Aalto University, School of Electrical Engineering
Automation and Electrical Engineering (AEE) Master's Programme
ELEC-E8004 Project work course
Year 2019

Final Report

Project #30
Intelligent Mobile Production Platform

Date: 30.5.2019

Markus Roni
Konsta Leino
Roman Rumiantcev
Mika Kuusisto
Vuong Vo

Information page

Students
Markus Roni
Konsta Leino
Mika Kuusisto
Roman Rumiantcev
Vuong Vo

Project manager
Markus Roni

Official instructor
Valeriy Vyatkin

Other advisors
Udayanto Atmojo
Jan Blech
Vladimir Kuliaev

Starting date
10.1.2019

Completion date
30.5.2019

Approval
The Instructor has accepted the final version of this document
Date: 2.6.2019

Abstract

The project was a part of further development of the Aalto Factory of the Future (AFoF) by enabling "agile" production, based on so-called product-centric manufacturing paradigm, where the overall production processes are determined based on the requirements by the customer. (V. Vyatkin 2019, p.1). For this purpose, professor Valeriy Vyatkin with his team designed a mobile production platform by mounting ABB IRB 14000 Yumi robot to SEIT 100 AVG from Milvus Robotics. Delivery of the SEIT included also a lift where ABB robot was later mounted. The task for the project team was to integrate the parts of the MPP together and enable the robot to be one part of the factory cell. Factory side interface was the ethernet network and IceBlock IoT smart controllers including I/O-interface.

As a result of the project, the team was able to create the first demo of a product that widens the usability of the traditional industrial robot. Normally robots are stationed in one place and the production facilities are brought to the robot. This solution creates flexibility for the production when the place of the robot is easy to change, and no personal or other equipment are needed for moving it around the factory floor. The platform provides power for movement and also for the two armed robot as well. At the same time MPP is able to navigate autonomously around the factory floor. Choosing ABB YuMi robot makes it possible to use the MPP in direct collaboration with factory personnel without fencing the work area. The achieved solution is possible to integrate to already existing wifi network in the factory without any major infrastructure changes, when even the charging of the MPP is possible with normally used outlet.

The project creates a great base for further research and development of the MPP and agile production. The project team was able to prove the possibility of combining these normally separately used devices and to prove their performance abilities. Work included also the research of limitations and finding opportunities for future development, which will be handed over to Aalto University research team and everyone who are interested in the field of study.

Table of Contents

1. Introduction

Aalto University is participating in global development and research of new trends in automation and manufacturing. One part of that work is to create new agile manufacturing methods to achieve the so-called fourth industrial revolution. With this goal in mind, Professor Vyatkin wanted to research opportunities of combining an autonomous mobile platform and a industrial robot.

The project team was build up from the Aalto University students participating in masters level project work course. The five-person team was joined together from the larger group according to the participants own interests. The project started at the beginning of January 2019 and ended at the end of May 2019 having five months working time between other studies. This report adds up all the results of the project around the idea of Mobile Production Platform.

2. Objective

This project aims development of a mobile production platform (MPP) by mounting the ABB IRB 14000 Yumi robot into an autonomous mobile robot (AMR) SEIT 100 of Milvus Robotics. The MPP should be able to perform its operations at various locations of a flexible production line and meanwhile cooperate and interoperate with the rest of the production units and robots in the factory floor.

The platform consists of a Yumi robot, a lifting mechanism and SEIT 100. The MPP allows Yumi robot perform the full range of its operation and adjust its behavior based on the actual position of SEIT and height of the lifting mechanism. To do this, Yumi and SEIT should be able to communicate to each other to finish the tasks which require a very fine positioning. The MPP also must be capable to interact wirelessly with the distributed control infrastructure of the other production units. During its production operation, MPP must ensure the safety for other objects and workers from the collision with it. Due to the fact that YuMi can only be powered from the battery of SEIT, an energy management is required to monitor the energy state of the MPP and to predict the time of its operation, as well as to plan for recharging the battery.

In the end of the project, the MPP operation will be demonstrated in a scenario where will complete a certain assembly task on one working station then moving to a different location to perform another task. Iceblock will be used to communicate with MPP to request the target location. In addition, Iceblock input/output will be connected to switches, sensors or lighting control to improve the demonstration scenario. If possible, the MPP can be integrated with the EnAS assembly system from project group 29 where they can cooperate and perform a small scale flexible intelligent manufacturing.

Yumi has to be able to communicate with the Java program translating product description into assembly operations. Yumi programming is done by Final thesis student same time with this project ends the end of February. This project is expected to take over and use results of the master thesis project.

3. Project plan

To reach the objectives and expectations, defined in the project plan, milestones had to be defined. Also, to each milestone there were several tasks to be done. This made working structured and progress was easier to follow.

The most important milestones were to finish the project plan, get the SEIT 100 delivery, mounting of the ABB Yumi robot, communications between Yumi and SEIT 100, communications between IceBlock and MPP, Yumi and SEIT 100 programming, business aspects and the final report. Rest can be found from the project plan.

In the project plan were also specified how should the MPP work, so that the project can be defined as a successful project. ABB Yumi robot should be able to perform at full range of its operation based

on the position of SEIT 100. The platform should be able to wirelessly communicate with other production units. Software is tested via testing scenarios such as assembly tasks with difficulties or obstacles.

4. Communications

Communication between different robots in the Mobile production platform and factory were implemented with a wireless local network. Main components of communication network were wireless router, wifi-usb dongle and some ethernet cable. Devices connected to the communication network were ABB Yumi robot, Milvus SEIT100 robot, Milvus fleet management system, PC and IceBlock PLC.

ABB Yumi had several different communication options in its main computer which were digital inputs and outputs, DeviceNet, PROFIBUS, USB and four ethernet ports. For this project ethernet ports were chosen and differences between these four port had to be researched. LAN 1 and LAN 2 are private ports and they are not capable to connect communication networks. They may only be used to add additional devices to Yumi itself, such as camera for integrated vision. LAN 3 is an isolated port which may be used for connecting device to industrial network. WAN port is an public port and it can be used to connect the device to the communication network.

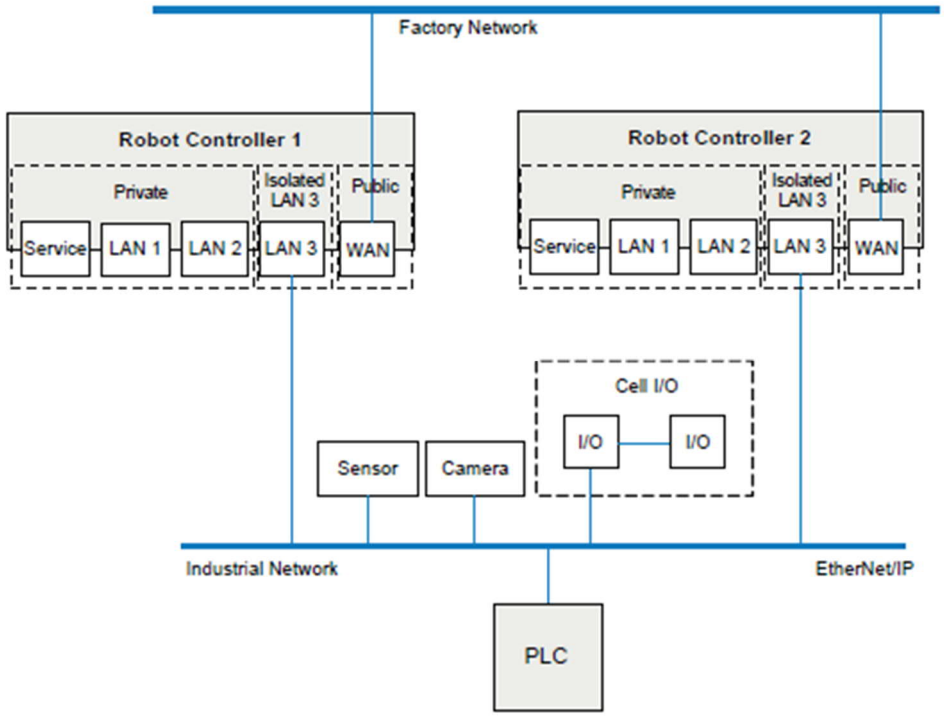


Figure 1. ABB Yumi ethernet ports.

To be able to connect ABB Yumi via wireless network, wifi-usb dongle needs to be installed to the Yumi's WAN port and Yumi's USB port can be used to power the dongle. Before connecting the dongle needs to be configured to the local network with a PC. Configuration is carried out by connecting dongle to the PC and accessing its preferences via default gateway. From the dongle's preferences wireless networks SSID, security settings and passwords can be set. After the dongle is configured it can be removed from the PC and connected to the Yumi. Connections can be tested with RobotStudio software after the Yumi and the dongle is powered on. Yumi's IP address should be visible in the "Recent Controllers" section.

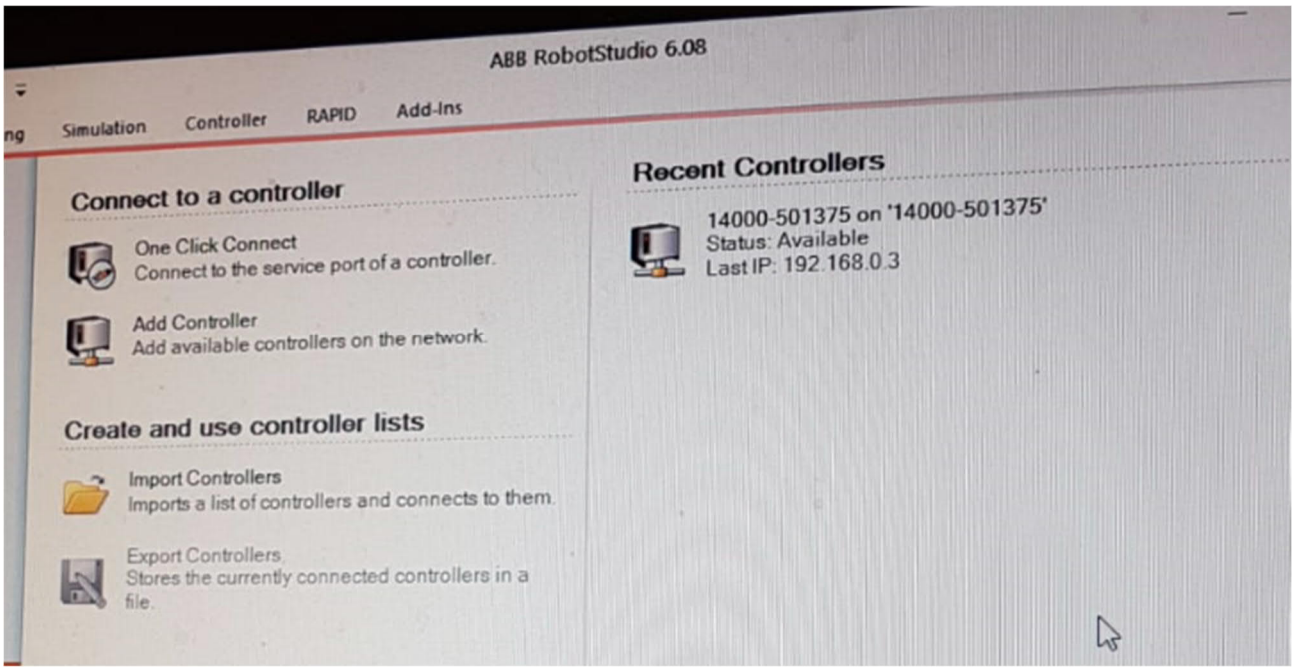


Figure 2. RobotStudio view when ABB Yumi is connected.

Milvus SEIT100 was connected via wireless network and fleet manager system was connected via ethernet cable to the router. IP addresses of these two devices were static: 192.168.7.3 for SEIT100 and 192.168.7.2 for fleet manager system. Fleet manager system were accessed by typing it's IP address to a web browser in a PC connected to the local network. IceBlock was connected via wireless network to the router with a static IP address 192.168.0.30.

Communication architecture between different devices were OPC and OPC UA. All of the devices used OPC UA architecture for communication except ABB Yumi which supported only OPC architecture. To be able to communicate with these two slightly different architectures special gateway program were needed. We used OPC Gateway program for this and we ran it with the same PC which we used for running the OPC UA and ABB OPC servers.

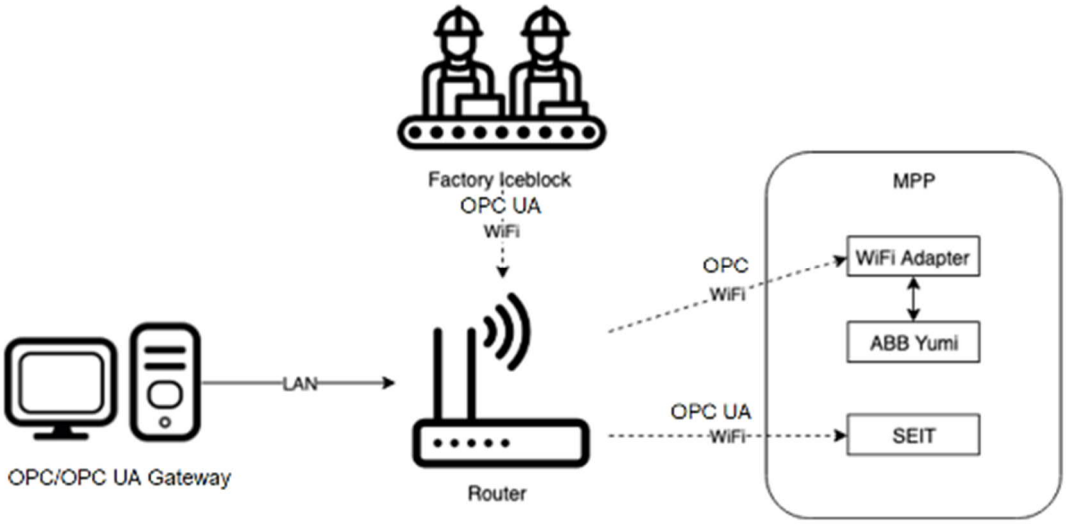


Figure 3. Communication network.

5. SEIT 100

5.1. SEIT 100 features

SEIT 100 is an autonomous mobile robot manufactured by Milvus Robotic. The idea behind the creation of SEIT was to provide people with an opportunity to safely and efficiently carry products and materials around warehouses and factories. It was designed to be loaded with products on it, or simply pull a cart full of load (Milvus Robotics, nd). SEIT 100 has the capacity of 100 kilograms payloads and can be installed with external components such as conveyor, lift for loading. In this project, SEIT 100 was delivered with a built-in lifting mechanism to help to adjust the height of Yumi robot.

SEIT transports materials autonomously by means of sensory capabilities and therefore without any additional infrastructure. SEIT's navigation does not require any physical markers such as magnets, beacons, wires or tapes. SEIT uses LIDAR, laser scanner and cameras to create the mapping of the environment, to plan the dynamic path and to determine obstacles. A remote server is used to help the user to access all the functionalities of SEIT. Robot monitoring, mapping, and missions are observed and controlled through this server. Figure 4 and Figure 5 below show the specification of SEIT 100. More information can be retrieved from SEIT's manuals.

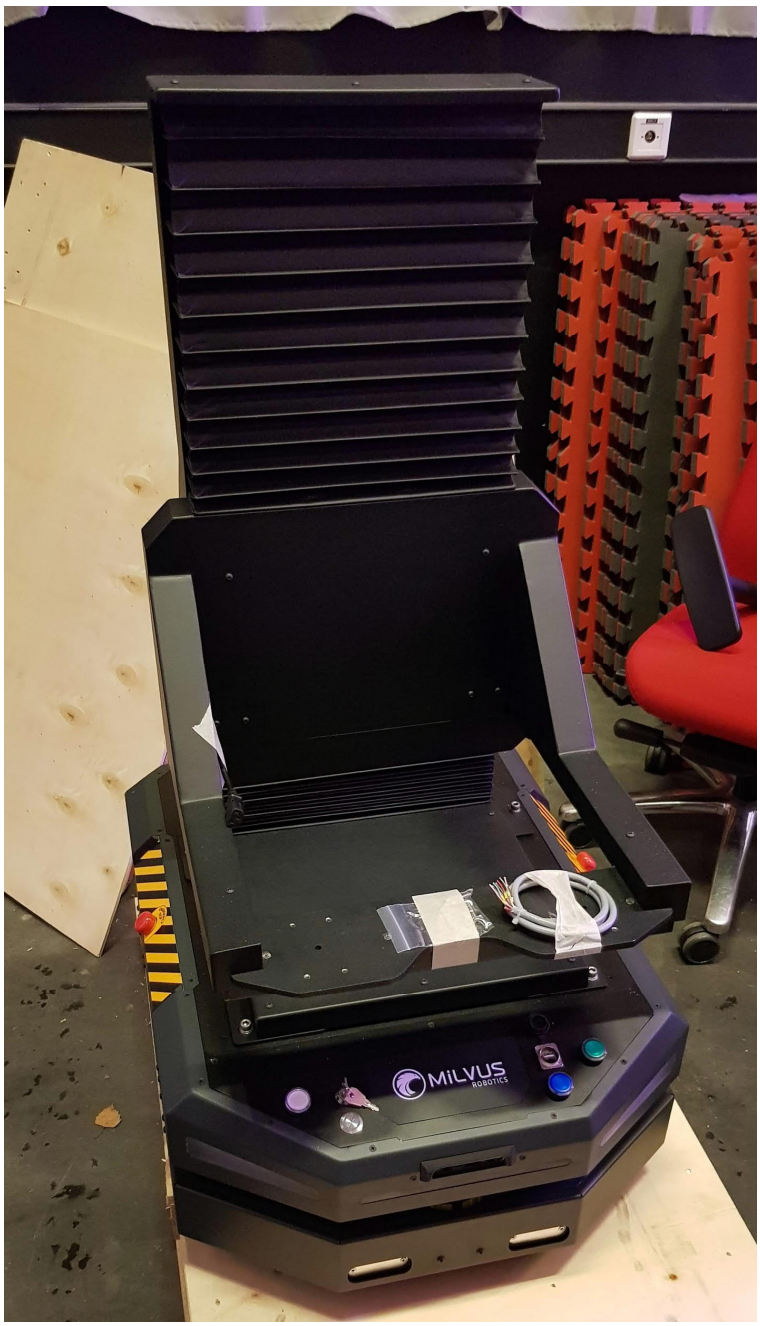


Figure 3. SEIT 100 with lifting mechanism.

SEIT100

KEY SPECS:

PAYLOAD:	100 KG
LOAD TYPE:	APPLICATION SPECIFIC - ON TOP LOADING, CONVEYOR, LIFT
MAX SPEED:	1.5 M/S
DIMENSIONS (LXWXH):	890 MM X 650 MM X 297 MM
NAVIGATION SYSTEM:	NATURAL NAVIGATION
COMMUNICATION:	WIFI
POSITIONING ACCURACY:	±1" (2.54 CM) LONGITUDINAL, ± 1" (2.54 CM) LATITUDINAL IN STANDS
INTERFACE:	WEB BASED SOFTWARE INTERFACE
SAFETY FEATURES:	SAFETY LIDAR, EMERGENCY STOP BUTTONS, INDICATOR LIGHT BARS

Figure 4. SEIT 100 specification (Milvus Robotics, nd).

SEIT 100 has built-in WiFi module and several other interfaces such as one USB port, 9 pins digital output and input, and Auxillary 12-pin connector. The USB port can be used to connect to wired and wireless USB dongles or Bluetooth port of the remote controller. Besides of SEIT100 itself, it was also delivered with charging station an accessory box. Figure 6 and Figure 7 show all devices and accessories of SEIT 100.

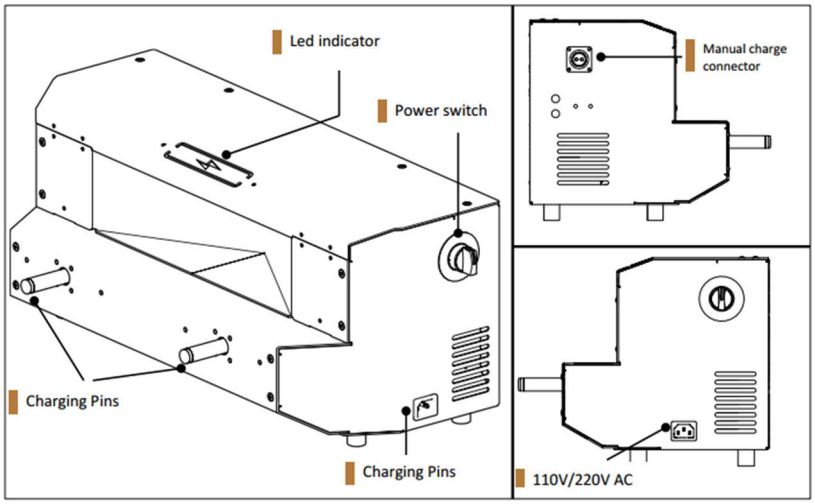


Figure 5. SEIT 100 charging station (Milvus robotics, nd)

Figure 6. SEIT 100's server and accessories (Milvus robotics, nd)

5.2. Network configuration for SEIT100 and its server

SEIT 100 is monitored, controlled and programmed by a remote server, so-called Fleet management server which is accessed via a web-based server from user computers. To establish the communication between the robot, the server and user computers (PCs), it is necessary to have a network which is connected to all those devices. In this project, we created a personal wireless network by a WiFi router. To obtain the connections, there was a procedure to configure the network. Network Manager is the application developed by Milvus robotics which is used to manage the network configuration of the robot and the server. The latest version can be downloaded from the link below or from our project google drive https://www.dropbox.com/sh/rncgjxzuomezqxz/AACOkARO25namKHRCdxwEJDea?dl=0

There are two ways to establish communication among the robot, the server and user PC: wired and wireless connection. In this project, it is reasonable for the robot to connect wirelessly to the network, therefore, it can freely move on the floor. However, to avoid disconnection between the server and the network, it was recommended by Milvus to connect the server to the router via ethernet cable (through LAN ports, not WAN ports). Therefore, there were three main procedures of network configuration: the wireless network for the robot, the wired network for server and host entries for both server and robot.

5.2.1. IP configuration of Server, Robot and host PC

The server and the robot have their own static IP address when the host PC connects to them via wired USB dongle. The wired USB dongle was delivered in the accessory box. On this USB dongle, the static IP address of the server and the robot are noted as figure 8. These static IP addresses are for the wired connection between the host PC and the agents (robot and server). As can be seen from figure 8, the wired static IP address of the server is 192.168.7.2 and the IP address of robot is 192.168.7.3, therefore, the user PC's IP address also need to be manually changed so that they can recognize each

other. The PC's static IP address can be in the 192.168.7.1/24 poll, and it has to be different to the robot and server IP address (cannot be 192.168.7.2 or 192.168.7.3). An example of a network configuration for the user PC is shown by figure 9.

Figure 7. Wired USB dongle and the static IP addresses labelled for SEIT100 and its server

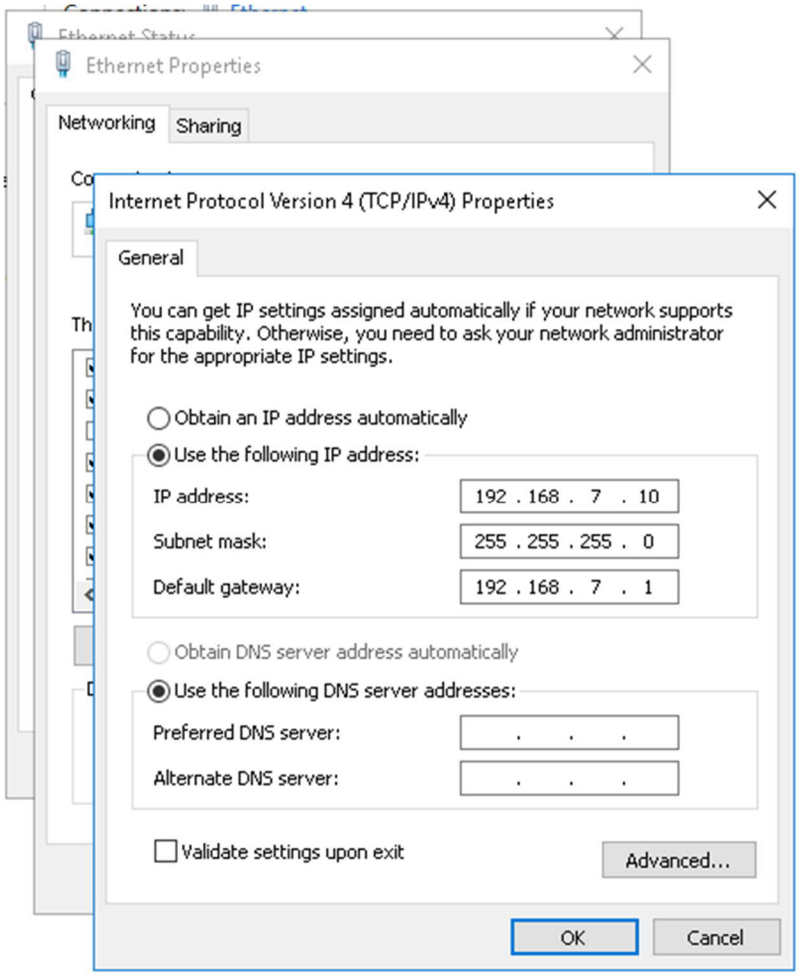


Figure 8. Example of static IP configuration applied to user PC.

