

# **Task with Robotino 4.0 robot for demonstra- tion purposes**

LAB University of Applied Sciences

Bachelor of Engineering, Mechanical Engineering (Studies for incoming exchange stu-  
dents)

2023

Lenka Hájková

## Abstract

Author(s) Lenka Hájková	Publication type Thesis, UAS	Completion year 2023
	Number of pages 33	
Title of the thesis <b>Task with Robotino 4.0 robot for demonstration purposes</b>		
Degree, Field of Study Bachelor (UAS), Mechanical Engineering		
Organisation of the client LAB University of Applied Sciences		
Abstract <p>The objective of this thesis is to create a guide for working with Robotino 4.0 robots based on the example of one task with demonstration purposes. The research part of this work is dedicated to a brief summary of Robotino 4.0 hardware and software and the theory behind planning and navigation. Furthermore, the process of map and path creation in Robotino Factory is described, basic orientation in Robotino View 4 is outlined and principles of coding in this software are introduced on a demonstration task. This guide's main purpose is to help with recreating the task in a new environment and as some help to integrate Robotino robots into teaching at LAB University of Applied Sciences.</p>		
Keywords Robotino, Festo, path, mobile robotics		

## Contents

1	Introduction.....	1
1.1	Robotino 4.0 .....	1
1.1.1	Hardware.....	1
1.1.2	Software .....	4
1.2	Planning and Navigation.....	6
1.2.1	Degrees of Autonomy .....	6
1.2.2	Generations of mobile robotic system control .....	6
1.2.3	Navigation approaches .....	7
2	Map and path creation .....	10
2.1	Localization map creation .....	10
2.2	Postprocessing and navigation map creation.....	12
2.3	Path creation .....	13
2.4	Path for the demonstration task .....	15
3	Code creation .....	17
3.1	Orientation in Robotino View 4.....	17
3.2	Code for the demonstration task.....	19
3.2.1	Global variables.....	20
3.2.2	Main program diagram.....	20
3.2.3	Init Subprogram .....	21
3.2.4	Position Subprogram .....	22
3.2.5	QR Subprogram .....	22
3.2.6	Drive Subprogram .....	23
3.2.7	Lua code for QR positioning .....	25
3.2.8	Possible task expansions .....	26
4	Encountered difficulties.....	28
4.1	Connection to RoboTimo not possible .....	28
4.2	Missing manual for Robotino View 4 .....	28
4.3	Automated rack output element not compatible with Robotino gripper .....	28
4.4	Motor of the gripper not able to be configured.....	29
5	Summary and conclusions.....	30
	References .....	31

## Appendices

Appendix 1. Program in Robotino® View

Appendix 2. Lua code

## **1 Introduction**

### **1.1 Robotino 4.0**

Robotino is a robotic platform from Festo Didactic. The goal of this platform is to help understanding technology using an entirely different approach. It is used for educational, training and research purposes.

Robotino 4.0 helps understand topics such as mechatronics, programming, sensors, motor and drive technology, closed-loop control technology and image processing in a simple and straightforward manner. (Festo Corporation a.)

#### **1.1.1 Hardware**

Robotino uses omnidirectional wheels, and they allow movement in all directions. The interfaces offer the connection of further modules. Sensors get data from the components, drives and environment to the controller and rechargeable batteries make long-term operations possible. Robotino 4.0 also offers the possibility of expansion modules for specific applications. (Festo Corporation b.)

#### **Control unit**

The control unit can be divided into two parts. Main switch and Bridge. The function of the Main switch has a ring-shaped illumination that visualizes the operating status of Robotino. To switch the system on the Main Push Button is pressed. This action should last around three seconds to turn Robotino on or off. The indicator of a successful switch-on is a blue light. If the need to replace the battery during operation arises, the main push button must be pressed for one second beforehand so Robotino can automatically detect which battery has been removed and can draw power from the others. Another function of this button is the ability to interrupt the currently running control program or the connection to an external control computer. (Festo Corporation c.)

The bridge contains the control board with the embedded PC, the microcontroller and interfaces. The case of this unit should only be opened if the expansion card is to be inserted. Attention must be paid to the connection cable of the main switch as it can get damaged otherwise. The overall specifications are mentioned in Table 1. (Festo Corporation d.)

Table 1 - Specifications of Robotino 4.0 Control unit (Festo Corporation e.)

Parameter	Value
Type	Embedded PC to COM Express specifications
Operating system	Linux Ubuntu 18.04 LTS (64 Bit)
CPU	Intel i5 8th generation, 2.5 GHz clock, up to 4.2 GHz in Turbo mode, 4 physical cores with hyperthreading
Random access memory	8 GB RAM
Hard disc	64 GB SSD
Motor control	Microcontroller with 32 Bit-Microprocessor

### Drive system

The drive system uses the omnidirectional drive, which allows Robotino to drive in all directions and turn in a fixed position. This omnidirectional drive is made of three independent drive units consisting of motors, incremental encoders, gears and wheels. (Festo Corporation f.)

Each DC motor of type Dunkermotoren GR 42x40 drives an omnidirectional wheel and has an incremental encoder to measure the rotation. Their maximum speed and acceleration can be configured in the Webinterface which is part of Robotino software. The motor is controlled by the microcontroller and while coding the setpoint for the motor speed is specified. (Festo Corporation g.)

Based on values of the incremental encoders of the type Dunkermotoren RE 30-2-500, the motor controllers can adjust and regulate the real motor speed. They are also used to determine the position of the system. (Festo Corporation h.)

The ratio between each motor and wheel is 32:1 and this allows high accuracy while maintaining low speed. This ratio is achieved by using the S3M flat toothed belt from ConCar and the PLG 42 S planetary gear from Dunkermotoren. (Festo Corporation i.)

The three R3-1258-85 all-side wheels from Rotacaster on the sides of the Robotino allow movement in all directions and the rotation on spot. The running surfaces of the wheels are metallized to prevent electrostatic charges. (Festo Corporation j.)

## Sensors

The sensors grant Robotino the ability to recognize objects, simultaneous localization, and navigation. There are multiple types of sensors built into the mobile system.

The SK EKS011 bumper bar with sensors is attached to the bottom edge of the Robotino and its function is to interrupt the program execution and movement in the event of a collision. The collision protection sensor serves as a so-called safety edge. The two separate conductive areas in the switching chamber are short-circuited when the safety edge is pressed, generating a digital signal for the evaluation device. (Festo Corporation k.)

There are nine GP2Y0A41SK0F infrared distance sensors installed in the lower area of the chassis. They are arranged at an angle of 40° to each other. These sensors allow accurate detection of objects in a distance between 4 cm and 30 cm. They have an analogue output signal. (Festo Corporation l.)

The increase in the accuracy of positioning is achieved by equipped CruzCore R6093U gyroscope. There is no need for user programming as Robotino's operating system can recognize the gyroscope and uses its signal to correct the position without user input. (Festo Corporation m.)

The Intel Realsense D435 depth measurement system consists of several camera modules. The conventional webcam delivers a live image for evaluating navigation and obstacles and object detection. It also has an infrared projector and 2 infrared imagers. These help determine the distance to obstacles like in a 2D laser scan. (Festo Corporation n.)

The two SOEG-L-Q30-P-A-K-2L so-called reflected light sensors from Festo allow recognition of different colours and surfaces. They let the mobile system go along a defined path or to stop in a position with high accuracy. They are evaluated by a digital input of the I / O interface. (Festo Corporation o.)

For the detection of metallic objects on the ground is used the SIEA-M12B-UI-S inductive sensor from Festo. This can be utilized for path control and fine positioning. It is evaluated by an analogue input of the I / O interface. (Festo Corporation p.)

## Interfaces

The Robotino system allows connections by means of these seven interfaces:

- WLAN Access Point
- E/A-Interface
- Motor/Encoder

- USB
- PCI Express card slots
- Ethernet
- HDMI. (Festo Corporation q.)

### **Power Supply**

Up to four lithium-ion high-performance rechargeable batteries from Festool are accessible in the back of the robot and allow replacement. The promised battery life is up to 10 hours. The batteries can be charged with the Festool TCL 6 or SCA 8 quick chargers. Once the battery is recharged it can be reconnected during operation by sliding it into the provided guides until it clicks into place. Only one battery is required for operation. (Festo Corporation r.)

### **Modules**

Multiple modules can be mounted on the chassis. This includes the Tower and Segment, Laser range finder, Legacy: Electric gripper, Forklift, Electric gripper, Height adjustment, Interfacebox, LED signal lamp, Sensor package and Electrical gripper package. (Festo Corporation s.)

#### **1.1.2 Software**

The Robotino system is programmable in multiple languages, including C, C++, Java, and Matlab. The recommended programming platforms are Robotino View, Robotino Factory, Matlab Simulink, or LabVIEW graphical programming. The basis for more complex projects includes the frameworks Microsoft .NET, SmartSoft and the Robot Operating System (ROS). (Festo Corporation t.)

### **Webinterface**

This interface provides the user with functions for the control, configuration and maintenance of the robot system. For access to the Webinterface from Smartphone, tablet, or PC/Notebook the user has to connect to the Robotino Wi-Fi network, start an internet browser and enter the IP address of Robotino in the address bar. (Festo Corporation u.)

On the Home page, the user can view the currently installed hardware and software version and, in the power management, check the current state of charge of the batteries. (Festo Corporation v.)



On the Program page, the user can manage programs created with Robotino View. This management consists of starting the program, stopping the program, uploading the program, saving the program and deleting the program. (Festo Corporation w.)

On the Config page, the user can make settings as the program is executed when Robotino is started. (Festo Corporation x.)

On the Network page, the user can check and customize Robotino's network settings. This customization consists of adjusting the settings of the Ethernet interface and adjusting Wi-Fi Connection Settings. (Festo Corporation y.)

On the Control page, the user can manually control Robotino. This control includes Camera image and Manual movement control. (Festo Corporation z.)

On the Service page, the user can view the running processes on the Robotino and start and stop them. The changes are only displayed by clicking on the 'Reload' button. (Festo Corporation aa.)

On the IO page, the user can display the current information of the occupied inputs and outputs. The digital outputs and relays can be set manually. (Festo Corporation ab.)

## **Programming**

The programming can be done in multiple different environments and previously mentioned frameworks, and languages.

Robotino View is an interactive graphical programming environment for Robotino. This environment allows the user to create and execute the control programs. Its main function includes sequence programs that are displayed as Graftet, simultaneous control of more than one Robotino, representation of hardware components as function blocks and downloading and starting Robotino View programs directly on Robotino. (Festo Corporation ac.)

