

**FESTO**

**Robotino®**

Instructor volume



544307 EN

## **Intended use**

The mobile robot system Robotino® has been developed and produced solely for vocational and further training purposes in the field of automation and technology. The company undertaking the training and/or the instructors is/are to ensure that trainees observe the safety precautions specified in the manuals provided. Festo Didactic herewith excludes any liability for damage or injury caused to trainees, the training company and/or any third party, which may occur if the system is in use for purposes other than purely for training, unless the said damage/injury has been caused by Festo Didactic deliberately or through gross negligence.

Order No.: 544307  
Status: 10/2007  
Authors: Bliesener, Weber, Karras, Kling, Zitzmann  
Graphics: Doris Schwarzenberger

© Festo Didactic GmbH & Co. KG, 73770 Denkendorf, 2007  
Internet: [www.festo-didactic.com](http://www.festo-didactic.com)  
E-Mail: [did@de.festo.com](mailto:did@de.festo.com)

The copying, distribution and utilisation of this document as well as the communication of its contents to others without express authorisation is prohibited. Offenders will be held liable for the payment of damages. All rights reserved, in particular the right to carry out patent, utility model or ornamental design registration.

# Contents

Instructor volume for two parts corresponds to the trainee volume. The trainee volume consists of a complete set of exercises (Part A) and the theoretical fundamentals (Part B) in loose-leaf form for each student.

1.	Robotino® – a training system for mobile robotics and automation technology _____	7
2.	The Robotino® learning system _____	11
3.	Tuition in an entirely different way _____	15

## Part A – Exercises

### **Project 1**

Inspection of supplied components and commissioning of the Robotino® \_\_\_\_\_ A-3

### **Project 2**

Linear travelling of a mobile robot system in any direction \_\_\_\_\_ A-13

### **Project 3**

Linear travelling and positioning of a mobile robot system \_\_\_\_\_ A-43

### **Project 4**

Path tracking of an automated guided vehicle system  
using two diffuse sensors \_\_\_\_\_ A-67

### **Project 5**

Accurately positioned approach of a loading station \_\_\_\_\_ A-89

### **Project 6**

Approaching an obstacle and maintaining a defined distance \_\_\_\_\_ A-107

### **Project 7**

Circling a station and approaching various transfer positions \_\_\_\_\_ A-117

### **Project 8**

Path tracking of an automated guided vehicle system using an  
analogue inductive sensor \_\_\_\_\_ A-123

### **Project 9**

Determining the optimal motion behaviour \_\_\_\_\_ A-139

### **Project 10**

Path tracking of an automated guided vehicle system  
with the help of a webcam \_\_\_\_\_ A-147

### **Project 11**

Searching and approaching a coloured object  
with the help of a webcam \_\_\_\_\_ A-159

## **Part B – Theory**

### **Drive**

1	Closed-loop control/PID controller	B-3
2	Robot subsystems: Drive	B-21

### **Sensors**

3	Characteristic	B-35
4	Infrared distance sensor	B-37
5	Optical proximity sensor	B-41
6	Inductive sensor	B-51
7	Safety strip, collision sensing	B-53
8	Webcam	B-55

### **Robotino® View**

9	Generators	B-57
10	Oscilloscope	B-61
11	Line detection	B-65
12	Segmenting	B-67
13	Segment extraction	B-71
14	IF function	B-73
15	Sign reversal	B-75
16	Sequence control	B-77

## **Part C – Solutions**

### **Project 1**

Inspection of supplied components and commissioning  
of the Robotino® – solution \_\_\_\_\_ C-3

### **Project 2**

Linear travelling of a mobile robot system in any direction – solution \_\_\_\_\_ C-9

### **Project 3**

Linear travelling and positioning of a mobile robot system – solution \_\_\_\_\_ C-45

### **Project 4**

Path tracking of an automated guided vehicle system  
using two diffuse sensors – solution \_\_\_\_\_ C-69

### **Project 5**

Accurately positioned approach of a loading station – solution \_\_\_\_\_ C-91

### **Project 6**

Approaching an obstacle and maintaining a defined distance – solution \_\_\_\_\_ C-107

### **Project 7**

Circling a station and approaching various transfer positions – solution \_\_\_\_\_ C-115

### **Project 8**

Path tracking of an automated guided vehicle system  
using an analogue inductive sensor – solution \_\_\_\_\_ C-121

### **Project 9**

Determining the optimal motion behaviour – solution \_\_\_\_\_ C-137

### **Project 10**

Path tracking of an automated guided vehicle system  
with the help of a webcam – solution \_\_\_\_\_ A-149

### **Project 11**

Searching and approaching a coloured object  
with the help of a webcam – solution \_\_\_\_\_ A-161