Network Manager has no support for IPv6 Addressing yet. In this project, IPv4 Addressing was used.

Figure 9. Wired connection of network configuration for server.

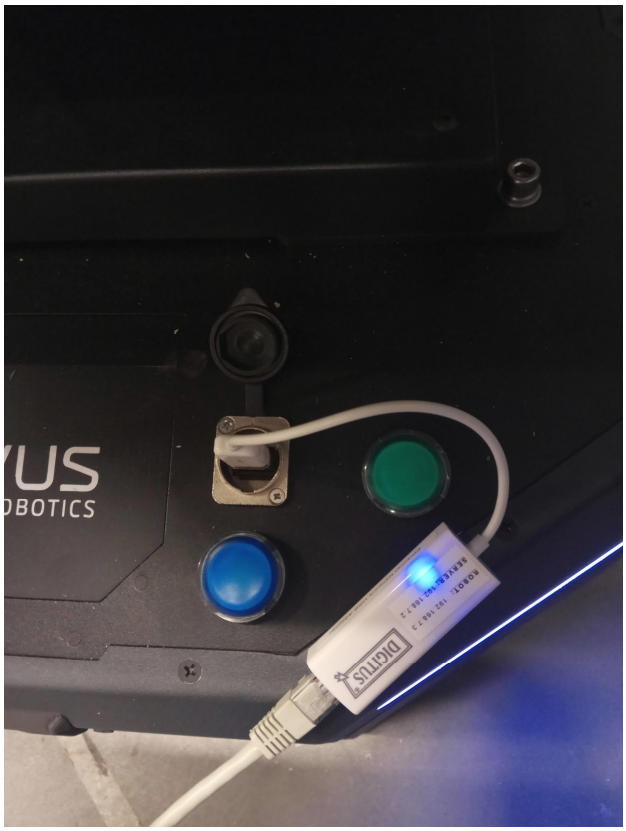


Figure 10. Wired connection of network configuration for robot

After changing the static IP address of the PC, the wired connection between the user PC and the agents were set up. An agent was configured at each time. There is no need to connect them at the same time to the user PC. One end of the ethernet cable was connected to the user PC and the other

end was connected to the wired USB dongle. The other end of the dongle connected to a USB port of the server or the robot. To test the connection, Command Prompt window in the user PC could be used to ping the IP address of the server.

After the connection was established, Milvus Robotics Network Manager application was launched in the user PC. To access the network management of the server, the server's IP address was entered in the welcome page. To access the network management of the robot, the robot's IP address was entered.

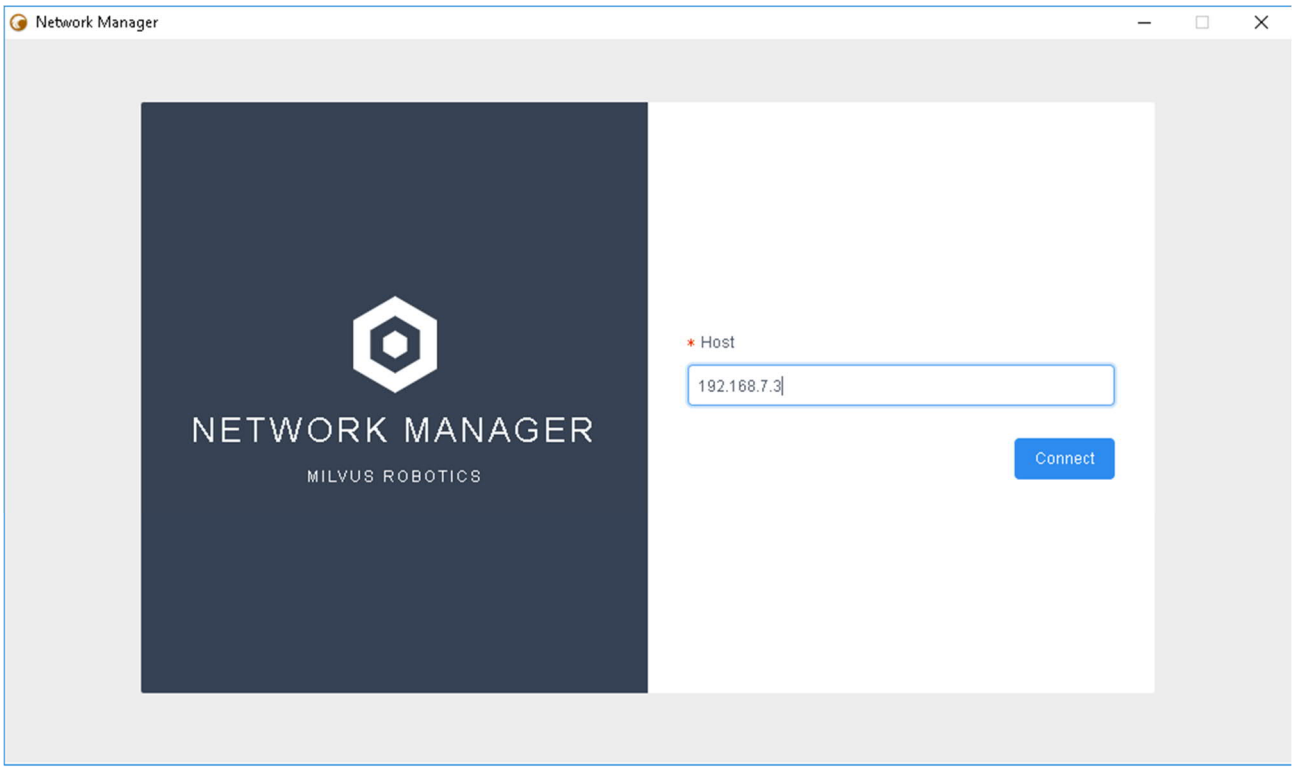


Figure 11. Welcome page of Network Manager

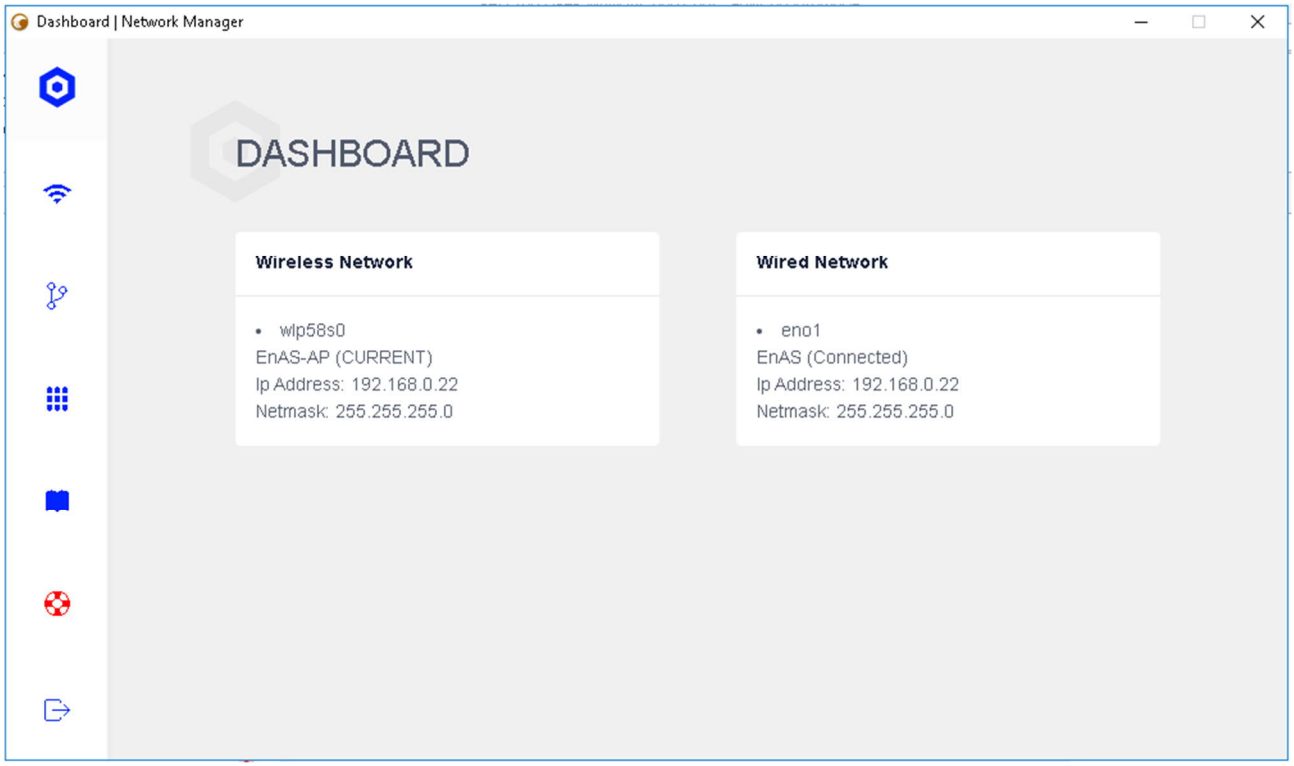


Figure 12. Dashboard of the network manager of the server.

5.2.2. Wired network configuration for the server

The wired network configuration of the server was done in the wired section of Network Manager of the server. To connect the server to the network, a new wired network need to be added. The information of the network configuration was entered as shown in figure 14..

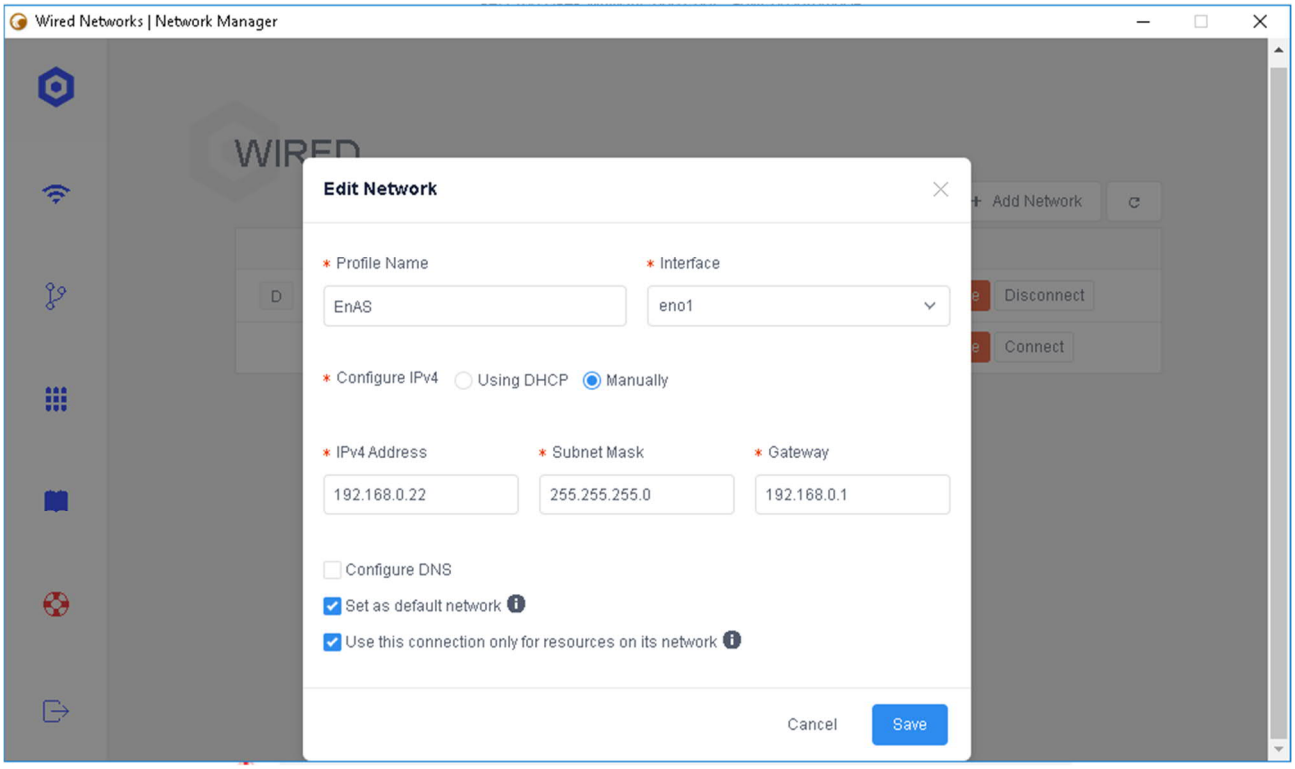


Figure 13. Router network information in the wired network configuration.

Note: In this project, we used a private network which has the configuration as:

Network SSID name: EnAs

Password: aic3ltu123

Router address: 192.168.0.1

Subnet Mask: 255.255.255.0

Gateway: 192.180.0.1

To connect to this network, the IP address of the server, the robot and user PC was set as:

IPv4 address for server: 192.168.0.22

IPv4 address for robot: 192.168.0.21

IPv4 address for PC: 192.168.0.2/20 (selected from 2 to 20)

5.2.3. Wireless network configuration for the robot

The wireless network configuration of the robot was done in the wireless section of Network Manager of the robot. A wireless network was created by pressing “Add Network”. The first step was to select the network interface. In this project, “wlo1” was selected. The second step was to select the wireless network based on SSID. Scanned results for the selected network would be shown in the list. If the target network was hidden network, “Join Other Network” was pressed. The third step was to configure the network. The network information was entered as shown in figure 15 and figure 16.

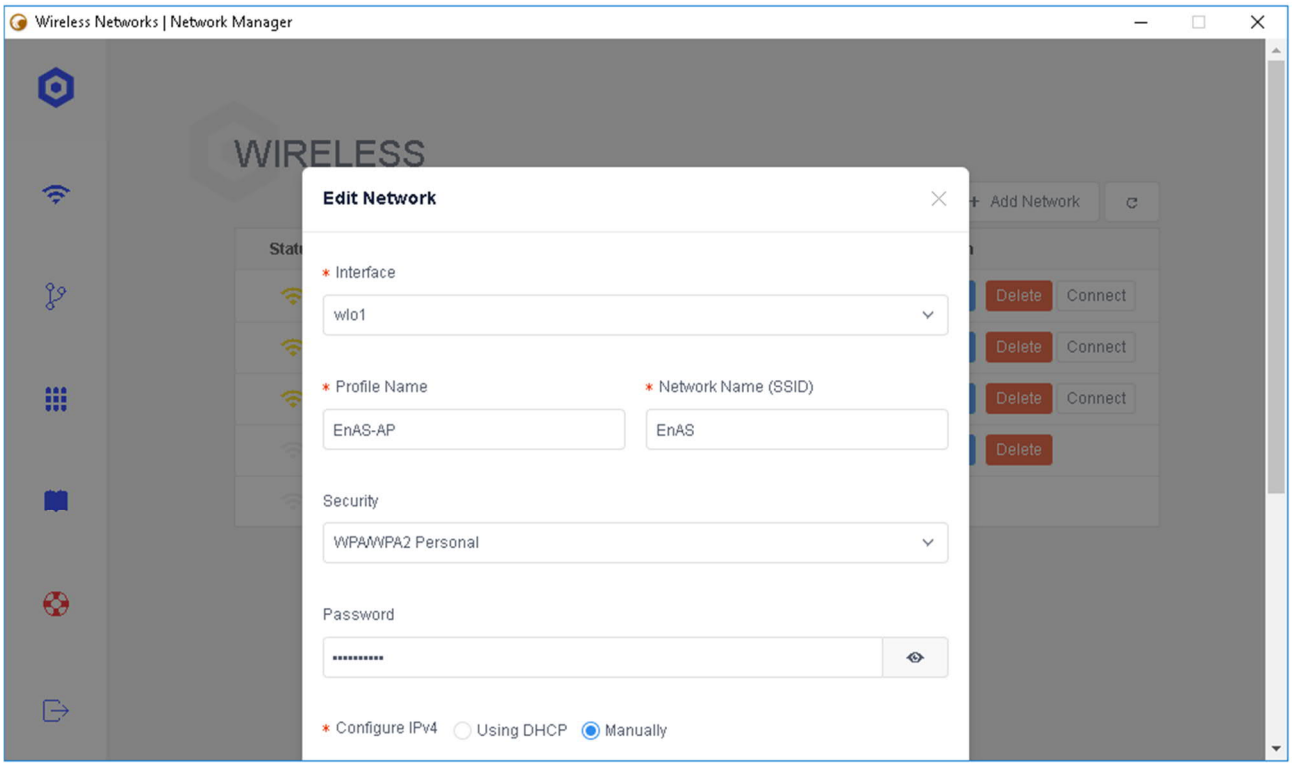


Figure 14. Router network information in the wireless network configuration of the robot

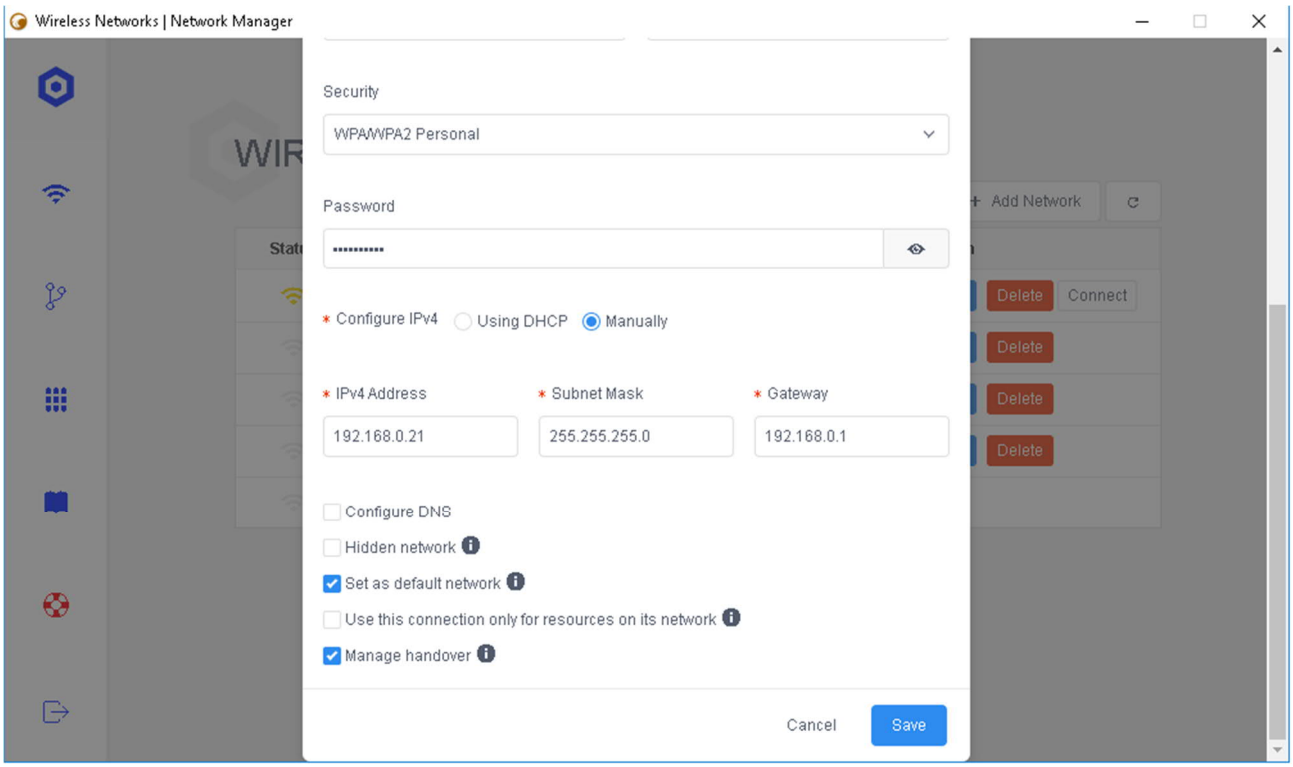


Figure 15. Router network information in the wireless network configuration of the robot

5.2.4. Host entry configuration for server and robot

Host entries are needed to set up the communication between the server and robots. The communication is provided by name resolution protocol. In order to name resolution work, each agent should have specific host entries in their database. The host entries can be added or modified in the Hosts section. Table 1 and table 2 show which entries should be in robot and server agents database. Figure 17 and figure 18 show the host entries which were added to the robot and server database.

IP Address for Host Entry	Hostname for Host Entry
`127.0.0.1`	**Description:** Lowercase serial number of current robot. **Example**: `rXXXXXXXbX`
`Static IPv4 address assigned to the Current Robot`	**Description:** Robots' serial number in lowercase concatenated with "`.external`" word **Example**: `rXXXXXXXbX.external`
`Static IPv4 address assigned to the Server`	**Description:** Lowercase serial number of server **Example**: `sXXXXXXXXX`
`Static IPv4 address assigned to the Server`	**Value:** `ntp-server`

Table 1. Required host entries for robots.

IP Address for Host Entry	Hostname for Host Entry
`127.0.0.1`	**Description:** Lowercase serial number of server. **Example**: `sXXXXXXXXX`
Static IPv4 address assigned to the Server	**Description:** Servers' serial number in lowercase concatenated with "`.external`" word **Example**: `rXXXXXXXbX.external`
Static IPv4 address assigned to the each robot	**Description:** Lowercase serial number of robot. There should be an entry for every robot in operation. **Example**: `rXXXXXXXXX`

Table 2. Required host entries for server.

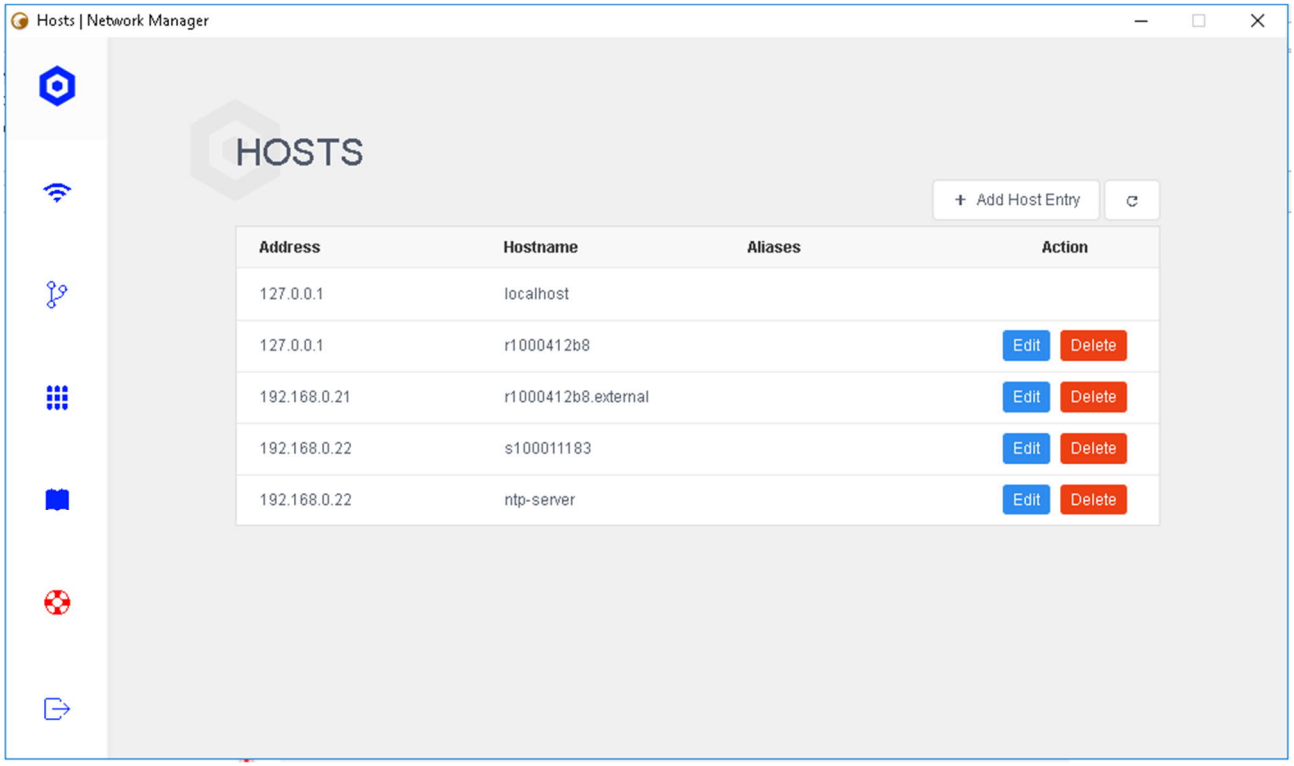


Figure 16. Host entries of the robot.