Robotino Factory is a touch-optimised App that allows the user to interact with Robotino's Mapping and Localisation features. (Openrobotino 2021.)

LabView is a graphical programming environment used to develop automated research, validation, and production test systems. (National Instruments)

## **Simulation**

Robotino SIM is a simulation environment used for experimenting with Robotino. This environment offers the user a virtual Robotino that the user can control with Robotino View. Its features include simulation of the behaviour of Robotino, Physics simulation and Integrated simulation of sensors. (Festo Corporation ad.)

## 1.2 Planning and Navigation

*In the case of a mobile robot, the specific aspect of cognition directly linked to robust mobility is navigation competence. Given partial knowledge about its environment and a goal position or series of positions, navigation encompasses the ability of the robot to act based on its knowledge and sensor values so as to reach its goal positions as efficiently and as reliably as possible. (Siegwart & Nourbakhsh 2004, 257a.)*

There are two competencies required for mobile robot navigation: path planning and obstacle avoidance. Path planning needs a map and a goal location and involves creating a trajectory that will allow the robot to reach the location. It requires the robot to decide what to do over the long term in order to achieve its goals, while the obstacle avoidance needs real-time sensor readings and involves changing the trajectory of the robot in order to avoid collisions. (Siegwart & Nourbakhsh 2004, 257b.)

### 1.2.1 Degrees of Autonomy

When speaking about control and navigation, there are three degrees of autonomy.

#### **The first degree of autonomy**

The mobile robot system is energetically independent. It does not need an external source of energy. This source is a part of the mobile robot system. (Řízení a navigace 1a.)

#### **The second degree of autonomy**

The mobile robot system is adaptable. It can adapt to dynamic changes in conditions and the work environment. These systems are able to complete the goal without or with minimal interference from the operator. (Řízení a navigace 1b.)

#### **The third degree of autonomy**

The mobile robot system is self-reconfigurable and has the ability to identify, localize and diagnose its own malfunctions and reconfigure itself to post-malfunction mode without or with minimal interference from the operator. (Řízení a navigace 1c.)

### 1.2.2 Generations of mobile robotic system control

The control of the mobile robot system requires parallel processing of different tasks:

- Collection and evaluation of data from sensor subsystems

- Transformation of data from external sensor system to data for creating an inner representation of the environment
- Planning of goal position of the mobile robot system
- Controlling of execution of transportation and work tasks of the mobile robot system
- Controlling of the functional group of actuators in the subsystem of mobility, action subsystem and end gripper. (Řízení a navigace 2a.)

### **First generation**

With programme control – mobile robot systems are equipped with flexible control. They are applied in an environment with precisely structured operational conditions and special adaptation based on the technological determination of the mobile robot system. Elements and control systems are deterministically arranged, and they are not connected to the environment by feedback. (Řízení a navigace 2b.)

### **Second generation**

With adaptable control – automatic control – mobile robot systems are equipped with extensive sensory subsystem allowing the adaptation to the incomplete determination of the environment in the implementation of mobile robot system tasks. Elements and control systems are self-acting and self-regulating systems with feedback. They are systems with controlled (regulated) dynamics of the system. (Řízení a navigace 2c.)

### **Third generation**

With artificial intelligence – intelligent control – mobile robot systems are equipped with extensive sensory subsystem supplemented with the ability to create knowledge that allows understanding and recognition of the operating environment, create a model of the environment, accept decisions about behaviour change in the performance of activities. Elements and control systems can change their arrangement but also their own principles. (Řízení a navigace 2d.)

#### **1.2.3 Navigation approaches**

The navigation of a mobile service robot represents a path planning problem of an autonomous mobile robot system from the starting position to the target position, position includes orientation in the work environment, without collisions with any obstacle occurring on the traffic route. (Řízení a navigace 4a.)

Additional requirements, for performing movement of the mobile service robot, can be:

- Path parameters (time, speed, shortest distance, etc.)
- Safe obstacle avoidance distance
- Movement direction parameters (directly, bypassing obstacles from the right, etc.).  
(Řízení a navigace 4b.)

Path planning, regarding the general movement of the body in general space, is performed on one of the two levels.

### **Global navigation**

The goal is to plan the path of the movement from starting position to the goal position based on assigned parameters. The position of the mobile service robot is determined in a global coordination system using absolute coordinates. Knowledge of the geometric and topologic properties of the objects that are placed in the work environment is used as well as the knowledge of the movement of objects in the defined environment. (Řízení a navigace 4c.)

### **Local navigation**

The goal is to plan the path of movement in relation to another object on the path of its movement whether it is standing or moving. Therefore to determine the position of the mobile service robot in relation to the second object and subsequently plan the local conflict-free movement trajectory and the behaviour of the mobile service robot in such a way as to avoid conflict situations. (Řízení a navigace 4d.)

These methods can be used for generating local collision-free trajectories:

- environment-based path planning
- model-based trajectory planning. (Řízení a navigace 4e.)

Global navigation is based on knowing the location and orientation of the mobile service robot in the global coordinate system and determining the optimal movement trajectory to the goal position. (Řízení a navigace 4f.)

### **Relative navigation**

This method utilizes parameters that can be measured directly on the mobile service robot without their direct relationship to the environment. The zero point of the global coordinate system is usually identical to the starting position of the mobile service robot. The down-

side of this method is that the position error increases continuously. (Řízení a navigace 4g.)

### **Two methods used to realize relative navigation.**

Odometry, which is based on kinematic models of mobile service robot, stored in the control system, based on which it is possible to determine the change in the position of the mobile service robot depending on the change of the position of the action elements of the locomotion system (e.g., wheels), measured by the corresponding sensor (e.g., incremental sensor). The accuracy of the method depends on the stability of the contact with the base and the accuracy of the size of the action element of the locomotion system. The method is used in differentially controlled mobile service robot, or Ackerman steering. (Řízení a navigace 5a.)

And Inertial navigation, which is based on acceleration measurements of the mobile service robot. Linear acceleration is measured by an accelerator and rotational gyroscope. Position, or position change increments are calculated by twice integrating the measured value. The method is not usable for mobile service robots moving at low speeds. (Řízení a navigace 5b.)

### **Absolute navigation**

This method uses reference points with a known position in the global coordinate system to determine the position of the mobile service robot. The position is determined to these points. (Řízení a navigace 4h.)

### **Two methods used to realize absolute navigation.**

Trilateration, which is based on defining the position of the mobile service robot by its distance from reference points (GPS systems). (Řízení a navigace 5c.)

And triangulation, which is based on determining the position of the mobile service robot by using measurements of three angles between reference points. On basis of these values, the mobile service robot control system calculates its position in the global coordinate system. Sensor systems working with the laser effect, electromagnetic radiation, or acoustic waves are used for these calculations. The calculations of the change in orientation are possible in combination with the relative method or are applied as a separate orientation measurement system. (Řízení a navigace 5d.)

Listed options are burdened with inaccuracies, or are complex, therefore special compasses are being intensively put into practice. (Řízení a navigace 5e.)

## 2 Map and path creation

Robotino Factory is a software that allows mapping and localisation. To be able to use it, there needs to be a Robotino Robot connected first. After connecting to the Robot's Wi-Fi, there will be a possibility to connect the software to the robot. Figure 1 shows the icon under which the space to input the IP address of the Robotino is located. This address might differ from the one on the plate of the Robotino. For the case of RoboMikko, the address is 192.168.0.1

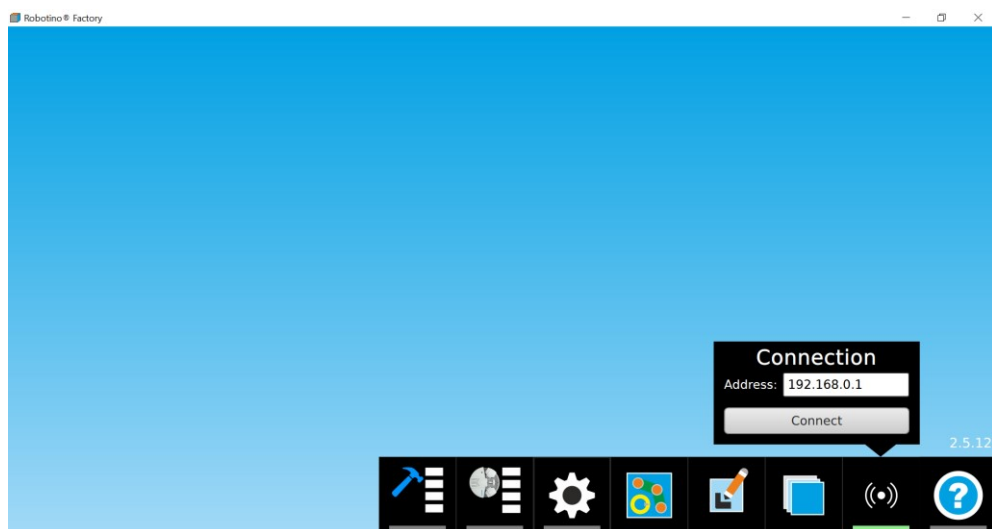


Figure 1 - Connecting to Robotino Factory

### 2.1 Localization map creation

After a successful connection, the software allows access to the maps already saved within the robot and the possibility to create new ones. These options are accessible under the highlighted icon of the Map Management dialog shown in Figure 2.

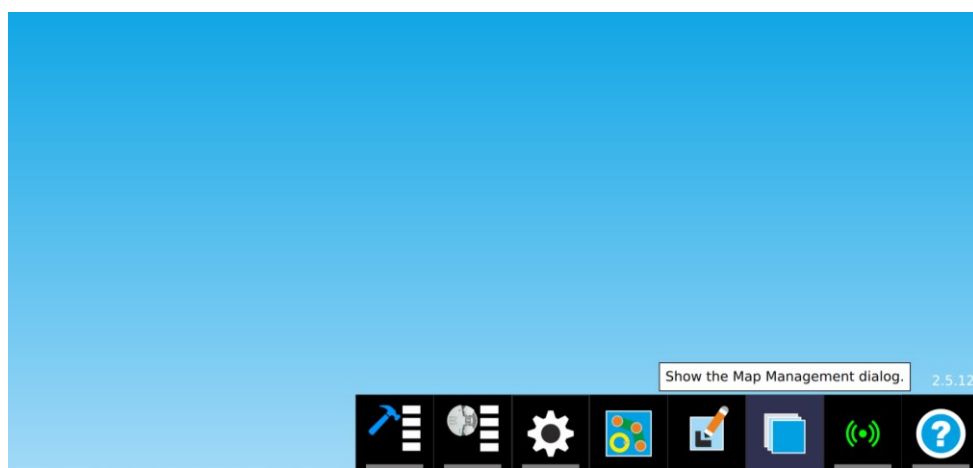


Figure 2 - Icon of the Map Management dialog

The Map management dialog shown in Figure 3 allows the user to scroll through the list of already created maps on the left side of the dialog and then either set it as a default map, delete the map or if there is a need for a new one, create it. To exit the dialog, in the down right corner is an exit button.