## Content

## 1. Robotino® – a learning system for mobile robotics and automation technology

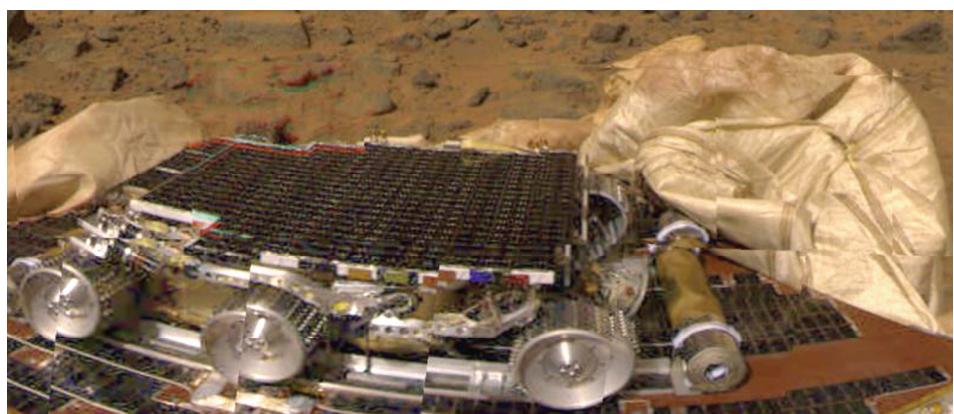
Mobile robots

They respond to commands, detect objects three-dimensionally and locate these with sensors, such are mobile robots.

Previously robot systems were restricted to a stationary position. Mobile robots represent the next step in the development of robotics in that they can execute the same tasks as their stationary predecessors but, in addition, can move away from a position.

This provides the prerequisites for dealing with countless additional tasks.

As a result of the robot Sojourner landing on Mars with the Pathfinder probe, mobile robots have made headlines in every newspaper. Furthermore, through this NASA project it has also become clear just how important navigation is for mobile robots. The fact that the robot moved just 10 cm from its space capsule was already celebrated as a huge success.



## 1. Robotino® – a learning system for mobile robotics and automation technology

Mobile robots are, however, also very useful in other areas. They can be used to explore canal systems, underwater worlds and volcanoes, in other words environments difficult to access by man.

### 1.1

#### Areas of application for mobile robots

The motivation behind the development and analysis of mobile robots is largely due to the necessity and desire to use robots that operate with and for people in their daily environment - in offices, hospitals, museums, libraries, supermarkets, sports facilities (lawn mowing), exhibition halls, airports, railway stations, universities, schools and eventually also in domestic use.



For disabled or older people, a means of mobile transport means more freedom of movement and independence. This is where the possibilities of orientation, navigation and autonomous obstacle recognition and avoidance are of great significance.

The research centre for automation in Karlsruhe developed James, a mobile service robot. Exactly like its siblings Stan and Ollie, they can receive orders from a central station and plan and execute these autonomously. Different sensors such as laser scanners, acoustic distance sensors and cameras enable the robots to sense their environment to flexibly react to any potential obstacle. The planning and execution of their task is executed via various computer cards and a correspondingly developed program. The wheels provide the robots with a wide range of different directions of motion.

## 1. Robotino® – a learning system for mobile robotics and automation technology

For instance, if you specify the outline of a building to robots of this type, they can perform errands autonomously. Areas of application can be found, for example, in hospitals or large hotels, where robots can transport bed linen and towels to the laundry or deliver meals. They can also conceivable sweep floors autonomously.

A popular use of mobile robots is as security guards in museums. They are small, quick, quiet and invisible in the dark. Equipped with heat and motion sensors, they are able to immediately locate unwanted guests and trigger the alarm.

### The robot as a home help

Tokyo (AP) – Its movements are still somewhat stiff and slow and the voice rather monotonous but, via its remote control, it readily turns towards the window and also brings something to drink. The HRP-2 robot currently being developed by a Japanese research laboratory should become a passable home help within just a few years.

The robots – called Promet – are being developed by the National Institute for Advanced Industrial Science and Technology. They respond to spoken commands, are able to detect objects three-dimensionally and locate these using infrared sensors. “We hope to make them into something akin to police or security dogs”, explained Isao Hara, head of research of the Institute in Tsukuba north-east of Tokyo, referring to two blue, metallic robots. “I believe that they can collaborate with humans. We are currently investigating how they could be integrated into human society.”

### „What can I do for you?“

When Hara called one of the robots: “Please come here”, it replied: „What can I do for you?“ When asked to switch on the TV, the Promet replies: „I shall switch on the TV“, and proceeds to do so. When Hara asks for a bottle of fruit juice, one of the robots passes on this task to the next one, saying: “Please take care of this.” Hara explains, these robots can copy virtually any human movement apart from running. This would cause too much noise and also jolt the metallic robots too much. They therefore only move at a measured pace. Above all, robots need to be able to communicate with humans, locate objects and act autonomously, said Hara. “They can help in the same way as dogs.”

### A flat battery halts production

Japan is regarded as the leading authority in robotics. Companies such as Sony, Hitachi and Honda have developed robots which are primarily intended for entertainment. In industrial production they are already ubiquitous. If robots stop responding to human commands, then this is due to the fact that the batteries are flat. This was exactly the case with a Promet, which stopped working in the middle of its demonstration and had to be recharged on a special device.

## 1. Robotino® – a learning system for mobile robotics and automation technology

### 1.2

#### Tasks in industry

#### Automated guided vehicle system

Automated guided vehicle systems can be found increasingly in use in production plants and hazardous areas. These are mobile robots that are floor-bound; in other words, a driverless conveyance system moving along the floor. The automatic tracking either runs along predefined lanes or freely definable routes within a store or factory premises. Differentiation is therefore made between line-bound and line-free tracking.