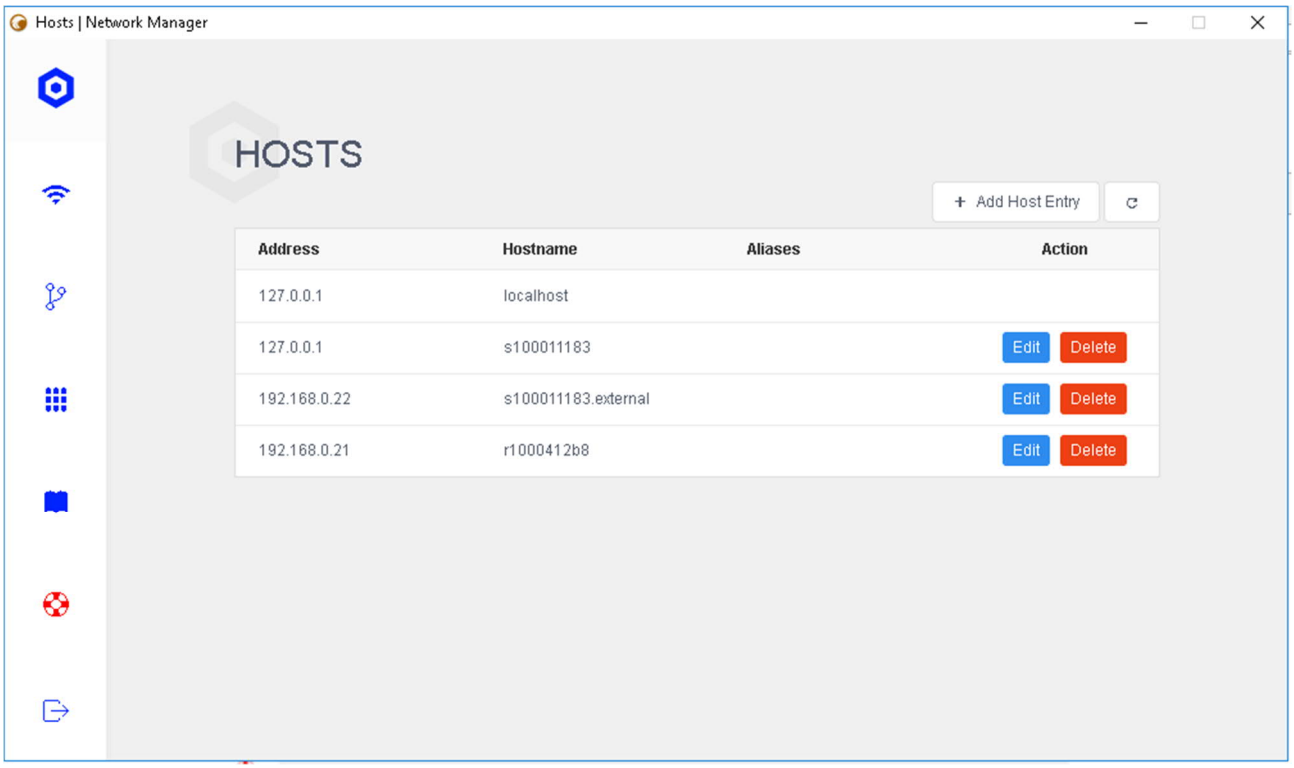


Figure 17. Host entries of the server.

After the network configuration processes were done, a network which carries the communication among SEIT100, the server and user PC was established. To activate the connection to the network, “connect” buttons were pressed in the wired network list of the server and in the wireless network list of the robot. The status went to the connected after this action.

Note: After the configuration, unplug the Ethernet cable from the user PC and take out the USB dongle. The user PC and robot connected wirelessly to the network and the server connected wired to the network via ethernet cable to the router. The IPv4 address of the user PC also needs to be changed to 192.168.0.2/20 to be able to connect to the wireless network.

5.3. Maps and Missions

5.3.1. Fleet Management web-based server

To access the fleet management server, a browser was opened and navigated to http://IP_ADDRESS_OF_THE_SERVER:8080. In this project, the IP address of the server when connected to the network was 192.168.0.22. Therefore, the navigated address was http://192.168.0.22:8080

To sign in to the server, enter the user name as ADMIN and password as ADMIN. After sign in, the fleet management server is accessed. Figure 19 shows the user interface of the web-based server.

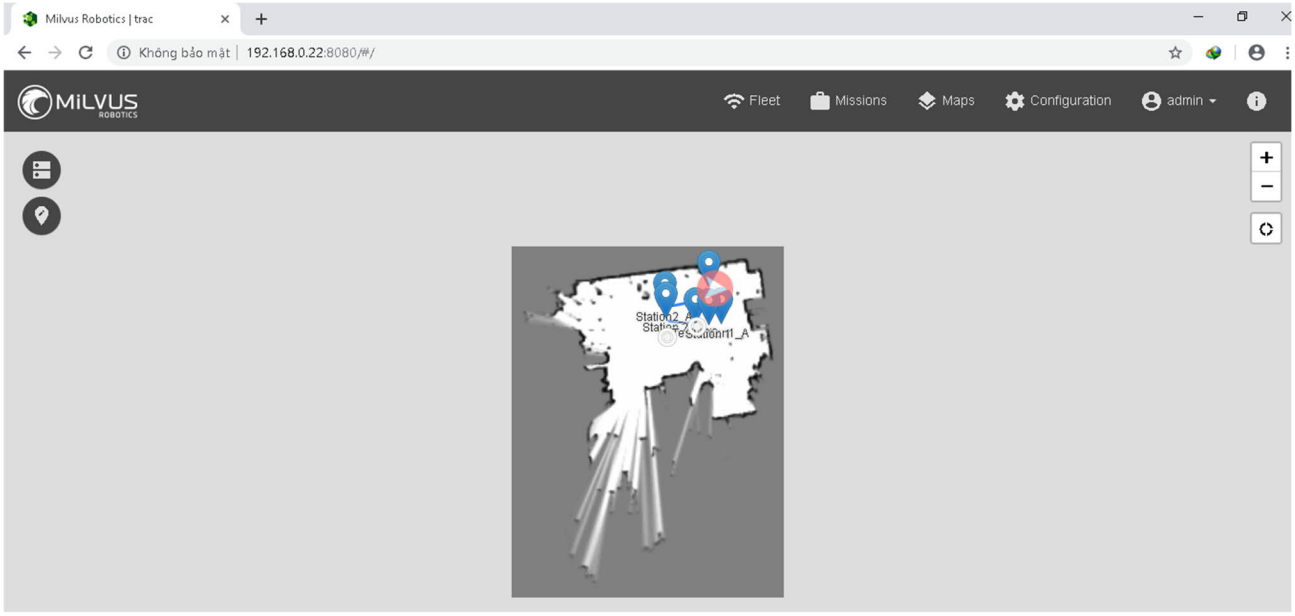


Figure 18. Milvus Robotics Fleet Management Server user interface.

To add the robot on the fleet management server, go to “Fleet” then “Add Robot”. Enter the robot information to the window. Robot UID and internal name are used by the system and serial number (lowercase) and the IP address of the robot must be entered. After this, the robot will display in the “Fleet” section.

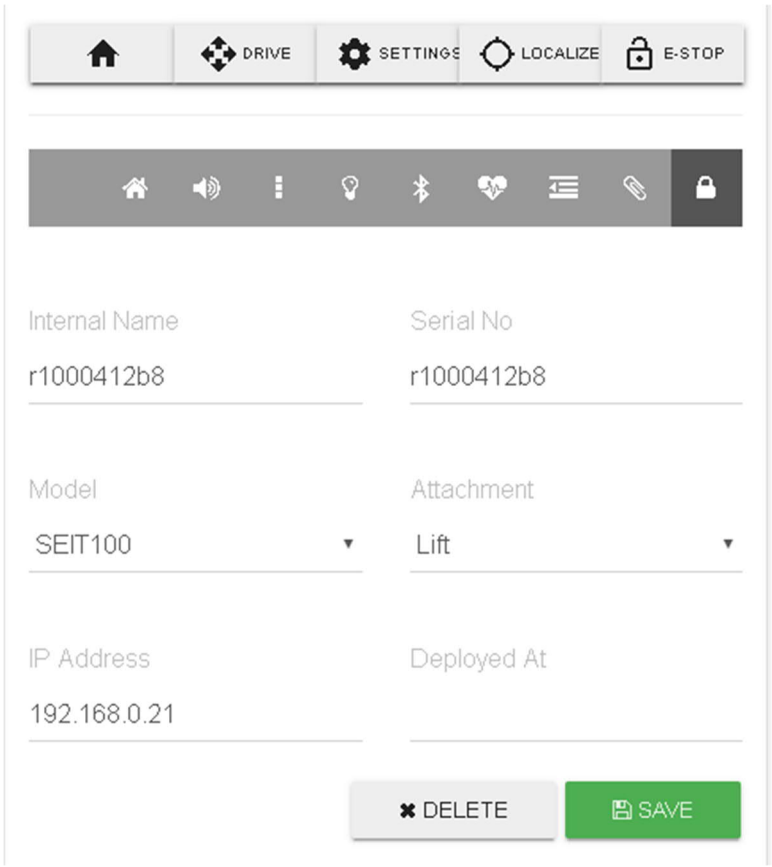


Figure 19. Robot information.

To manually drive the robot, there are two ways: drive manually by the Fleet management server or drive by the remote controller via Bluetooth. To drive by the Fleet management server, select the robot and go to “Drive”. Enter the desired speed and use W, S, A, D keyboard keys. To drive by the remote controller, plug the Bluetooth port to the USB port on robot then activate or deactivate the controller by pressing key A. Use the joysticks and buttons to drive the robot. The manually drive is to drive the robot to an unknown station to scan the factory floor maps creation.

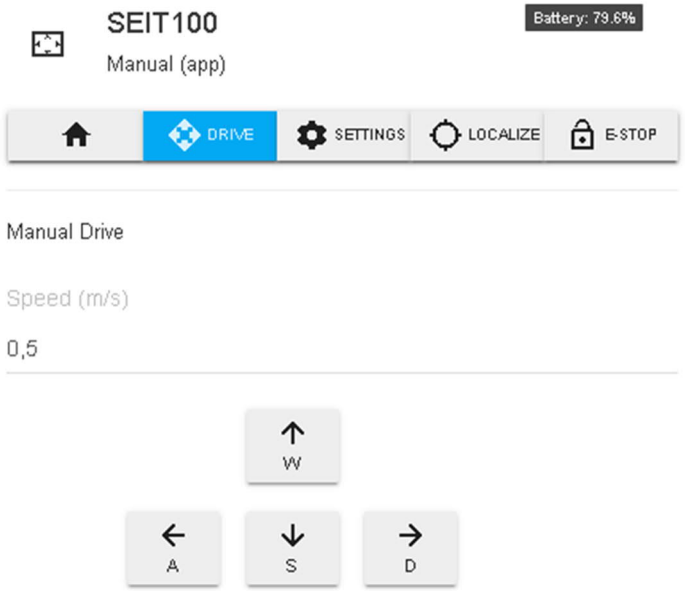


Figure 20. Manual drive window

To manually control the height of the lift, select the robot then click on “Setting” and choose the lift icon on the “Setting” tab. Enter the desired position and speed then press “Set Lift Level”.

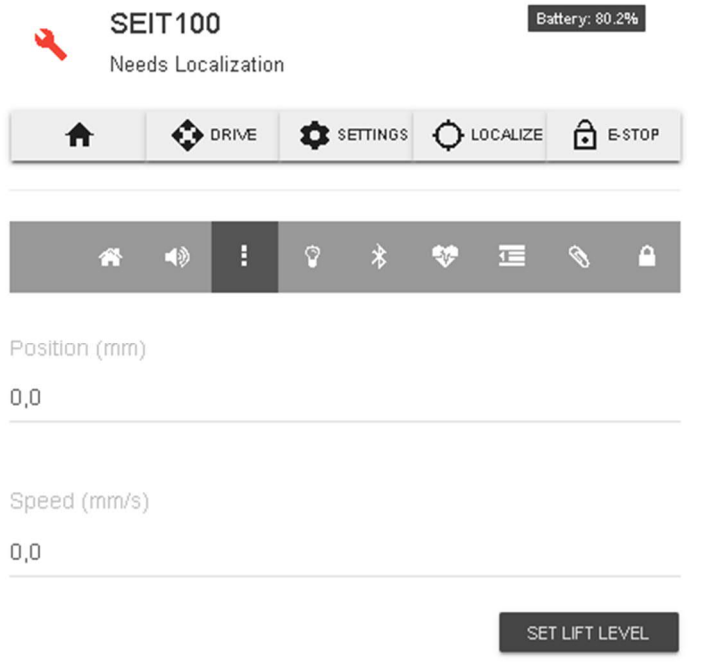


Figure 21. Set lift level window.

There are also other settings for the robot can be accessed and modified, such as LED indicator, sound alarm or Bluetooth connection, etc.

5.3.2. Maps

To create a new map, click “Maps” on the main navigation, then select “New Map”. There are two options to create a map: upload previously created and used maps or explore the area by manually driving the robot. Follow the instruction then enter the name of the map (do not use dot in the name, it will cause problems when editing and exporting the map). After the map is created or uploaded, click “Finish & Save”.The created map will show on the lists of maps. To active the maps for current use, click “Activate” on the desired map. The current activated map will be marked by a star label.

Figure 22. Map creation window.

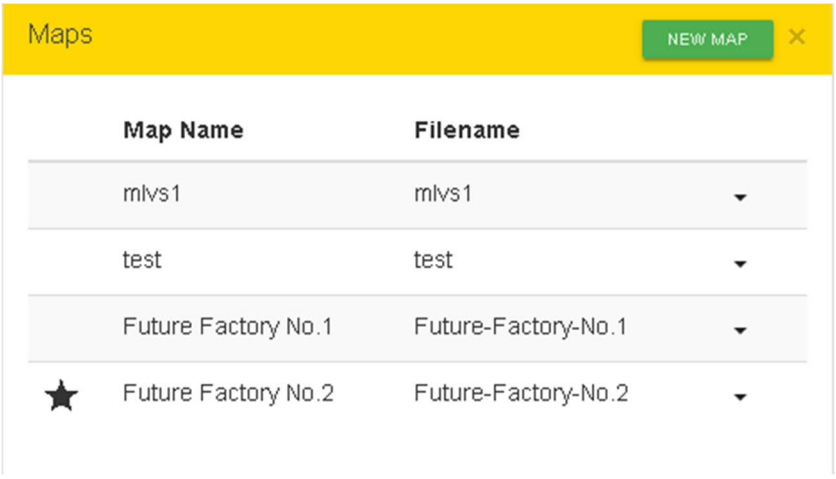


Figure 23. List of created maps.

After creating a new map, stations, zones and navigation helpers can be defined on the map for the use cases. Stations, zones and navigation helpers can be added, modified and deleted in the “Map editor” section. Click on “Map editor” then select “New Station” to create a station. There are two ways to define stations. One way is to enter the coordinates, heading and other configuration in the

station's parameters window. The other way is to manually drive the robot to the desired location and select the robot from the “Get Position From Robot” menu. The station coordinates and heading will be automatically defined according to the current position of the selected robot.

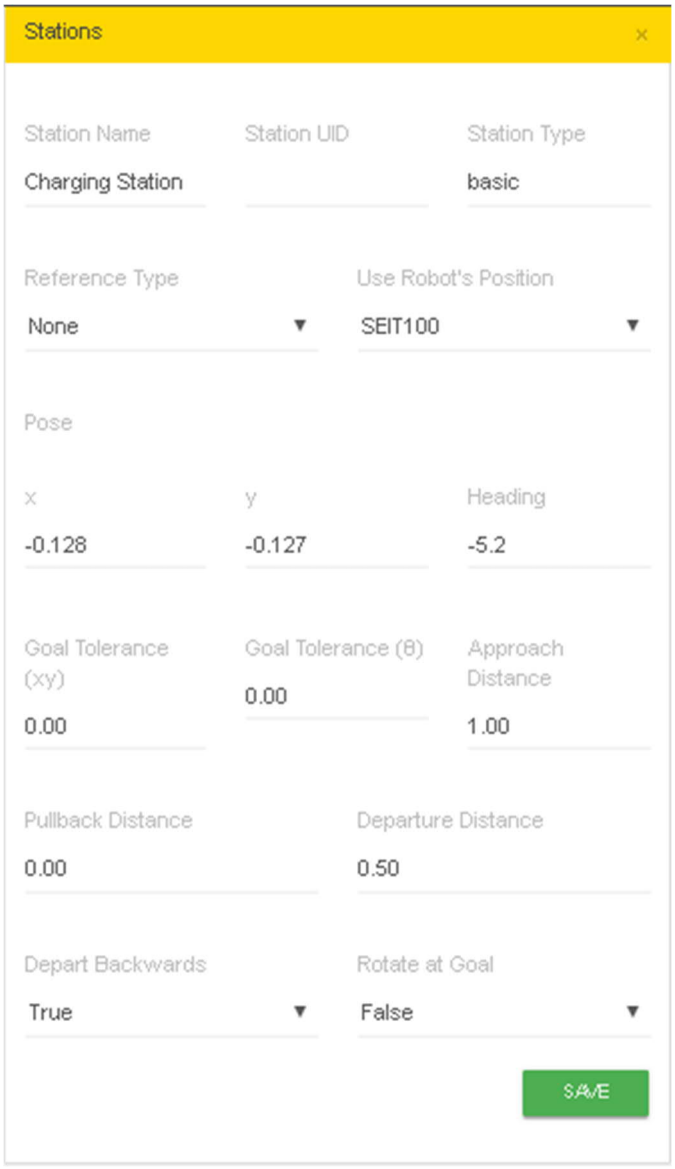


Figure 24. Example of adding station.

As can be seen from figure 25, there are some other optional parameters that can be specified while adding a station. According to the manual of Milvus robotics, the explanation of the parameters is shown in table 3.

Goal Tolerance (xy)	The tolerance in meters for the robot in the x & y distance when achieving a goal.
Goal Tolerance (θ)	The tolerance in radians for the robot in yaw/rotation when achieving its goal.
Approach Distance	Distance to goal station when the robot starts decelerating.
Pullback Distance	The distance in meters for the robot to arrive a specific station with an offset in x direction.
Departure Distance	Distance from initial station when the robot starts accelerating.
Depart Backwards	A boolean value (true/false) for the robot to depart backwards from a station.
Rotate at Goal	A boolean value (true/false) to allow the robot to rotate in place even if it achieved x-y distance tolerance.

Table 3. Station parameters.

After click “Save” button, a marker of the created station will be added to the map. Figure X shows the list of stations that we used in the final demo.

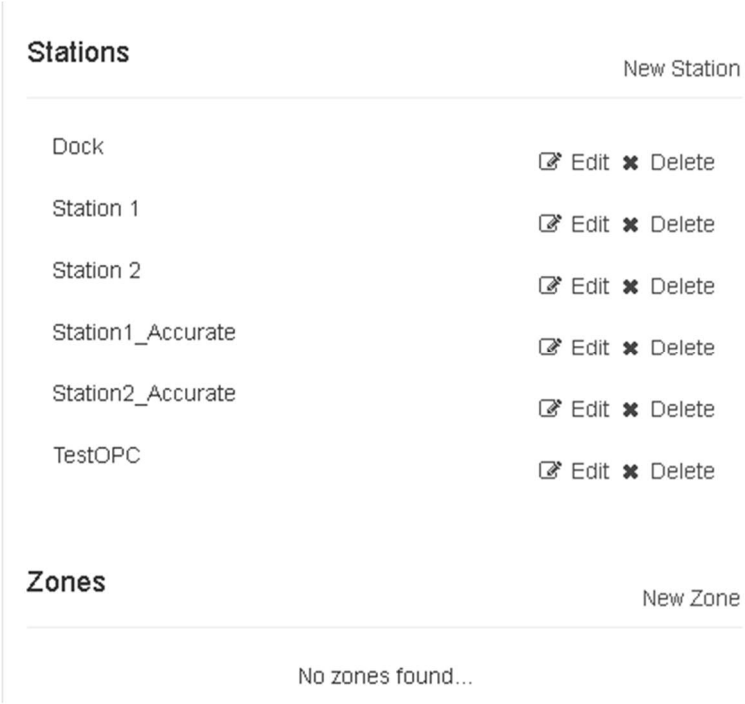


Figure 25. List of created stations.

In the “Map editor” section, zones and navigation helpers can be created. Zones are special areas in the operating environment that the user may have additional control over the mobile robot. Navigation helpers are to help the robot follow the predefined paths or passing points. There are two types of helpers: waypoint and path segments. To create a predefined path between two locations on the map, select “New Helper”, type the name and select “Path Segment” on the “Type” menu, then draw a line between the two stations on the map. If there is more than one path segment between the stations, remember to add them in order in the mission planning. Select “Waypoint” on the “Type” menu and draw a point on the map to create a waypoint helper. More details can be reached in SEIT100 manual. Figure 27 shows the list of navigation helpers and figure 28 shows the created map with stations and navigation helpers.

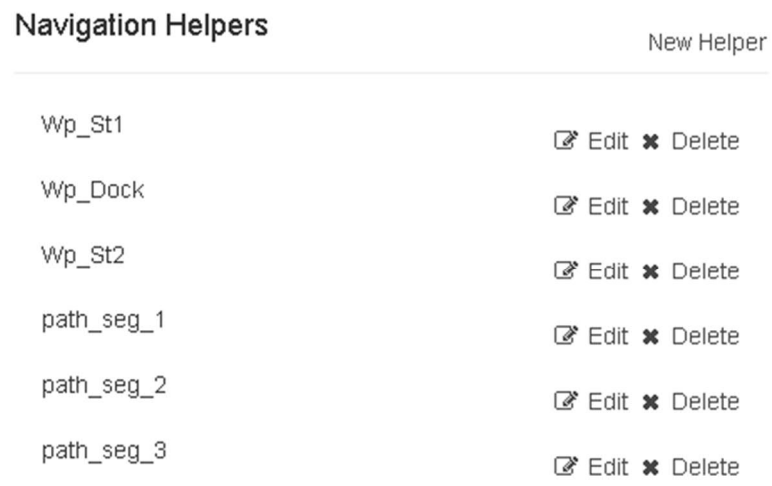


Figure 26. List of created navigation helper

Figure 27. Map of the factory floor after created with stations and navigation helpers.

5.3.3. Missions

The missions can be created, edited, deleted and added to the queue in the “Missions” section. To add a new mission, select “New mission”. The “Mission Planner” window will pop up.

A mission consists of many routines. Each routine is defined by sequential steps. In every step, the robot is sent to a station and is triggered actions before and after arriving at the station. There is also a navigation helper section which can be defined for each step. Note that if there are more than one path segment or more than one waypoint between two stations, those path segments and waypoints are required to add in order.