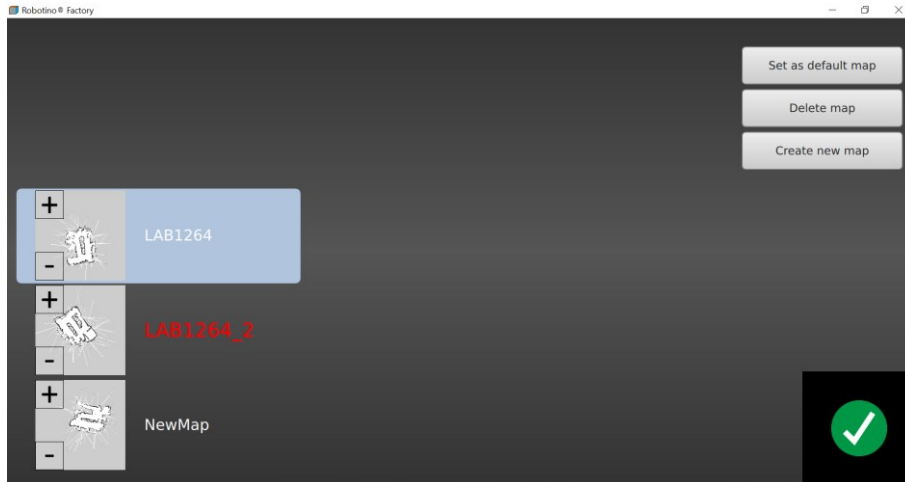


Figure 3 - Map management dialog

To create the new map, the user must put in a unique name and then continue the process. Because there can be multiple robots connected to Robotino Factory, there is part of the process to choose one of them which will do the mapping. Once selected, the user needs to move the robot around the room, so the integrated camera can take the layout in. Robotino Factory offers a way to directly control the robot, but this can be also done through Browser UI from another device like a laptop, tablet, or even mobile phone. Figure 4 shows how the process looks like. The user can see the progress in real time and can adapt the movements accordingly. To get the information needed, slower movements and periodical 360° rotations can create an accurate map.

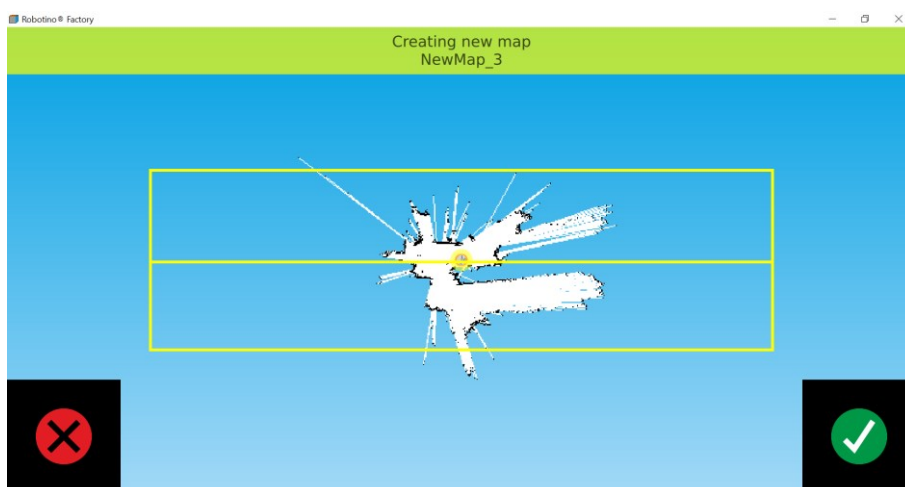


Figure 4 - Map creation process

The process can always be exited by the red button in the down left corner and once the map is completed, the green button in the down right corner is used to save the new map.

## 2.2 Postprocessing and navigation map creation

After the map is complete, the Robotino Factory software offers options for postprocessing. Figure 5 shows the highlighted icon of the Navigation Map Editor.

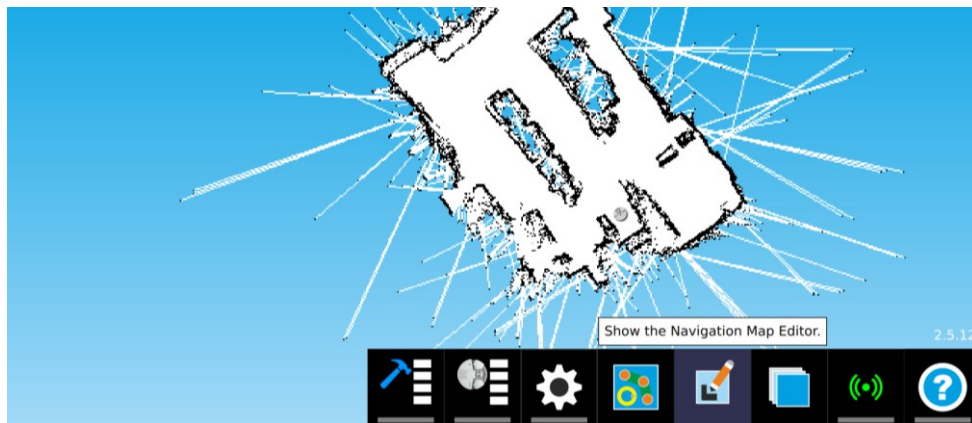


Figure 5 - Icon of the Navigation Map Editor

This editor shown in Figure 6 offers five different features. From the top, the first feature is to activate the pointer mode. In this mode, the user can move and scale the map. The second feature is the pencil mode. In this mode, the user can paint forbidden areas. The third feature is the rubber mode. In this mode, the user can mark areas as free of obstacles. The fourth one is the reset the navigation map feature. This feature allows the user to reset all the changes and return to the blank created map. The last feature is used to filter out small regions marked as obstacles, making the map clearer.

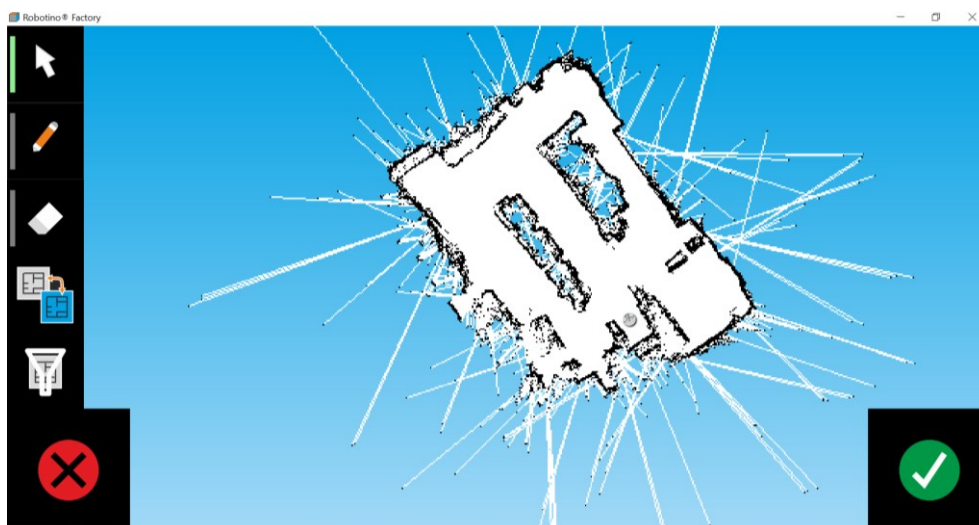


Figure 6 - Navigation Map Editor



Figure 7 shows the edited map. The green button in the down right corner saves the changes while the red button in the down left corner discards them.

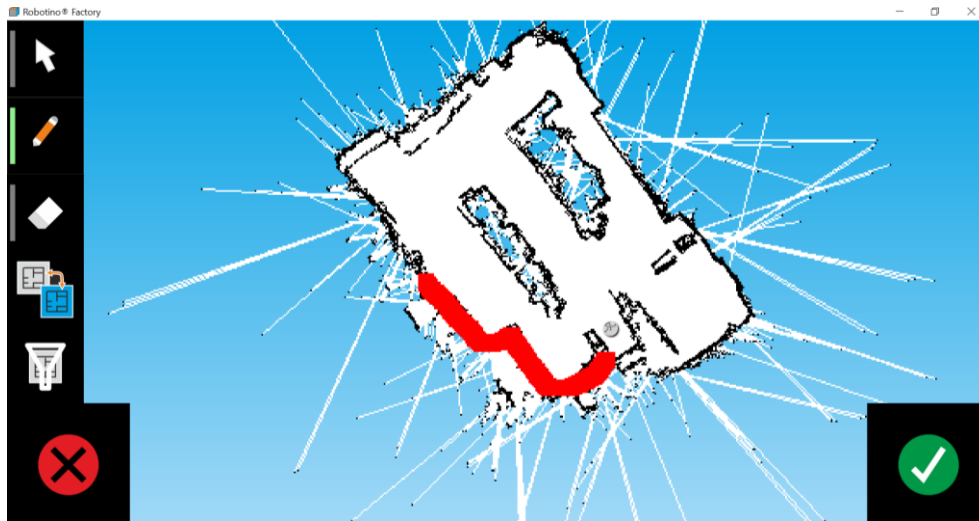


Figure 7 - Edited map

### 2.3 Path creation

Once the map is complete and edited, the Robotino Factory also offers the Path Network and Pose Editor. Information from this editor can be shared with Robotino View and used for coding. Figure 8 shows the highlighted icon of this Editor.