Automated guided vehicle systems are ideally suited for the loading and unloading of assembly lines, packaging conveyors and for the configuration of assembly devices for use in commissioning and assembly lines.

## **2. The learning system Robotino®**

The following are special characteristics of and special requirements for all mobile robots:

- Mobile machines with autonomous orientation, navigation, obstacle recognition and avoidance
- Autonomous power and computer supply
- Incorporation of own sensors and actuators

The Robotino® learning system meets all these requirements and enables you to familiarise yourself with the multifaceted technical areas of knowledge of mobile robotics.

### **2.1**

#### **Target groups and topics**

Vocational and further training:

- Commissioning of a mechatronic system
- Acquisition and scaling of miscellaneous sensor data
- Electrical motor control/drive unit
- Electrical drive technology
- Closed-loop control of a mechatronic system
- Graphic programming of applications for a mobile robot system
- Analysis of sensor data for various applications
- Introduction to image processing

In particular for technical colleges and universities:

- C++ programming of mobile robot applications on the basis of the API provided
- Remote control via WLAN
- Integration of a camera system
- Programming of autonomous navigation

### **2.2**

#### **Interesting facts about the Robotino®**

- It does not hide its technology, but displays it through the open chassis
- It is fun because trainees can control it themselves by making it intelligent
- It is technology that encourages trainees to understand and use it
- It is industry-focuses since it consists of components used in industry
- It is flexible, easy to transport and space saving

## 2. The learning system Robotino®

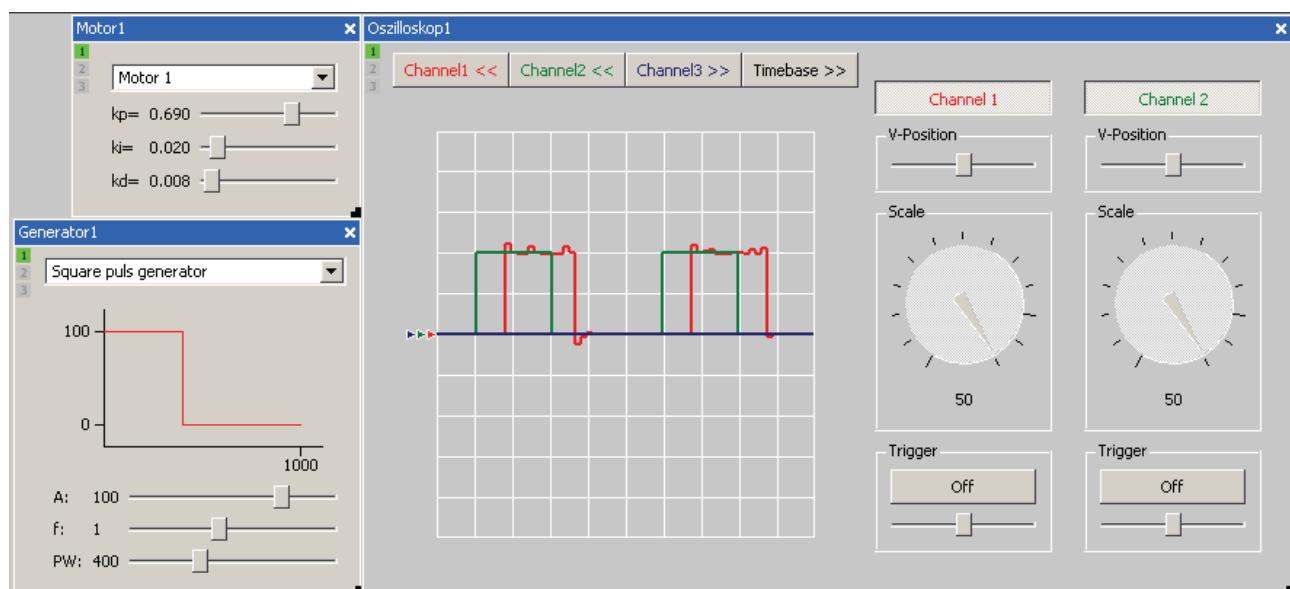
### 2.3 Experimental procedure

With the help of interesting experiments with the Robotino®, trainees come into contact with the mechatronic system and the associated topics. They can practise and acquire the necessary technical information in the integrated theory.

The Robotino® View software not only enables trainees to program the behaviour of the system, but also to modify and test it interactively online via WLAN.