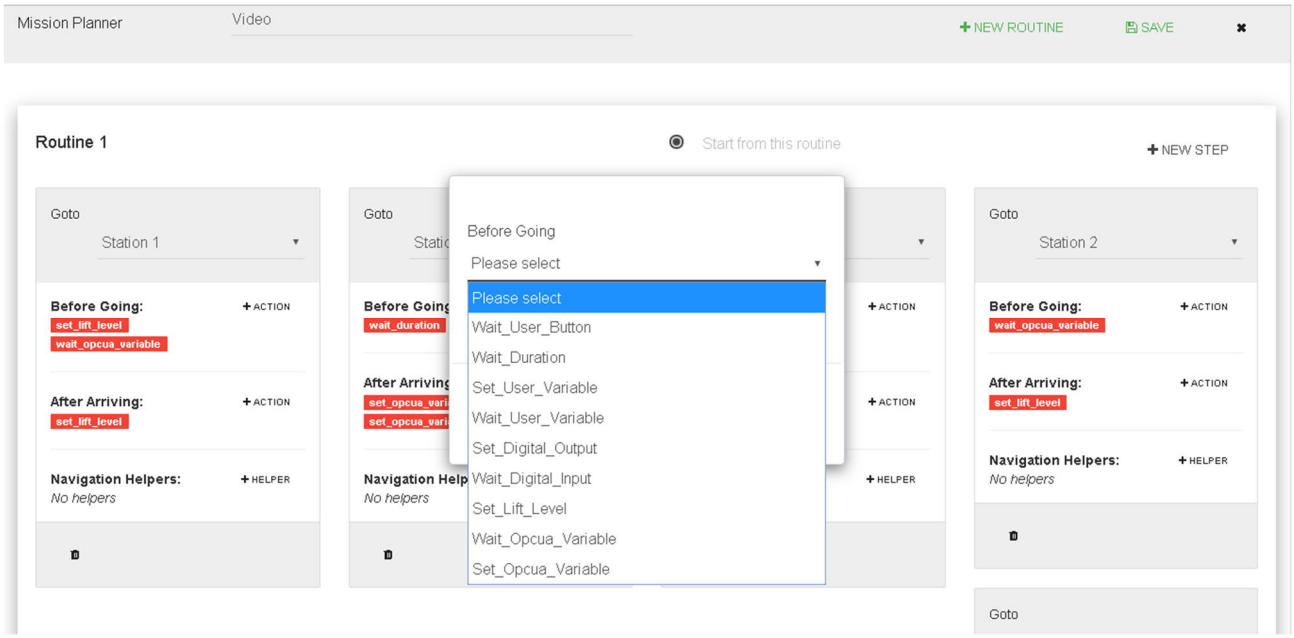


Figure 28. Mission planner, routine, steps and actions.

The available actions are shown in the table 4.

Wait User Button	SEIT waits for user button 1 to be pressed.
Wait Duration	SEIT waits for specified time duration.
Set User Variable	SEIT sets a configured user variable
Wait User Variable	SEIT waits for a user variable to have a specific value
Set Digital Output	SEIT sets a digital output to ON or OFF
Wait Digital Input	SEIT waits for a digital input to be ON or OFF
Set Lift Level	SEIT commands the lift motor to position at specific height
Wait OPC UA Variable	SEIT waits for an OPC-UA variable to have a specific value
Set OPC UA Variable	SEIT sets a configured OPC-UA variable

Table 4. Available actions in Mission Planner

In this project work, to demonstrate the operation of MPP, a mission was created named “Video”. In this mission, there was one routine. At the beginning of the routine, the robot was called to go to Station 1 by an OPC UA signal from IceBlock was triggered by switch 1. Before going to station 1, the lift of the robot was set to the level of 50mm. After arriving to the station 1, the lift was set to the level of 270mm to reach the desired working height. After that, the robot will wait for 11s before going to Station1_Accurate. When arriving the Station1_Accurate, the robot will set OPC UA signals to Yumi to active the tasks of two arms. After that, SEIT100 was waiting for the completing task

OPC UA signals from Yumi to back to Station 1 and set the lift back to the level of 50 mm. After completing an operation at Station1_Acurrate. MPP was called to execute another operation at another station which was triggered from IceBlock by switch 2. The similar actions were executed according to Station 2 and Station2_Accurate. After finishing the operation there, SEIT will wait for several seconds before going to Dock to charge the battery. Figure 30 below describes the “Video” mission

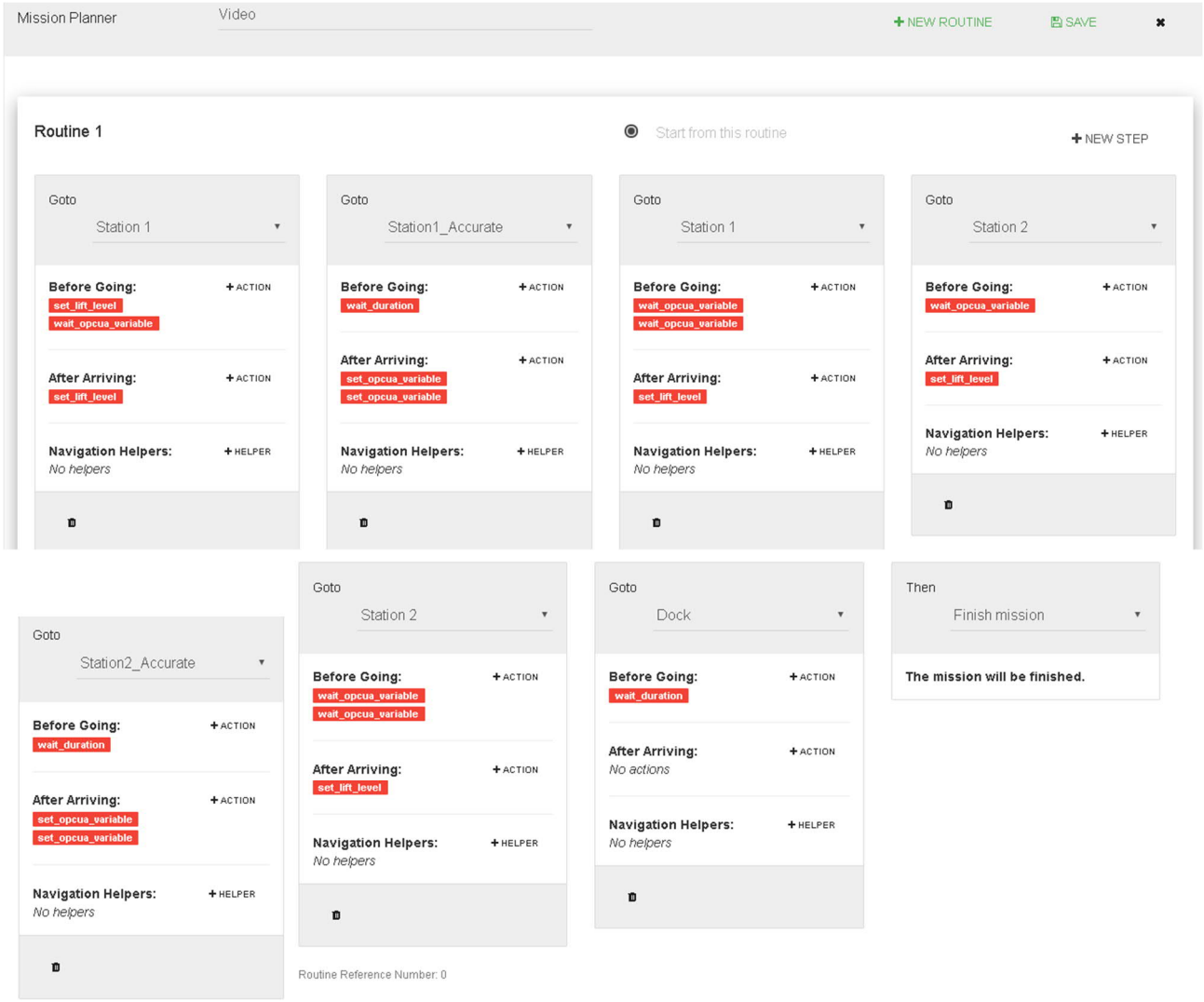


Figure 29. Mission “Video” of the final demo.

To add an action of OPC UA signals, the parameters of the signal are required to fill in the action configuration window such as the OPC UA external server on the “Device” menu, node ID of the signal, qualified name, data type and value of the signal. Figure 31 shows an example action configuration of an OPC UA signal.

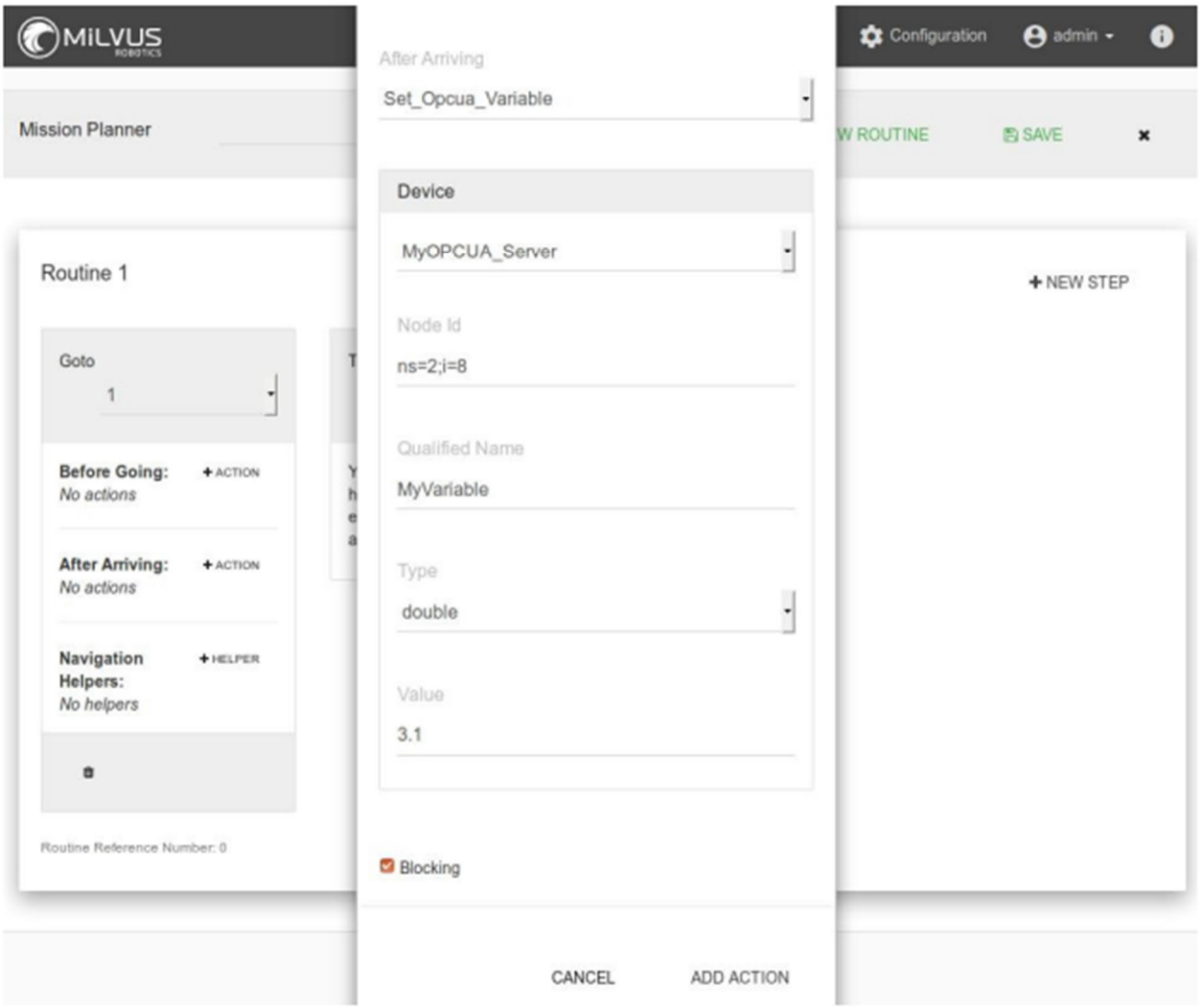


Figure 30. Example of OPC UA signal action.

5.4. OPC UA communication with SEIT

SEIT does not have its own OPC UA variables, but it can read and write OPC UA variables in other devices. This means SEIT has OPC UA client and it is communicating with OPC UA servers. In our case the servers were Iceblock and YuMi via UA gateway.

To use the OPC UA in SEIT we needed to know the OPC UA server addresses and ports. And also wee needed to know node_id of the specific variable. To get the node_id, we had to use UA expert. Here is short explanation how and where configuration is need to make it work.

First of all You need to configure OPC UA server as External devices in SEIT Fleet manager. In Following picture are our project setting to connect to Iceblock and YuMi.

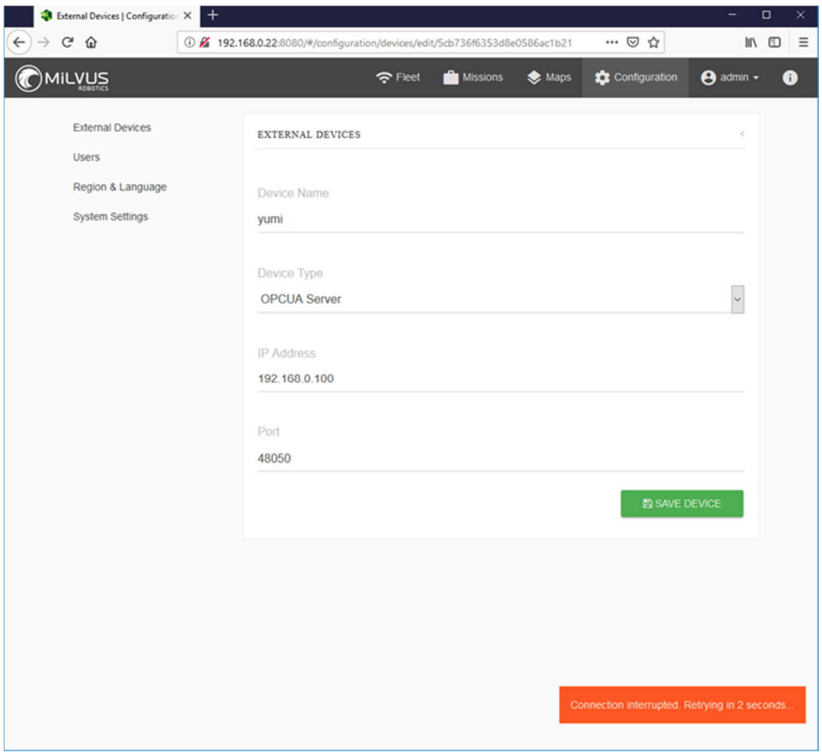


Figure 31. OPC UA settings for YuMi in the SEIT fleet manager.

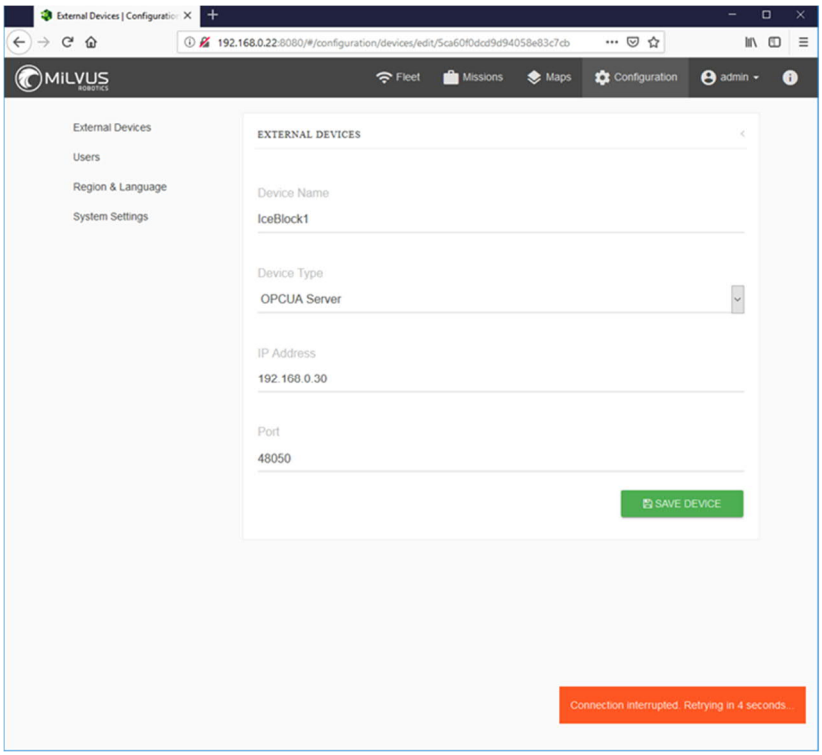


Figure 32. OPC UA settings for IceBlock in the SEIT fleet manager.

When using the OPC UA variables inside the SEIT mission, user need the Node_id of the specific signal. For defining specific variable node_id's and testing the OPC UA connection we used UA Expert program. After installation of the UA expert, we had to connect to two OPC UA servers. One server was the UA gateway for YuMi connection and another was the IceBlock OPC UA server. Server addresses in UA Expert are show in the following figures. UA expert file is available at the AEE Project website.

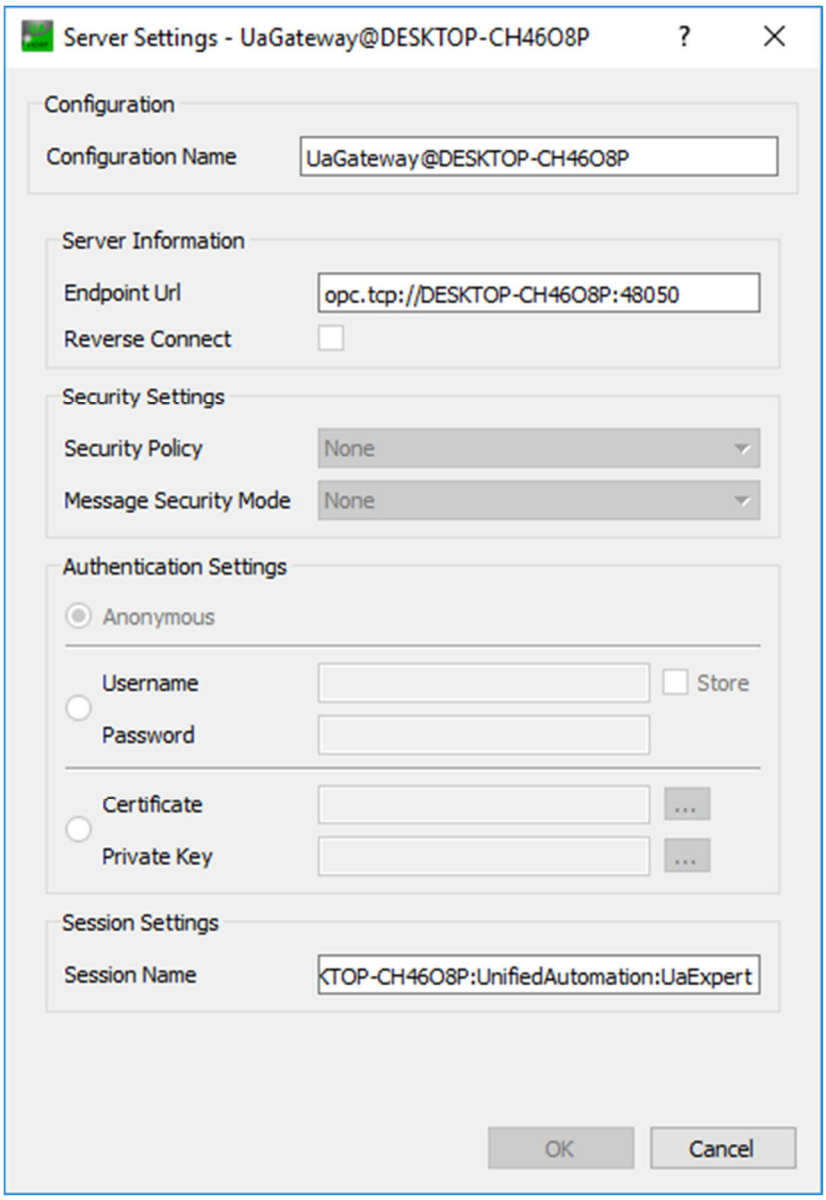


Figure 33. View from UA Expert server setting for UA gateway server the was our link to ABB YuMi OPC server.

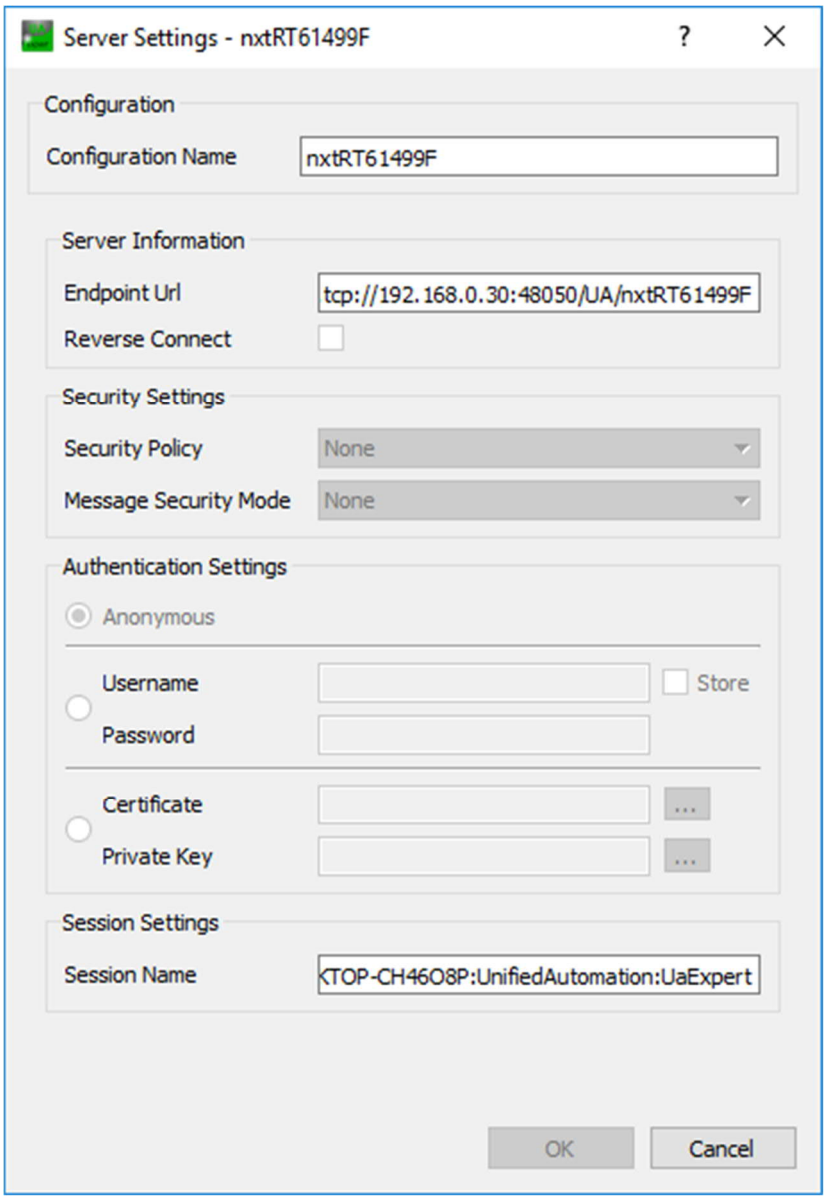


Figure 34. View from UA Expert server setting for IceBlock OPC UA Server. IP is the IceBlock IP and port is definer inside the software in OPCUASERVER FB.

Figure 35. When connecting to OPC UA server with UA Expert comes window where you need to mark the checkbox in left bottom corner.