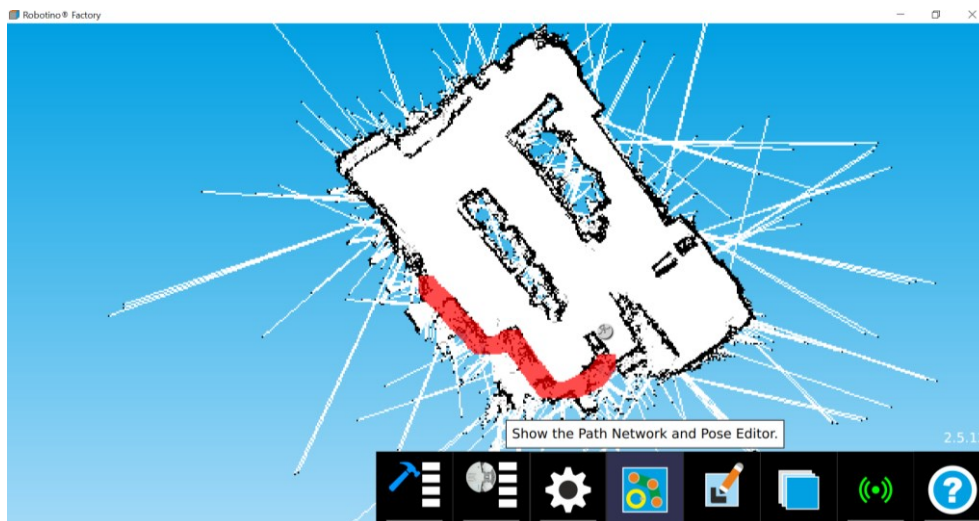


Figure 8 - Icon of the Path Network and Pose Editor

This editor shown in Figure 9 offers five different features. From the top, the first feature is to activate the pointer mode. In this mode, the user can move and scale the map, move path nodes, click path nodes to display the path nodes dialogue and drag path width arrows to change the path width. The second feature is the path creation mode. In this

mode, the user can click into the map to place the first node of the new path, click into the map to place the second and more path nodes and when the path is finished, click the accept button next to this button. By entering the pointer mode during the path creation, the process of the new path is discarded. The third feature is the add pose mode. In this mode, the user can click on the map to create a new pose. The fourth one is the add station mode. In this mode, the user can click on the map to create a new station. The last feature is a create navigation area mode. In this mode, the user can click into the map to create a polygon defining an area where the robots are using the free navigation.

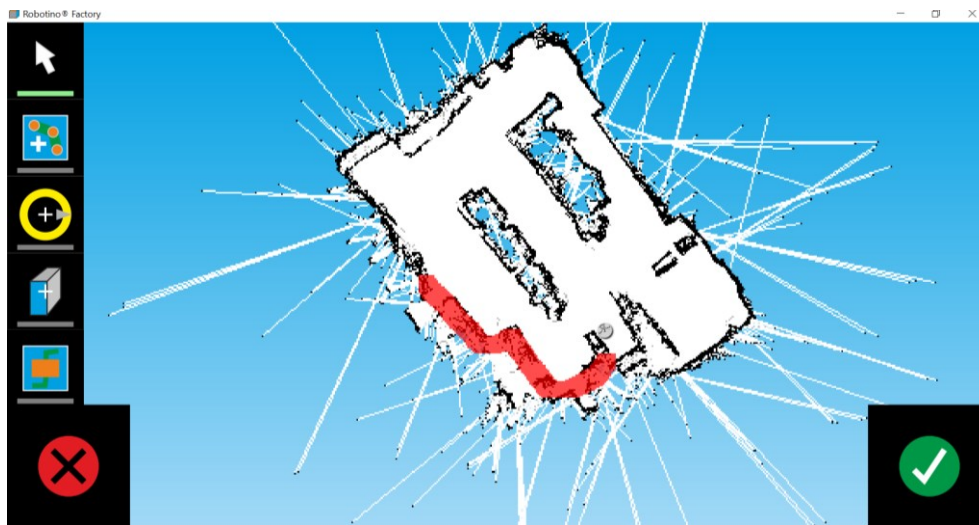


Figure 9 - Path Network and Pose Editor

The path node dialogue shown in Figure 10 allows the user to remove the path node, remove the whole path, insert the path node, reverse the path, connect or split the path and choose the path type.

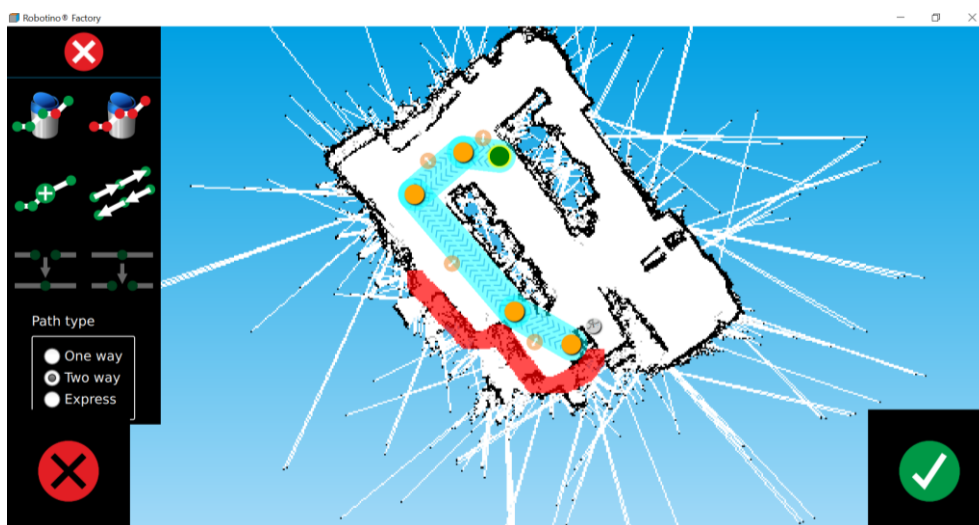


Figure 10 - Path node dialogue

The added poses can be edited in the Pose dialogue shown in Figure 11. The user can get the view centred on the pose by clicking the magnifier icon, enter move pose mode, in which it is possible to move and rotate the pose, remove the pose, and select the type of the pose.

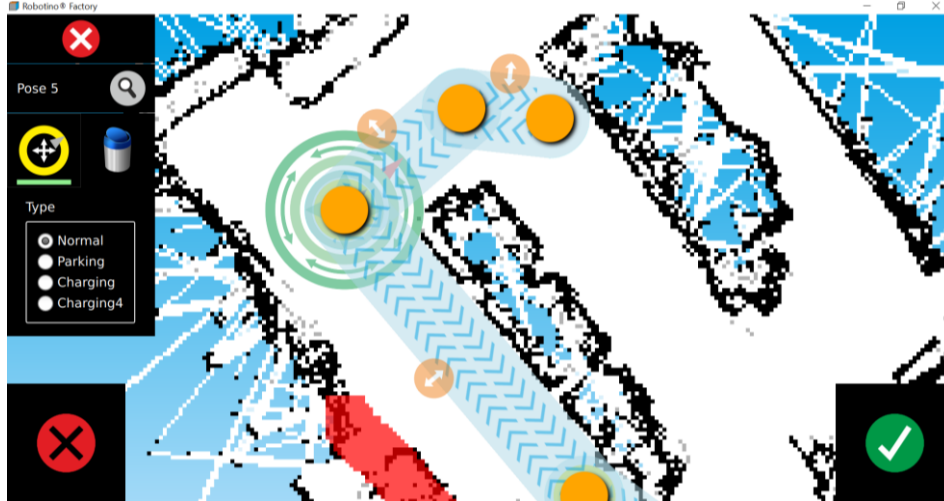


Figure 11 - Pose dialogue

It is notable that if the red button in the down left corner is selected, all paths get discarded. The poses stay, but the path connecting them is lost. Even if the editor is accessed for the second time, all paths even from the previous work are lost.

## 2.4 Path for the demonstration task

The finished path for the demonstration task is shown in Figure 12. It is a closed loop of one directional path with four positions. This path is also saved alongside the map in the Robotino RoboMikko and can be accessed after connecting to this robot from any device with Robotino Factory installed.

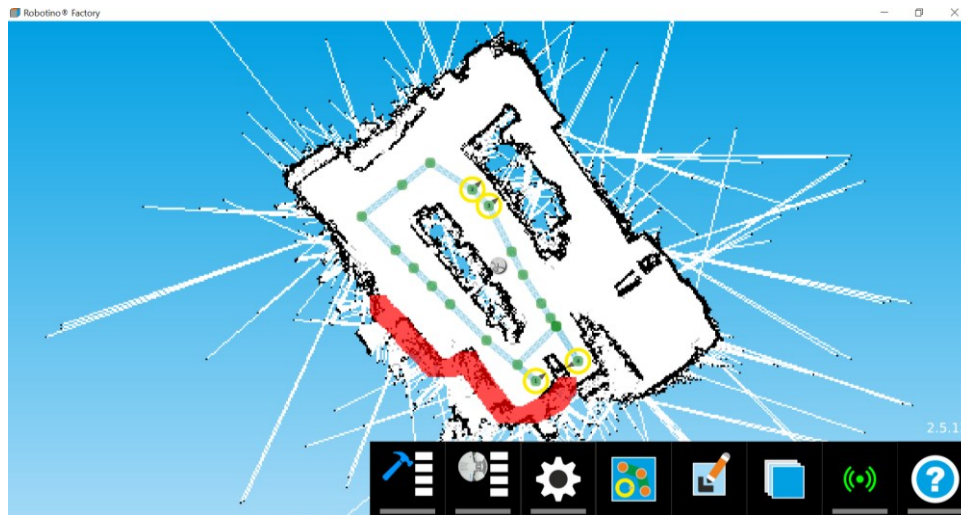


Figure 12 - Finished path

### 3 Code creation

#### 3.1 Orientation in Robotino View 4

Robotinos can be directly coded through the Robotino View software. This programming works through function blocks and is divided into several subprograms connected by the diagram of the main program shown in Figure 13. In this tab, there is the possibility to add more blocks, branches and jumps into the diagram by clicking the buttons on the left side of this tab. Each of these blocks can be assigned a subprogram that can be either chosen from already created subprograms or the user can create a new one. This assignment is done by clicking on the second row in the block. There is also the possibility to change the conditions connected to the blocks by clicking on them and simply rewriting them.