Robotino® View: an example



You set the controller parameters

You set the setpoint signal interactively

Online display of setpoint and actual data via a virtual oscilloscope

## 2. The learning system Robotino®

### 2.4 Exercises

The exercises are based on industrial tasks in automation technology

Experiments covering all aspects of the Robotino®

- provide suggestions to make a particular technology more easily understandable
- are useful, interesting, clearly explained and hands-on
- and therefore facilitate an affective and haptic approach to topics in automation technology and mobile robotics

### 2.5 Topics and contents

Training contents from the following areas can be dealt with:

- Mechanics
  - Mechanical construction of a mobile robot system
- Commissioning
  - Commissioning of a mobile robot system
- Electrotechnology
  - Motor actuation
  - Measurement and evaluation of different electrical values
- Sensors
  - Sensor-guided path control
  - Collision-free path control by means of distance sensors
  - Path control by means of image processing of camera images
- Closed-loop control technology
  - Actuation of omnidirectional drives
- Programming
  - Intuitive via graphic wiring of predefined function blocks
  - C++ programming on the basis of a Windows API and Linux API (functions libraries)
- Fault finding
  - Systematic fault finding on a mobile robot system

## 2. The learning system Robotino®

### 2.6

#### Training aims

The following training aims can be achieved with the Robotino®:

##### Trainees

- learn to handle an electrically controlled motor actuation
- are familiarised with the fundamentals, construction, measurement of values and parameterisation of DC motor control
- are familiarised with the fundamentals of electrical drive technology
- understand an omnidirectional 3-axis drive and are able to commission and operate this
- are able to commission (software and hardware) a mobile robot system using the Robotino® as an example
- are able to move the mobile robot system Robotino® in different directions
- are able to realise sensor-guided path control for the Robotino® along a predefined path by means of software support
- are able to realise the integration of image processing into the control system of the Robotino®
- are able to develop a sensor-guided autonomous path control of the Robotino® using object recognition and simple exploratory behaviour

Furthermore the following additional training aims can be achieved:

##### Trainees

- are able to realise the integration of additional sensors
- are able to integrate additional mechanical devices into the system such as handling equipment
- are able to realise the programming (C++) of their own navigation and control algorithms
- are able to realise autonomous navigation of the Robotino®

### **3. Tuition in an entirely different way**

Autonomous and mechatronic systems are becoming increasingly more important. The learning system Robotino® enables you to familiarise yourself with the multifaceted topic of mobile robotics. A particularly interesting aspect of the learning system Robotino® is that it covers the entire range of the latest developments.

The same also applies for the use of a WLAN. You are able to experience the technology first-hand in that the program entered directly communicates with the Robotino® via WLAN.

Topics	Process-oriented topics (e.g. maintenance, process control) as well as technology-oriented topics (e.g. control technology, programming) can be dealt with. Individual subareas of these such as sensors, controllers, can be excerpted for tuition.
Experimental learning	<p>Unlike the usual method, training doesn't start with theory but with practice. Trainees are able to practise and acquire the necessary technical background information. Consequently the topics of this book of exercises are set out in the form of experiments.</p> <p>These experiments comprise the traditional contents of the previous syllabus, but are more activity-orientated than previous purely theoretical tuition and therefore tie in with the training areas.</p> <p>Since theory therefore only features in the background, the mobile Robotino® represents the training medium. The theory to be taught will be solely that required by trainees for experiments.</p> <p>Training with the learning system Robotino® therefore meets the requirements of activity-orientated tuition and enables trainees to become competent through successful practice.</p>

### 3. Tuition in an entirely different way

**Advantages for the trainee** Trainees are given a hands-on introduction to mobile robotics by means of interesting experiments. They are therefore more attentive, eager to learn and capable.

The level of learning is gradually raised in the exercises so that trainees can see the initial measurable success of training after each exercise. The knowledge imparted can then be used again in a different exercise covering the same subject matter in order to consolidate the knowledge acquired. The book of exercises is predominantly practice-related, dealing with problems occurring in industry are thereby providing trainees with even greater incentive to find a solution for the exercise. The fact that trainees are not only listening and observing, but are actively involved in what takes place as part of tuition arouses greater interest and motivates trainees to address these topics and problems. This ensures a successful training outcome.  
Robotino® helps trainees to gain a better understanding of the technologies dealt with.

**Advantages for trainees/the training centre** Higher motivation and a better understanding of the technology enable instructors to teach the required subject matter at a more rapid pace. Consequently instructors are faced with less disruption during tuition.

Equally, instructors receive greater recognition from students, college and training establishments since this type of tuition could hardly be more practice-oriented. Tuition can be prepared and structured with the help of the problem definitions and the practice-related exercises can also be used for written exam papers.  
Robotino® can also be used for interdisciplinary tuition.

**Instructor tasks** One of the tasks of the instructor is to impart theoretical fundamentals. This can be instructor-orientated. On the other hand, it is important to assist students with advice and support during experiments and in this case the role of the trainer is rather that of a moderator.