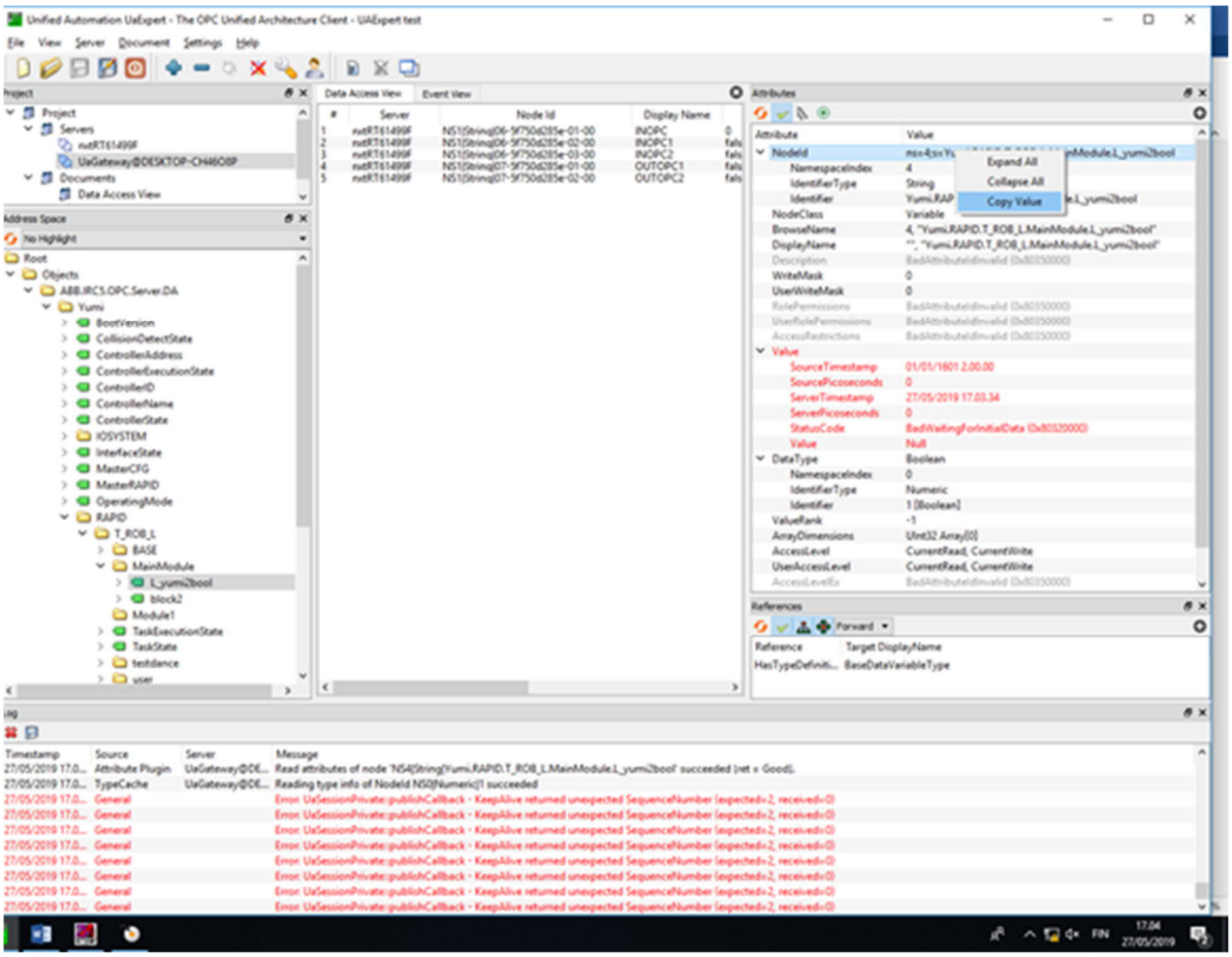


Figure 36. From the UA Expert view it is possible to find the wanted variable on tree structure at right side. When correct variable is selected. it is possible to copy the Node_id from right side view.

5.5. REST API Testing

Milvus Robotics has provided the REST API interface for SEIT platform. During project we tested its functions with using Postman software. By using postman it is possible to test the basic functionalities. Our team's time and knowledge resources were not enough to go further in integrating REST API. Milvus Robotics provided short manual for SEIT REST API functions. According to the manual it is possible to queue a mission, create, load and unload mission, monitor jobs, cancel job, get station list, get robot list, get map list and get mission list. Deleting job was the only thing that we were not able to perform. Below shown is the user interface view from Postman. There it is possible to see the mission queue functionalities set for sending. Used mission_ID is first got with an another command. In the lower part it is shown respond from earlier use of this Queuing mission feature. Response show the job id of the created job. In the AEE Project website is a short document available about the REST API testing and responses to reproduce the Postman settings for each feature.

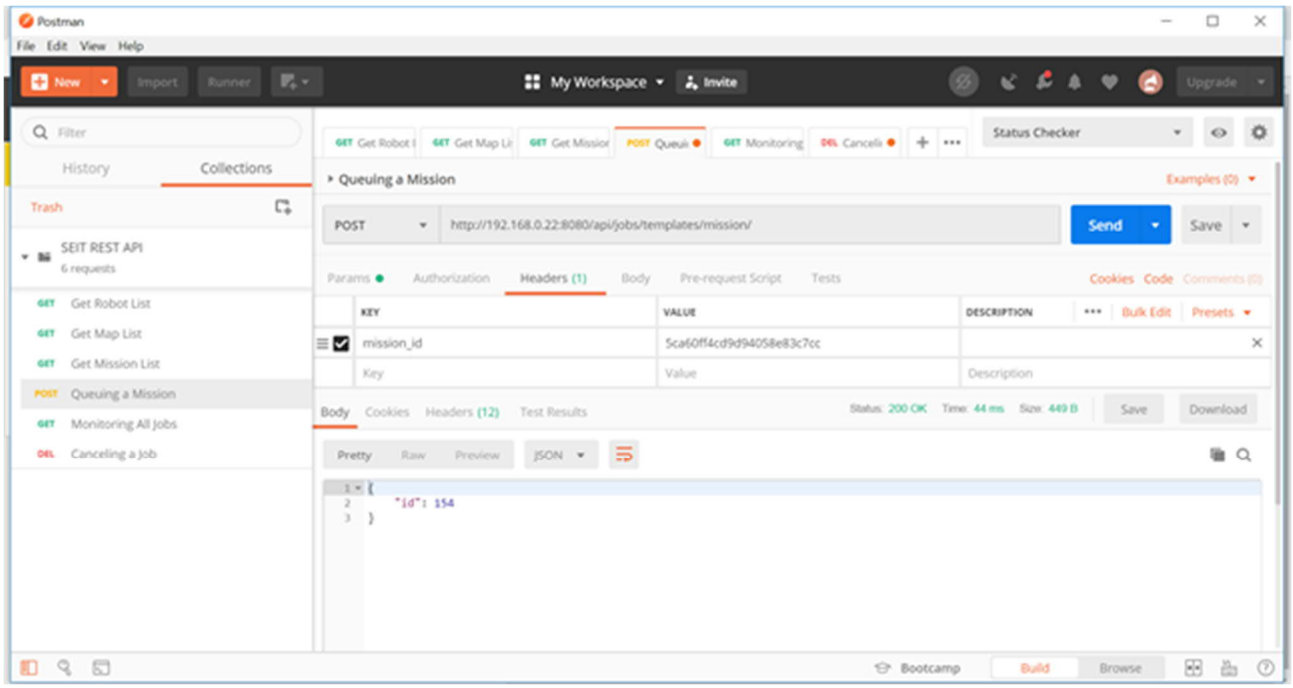


Figure 37. Postman view of the created REST API functionality.

6. ABB YuMi results

ABB Yumi robot is well documented, easy to handle and widely used around the world. This means that it should be easy to implement Yumi to our MPP, to add value and flexibility. It is designed to do small assemblies with accuracy of 0,02mm, and work safely alongside with humans.

Our challenges related to Yumi robot was communications with SEIT 100 and IceBlock, accuracy and programming. These challenges are not only Yumi's fault, it's clear that every challenge is about the environment that Yumi is implemented to.

Yumi can read and write OPC signals but when IceBlock and SEIT 100 is using OPC UA, a gateway server is needed. This means that the factory must have a separated computer that runs the gateway all the time, and it was noticed that the server must be restarted once in a while. This can be avoided with a license.

As mentioned earlier, SEIT 100 is not really accurate in case of orientation. That refers to Yumi's abilities to assemble small objects. When SEIT arrives to station every time in different angle, Yumi cannot calculate the new positions and angles of the joints. To reach the needed accuracy, SEIT must be more accurate and camera vision should be used. In this project the problem was solved by installing a small tray for carrying the objects that Yumi handles.

It is easy to program simple movements with the robot, but Yumi allows only to run one program at the time per hand. When flexible manufacturing working cell need's several stations with different assembly tasks, every assembly program must be called inside the same program. So, all the tasks must be inserted inside one program. Also, the hands don't know about each other unless they are colliding. User variables cannot be written so that the other hand would know when to move. This makes timing and working with two hands challenging.

6.1. Setting up OPC communication

OPC variables enable the communication between the Yumi robot and SEIT. The key elements when defining the variable is that they must be global and persistent. Also Robot has to be in AUTO-mode and motors running to make the variables writeable via OPC. OCP server is always restarted after changing RAPID code, specially after adding a variable to program. Variables are defined in the program data view in the flexpendant.

To enable the OPC in YuMi, it is needed to install ABB IRC5 OPC server to same computer as robot studio. ABB OPC server installation and configuration is quite easy with its own guides. When Robot Studio finds the robot from network then the OPC server finds it too. In the following figures a screenshot from the ABB IRC5 OPC server interface is shown. When OPC server running it might be needed to have the Robot studio also running same time.

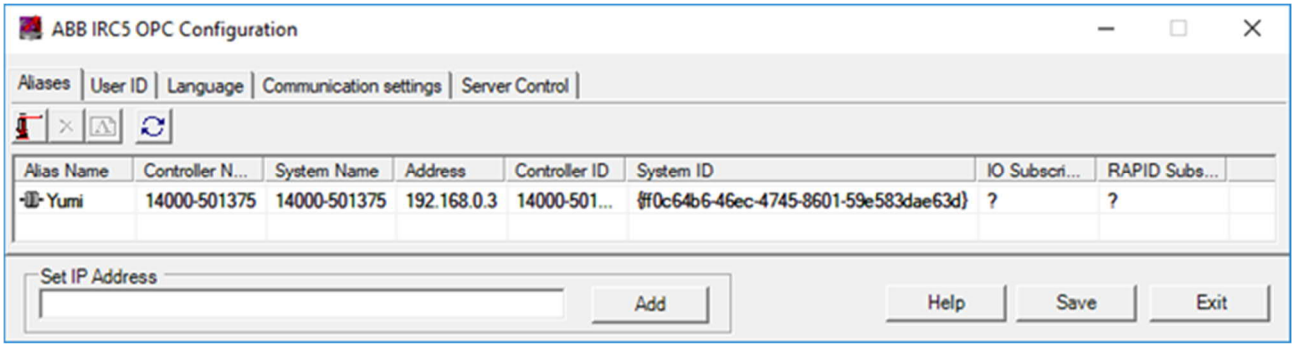


Figure 38. View from ABB OPC server, with one YuMi robot configured using automatic setup. First row shows the ID and other information created to be used in OPC communication. when robot is defined correctly.

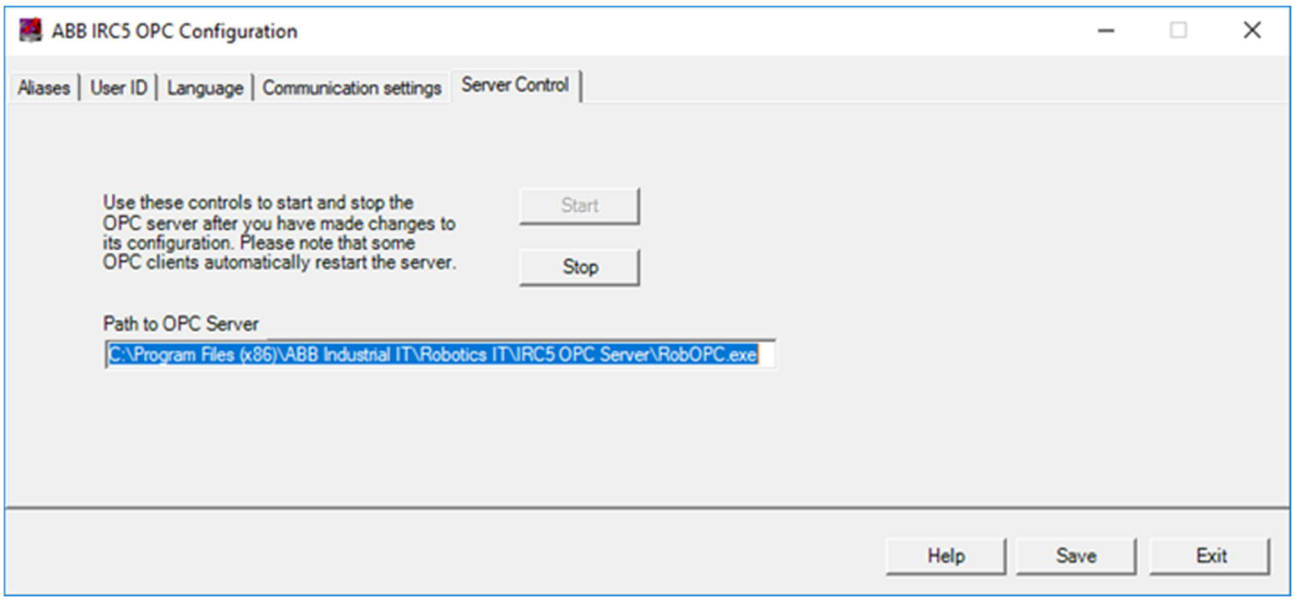


Figure 39. View from ABB OPC server interface from Server Control sheet where server is started.

We have to remember that OPC and OPC UA are different things. So to be able to see and write the YuMi variables in the SEIT, we needed the software in the middle. For this reason we use UA Gateway program that helps to interact between OPC and OPC UA clients. We installed the the UA Gateway on same computer with ABB OPC server. Installation of the UA gateway goes quite simple with the automatic installation. After installation, one needs to pay attention to what is the address of the OPC UA server of the UA gateway and search the OPC server that is running on the same computer. UA gateway file is available at the AEE Project website.

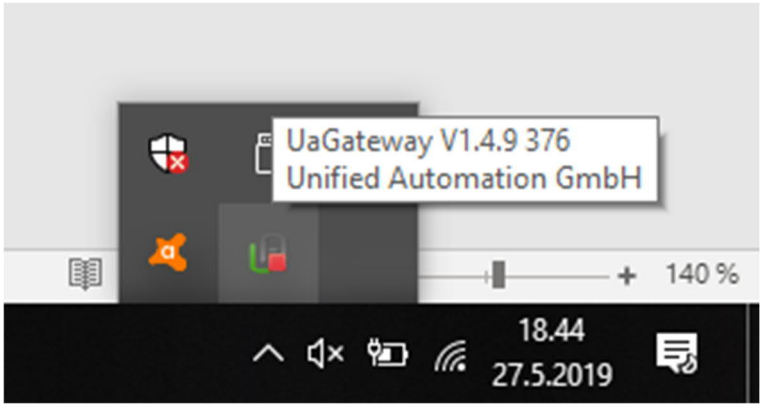


Figure 40. UA Gateway configuration after installation is easiest found from background programs in bottom corner in Windows.

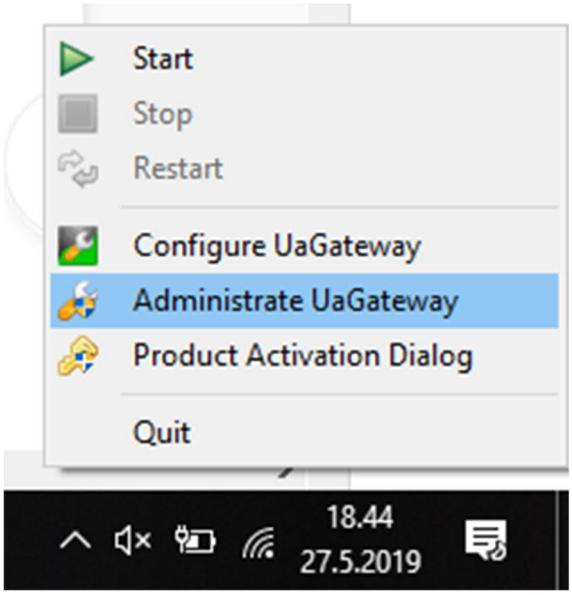


Figure 41. With right-click on mouse opens the selection where is possible to start the server, open the configuration tool for setting the ABB OPC server or open the administrate view to check the server address.

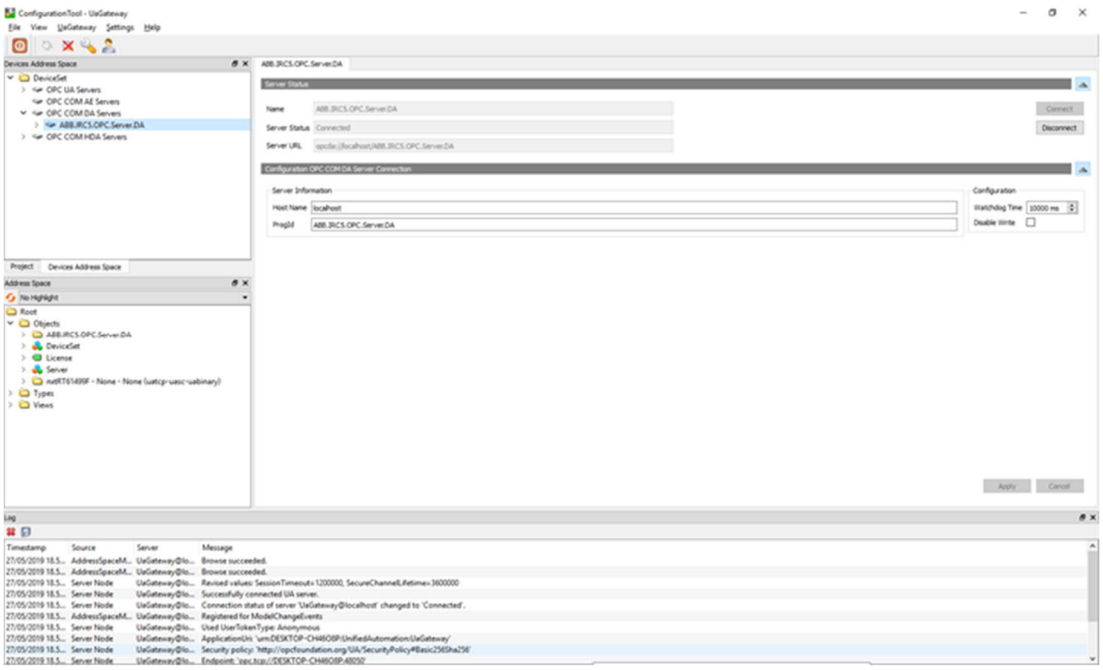


Figure 42. View from UA gateway settings for OPC DA were ABB OPC server is connected

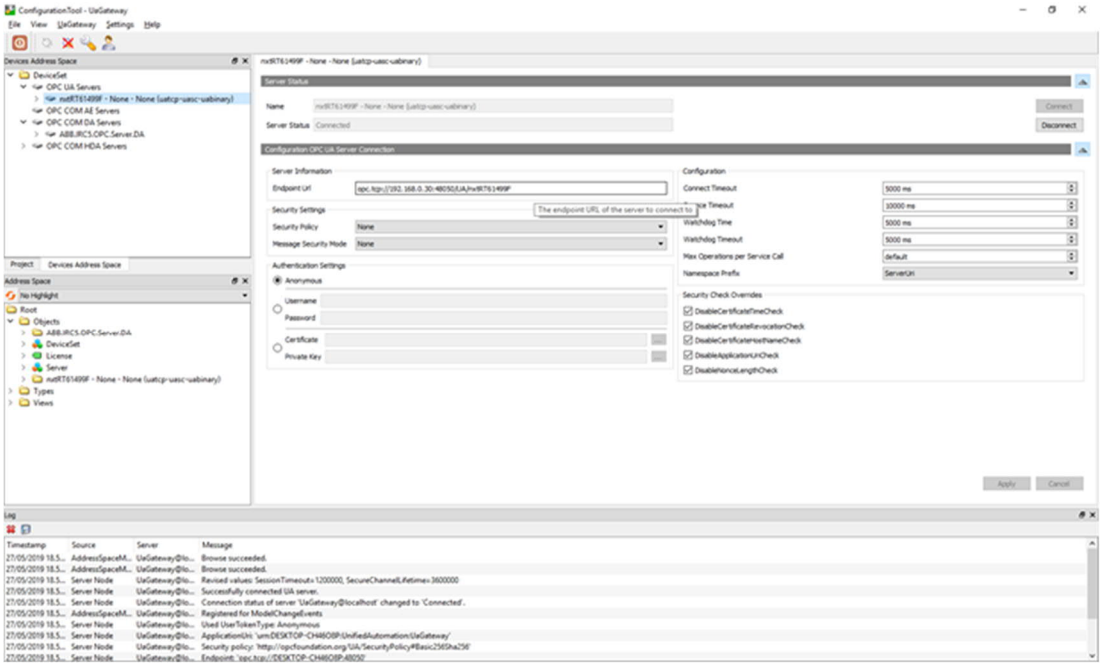


Figure 43. View from UA Gateway from OPC UA settings where Iceblock server is connected.

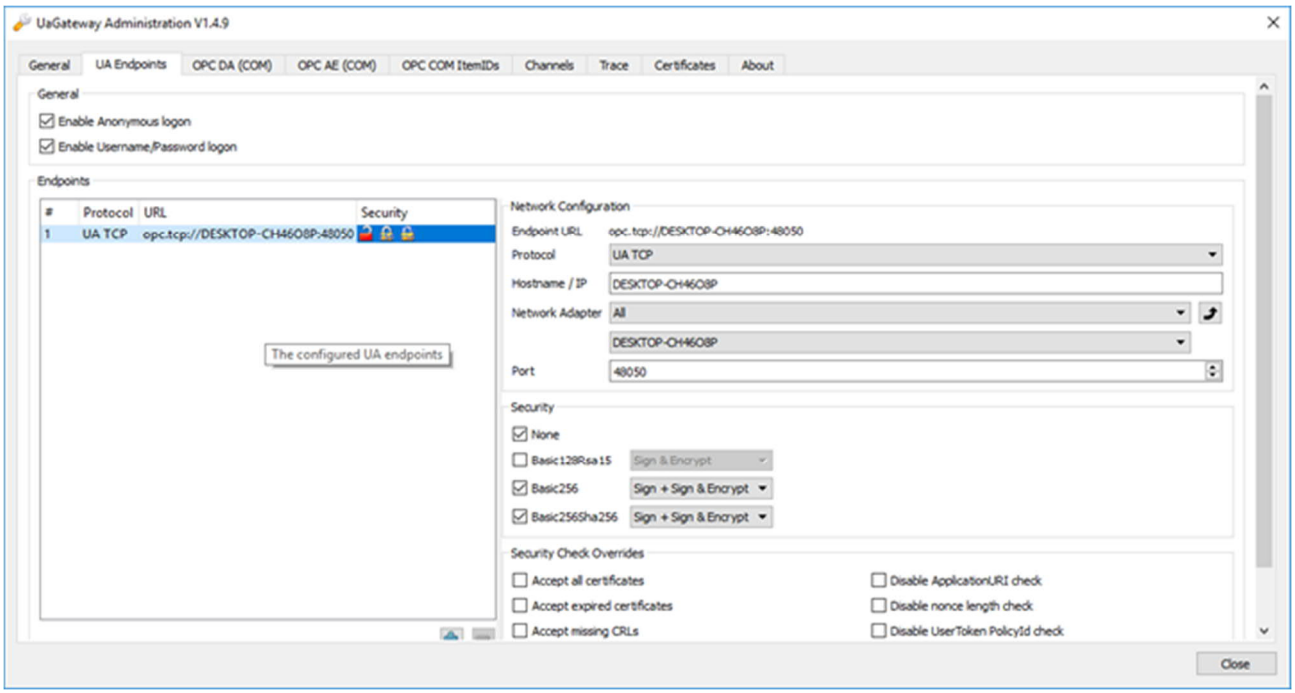
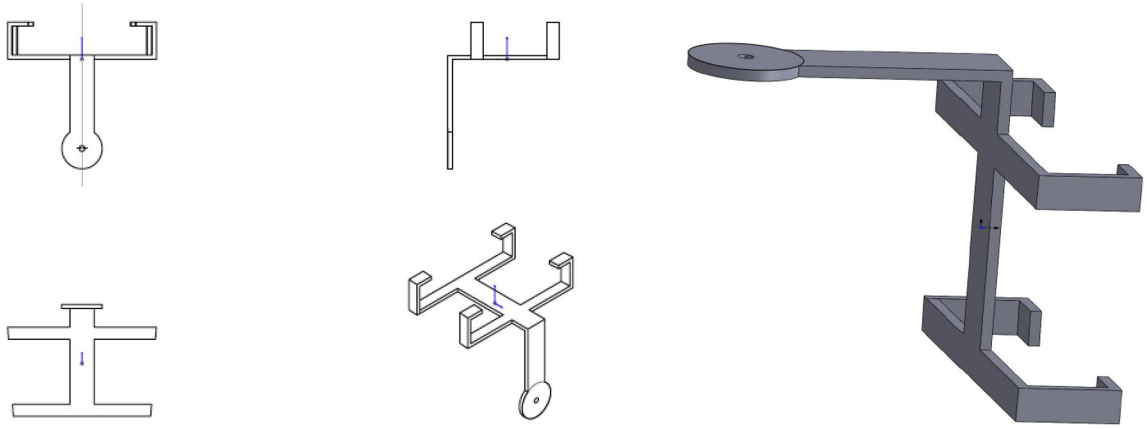


Figure 44. View of UA gateway administration tool where port and hostname can be edited.

7. Mechanical results

In the project, besides establishment of communication and programming of the robots, several mechanical issues were solved. Main task from the mechanical point of view was mounting Yumi on top of the SEIT. Special mounting system, similar to the one used in the forklift systems, was introduced. Forks containing energy supply and DIO wires were connected between each other with the plate. Fixture of the Yumi was done with the screws and premade holes in the forks bridging plate.