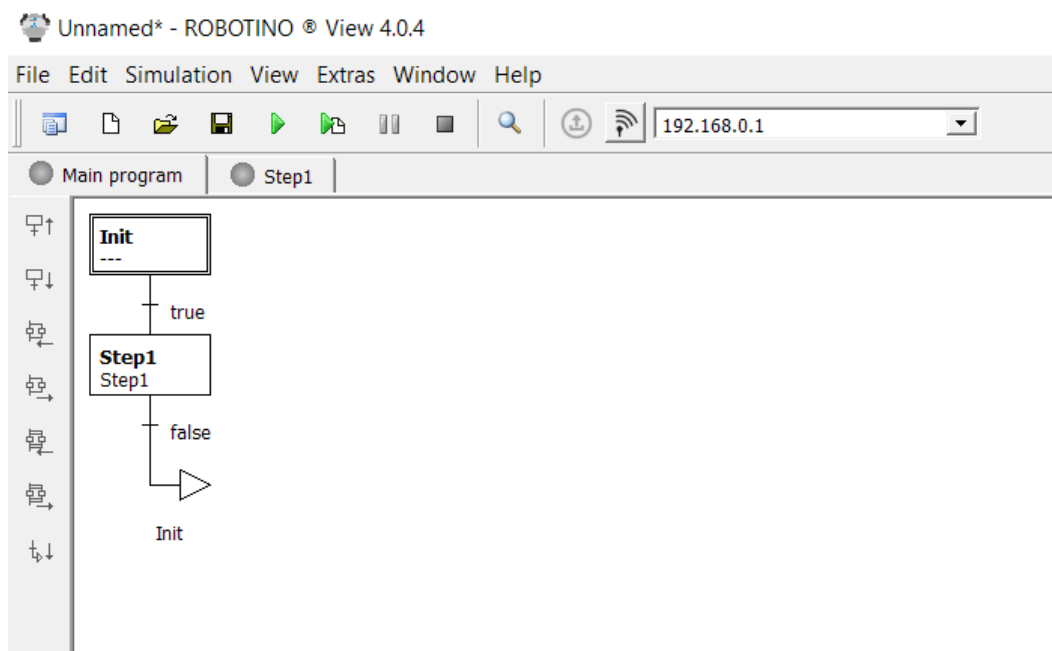


Figure 13 - Main program tab

In addition to the program diagram, there is also the possibility to add global variables in this tab. Figure 14 shows the right side of the tab where this option is located. By clicking on the Add button, the 'Add variable' dialogue gets open. In this dialogue, the user can choose the name for the variable and also its type. In Robotino View 4 this choice is between four different types. The float type that stores both whole numbers as well as fractions. The pose type offers the opportunity to store three float values. The first number represents the x coordinates, the second one represents the y coordinates and the third one represents the orientation in degrees. The string type lets the user to store a string of text and the floatvector type allows to store multiple float numbers. (Openrobotino.)

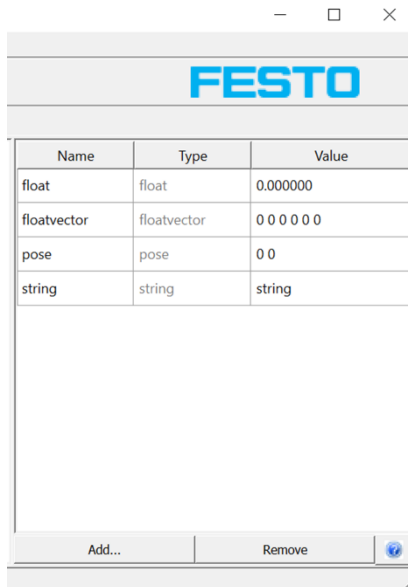


Figure 14 - Global variables

The subprograms are accessed from the top part of the tab by simply clicking on the name of the subprogram. All subprograms start as a blank sheet upon which the user can click and drag function blocks. These function blocks can be found on the right side of this tab divided into files as shown in Figure 15.

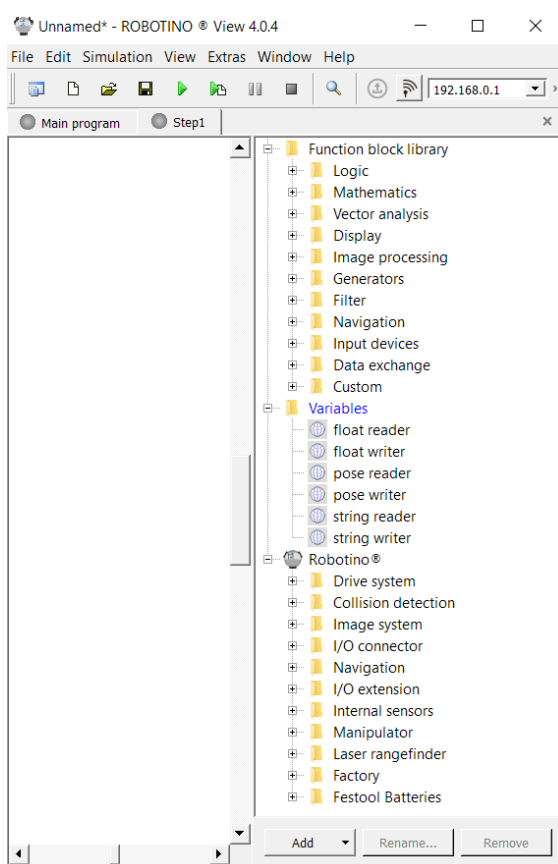


Figure 15 - Function blocks list

In these files are function blocks that can be connected together to form the code of the program as well as previously defined global variables. To connect these blocks, the user only needs to click on the output square of one block and then the input square of the second block, and these blocks will get connected with a line. By hovering the mouse over these squares, the user can gain insight into what each input and output represents. There is also a possibility to add more Robotinos by clicking the Add button and choosing Robotino in the list of options. Right click on the function block located in the subprogram and choosing the Help button opens a Help window.

It is also possible to show what values each input and output has. By using the keyboard shortcut Ctrl+D, the values are displayed as shown in Figure 16.

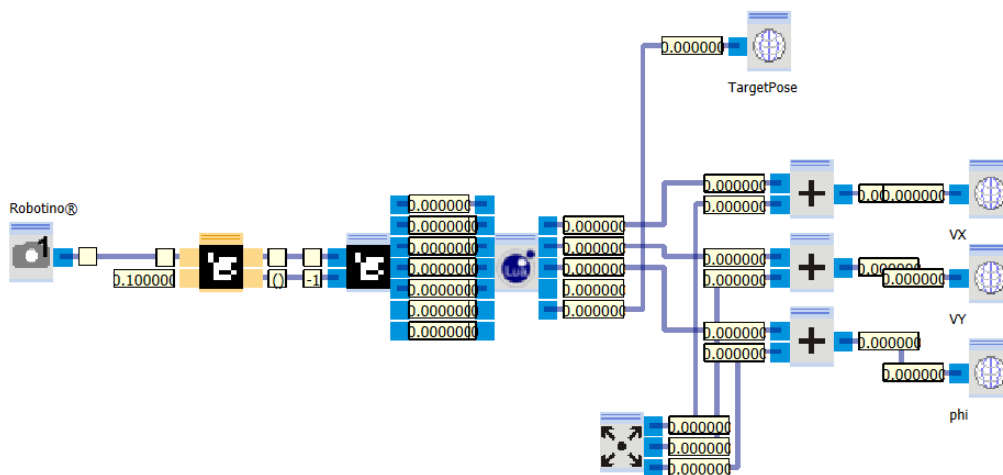


Figure 16 - Displayed values

### 3.2 Code for the demonstration task

The task is for the robot to follow the created path on which are two pick-up and two drop-off positions. These positions are clearly marked in Robotino Factory and serve as a rough estimation of the required position. Once the robot reaches this position, it needs to recognise a QR code and position itself based on this mark. After it position itself, it moves to the next position. The code works in an endless loop so after it reaches the last position, the code has to reset. The structure of the code has been inspired by the code used in the thesis: Installation of Robotino 4.0 (Sairanen 2021, 34-40.).

### 3.2.1 Global variables

Global variables needed for this task are shown in Figure 17. Variable CycleDone serves as a bool type variable and it keeps track of whether the robot has reached the end of the path (value 1) or is still on its way (value 0 – default). Variable PositionPoints is there to keep track of how many defined positions are there on the path. Variable Step keeps track of which position point ID is next on the way. StepIncr is another bool type variable and when it gains the true value (value 1), it triggers the increase in variable Step. Variable TargetPose keeps track of how many requirements of the position in relation to the QR code the robot actually fulfils. Variables VX, VY and phi are used to set the velocity, in the x direction, y direction and rotation respectively, while positioning from the roughly estimated position to the required position based on the QR code.


Name	Type	Value
CycleDone	float	0.000000
PositionPoints	float	0.000000
Step	float	0.000000
StepIncr	float	0.000000
TargetPose	float	0.000000
VX	float	0.000000
VY	float	0.000000
phi	float	0.000000
Add... Remove 		

Figure 17 - Global variables of the demonstration task

### 3.2.2 Main program diagram

The diagram itself shows when each subprogram will be running, and after which condition it ends. In this case shown in Figure 18, the subprogram of Initiation (Init) will start first and after it runs through, the subprograms located in blocks of code Position 1 and Drive will start. From the diagram, it is clear that they both run simultaneously. While the Drive subprogram runs for the whole duration of the loop, the block of code Position 1 has an end condition. When the variable Step is equal to two, the subprogram called by the block Position 1 ends, and the subprogram called by the block QR 1 starts and will run until the variable TargetPose is equal to three. It is possible to use the same subprograms in different blocks of the diagram, however these blocks do need to have different names. The whole diagram is shown in Appendix 1. The end condition is set as true, so the program



continues in the loop. Should it be set as false, the program will run through the path once and then stop.

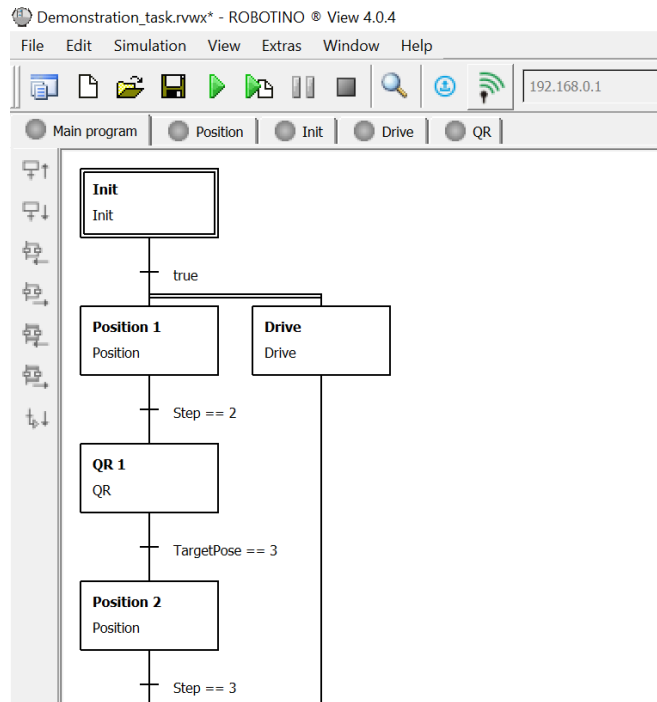


Figure 18 - Main program diagram for the task

### 3.2.3 Init Subprogram

The function blocks used for this subprogram are shown in Figure 19. The Sample and hold element block was dragged and dropped from the Function block library → Logic → Triggered operations file, and the Constant block from the Function block library → Generators file in the function block list.