### 3. Tuition in an entirely different way

<b>Areas of application for tuition</b>			
<b>Areas of application</b>	<b>Topics</b>	<b>Training material</b>	<b>Learning style</b>
Vocational colleges	<ul style="list-style-type: none"> <li>– Sensors</li> <li>– Mechanics</li> <li>– Closed-loop control technology</li> <li>– Programming - graphic-visual, symbolic, online</li> <li>– Image processing (optional)</li> </ul>	<ul style="list-style-type: none"> <li>– Sensors</li> <li>– Assemblies</li> <li>– Electrical drive technology, motor actuation, measurement and evaluation</li> <li>– Robotino® View</li> <li>– Camera (optional)</li> </ul>	<ul style="list-style-type: none"> <li>– Individual and team work</li> <li>– Experimental learning with the help of practice-related problem descriptions</li> <li>– Instructor-orientated</li> <li>– Student-orientated</li> </ul>
Sixth form schools	<ul style="list-style-type: none"> <li>– Applied vector analysis</li> <li>– Omnidirectional drive</li> </ul>	<ul style="list-style-type: none"> <li>– Robotino® View</li> <li>– Assemblies</li> </ul>	<ul style="list-style-type: none"> <li>– Individual and team work</li> <li>– Experimental learning with the help of practice-related problem descriptions</li> <li>– Instructor-orientated</li> <li>– Student-orientated</li> </ul>
IT sector	<ul style="list-style-type: none"> <li>– C-programming</li> <li>– Image processing (optional)</li> <li>– WLAN</li> </ul>	<ul style="list-style-type: none"> <li>– C++</li> <li>– Camera</li> <li>– WLAN Robotino® and computer</li> </ul>	<ul style="list-style-type: none"> <li>– Individual and team work</li> <li>– Experimental learning with the help of practice-related problem descriptions</li> <li>– Instructor-orientated</li> <li>– Student-orientated</li> </ul>
Technical colleges/universities	<ul style="list-style-type: none"> <li>– C ++</li> <li>– Vector analysis</li> <li>– Programming of autonomous navigation</li> </ul>	<ul style="list-style-type: none"> <li>– C++</li> <li>– Libraries (software)</li> </ul>	<ul style="list-style-type: none"> <li>– Individual and team work</li> <li>– Experimental learning with the help of practice related problem descriptions</li> <li>– Instructor-orientated</li> <li>– Student oriented</li> </ul>

### 3. Tuition in an entirely different way

#### **Methodological help for the instructor**

##### **Example: Interdisciplinary tuition**

The Robotino® is ideally suited for interdisciplinary tuition. For example, it is possible to combine the programming with the Robotino® View software (graphical user interface) and sensors.

#### Training aims

The general training aim is to be able to use the sensor data for programming such as to enable the Robotino® to follow a line along an aluminium strip.

More specific training aims include familiarisation with the functions, characteristics and areas of application of inductive sensors, the ability to use Robotino® View as well as the symbols and their function.

#### Problem description

What is required to enable the Robotino® to travel along a predefined line?

#### Parameters

- How can a control concept be designed for Robotino®?
- Which sensors can be used?
- Why is the line created by means of an aluminium strip?

Programming	Robotino® View
	C-Programming
	WLAN
	Image processing

Sensors	Infrared distance sensors
	Incremental encoder
	Collision protection sensor
	Inductive proximity sensor, analogue
	Optical sensor, digital

#### Additional examples

Possible additional examples are the combination of closed-loop control technology with the programming of the Robotino®.

The possibility here is to measure to ask trainees to measure and evaluate the different electrical variables of the Robotino®.

Another possibility is to establish a connection between the technical mechanism and Robotino® View. This enables trainees to familiarise themselves with the effect of different drivers within the mechanism by mounting and then testing these in the program entered.

### 3. Tuition in an entirely different way

Social themes

#### Competitions

To organise competitions between various teams working on the same problem definition: Different approaches and alternative solutions promote creative and critical thinking.

Evaluation: correctness, quality, speed

#### Remote control of Robotino® in lessons

Use of one Robotino®

The Robotino® has its own WLAN server. When operating a Robotino®, you therefore only need a PC that can establish a WLAN connection.

In the case of this application, the WLAN server of the Robotino® is in AP (Access Point) mode.



Use of three to four Robotinos®

If three to four Robotinos® are to be controlled simultaneously, the application as described above can be used.

- Advantage

All Robotinos® can have the same IP-address since each one forms its own network.