The second problem appeared after familiarizing ourselves with the provided equipment, specifically working principles of the Yumi robot. The control panel attached to the robot with 5 meter long cable turned out to be essential for the robot to operate. To make things easier and smoother special panel holder had to be introduced. However, there was no premade solution available on the market, so it was decided to make one using additive manufacturing. The holder was installed using the thread located on the top cover of the SEIT.



.

Figure 45. Holder views.

Third problem appeared when we could not get accurate enough positioning of the MPP, making it impossible for the Yumi robot to find workpieces on the workstations. To solve this issue a special plate, carrying workpieces, was introduced to demonstrate the capabilities of the Yumi. Plate was attached to the forklift connector by screws. To keep the workpiece on its place, in our case it was a cup, a tray was made on the plate.

8. IceBlock

IceBlock is a small smart controller with digital inputs and outputs. IceBlock is a part of the current trend of IoT. Programming is possible in IEC61499 format, using for example NXTStudio. IceBlock requires 24V DC power supply and can handle the 24C DC small current signals on inputs and outputs.

In the project were used one IceBlock device as an interface for the MPP. the IceBlock was imitating the factory automation system where MPP was connected. A couple of reasons for selecting the IceBlock as the interface device, was its abilities in IEC 61499 programming and OPC UA communication. These two topics are currently researched globally to make the factories, for example, more robust and more agile.

8.1. Programming of the IceBlock

The following instructions assume that the user has basic skills to use NXT Studio. We had some experience, but a lot of time were consumed to figure out that simple things are working. Hopefully, these instructions help the next team to go on faster. Communication settings are the first thing to set up when working with IceBlock. When starting the IceBlock for the first time, it is in initial mode, and to get it working, it is required to follow instructions at IceBlock user manual /2/. Basically, the user wants the IceBlock to be connected to the factory wifi or to the wifi where the programming and testing are done. We connected first to a mobile phone hotspot, so the programming was possible where ever the programmer was working. Note! If you want to change the wifi network during a program is running, delete the boot project before setting the new network settings via Iceblock interface.

The main software is programmed using NXT Studio software tool. Project files created in this project are available at AEE Project website. Using NXT Studio with real devices requires a license for the program. More information from the NXT guide called HowTo_InstallLicenseNode /3/. Also correct library needs to be installed (nxtControl.IceBlock-3.0.0.0.fblib). Licenses and libraries were provided by Aalto University.

After this, the device needs to be added.

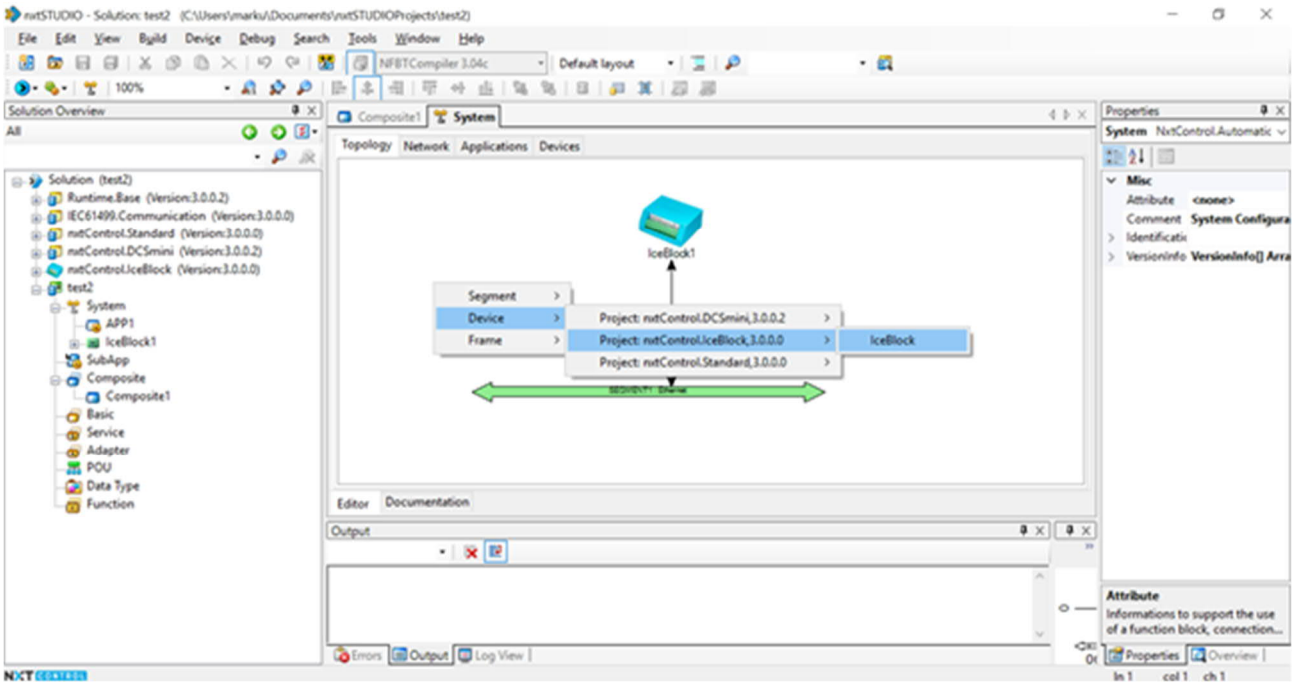


Figure 46. Creating IceBlock recourse in the NXT Studio.

Set IceBlock IP address and check Active network profile is as default.

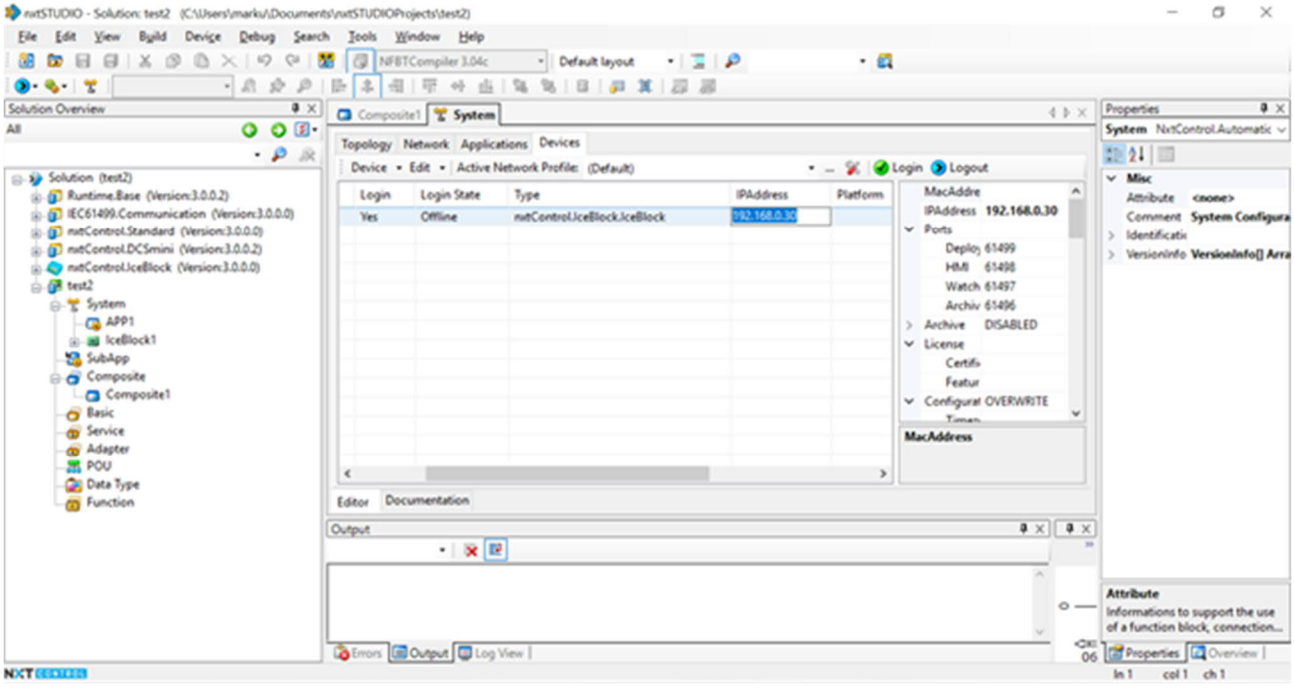


Figure 47. Set up the correct IP address to connect your device.

Login should be possible if the IceBlock is powered and connected to the correct wifi network. Make sure that your laptop is also connected to the correct network.

8.1.1. Enabling inputs and outputs of IceBlock

To be able to use inputs and outputs, a set up of the HW configuration is required.

Select under device RES0 and sheet HW configuration. Right-click with the mouse on empty rows and select add. It should propose correct HW. Select ICEBLOCKIO and OK in window Insert device master.

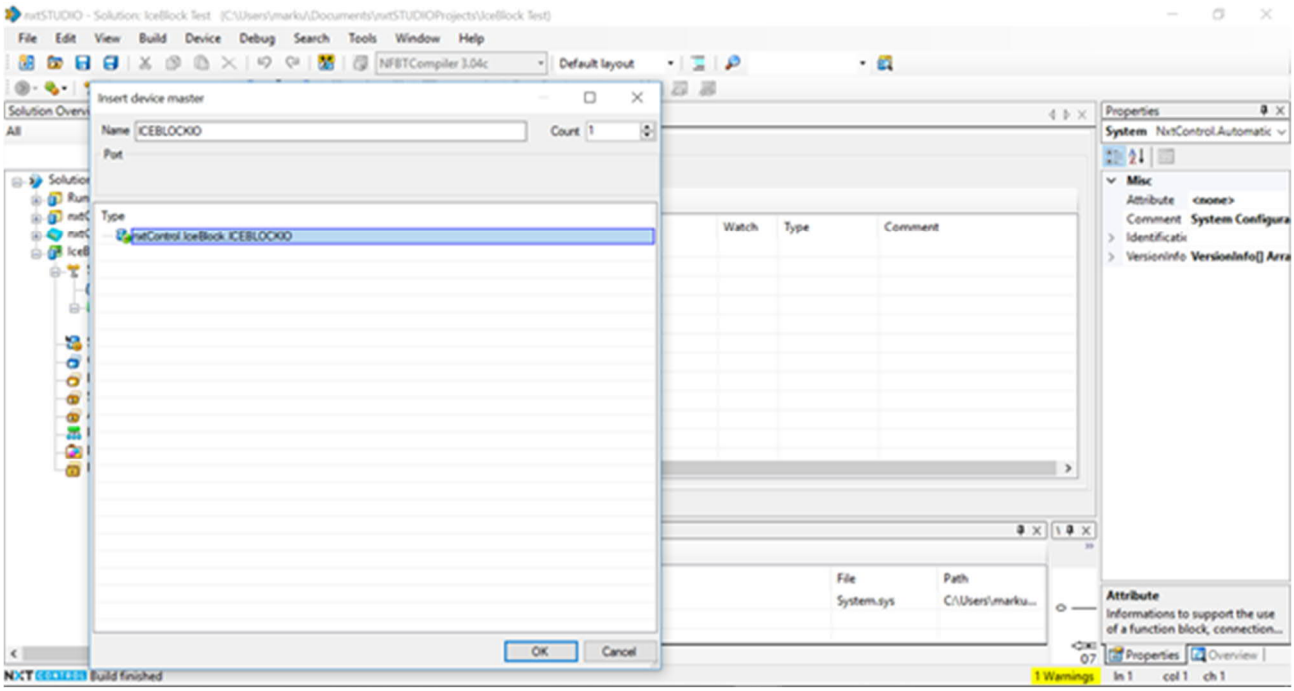


Figure 48. Add IceBlock configuration in the HW configuration view

Then correct HW configuration should appear to window.

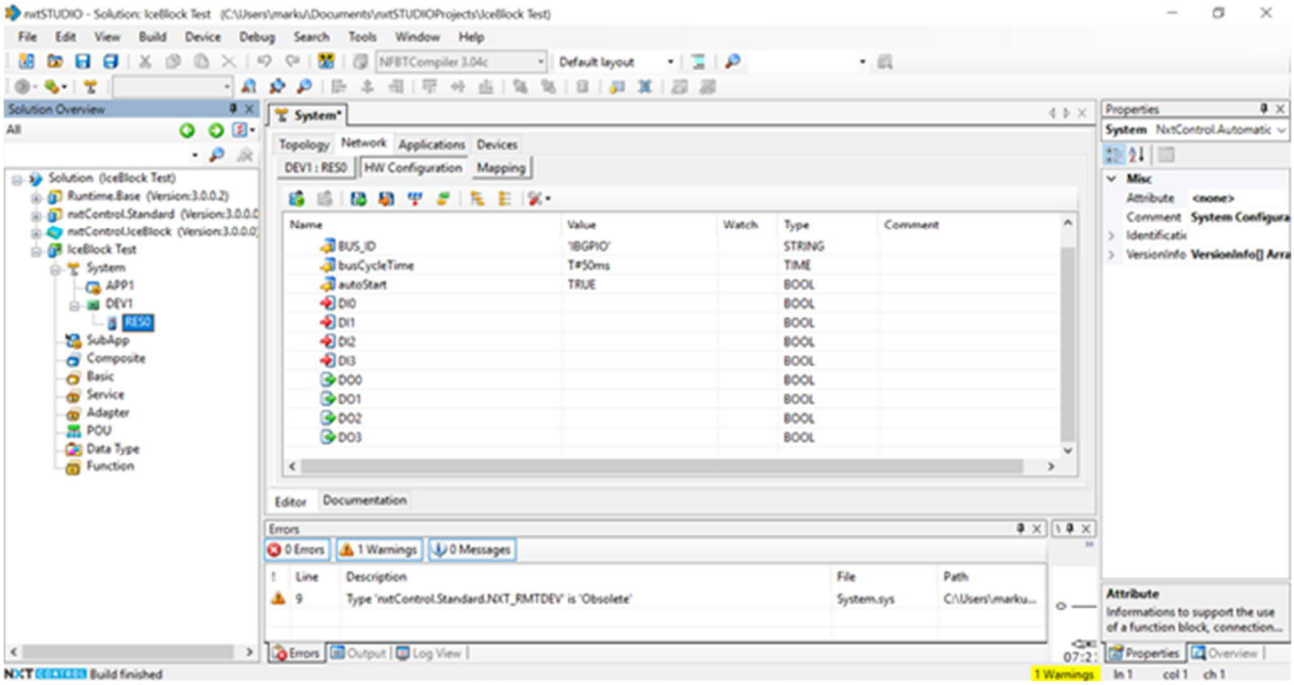


Figure 49. Correct HW setting are visible after successful creation.

If these I/Os are wanted to be used with FB's in the program, certain steps are required.
For inputs include FB SYMLINKMULTIVARDST to program.

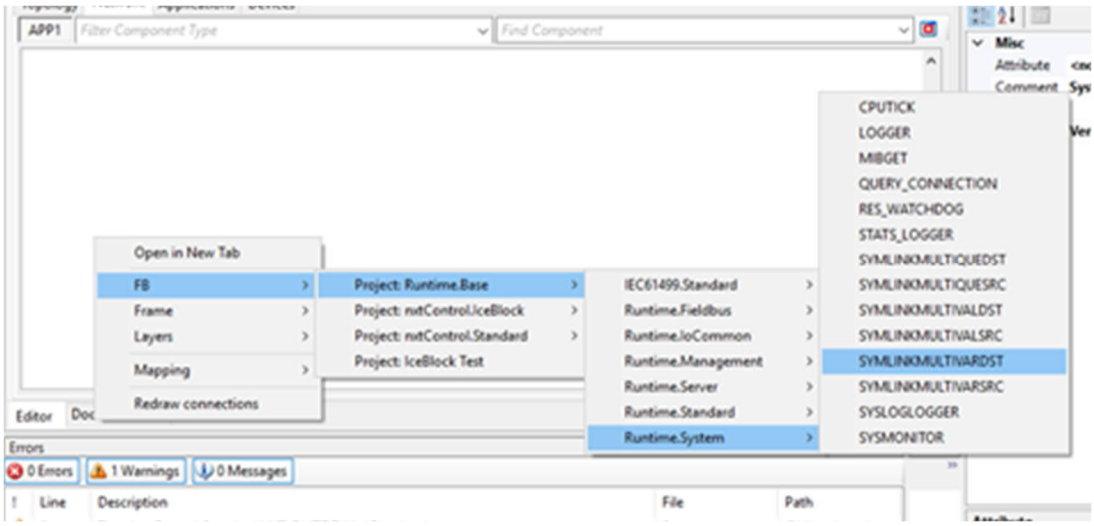


Figure 50. For input handling, add SYMLINKMULTIVARDST FB.

Set constant to the left side and add a description like the input1 figure below.

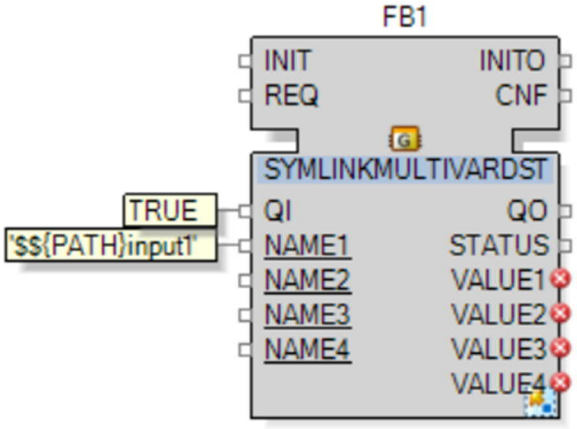


Figure 51. On the NAME fields are needed to add constant with the name that is used also at HW side

Use Interface editor to set VALUE outputs to BOOL type. Variable type can not stay ANY.

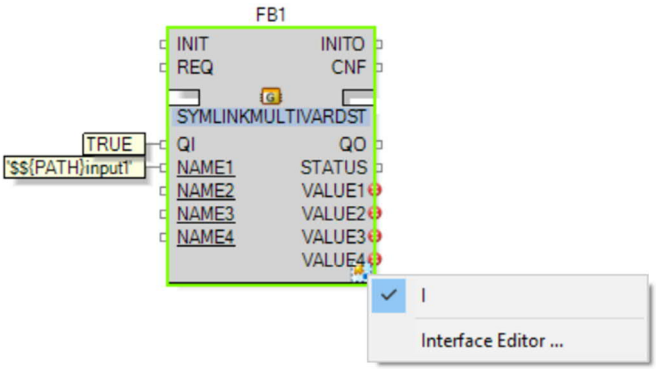


Figure 52. VALUE variable type is first ANY and must be changed in interface editor

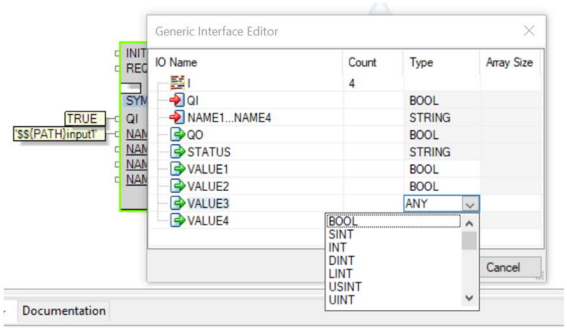


Figure 53. With IceBlock the variable has to be changed to BOOL.

Then use the same input1 naming in the HW configuration with correct indication on front.

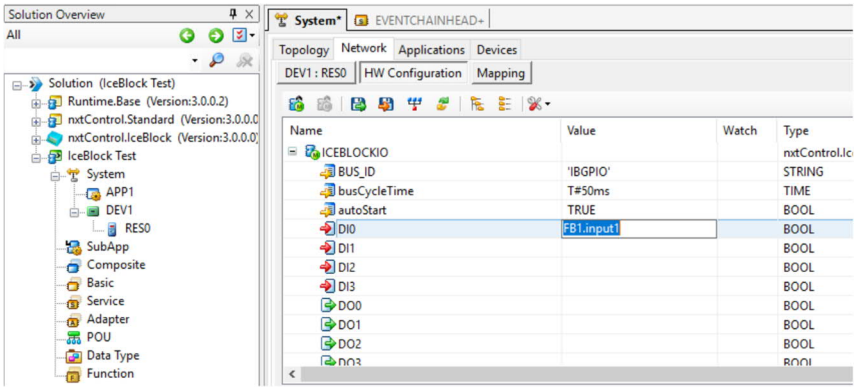


Figure 54. To link the HW to software, same name combined the fb is required to fill in.

For outputs use the FB SYMLINKMULTIVARSRC and change the datatype to bool in interface editor like earlier. Add your name to NAME places. The Figure below has the “output1” added. One way to make the FB correct size, is to drag from the right bottom corner the whole FB bigger or smaller

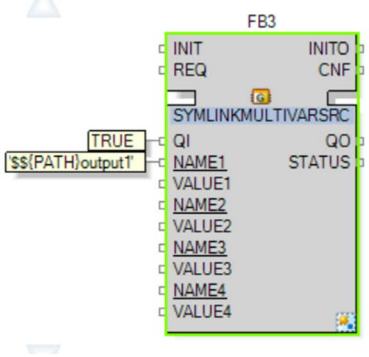


Figure 55. FB for outputs uses the same naming.

Add the same naming with correct FB information to HW configuration in Value field.

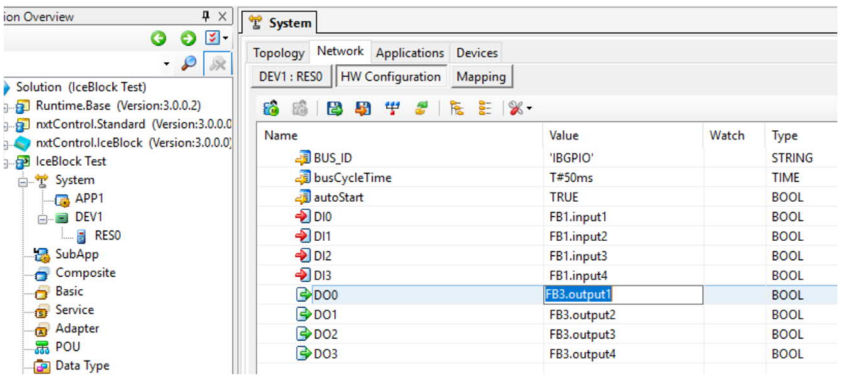


Figure 56. Add the fb and name to the Value field

Now it is possible to “wire” these variables in required places. In the example below the Activation of input, activates the output in the same IceBlock.

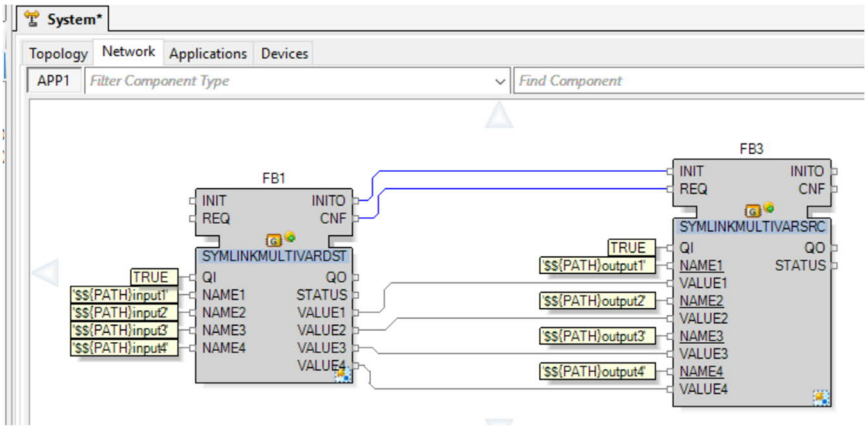


Figure 57. This wiring model makes the input to activate corresponding output.

8.1.2. Enabling the OPC UA communication

IceBlock has OPC UA server abilities. This means that IceBlock variables can be made visible to other devices, but IceBlock is not able to monitor or control other devices over OPC UA. In our project, the SEIT platform was following and controlling some of the variables from the IceBlock program. Enabling variables to OPC UA in IceBlock requires certain steps in the NXT Studio.

Add the OPCUASERVER FB to your program. Map the FB to your resource IceBlock.