This part of the code allows the Step variable to reset if the Cycle has been completed (CycleDone variable gains the true value) when the loop starts anew.

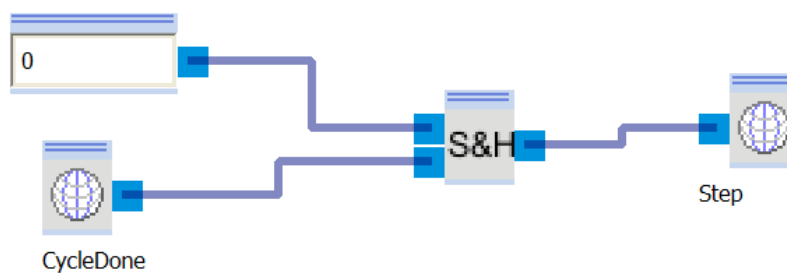


Figure 19 - Subprogram Init (Adapted from Sairanen 2021)

### 3.2.4 Position Subprogram

The function blocks used for this subprogram are shown in Figure 20. The GoToPosition block was dragged and dropped from the Robotino → Factory file in the function block list. This block allows the user to choose the ID of the Robotino this 'command' is for and the Position ID of the position to get to. As this demonstration task is done with only one Robotino, the first input can be left without changes, and the second input the Position ID is based on the variable Step, which keeps the value of the next unreached position. As an output, this block allows to keep track of progress, what should happen if there is an error encountered, what should happen after the job is finished and the job ID.

In the demonstration code, only the option of what will happen once the job is finished was used and when the program gets the signal of the job finished, the variable of StepIncr will be assigned the true value. As a default when encountering some obstacle, the Robotino will wait until this obstacle is removed and if this action takes too long, an error will appear and the Robotino will stop and abort the task.

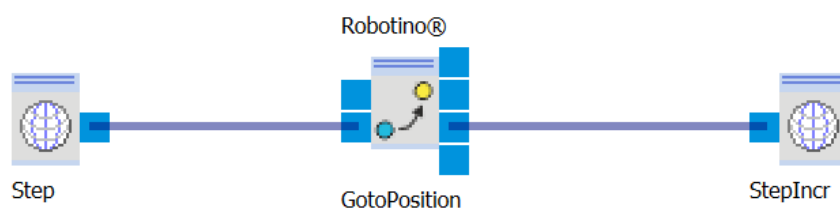


Figure 20 - Subprogram Position

### 3.2.5 QR Subprogram

The function blocks used for this subprogram are shown in Figure 21. The Camera #1 block was dragged and dropped from the Robotino → Image system file, the Marker detection and Marker position blocks were dragged and dropped from the Function block library → Image processing file, the Lua script block was dragged and dropped from the Function block library → Custom file, the Control panel block was Function block library → Input devices file and the Addition block was dragged and dropped from the Function block library → Mathematics → Arithmetic operations in the function block list.

This part of the code allows detection of the QR code generated by the Marker detection block by right clicking on it and choosing the properties button and then in the dialogue clicking on the Generate markers button. Further than that, this code detects its position in relation to this mark. The Lua code was added for processing these outputs into the

needed shift to the proper position. This shift is realized by setting the velocity in the x and y axes and the rotation speed. The code also keeps track of how many of the specifications of the target (distance in x and y axes and rotation) pose has been reached.

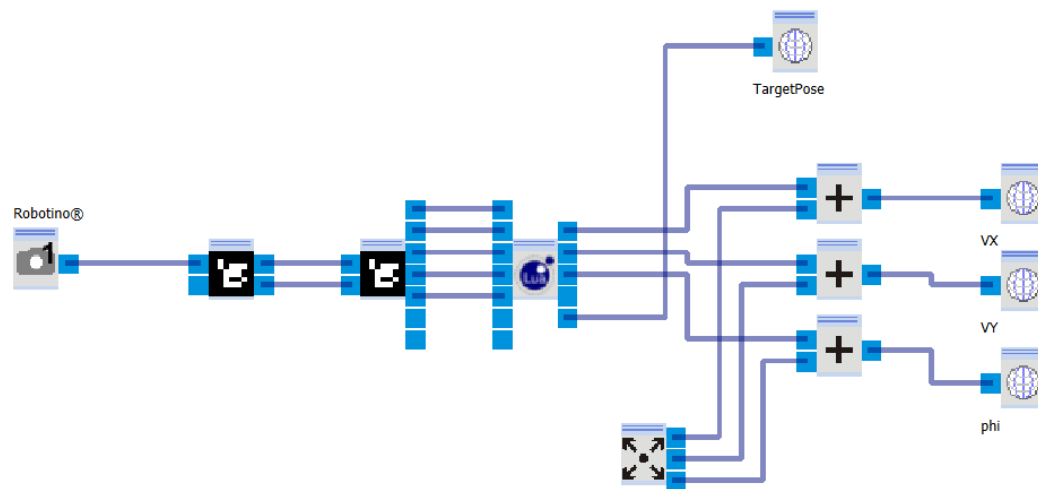


Figure 21 - Subprogram QR (Adapted from Sairanen 2021)

### 3.2.6 Drive Subprogram

This subprogram is running for the whole time in a parallel branch. It contains multiple separate parts of the code.

The function blocks used for the Drive system part of this subprogram are shown in Figure 22. The Omnidrive, Motor #1, Motor #2 and Motor #3 blocks were dragged and dropped from the Robotino → Drive system file, and the Bumper block from the Robotino → Collision detection file in the function block list.

This part of the code takes the values of velocity in axes x and y as well as rotation speed while positioning with the relation to the QR code and directs Robotino to move accordingly. If the bumper recognizes an obstacle, the brake is deployed.

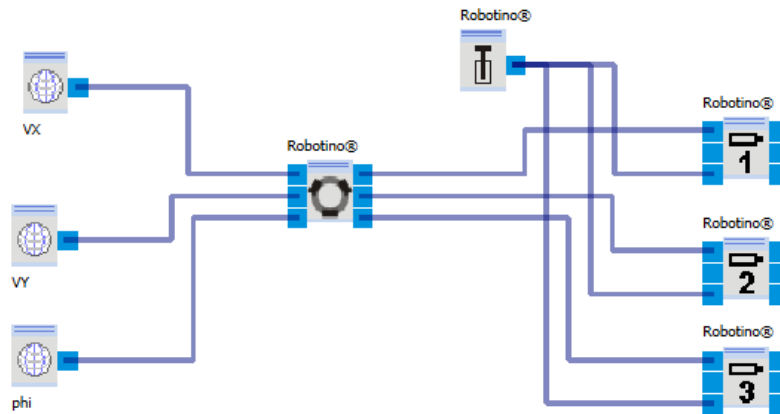


Figure 22 - Subprogram Drive - Drive system (Adapted from Sairanen 2021)

The function blocks used for setting the number of Position Points part of this subprogram are shown in Figure 23. The Constant block was dragged and dropped from the Function block library → Generators file in the function block list.

This part of the code sets the number of positions that make up the stops on the path. In this task, there are four of them. The number five was chosen as the code increases the value of the step once the Robotino reaches its destination so after the full cycle, the value equals five even if there is no position with the ID 5.

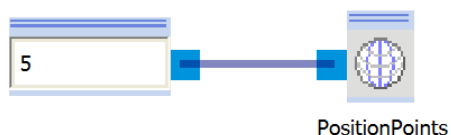


Figure 23 - Subprogram Drive - Position Points

The function blocks used for the Step increase part of this subprogram are shown in Figure 24. The Constant block was dragged and dropped from the Function block library → Generators file and the Counter Up block was dragged and dropped from the Function block library → Logic file in the function block list.

This part of the code increases the value of the Step variable by one when the StepIncr variable gains the true value.

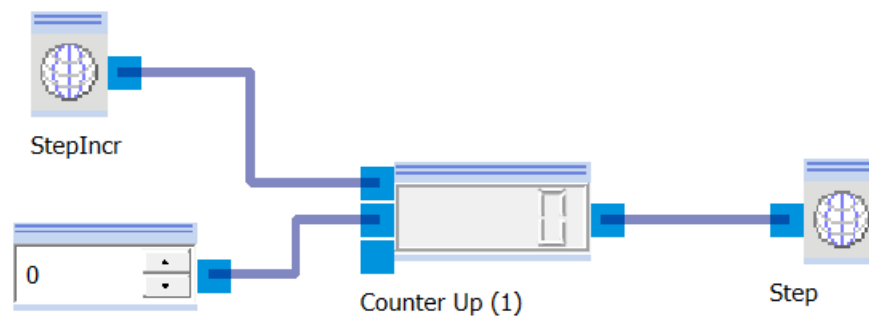


Figure 24 - Subprogram Drive - Step increase (Adapted from Sairanen 2021)

The function blocks used for the Cycle completion part of this subprogram are shown in Figure 25. The Equal block was dragged and dropped from the Function block library → Mathematics → Comparison operations file in the function block list.

This part of the code keeps track of whether the variables PositionPoints and Step are equal or not. Once they're equal, the variable CycleDone gets assigned the true value.

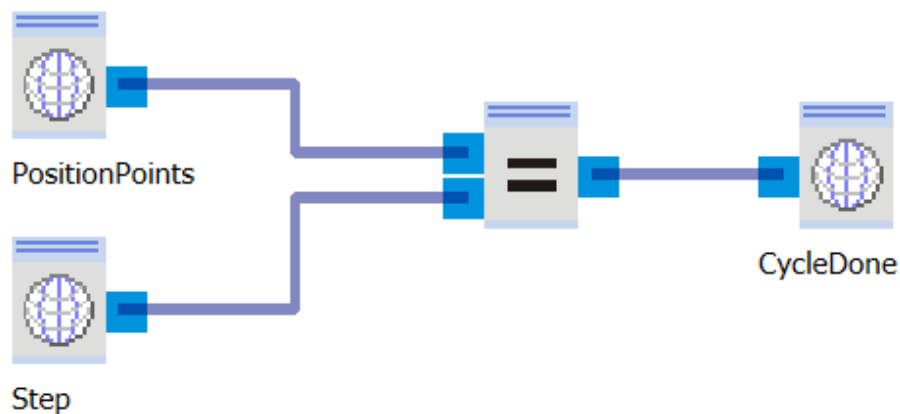


Figure 25 - Subprogram Drive - Cycle completion (Adapted from Sairanen 2021)

### 3.2.7 Lua code for QR positioning

Part of the QR subprogram is a custom function block that contains a code in the Lua language. This code doesn't work on the principle of object programming but has to be written in a specific language. Robotino View has an integrated script editor under the proper-

ties dialogue in the Lua script function block, shown in Figure 26. Here, the user can write the code they need this block to represent. In this case, the script has seven inputs, because the Marker position block to which it is connected has seven outputs and five outputs. The inputs are labelled as in1 to in7 and outputs as out1 to out5. These labels are case-sensitive.