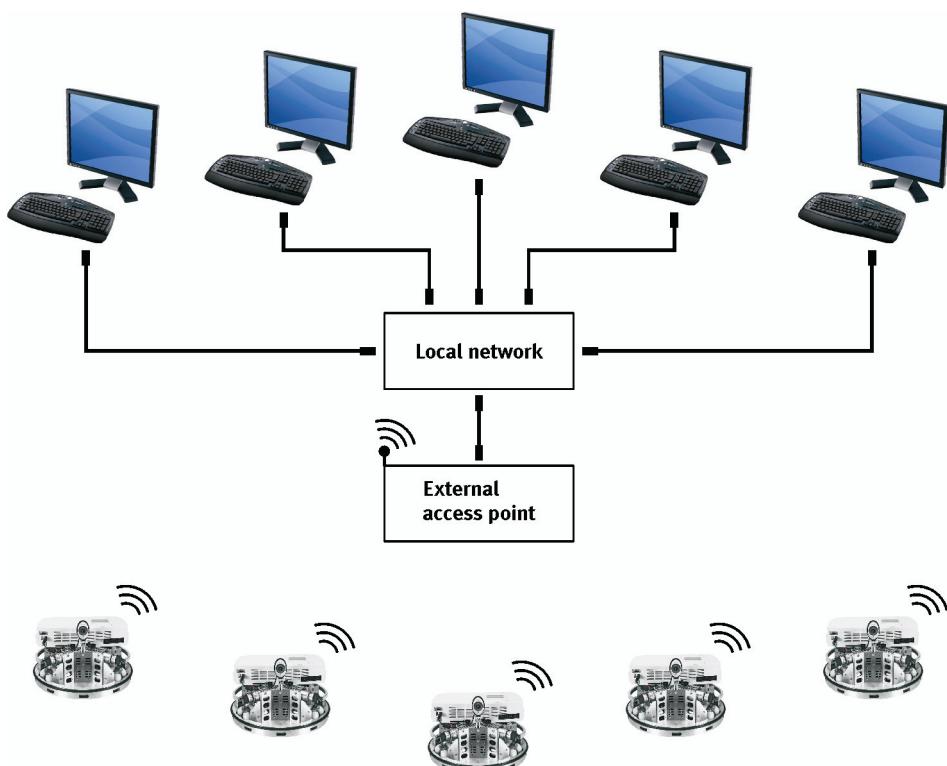
- Disadvantage

Different WLAN networks can cause collisions if their channels are too close together. A maximum of 11 channels is available and, for reasons of safety, it is advisable to leave at least three free channels between two active channels.

### 3. Tuition in an entirely different way

Use of several Robotinos® if the PCs are connected to a school network.

The access point of the Robotino® must be set to AP client mode via a switch directly at the Robotino® access point. A central WLAN access point is required in this case, which is directly connected to a local Ethernet-network.



- Advantage  
Any number of Robotinos® can operate on one network.
- Disadvantage  
Each Robotino® requires a special IP address that can, however, be input via the touch-sensitive keyboard.

**The local network can also be accessed via the unencrypted external access point.**

Settings	Value
SSID	RobotinoAPx.1
Channel	11
Encryption	None

### 3. Tuition in an entirely different way

Use of several Robotinos® in the absence of a school network

The WLAN of the Robotino® must be set to AP client mode via a switch directly on the Robotino®. A central, additional WLAN server is required in this case.



- Advantage

Any number of Robotinos® can operate on one network.

- Disadvantage

Each Robotino® requires a special IP address that can, however, be input via the touch-sensitive keyboard.

### 3. Tuition in an entirely different way

## Part A – Exercises

### **Project 1**

Inspection of supplied components and commissioning of the Robotino® \_\_\_\_\_ A-3

### **Project 2**

Linear travelling of a mobile robot system in any direction \_\_\_\_\_ A-13

### **Project 3**

Linear travelling and positioning of a mobile robot system \_\_\_\_\_ A-43

### **Project 4**

Path tracking of an automated guided vehicle system  
using two diffuse sensors \_\_\_\_\_ A-67

### **Project 5**

Accurately positioned approach of a loading station \_\_\_\_\_ A-89

### **Project 6**

Approaching an obstacle and maintaining a defined distance \_\_\_\_\_ A-107

### **Project 7**

Circling a station and approaching various transfer positions \_\_\_\_\_ A-117

### **Project 8**

Path tracking of an automated guided vehicle system  
using an analogue inductive sensor \_\_\_\_\_ A-123

### **Project 9**

Determining the optimal motion behaviour \_\_\_\_\_ A-139

### **Project 10**

Path tracking of an automated guided vehicle system  
with the help of a webcam \_\_\_\_\_ A-147

### **Project 11**

Searching and approaching a coloured object  
with the help of a webcam \_\_\_\_\_ A-159

#### Note

The exercises and solutions are based on Version 1.6 of Robotino® View.



## **Project 1**

### Inspection of supplied components and commissioning of the Robotino®

#### Training aims

#### Trainees

- are familiarised with the main components of a mobile system using the example of the Robotino®.
- are able to carry out the commissioning of a mobile robot system using the example of the Robotino®.
- are able to test and describe the motion behaviour of the Robotino®.

#### Problem description

The task is to carry out the inspection of supplied components and the commissioning of a complex mechatronic system.

#### Project assignment

Carry out the supplied components inspection and the commissioning of the Robotino®.

The supplied components inspection includes

- the creation and checking of a check list for visual inspection

The commissioning comprises

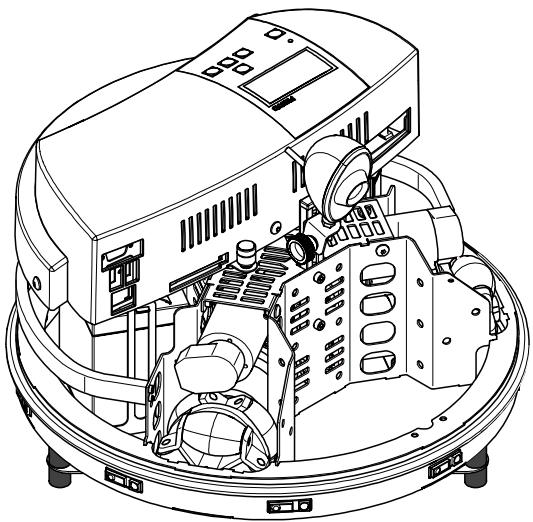
- the correct start-up sequence of the system
- the checking of the charge status of the rechargeable batteries
- the testing of the motion programs „circle“, „forward“, „rectangle“, „roam“
- the documentation of the results