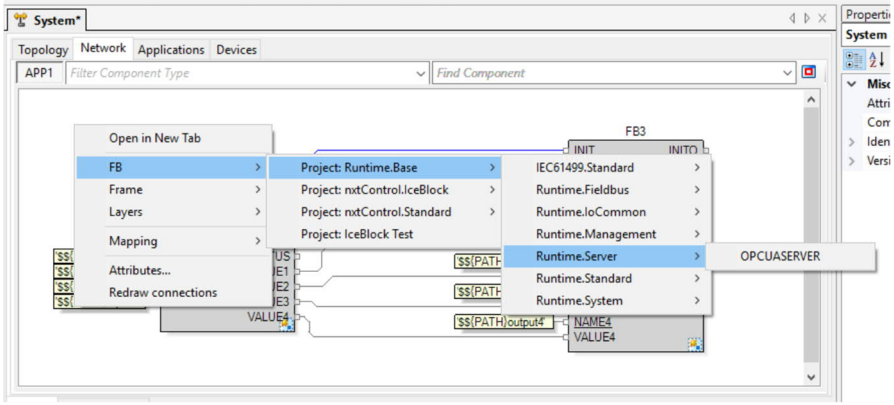


Figure 58. Add OPCUASERVER FB to program

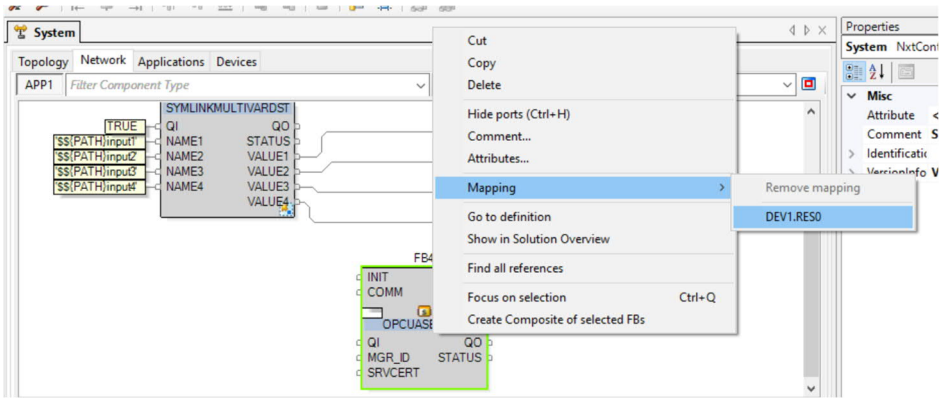


Figure 59. Create the mapping to correct IceBlock

Next thing is to activate the OPC UA feature in HW side. Select your device and change the status of FB row on right side list to OVERWRITE. This makes it possible to open the subselection under FB.

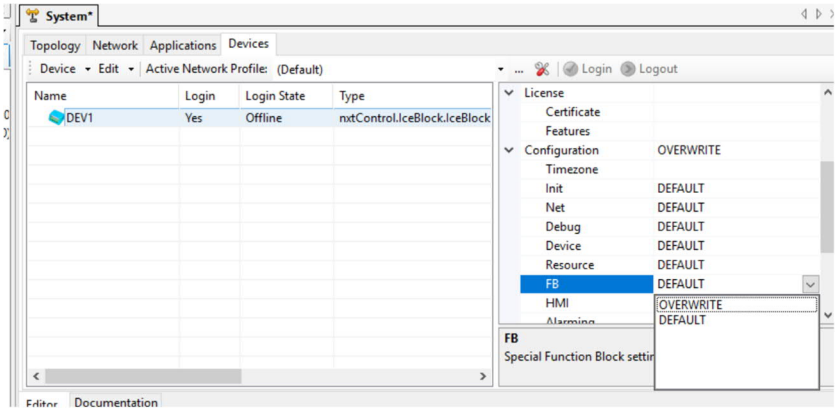


Figure 60. Change DEFAULT to OVERWRITE for FB line

Next select the OPCUASERVER and change it to OVERWRITE.

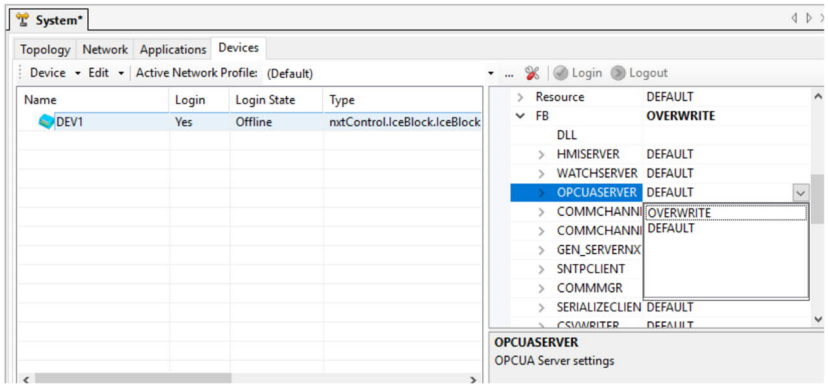


Figure 61. Change DEFAULT to OVERWRITE in OPCUASERVER line

Next thing is to open the subselection and change Enable field to TRUE

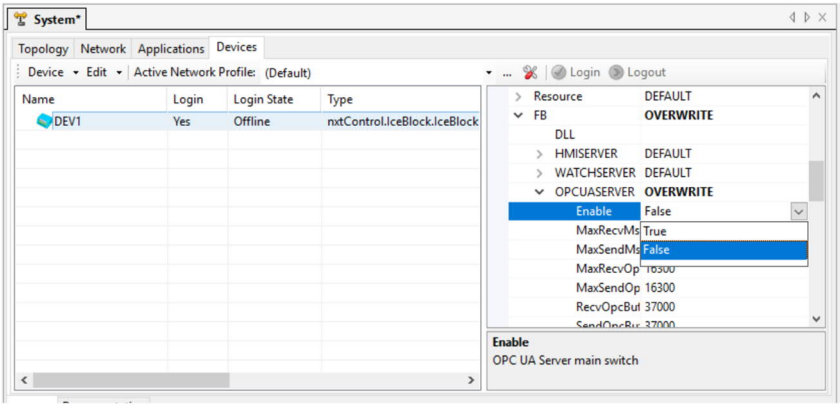


Figure 62. Change FALSE to TRUE in Enalbe line under OPCUASERVER

Add also the constants shown below figure. The values come automatically. The port number is the one that is needed in OPC UA client.

Now the hardware side is done. This does not mean that everything is working. Next, you need to determine what variables are visible in the OPC UA. This is done via attribute settings of certain FB's. It is not possible in all FB's. In our example, we want to see the status of the input and outputs of the IceBlock. This means we have to create composite FB of our earlier created FB to connect physical I/O signals. Remove mapping from the FB's which you want to create the composite FB. Select the FB and create the composite FB from the selection that opens with right click of the mouse.

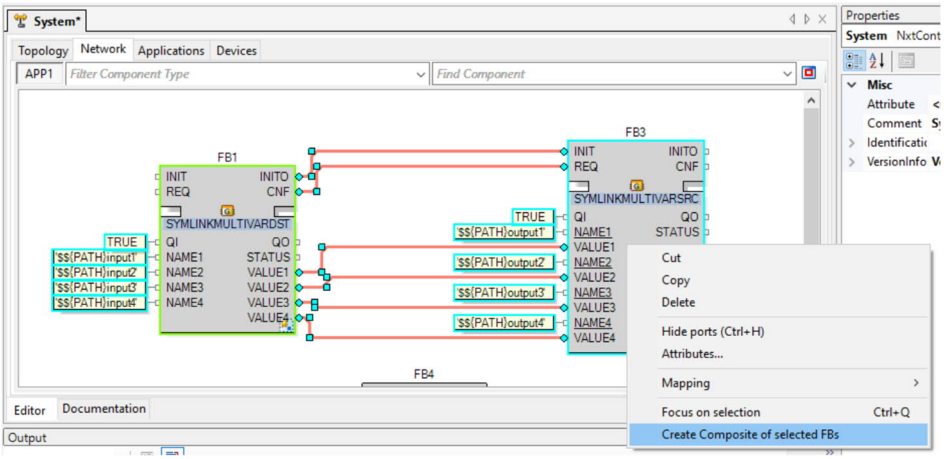


Figure 63. Creating composite FB of the selected FB's.

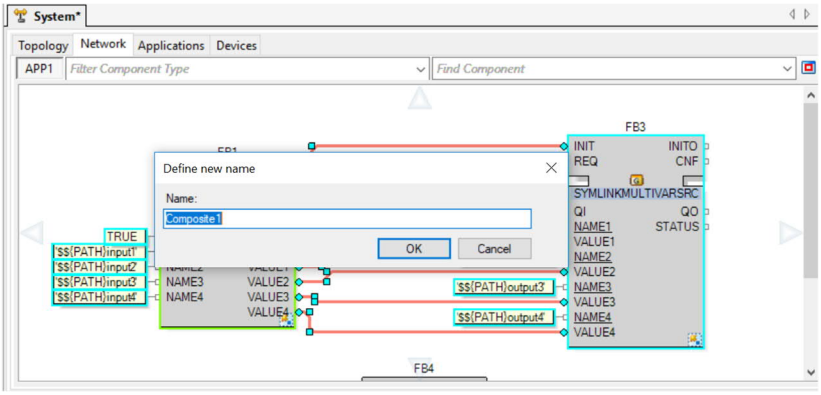


Figure 64. Giving a name for the new composite FB.

Add wanted interfaces to the created composite FB.

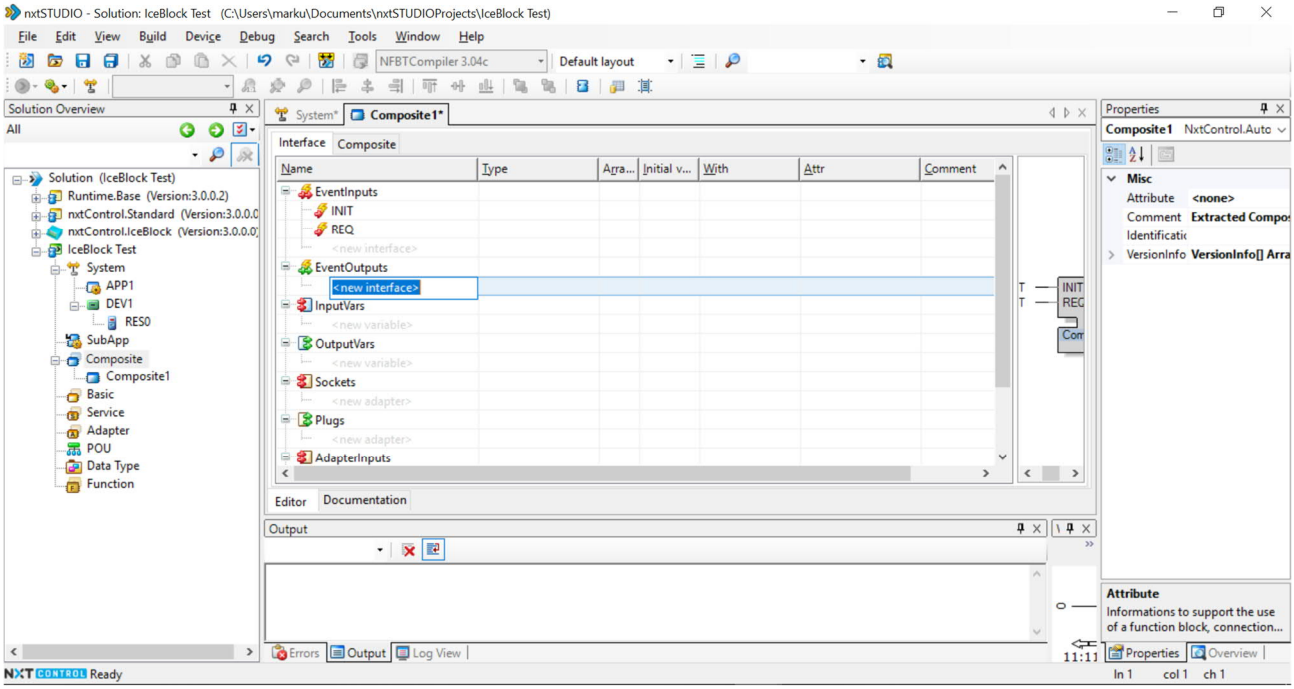


Figure 65. Adding inputs and outputs according to user needs.

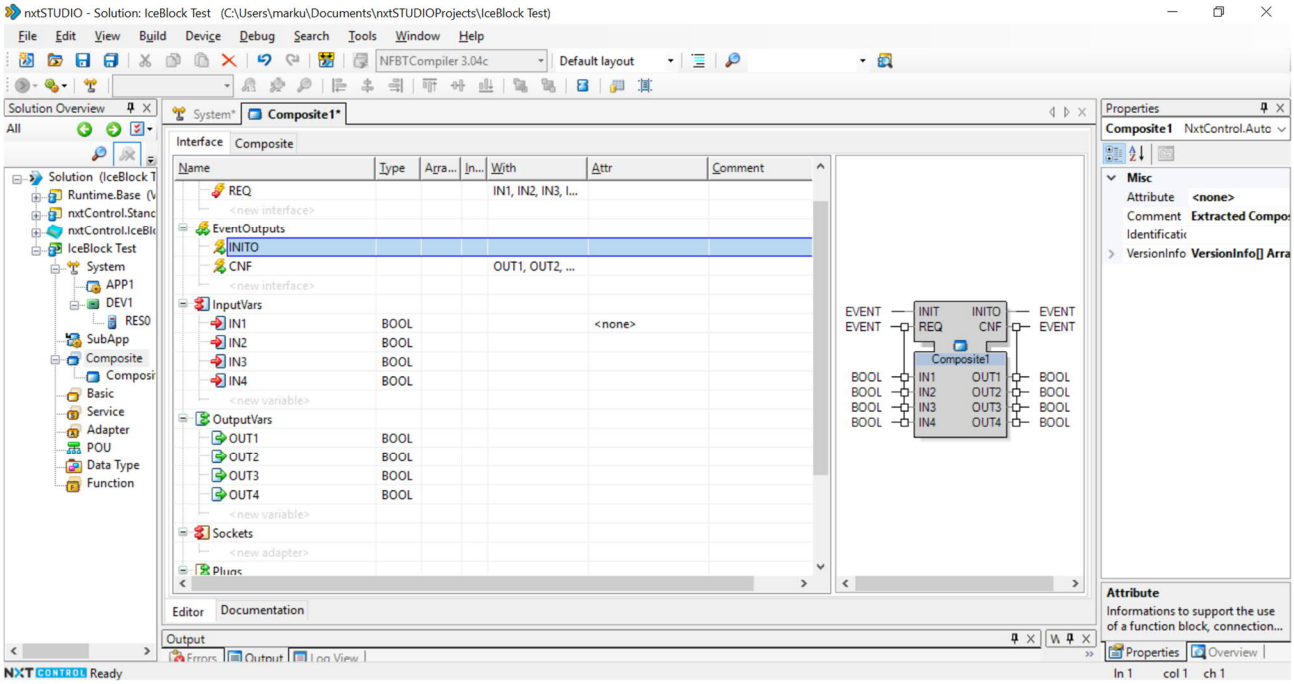


Figure 66. Inputs and outputs with event connection added to composite FB.

Now it is possible to activate the OPC UA features to variables of this new composite FB. Select the Attr on one of the inputVars and the attributes window opens.

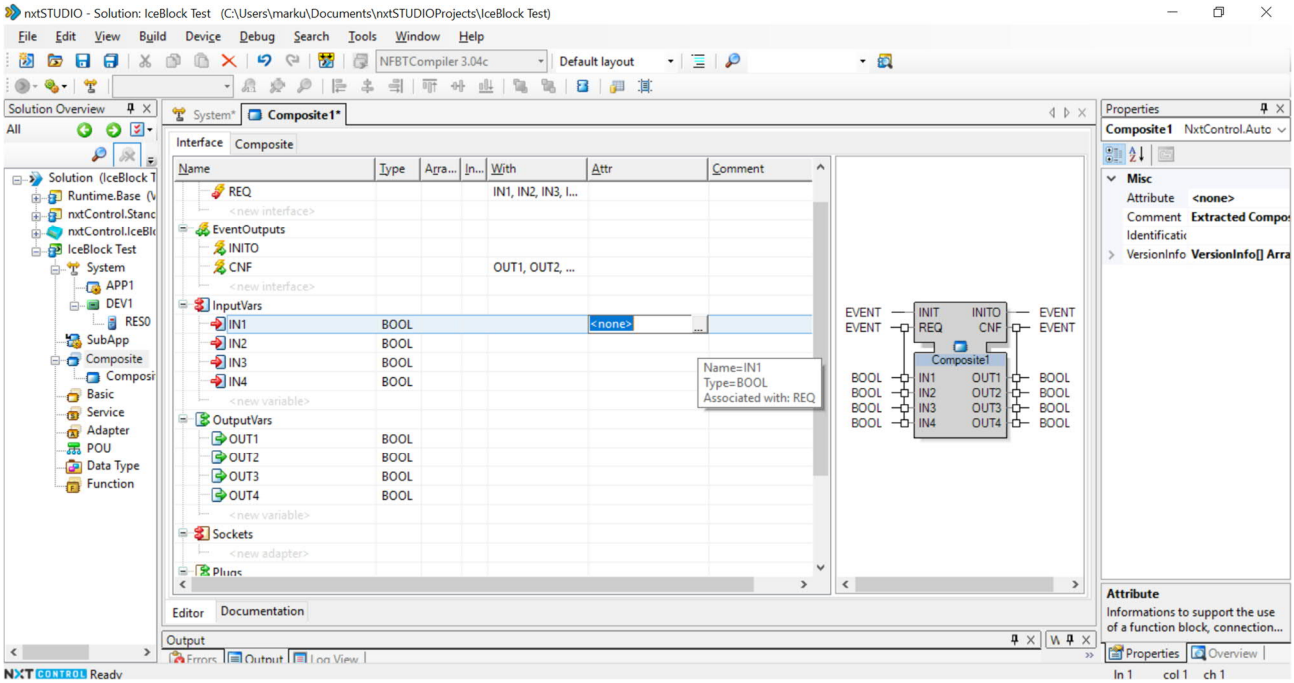


Figure 67. Select Attr cell to start activating the OPC UA.

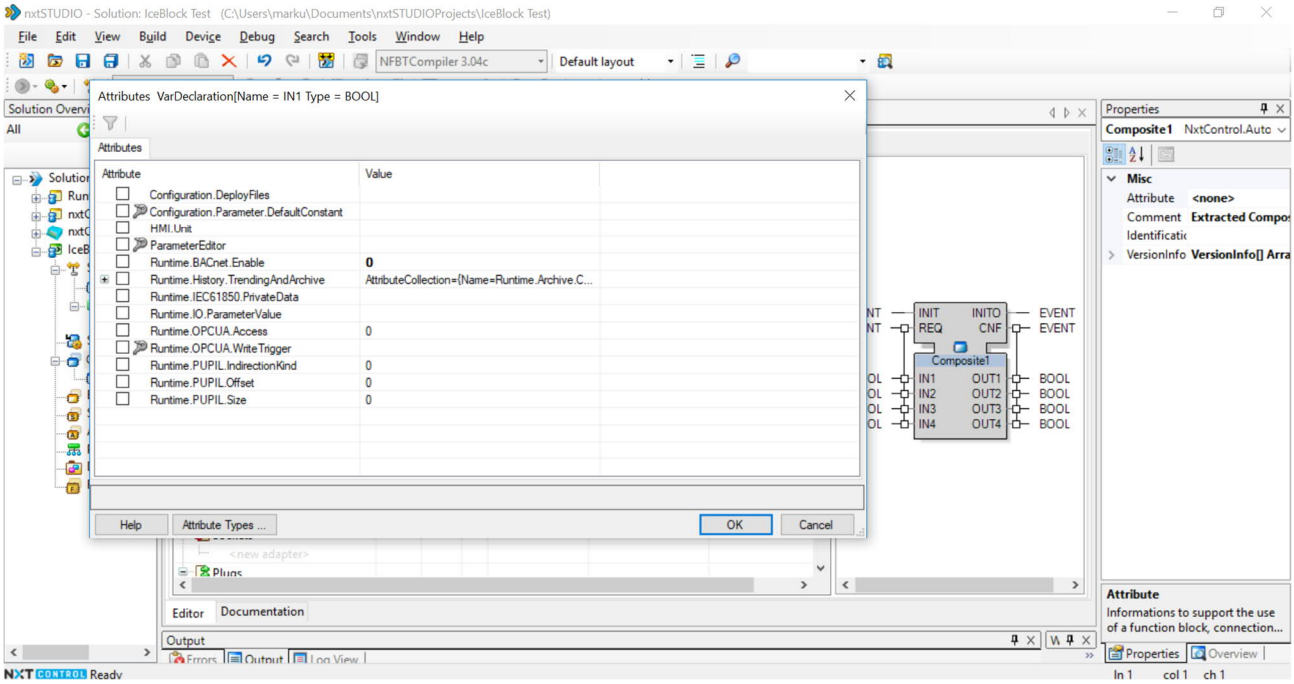


Figure 68. Attributes window opens.

Select the Runtime.OPCUA.Access and change the value to 2 if you want to read and write the value via OPC UA.

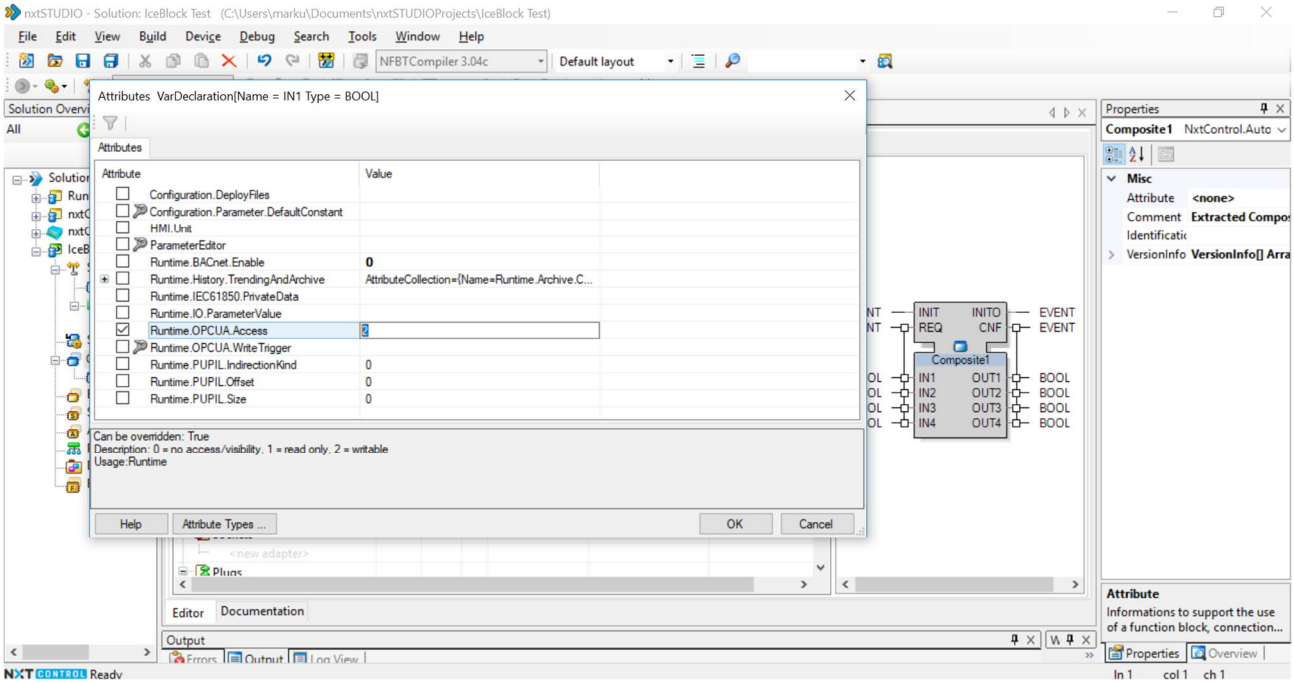


Figure 69. First, select the correct setting 0 or 1 or 2 for Runtime.OPCUA.Access line

Select the Runtime.OPCUA.WriteTrigger and add the name of the event that will be triggered by writing the value via OPC UA.