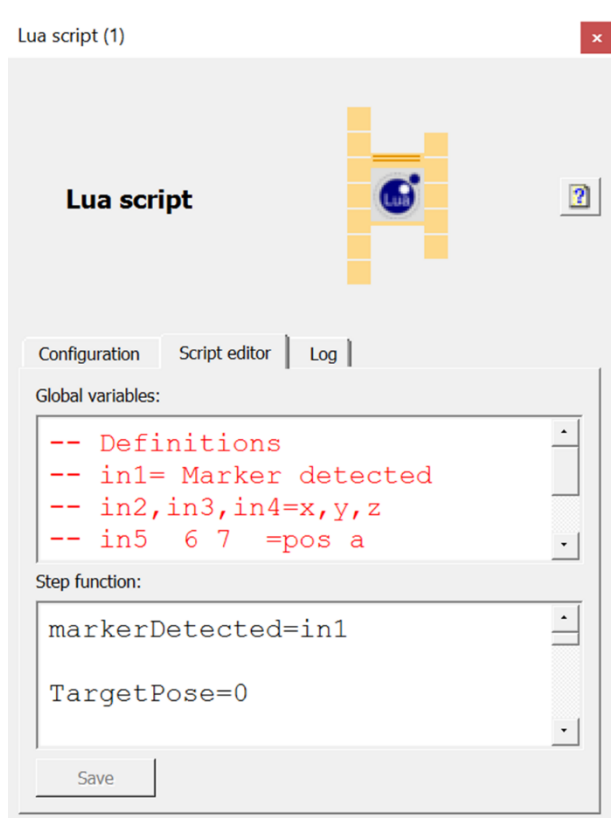


Figure 26 - Lua script dialogue

The point of this code is to decide whether there is a QR mark to be detected and if it is, to find out how far from it the robot is in each of the x and y axis as well as in regard to under what angle it is. With these inputs, the code creates outputs for the required speed to move the Robotino into the desired range. Once it is in the given range for each of the three watched parameters, the variable TargetPose has stored a value of three. The entire code used can be seen in Appendix 2.

### 3.2.8 Possible task expansions

This code comprise only a basic skeleton for many possibilities and its purpose is mainly as a starting guide. Once there is the possibility to configure the gripper motor, there can be added parts of the program featuring the gripper. There is the Gripper function block in Robotino View which accepts a true or false input to open or close the gripper respectively. It is also possible to keep track of whether the gripper is open or closed through its out-

puts. The sensor recognizing whether there is something located in the gripper can be reached by the function block Digital Input #5. A suggestion how such a code might look like is shown in Figure 27.

The transfer function block has integrated either graphics way or table that allows inputting which values should be transferred to which. In this case, the true value of the ObjectLocated variable into the false value of the Gripper. This will send the signal to the gripper to close when the object is located. And variables GripperOpen and GripperClosed can be used as conditions in the Main program diagram.

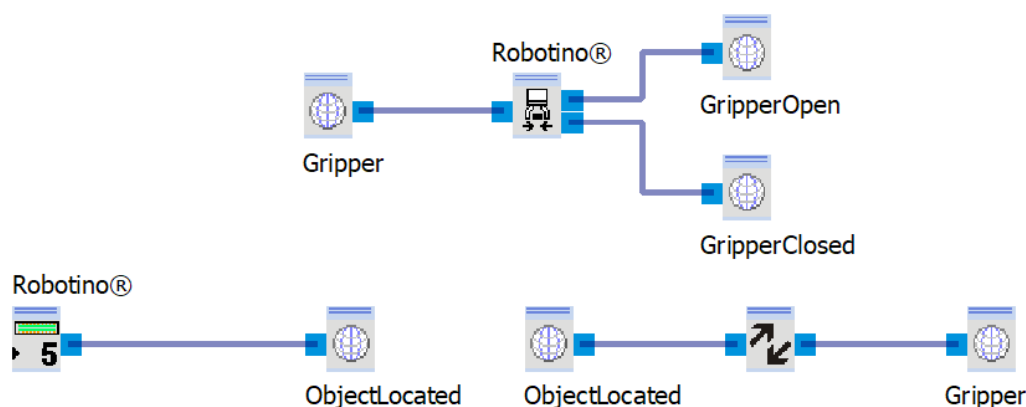


Figure 27 - possible Gripper part of the program

Another way to expand the task might be more complex navigation. As in added obstacle evasion or creating a code for the instance of the error. For these, Robotino offers the function blocks of bumper and distance sensors located all around the base or the Obstacle avoidance block.

## **4 Encountered difficulties**

### **4.1 Connection to RoboTimo not possible**

Attempts at connecting an external computer to RoboTimo – one of the available Robotino robots – through a Wi-Fi network and directly through cable were unsuccessful. The only way to get to the program of the robot was through its own computer and HDMI cable to a screen. This does not work for the purposes of the task.

#### **Solution**

The task was adjusted so only one of the robots is needed to complete it. The thesis can serve as a guide for future modification and possibly adding the second robot. Festo company has been contacted about the issue.

### **4.2 Missing manual for Robotino View 4**

There was no manual for the newest version of Robotino View software. Finding out how the software works without a manual was a lengthy task. There were troubles accessing even the manual for Robotino View 2 from the school network.

#### **Solution**

The externally downloaded manual for one of the previous versions of the software could be accessed and helped with some troubles. This did not, however, solve all issues encountered because of the missing manual and it took a trial-and-error approach to find the necessary information.

### **4.3 Automated rack output element not compatible with Robotino gripper**

Part of the task has been picking up an object from the automated rack. While this output element indeed is at the correct height for Robotino to be able to pick the object, it contains the ending that stops the object from falling and the Robotino gripper is not flexible enough to work around it.

#### **Solution**

There is a need for a construction of an alternative output element ending with a lower barrier and possibly a longer horizontal landing. For this task, it has been decided after consultation to only have the robot reach the space and, in the future, furthering the task to just putting the object into the gripper manually.



#### 4.4 Motor of the gripper not able to be configured

The task was meant to include opening or closing of the gripper at certain positions along the path. It was found out the wiring was probably not done correctly and the possibility to configure the motor in the browser UI was not available. The assembly manual is however no longer available to be found.

##### **Solution**

Festo company was contacted, and the wires were rewired according to instructions as well as the Help dialogue from Robotino View. This however did not solve the issue and the motor was still unable to be configured let alone accessed through the Robotino View software. Another question has been sent to Robotino Forum, where was a possibility to ask a Robotino specialist. The answer should solve the issue for future use.

## 5 Summary and conclusions

The thesis was supposed to serve as a creation of a demonstration task with Robotino 4.0 robots located in the room 1264 at LAB University of Applied Sciences. This task was supposed to connect to the already functional production line and widen this possible demonstration across the whole room by bringing an object from an automatized rack to a place on the other side of the room, with the vision of a conveyor at that place at the later date. Then this object was supposed to be picked up from the end of this imaginary conveyor and brought back to the rack.

There were a few obstacles that led to the task modification as well as shifting the focus of the thesis more towards being a guide than a report. The room is to be changed in the near future and as the task relies on the map that needs to be up to date, it was decided to focus on how to recreate the task so it will be available soon after the change and hopefully introduce the work with these mobile robots to more students so they can serve the educational purpose without each person having to figure everything out by themselves.

As of right now, the program does not fill out the objective of picking an object at one position and dropping it off at the next one. Otherwise, it fulfils the overall objective of expanding the demonstrational production line. The interaction with the unexpected obstacles leaves ample space to improve but is workable by default and the solution for the gripper is proposed, yet not tried.

The positioning in relation to the QR code has been left the same for all of the positions. For further task development, it is possible to set each desired position differently in relation to the QR code by just rewriting the numbers in the Lua code.

To ensure the program is running smoothly it is needed to reconnect the Robotino before each start of the program. It is also needed to start Robotino Factory and set the current position of the Robotino on the map.

## References

Festo Corporation a. Retrieved on 14 February 2023. Available at <https://www.festo-didactic.com/int-en/learning-systems/robotino-4-the-flexible-mobile-robotics-platform-for-vocational-training.htm?fbid=aW50LmVuLjU1Ny4xNy4xOC58ODU4fC4xMDMyOTc>

Festo Corporation b. Retrieved on 14 February 2023. Available at <https://ip.festo-didactic.com/InfoPortal/Robotino/Hardware/EN/index.html>

Festo Corporation c. Retrieved on 14 February 2023. Available at <https://ip.festo-didactic.com/InfoPortal/Robotino/Hardware/Controller/EN/PowerSwitch.html>

Festo Corporation d. Retrieved on 14 February 2023. Available at <https://ip.festo-didactic.com/InfoPortal/Robotino/Hardware/Controller/EN/ControlUnit.html>

Festo Corporation e. Retrieved on 14 February 2023. Available at <https://ip.festo-didactic.com/InfoPortal/Robotino/Hardware/Controller/EN/index.html>

Festo Corporation f. Retrieved on 14 February 2023. Available at <https://ip.festo-didactic.com/InfoPortal/Robotino/Hardware/DriveSystem/EN/Omnidrive.html>

Festo Corporation g. Retrieved on 14 February 2023. Available at <https://ip.festo-didactic.com/InfoPortal/Robotino/Hardware/DriveSystem/EN/Motors.html>

Festo Corporation h. Retrieved on 14 February 2023. Available at <https://ip.festo-didactic.com/InfoPortal/Robotino/Hardware/DriveSystem/EN/IncrementalEncoder.html>

Festo Corporation i. Retrieved on 14 February 2023. Available at <https://ip.festo-didactic.com/InfoPortal/Robotino/Hardware/DriveSystem/EN/GearUnits.html>

Festo Corporation j. Retrieved on 14 February 2023. Available at <https://ip.festo-didactic.com/InfoPortal/Robotino/Hardware/DriveSystem/EN/Wheels.html>