#### Work assignments

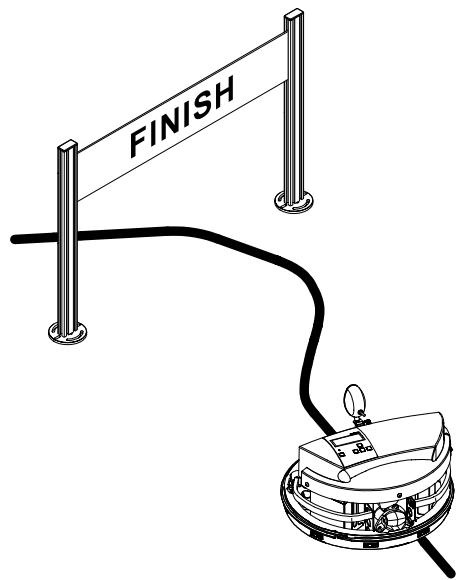
1. Carry out an inspection of the supplied components by means of a visual inspection:
  - Create a check list regarding the completeness of the system.
  - Check through the check list and tick it if complete
2. Commission the Robotino® hardware:
  - Check the functionality of the components and document the results
  - Check the motion behaviour of the Robotino® by testing and documenting the demo applications „forward“, „circle“, „rectangle“ and „roam“.

Project 1: Inspection of supplied components and commissioning of the Robotino®

Positional sketch



Working aid



Technical documentation of the Robotino®

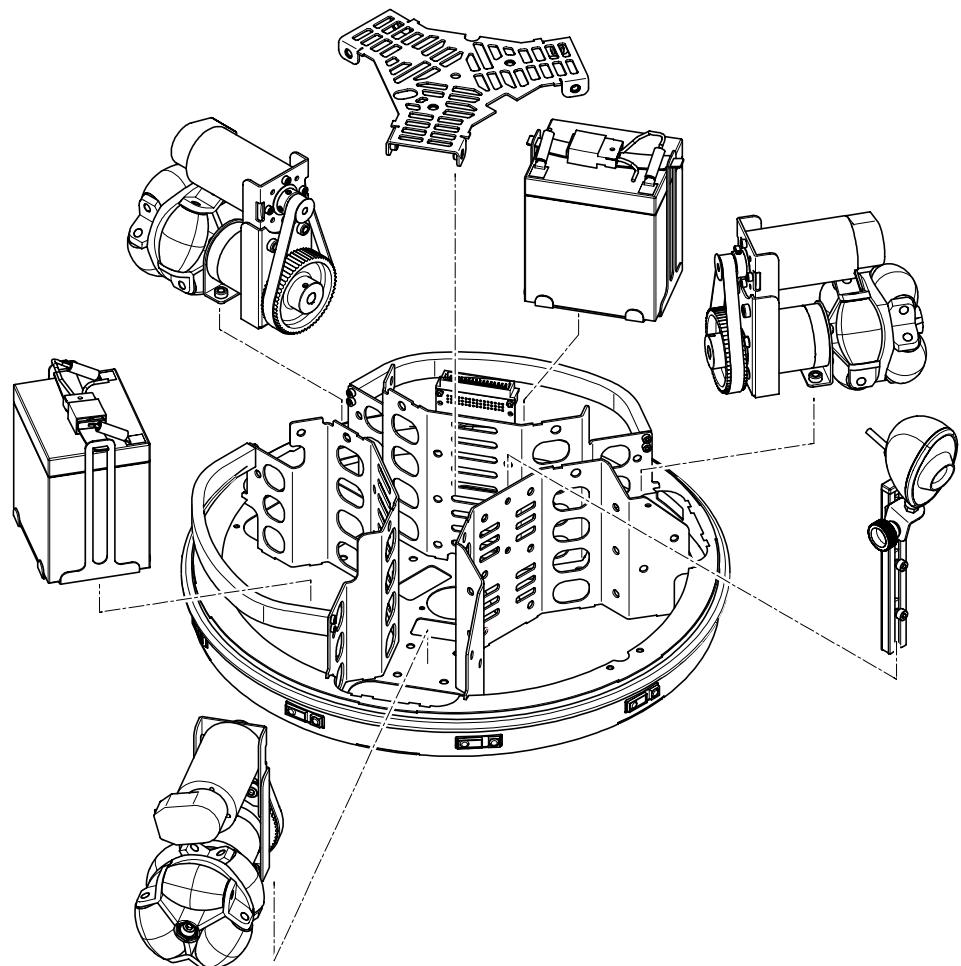
## Project 1: Inspection of supplied components and commissioning of the Robotino®

Project 1: Inspection of supplied components and commissioning of the Robotino®	
Name:	Date:
Creating a check list	Sheet 1 of 2

- Create a check list for the visual inspection to check whether the system is complete.

Note

Refer to the technical documentation and check what components the system needs to include.



Some of the main components are:

3 DC motors  
2 12 V rechargeable batteries  
Base plate with bumper  
Distance sensors  
Working platform with webcam (camera)  
Embedded controller

Project 1: Inspection of supplied components and commissioning of the Robotino®

<b>Project 1: Inspection of supplied components and commissioning of the Robotino®</b>	
Name:	Date:
Creating a check list	Sheet 2 of 2

- Complete the check list and tick it for completeness.

<b>Quantity</b>	<b>Description</b>	<b>ok</b>

Date \_\_\_\_\_

Signature \_\_\_\_\_

## Project 1: Inspection of supplied components and commissioning of the Robotino®

<b>Project 1: Inspection of supplied components and commissioning of the Robotino®</b>	
Name:	Date:
Commissioning of the Robotino®	Sheet 1 of 2

- Test the functionality of the components and document your findings.