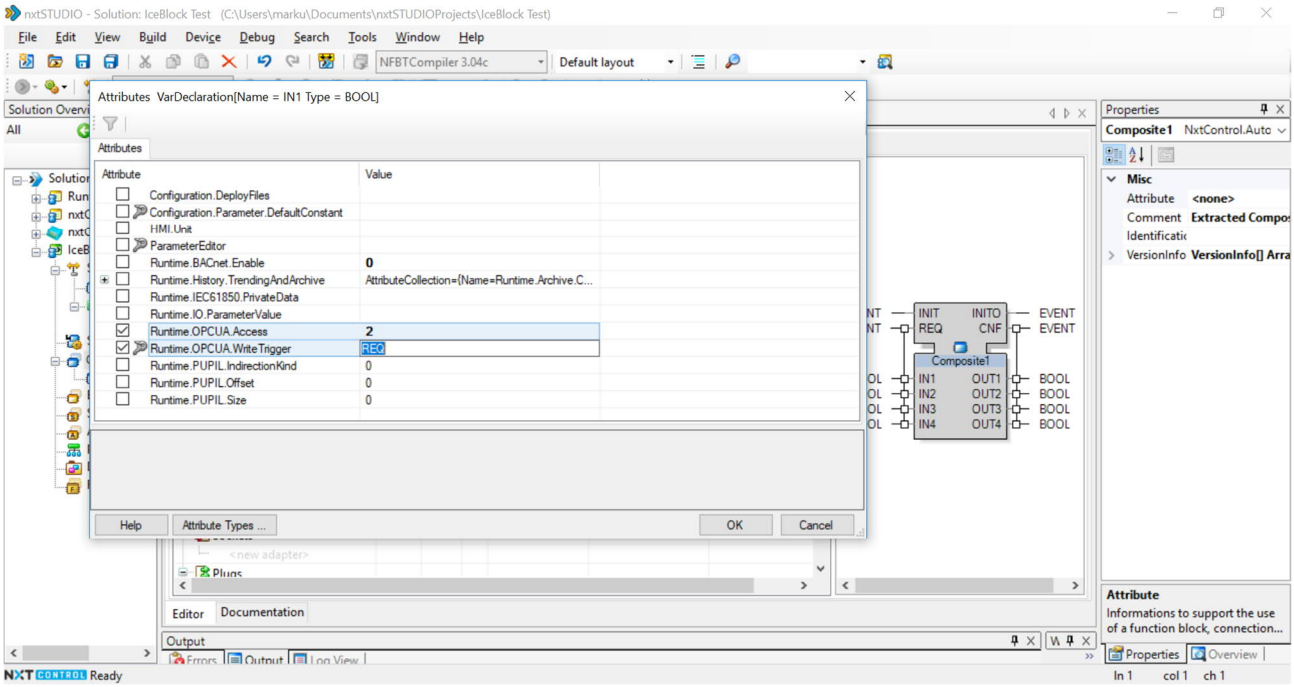


Figure 70. Then write the correct event input to be triggered to Runtime.OPCUA.WriteTrigger

On composite FB output signals, the selection is different.

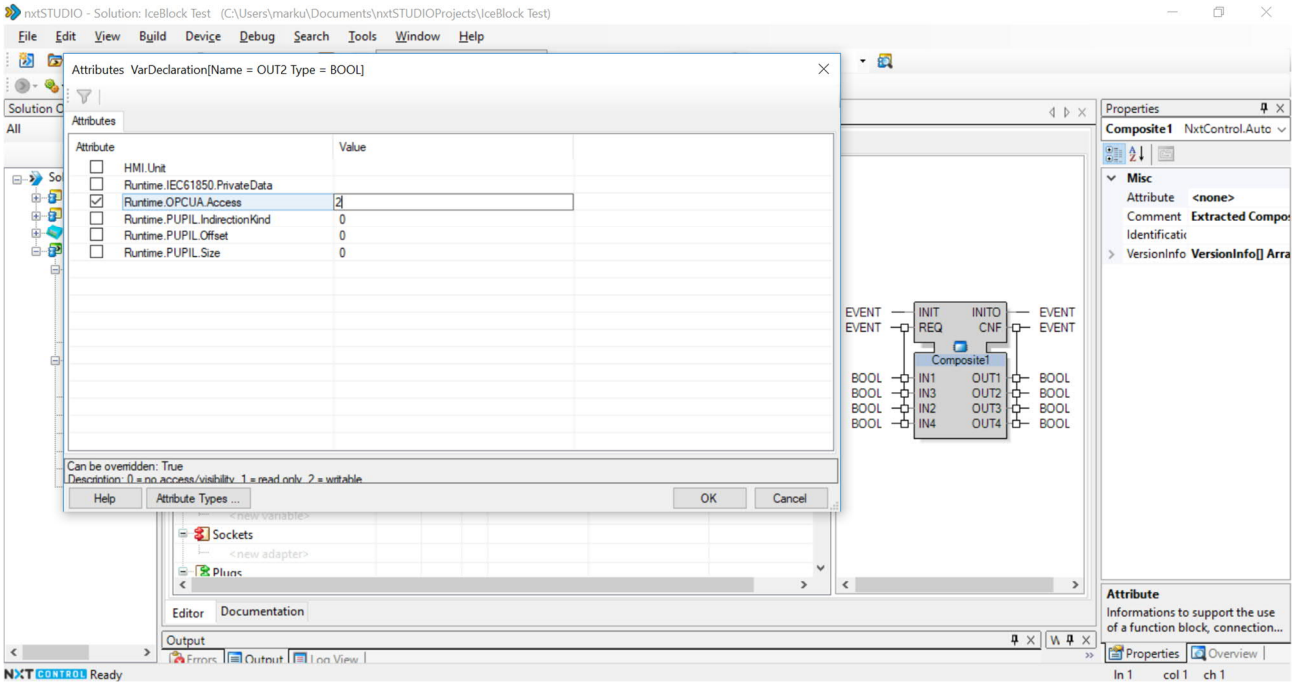


Figure 71. On output signal side there is only one setting to be selected for OPC UA

Make the attributes configuration to all variables that you want. In our example, the FB input has Name OUTx because wiring inside the FB is possible only one way.

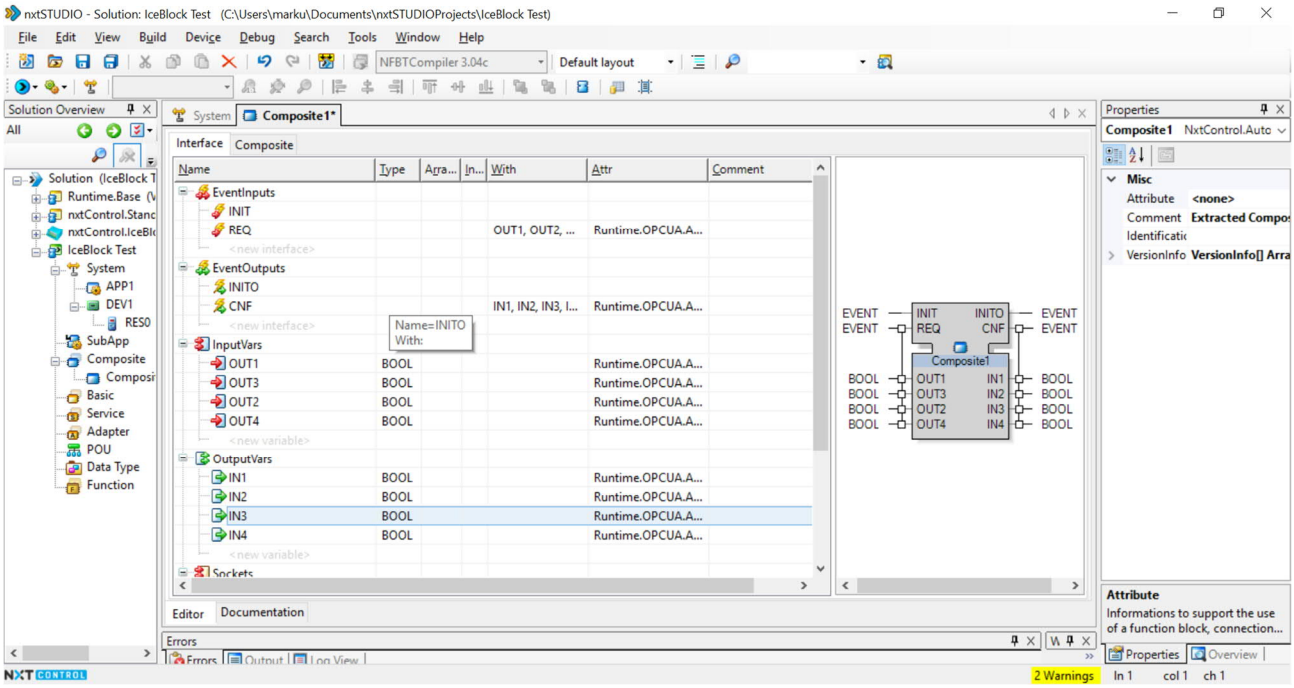


Figure 72. Attribute settings are needed to be done all needed variables separately.

Wiring inside composite FB is possible only in one way. This why the naming of the input and outputs are vice-versa.

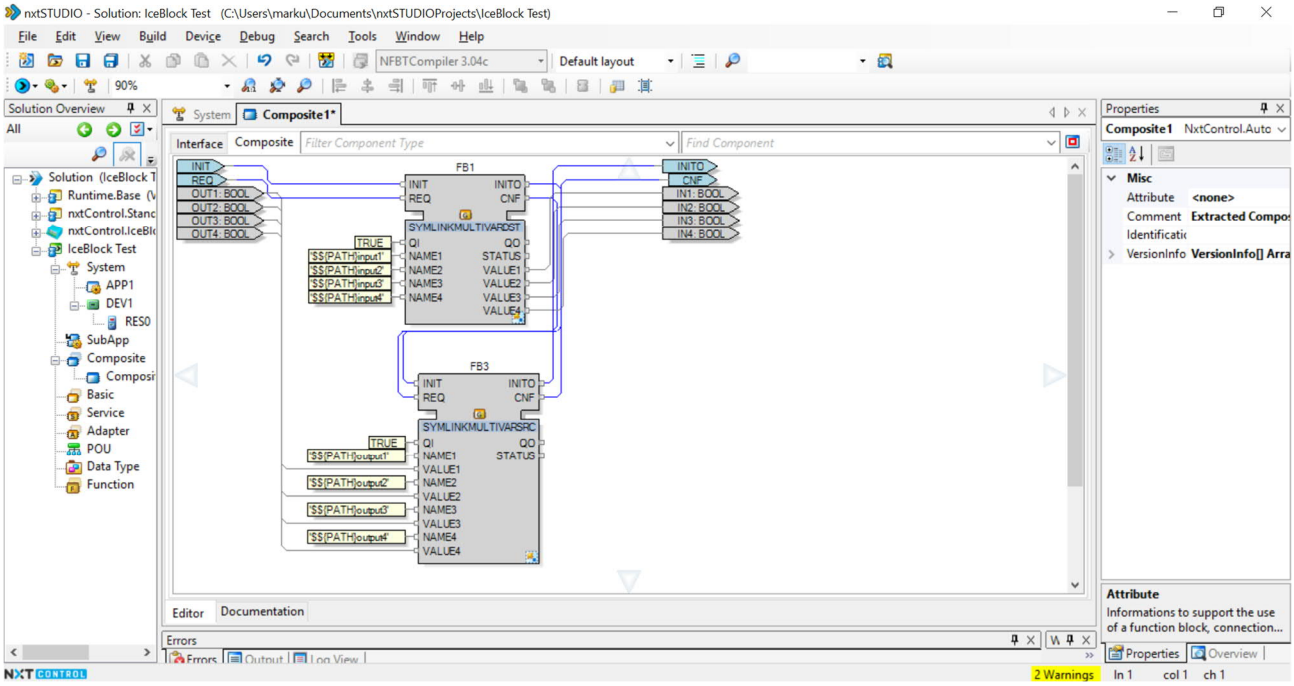


Figure 73. FB's inside composite FB needs to be wired to composite FB connections.

Now the variables of the composite FB are visible and writeable in the OPC UA.

8.1.3. Communication using TCPIO FB

We also tested the communication between IceBlock and YuMi Robot. For this, we used the socket communication. This way it is possible to send a string variable from device to another. On IceBlock side this requires the usage of the TCPIO FB. With this FB it is possible to create TCP server or client. In our project, we used the TCP server mode. This means that YuMi socket communication has to be created in client mode. YuMi program opens the communication to Iceblock and TCPIO FB replies to the same address where the last connection has come from.

This project we only shortly tested the functionality and verified the communication by sending short strings from the device to another. This is communication was needed to verify because the OPC and OPC UA communication are not possible between IceBlock and YuMi. IceBlock is not able to read the YuMi OPC signals or YuMi is not able to read IceBlock OPC UA signals. This is because both have server feature on these. There should be client features to enable communication via OPC UA or OPC. The following figure shows the settings needed for the TCPIO FB.

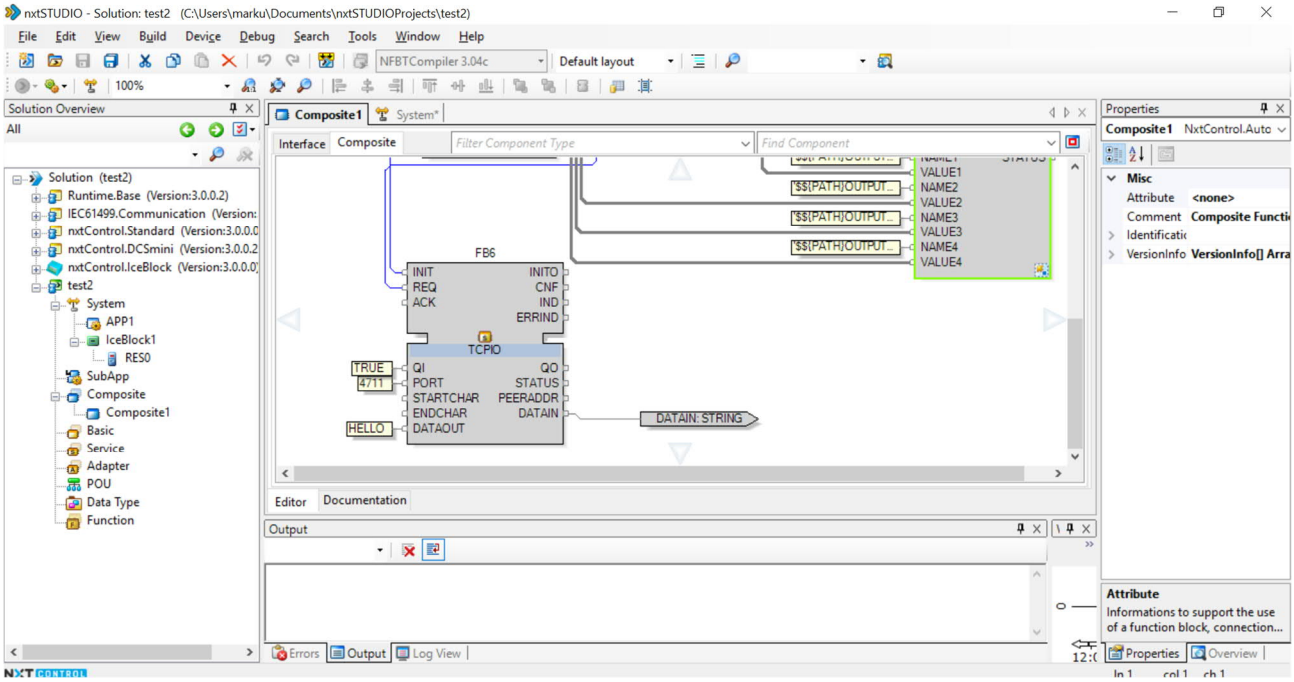


Figure 74. Example settings for TCPIO FB. DATAOUT is the string that will be sent.

At Yumi side, we need to use the Rapid commands related to the socket communication. In our tests, we used the following Rapid code.

```
MODULE MainModule
   CONST robtarget p10:=[[371.83,354.67,581.54],[0.105768,0.00768089,0.99132,-0.0777077],[0,0,0,4],[-
115.93,9E+09,9E+09,9E+09,9E+09,9E+09]];
   PERS bool yumi1bool:=TRUE;
   PERS bool yumi1bool2:=FALSE;
   VAR socketdev temp_socket;
   VAR socketdev client_socket;
   VAR string receive_string:="";
   PROC main()
                        SocketCreate temp_socket;
                        SocketConnect temp_socket, "192.168.0.30",4711;
                        SocketSend temp_socket\Str:="hello server";
                        SocketReceive temp_socket \Str:=receive_string;
                        TPWrite receive_string;
                        SocketClose temp_socket;
                         IF yumi1bool2 THEN
                                              MoveJ [[448.39,272.01,527.36],[0.0338877,-0.00978943,0.730647,-0.681844],[0,0,0,4],[-
137.916,9E+09,9E+09,9E+09,9E+09,9E+09]], v200, fine, tool0;
                                              MoveL [[452.60,159.13,538.93],[0.0152918,-0.0113091,0.724212,-0.689315],[0,0,0,4],[-
146.035,9E+09,9E+09,9E+09,9E+09,9E+09]], v100, fine, tool0;
                                              MoveL [[430.73,242.62,538.63],[0.039539,-0.0562581,0.7252,-0.685096],[0,0,0,4],[-
145.729,9E+09,9E+09,9E+09,9E+09,9E+09]], v100, fine, tool0;
                                              MoveJ [[78.84,183.29,217.54],[0.156812,-0.591869,0.668658,-
0.421898],[0,0,0,4],[109.442,9E+09,9E+09,9E+09,9E+09,9E+09]], v200, fine, tool0;
                                              MoveJ [[78.84,183.29,217.54],[0.156812,-0.591869,0.668658,-
0.421898],[0,0,0,4],[109.442,9E+09,9E+09,9E+09,9E+09,9E+09]], v200, fine, tool0;
                                              yumi1bool := TRUE;
                                              yumi1bool2 := FALSE;
                         ENDIF
   ENDPROC
ENDMODULE
```

The TCPIO FB was also used to test sending REST API commands to SEIT. This would have made the control of the whole MPP more flexible when SEIT jobs could be ordered separately and maybe even create mission from the Iceblock software. We tried to order certain job from SEIT missions, but we were not able to get through the HTTP error that occurred all the time. We got also some help from the computers science staff from the same building, but unfortunately, the outcome was not satisfying. This FB is working on a different layer than normal REST API. It would require more knowledge of how the FB really works and what form the REST API command should be written so it is understood at HTTP-level in SEIT.

9. Further paths to research

If it is decided to continue the development of the MPP, here are listed some of the practical issues to consider for creating better functionality.

- Further testing REST API via TCPIO FB or ABB Rapid socket commands. What would be the correct way to send POST command for creating the job for the SEIT. Some professional help who knows the TCP layer communication is required. Also, using the mission creation might have use when creating agile production.
- More testing of the accuracy of SEIT, when instructions for triangle reference tools are received from Milvus Robotics.
- More testing with REST API of SEIT. Deleting the job was not possible. Maybe some professional help that knows the REST API communication is required.
- Creating the functionality to stop working and get to the charger when battery level low. It Might be possible to get SEIT battery level information by using the REST API.
- Solving the problems with accuracy in the way that robots work object coordinates are enough accurate. Maybe coordinate adjustment is needed to be done via camera or some hardware that forces the MPP precisely on the correct place. We used the small tray to attached to MPP. This way is not possible in the all applications.

10. Reflection of the Project

The project task had variation during the project. in the beginning, we were not sure how much of the lift hardware was installed. It turned out the lift mechanism including 230V power supply, was all there. Also, the programming abilities of the SEIT were uncertain in the beginning. This reason we were not able to plan everything in advance. Mostly the time was used to learn the programming of SEIT, YuMi and IceBlock. Also, OPC UA and related software like, UA Gateway and UA Expert required some learning.

10.1. Reaching objective

At the end, the project was totally completed according to the adjusted objectives. Most of the expected outputs were obtained and the most important functions of MPP were well operated such as autonomous mobile positioning, wireless communication among components of the platform as well as with other production units in the factory. The final demo was successfully performed to professors, students, and visitors. Due to the constraint of time and limitation of the product as well as manual from the manufacturer, the objectives were adjusted to be suitable for the workload of the project and abilities of students and they were accepted by the instructors and advisors.

In the adjusted objectives, the degree of fine positioning requirement was lowered. There are internal and external reasons for the decision. The internal reason was the limitation of students for intensive robot programming for marketed products, such as advance ROS programming, webs and software programming languages which were necessary to access the robot data and control the robot base on those data. The external reasons were the limitation of SEIT100 and its documents from the manufacturer. To achieve the accuracy of ± 2.54 cm, SEIT100 requires triangle reference tools to be installed at desired stations. However, the CAD design was delivered at the end of the project work

and the manual for the tools was proposed to be delivered in July. The other way to obtain a very fine positioning is to add a feedback control for the MPP. At the moment, since it is not possible for us to access its positioning control signals from the source code, a potential solution can be adding a camera to the Yumi. The camera will help Yumi adjust its positioning according to the position of SEIT100. However, the development will require an amount of workload which will not meet the time requirement.

To handle with the moderate accuracy of the SEIT100, a solution was found to help Yumi to precisely picking workpieces. A tray was designed, 3D printed and installed to MPP to carry the workpieces for the robot. To adapt to the external component, the lift levels were monitored at different locations to avoid collision between the trays and the working environment. Besides, the energy management and Java program translation were excluded to meet the time constraints.

10.2. Timetable

Initial timetable was introduced long before the actual equipment arrived and thus most of the tasks which were planned turned out to be irrelevant. Such tasks as defining power consumption and controlling height of the lift were preset by the manufacturer, in other words, very little or even no time was spent on those. On the other hand, programming IceBlock and making it communicate with other production units appeared to be more time consuming than it was estimated at the very beginning of the project. To summarize, in general, workload was overestimated.

Besides defining the tasks, personal availability was not designed good enough, but it is an expected outcome, since planning all at once for the 5 month period is not realistic.

10.3. Risk analysis

Risks were categorized to internal and external risks. Many of them did not occur but some of them was noticed, fortunately none of them were critical and did not lead to failure of the project.

One of the internal risks was that MPP would not achieve required positioning. The expected workarounds were to do a mechanical fixture. More efficient way, for accurate positioning, is to use triangle method, where every station is equipped with an angled plate. SEIT could then position itself accurately with its Lidar sensors. Unfortunately, it was never tested because we discovered it too late during the course.

Another external risk was that Yumi could not be programmed well enough to do its job. Challenge was to create accurate assembly with Yumi because of SEITs inaccurate positioning. Problem was managed with 3D-printed table attached to MPP so that Yumi knew where the used pieces would be at all time. This problem can be fixed when machine vision or SEIT positioning plates are installed into the system.

10.4. Project Meetings

In the beginning, we had meetings every week. When the number of open points was getting smaller, the project team had mainly internal meetings and workshops. It was realized during the project that proper agenda and following it helped most the progress. Of course, the meetings ended up sometimes freely flying ideas about the possibilities of the project, until we noticed the next meeting was starting. Clearly named leader to keep up the order is helping a lot to get all needed issues done during the limited time of the meeting.

For this size project, all the project handling methods are oversized and creates only extra work to create stuff that no one follows. During the project, the extra work was minimized to get enough time to solve the obstacles in the project itself.

10.5. Quality management

In the Project plan we planned to measure quality in this project technically and operationally. Operational quality goal was met although documentation in the middle parts of the project were partially forgotten, but we still managed to increase project's outcome quality. This was achieved especially due our good verbal communication between group members and the project manager.

Technical quality goals were met and our product is working as planned considering the possibilities that systems that we used provided. Technical documentation succeeded by using Google Drive for documentation library. Main issue with achieving good technical quality was that we were unable to affect the accuracy of the SEIT100 robot.

11. Discussion and Conclusions

The project contained both hardware and software aspects. Hardware issues were easily solved on a go, while from the software side more troubles appeared. As it turned out neither of the team members had enough knowledge to establish communications between factory units, it could be said, that this topic was completely new for all of us.

Even though, as it was described in the previous chapters, all the goals have been more or less reached, there is still plenty of room for various improvements. Since there were some troubles with SEIT delivery at the beginning of the project, some time was lost and several possible improvements have not been implemented. To name some:

- Better positioning of the SEIT. In the last week of the project manufacturer introduced some markers for SEIT better positioning.
- Machine vision for Yumi. Introduction of some kind of camera for Yumi robot would make it less dependant on the positioning of the MPP.
- ROS program for SEIT. During the project we only worked with the interface provided by the manufacturer, which gives not much of a freedom for programming SEIT.
- Safety issues. Again, since SEIT interface appeared to be quite dumb, we did not manage to introduce the collision avoidance to the MPP.

Overall, the project was a nice experience, not only because it improved the project management skills, but because it was about making software and hardware work in tandem as it is done in real industry.

References

Milvus Robotics, nd. *Seit: Automated material transport* [viewed 26 May 2019]. Available from: http://milvusrobotics.com/products/seit

Milvus Robotics, nd. *Seit100 user manual*

Vyatkin, 2019. Flexible Intelligent Mobile Production Platform (MPP) in Industry 4.0-compliant Factory Floor, Project summary