation-using-computer-vision.

[2] Festo. *elektrický válec EPCO-16-100-3P-ST-E katalogový list*, 2015.

[3] Festo. *Ovladače motorů CMMO-ST - manual*, 2016.

[4] Siemens. *ET 200SP Communication module IO-Link Master CM 4xIO-Link(6ES7137-6BD00-0BA0) - manual*, 2015.

[5] Loziska henlich gfsm. https://www.hennlich.cz/produkty/kluzna-pouzdra-a-vedeni-kloubova-loziska-novinky-igus-7159/prirubove-lozisko-gfsm.html.

[6] Siemens, Siemens AG Division Digital Factory Postfach 48 48 90026 NÜRNBERG GERMANY. *ET 200SP Open Controller, Product information on CPU 1515SP PC*, 2015.

[7] Festo. *Kamerová čidla SBSI - manual*, 2015.

[8] Festo. *Vision Sensor User manual SBSI*, 2015.

[9] Nan Zhang Shuo Yang Lingtao Yu, Lixun Zhang. Kinematics simulation and analysis of 3-rps parallel robot on simmechanics. In *International Conference on Information and Automation*, 2010.

[10] Radu Bălan Ciprian-Radu Rad, Milos Manic and Sergiu-Dan Stan. Real time evaluation of inverse kinematics for a 3-rps medical parallel robot usind dspace platform. *IEEE*, 2010.

[11] Zhang Hongyan Ni Tao Zhao Dingxuan, Wei Haiiong. Explicit solution for inverse kinematics of 3-rps parallel link manipulator. 2010.

[12] Hamid Taghirad. *Parallel robots: mechanics and control*. CRC Press, 2013.

[13] Tomas Spacil. Testovaci plosina se tremi stupni volnosti. *Bakalarska prace*, 2013.

[14] Siemens. *SIMATIC NET PC software Commissioning PC Stations - Manual and Quick Start*, 2015.

List of Figures

Attachments

Volume of attached CD.

- CD
  - matlab
    - opcuainit.m
    - opcread.m
    - opcwrite.m
  - TIA
    - modelcomplet.zap13
  - camera
    - light.job
    - dark.job
  - documentation
    - manuals

Attachment 1 - How to start

In this attachment, I briefly described, how you can start programming model. All mentioned documents are in documentation folder on CD and uploaded on open controller.

Opening project in TIA portal

On CD you find in folder TIA portal file called complet.zap13, it is archive file type for TIA portal V13. You can open in by launch TIA portal, in Project view click on "Project"$\longrightarrow$"Retrieve...". This will unpack project into your TIA portal and you are ready to start.

Connecting to model

To connect to model, use ethernet port on Scalance swich next to open controller. Set your IP to 192.168.136.200 and subnet to 255.255.255.0. Also, you can let TIA portal to set your IP automatically, but if you want to use OPC server on your PC, use manual set.

Description of parts

Siemens hardware

If you are looking for information about Siemens hardware, go to `support.industry.siemens.com`, here you can find all manuals about hardware.
Also, on CD or on open controller on folder "documentation" are stored basic presentation about open controller functions. In folder "manuals" are actual manuals for all Siemens parts.

Festo hardware

All information about Festo hardware can be found on `festo.com/net/cs_cz/SupportPortal/default.aspx` also, in folder manuals are necessary manuals and handbook to start working with hardware.

List of part used on model

If you will be looking for information about parts on model, in this table are order numbers and full names that will identify parts correctly.

Part	Order number
Siemens	
ET200SP Open Controller	6ES7677-2AA41-0FB0
SCALANCE x204 IRT switch	6GK5204-0BA00-2BA3
ET200SP interface module	6ES7155-6AU00-0CN0
ET200SP IO-link master	6ES7137-6BD00-0BA0
TP700 Comfort HMI	6AV2124-0GC01-0AX0
Festo	
SBSI vision sensor	SBSI-Q-R3B-F6-W
EPCO linear motor	EPCO-16-100-3P-ST-E
CMMO motor encoder	CMMO-ST-C5-1-LKP

List of parts used on model.