Festo Corporation k. Retrieved on 14 February 2023. Available at <https://ip.festo-didactic.com/InfoPortal/Robotino/Hardware/Sensors/EN/Bumper.html>

Festo Corporation l. Retrieved on 14 February 2023. Available at <https://ip.festo-didactic.com/InfoPortal/Robotino/Hardware/Sensors/EN/DistanceSensors.html>

Festo Corporation m. Retrieved on 14 February 2023. Available at <https://ip.festo-didactic.com/InfoPortal/Robotino/Hardware/Sensors/EN/Gyroscope.html>

Festo Corporation n. Retrieved on 14 February 2023. Available at <https://ip.festo-didactic.com/InfoPortal/Robotino/Hardware/Sensors/EN/Camera.html> )

Festo Corporation o . Retrieved on 14 February 2023. Available at <https://ip.festo-didactic.com/InfoPortal/Robotino/Hardware/Sensors/EN/OpticalSensors.html>

Festo Corporation p. Retrieved on 14 February 2023. Available at <https://ip.festo-didactic.com/InfoPortal/Robotino/Hardware/Sensors/EN/InductiveSensors.html>

Festo Corporation q. Retrieved on 14 February 2023. Available at <https://ip.festo-didactic.com/InfoPortal/Robotino/Hardware/Interfaces/EN/index.html>

Festo Corporation r. Retrieved on 14 February 2023. Available at <https://ip.festo-didactic.com/InfoPortal/Robotino/Hardware/Supply/EN/Akkus.html>

Festo Corporation s. Retrieved on 14 February 2023. Available at <https://ip.festo-didactic.com/InfoPortal/Robotino/Hardware/Modules/EN/index.html>

Festo Corporation t. Retrieved on 15 February 2023. Available at <https://ip.festo-didactic.com/InfoPortal/Robotino/Software/EN/index.html>

Festo Corporation u. Retrieved on 15 February 2023. Available at <https://ip.festo-didactic.com/InfoPortal/Robotino/Software/Webinterface/EN/index.html>

Festo Corporation v. Retrieved on 15 February 2023. Available at <https://ip.festo-didactic.com/InfoPortal/Robotino/Software/Webinterface/EN/Home.html>

Festo Corporation w. Retrieved on 15 February 2023. Available at <https://ip.festo-didactic.com/InfoPortal/Robotino/Software/Webinterface/EN/Program.html>

Festo Corporation x. Retrieved on 15 February 2023. Available at <https://ip.festo-didactic.com/InfoPortal/Robotino/Software/Webinterface/EN/Config.html>

Festo Corporation y. Retrieved on 15 February 2023. Available at <https://ip.festo-didactic.com/InfoPortal/Robotino/Software/Webinterface/EN/Network.html>

Festo Corporation z. Retrieved on 15 February 2023. Available at <https://ip.festo-didactic.com/InfoPortal/Robotino/Software/Webinterface/EN/Control.html>

Festo Corporation aa. Retrieved on 15 February 2023. Available at <https://ip.festo-didactic.com/InfoPortal/Robotino/Software/Webinterface/EN/Service.html>

Festo Corporation ab. Retrieved on 15 February 2023. Available at <https://ip.festo-didactic.com/InfoPortal/Robotino/Software/Webinterface/EN/lo.html>

Festo Corporation ac. Retrieved on 15 February 2023. Available at <https://ip.festo-didactic.com/InfoPortal/Robotino/Software/Programming/EN/RobotinoView.html>

Festo Corporation ad. Retrieved on 19 February 2023. Available at <https://ip.festo-didactic.com/InfoPortal/Robotino/Software/Simulation/EN/index.html>

National Instruments. Retrieved on 16 February 2023. Available at <https://www.ni.com/fi-fi/shop/labview.html>

Openrobotino 2021. Retrieved on 16 February 2023. Available at [https://wiki.openrobotino.org/index.php?title=Downloads#Robotino\\_Factory](https://wiki.openrobotino.org/index.php?title=Downloads#Robotino_Factory)

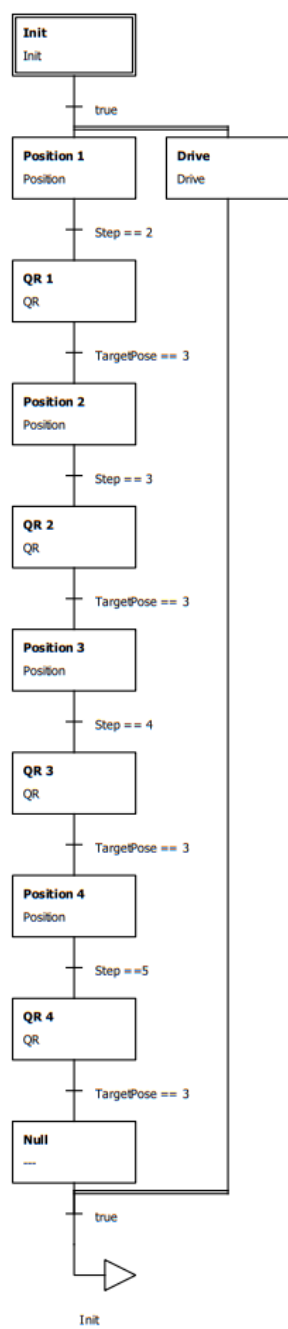
Openrobotino. Retrieved on 16 February 2023. Available at [https://doc.openrobotino.org/download/RobotinoView/en/globalvariables\\_general.htm](https://doc.openrobotino.org/download/RobotinoView/en/globalvariables_general.htm)

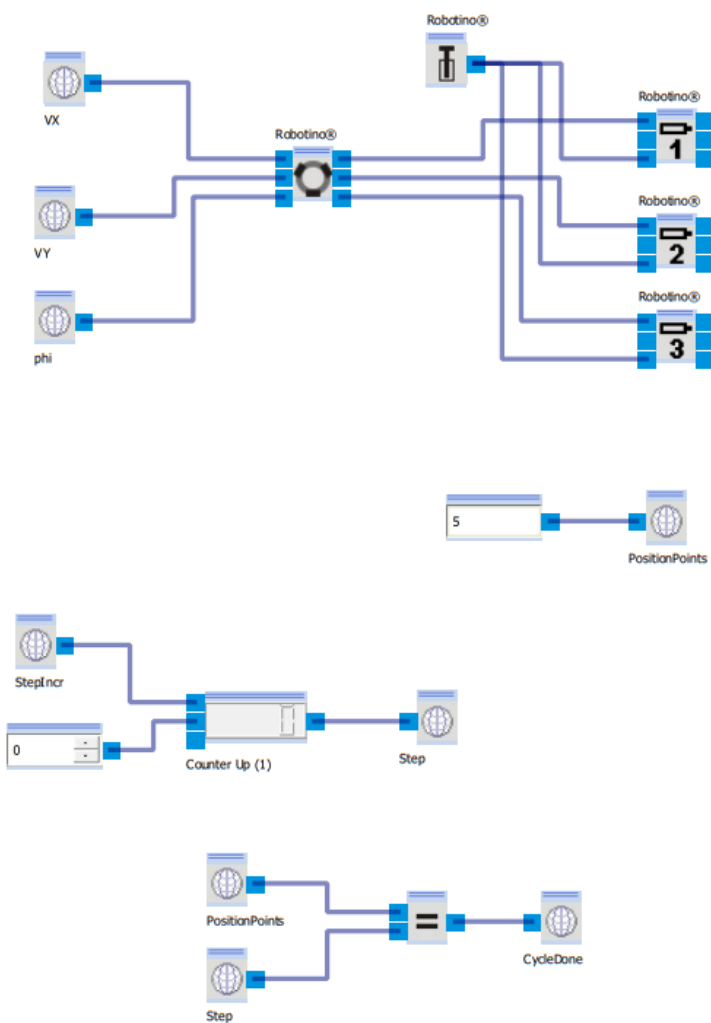
Řízení a navigace. LMS.VSB. Retrieved on 18 February 2023. Limited availability at <https://lms.vsb.cz/mod/folder/view.php?id=770759>

Sairanen, A. 2021. Installation of Robotino 4.0. LAB University of Applied Sciences. Thesis (Bachelor's degree). Retrieved on 30 March 2023. Available at <https://www.theseus.fi/bitstream/handle/10024/502428/Installation%20of%20Robotino%204.0%20updated.pdf?sequence=2>

Siegwart, R. & Nourbakhsh, I.R. 2004, Introduction to Autonomous Mobile Robots. London: The MIT Press

## Appendix 1. Program in Robotino® View

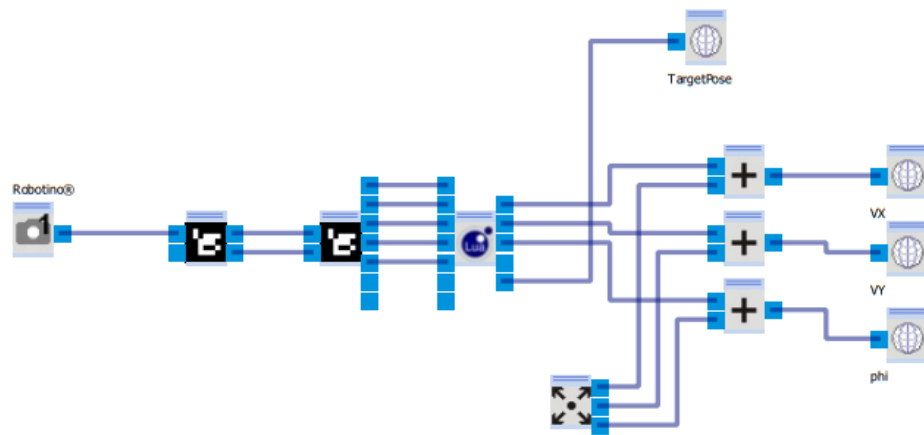












## Appendix 2. Lua code

```

markerDetected=in1
TargetPose=0

if markerDetected == true then
--x correction of camera->VY of Robotino
if in2 > 0.002 or in2 < -0.002 then out2 = in2 * -10000
else TargetPose = TargetPose + 1
end

--Y correction of camera ->VX of Robotino

if in4 < -0.065 then out1 = 100
elseif in4 > 0.01 then out1 = -40
else TargetPose = TargetPose + 1
end

--A correction of camera(horizontal rotation)-> Phi of Robotino
if in5 > 5 then out3 = -20
elseif in5 < -5 then out3 = 20
else TargetPose = TargetPose + 1
end

end

out5=TargetPose

```