### Note

Proceed as described in the technical documentation under „commissioning“ for the work assignments below.

- Jack up the system so that the wheels are freely movable.
- Connect the Robotino® to the power supply and switch on the system controller.
- Check whether the system signals correctly via the control panel display whilst observing the LED on the control panel.

Display	Description

- Check the charge state of the batteries via the control panel display.

Charge state of batteries

Idle state, no electrical malfunction

Project 1: Inspection of supplied components and commissioning of the Robotino®

<b>Project 1: Inspection of supplied components and commissioning of the Robotino®</b>	
Name:	Date:
Commissioning of the Robotino®	Sheet 2 of 2

- Document your results on the worksheet.

Commissioning of \_\_\_\_\_

Commissioned by \_\_\_\_\_

Power supply and status display \_\_\_\_\_

Charge state of batteries \_\_\_\_\_

Date \_\_\_\_\_

Signature \_\_\_\_\_

<b>Project 1: Inspection of supplied components and commissioning of the Robotino®</b>	
Name:	Date:
Testing of motion behaviour	Sheet 1 of 3

- Test the motion behaviour of the Robotino® by testing the demo applications „forward“, „circle“, „rectangle“ and „roam“.
- Observe the motion behaviour in the jacked-up and moving state.



Make sure that, in the motion program „roam“, the Robotino® only avoids obstacles at floor level, otherwise damage may be caused.

- Start the programs „circle“, „forward“, „rectangle“, „roam“, once each in the jacked-up state and during motion.

Note

Proceed as described in the technical documentation under „testing of demo programs“. Select the appropriate program in the display menu.

- Describe the behaviour of each of the three multidirectional casters with regard to motion and direction of movement during the motion programs „forward“, „circles“, „rectangle“ and „roam“.

Observe the „line of vision“ of the Robotino®.

## Project 1: Inspection of supplied components and commissioning of the Robotino®

<b>Project 1: Inspection of supplied components and commissioning of the Robotino®</b>	
Name:	Date:
Testing of motion behaviour	Sheet 2 of 3

- Which sensors respond?
- Explain the behaviour. What is the correlation between the movement of the wheels and motion behaviour?

<b>Description: Behaviour of „forward“ demo</b>	
Jacked-up Behaviour of casters	
Travelling Motion behaviour Sensors Behaviour of casters	
Additional observations	

<b>Description: Behaviour of „circle“ demo</b>	
Jacked-up Behaviour of casters	
Travelling Motion behaviour Sensors Behaviour of casters	
Additional observations	

# Project 1: Inspection of supplied components and commissioning of the Robotino®

<b>Project 1: Inspection of supplied components and commissioning of the Robotino®</b>	
Name:	Date:
Testing of motion behaviour	Sheet 2 of 3

<b>Description: Behaviour of „rectangle“ demo</b>	
Jacked-up Behaviour of casters	
Travelling Motion behaviour Sensors Behaviour of casters	
Additional observations	

<b>Description: Behaviour of „roam“ demo</b>	
Jacked-up Behaviour of casters	
Travelling Motion behaviour Sensors Behaviour of casters	
Additional observations	

**Project 1: Inspection of supplied components and commissioning of the Robotino®**

## Project 2

### Linear travelling of a mobile robot system in any direction

#### Training aims

#### Trainees

- are able to describe and program simple linear movements of a driven multi-axis system.
- are able to use function blocks in Robotino® View and create operating sequences for the actuation of the motors.
- are familiarised with the degrees of freedom of a driven multi-axis system.
- are able to factor in safety-relevant aspects in that the Robotino® stops in the event of a collision.
- are able to realise the programming of an omnidirectional drive and describe the basic functions.
- are familiarised with the main aspects of drive technology and are able to apply these.

#### Problem description

A robot system is to carry out feed tasks in storage systems and therefore needs to exhibit all-round flexibility and mobility. It is important for the robot system to be able to move at different speeds in any direction. To enable travel in any direction, it must be possible to drive all wheels of the system in any direction, i.e. they must be omnidirectional. This is effected by means of an omnidirectional drive. To ensure safety and so that the robot system does travel full force when approaching an obstacle, the system is to stop travelling in the event of collision with an obstacle.

#### Project assignment

1. Actuate the Robotino® motors such that the mobile robot system moves forward and backward on a level plane. Test the actuation of the motors using different speeds of revolution.
2. Test the mobility of the Robotino® by moving it in different directions and around its own axis without a change in orientation and speed. To do so, create and test the appropriate programs using the Omnidrive-function blocks.

During these tests the robot system may travel unforeseeable paths and it is therefore essential to protect the robot system environments against damage due to collisions. Create and test this program.

Basic conditions

- Robotino® View is installed on the PC and the software is started.
- A W-LAN connection is established to the Robotino® (technical documentation).

Work assignments

**To actuate the motor, to travel forward**

1. Move the Robotino® forward to see which motors need to be actuated.  
Observe the direction of rotation of the multidirectional wheels.
2. Create a function block diagram in Robotino® View whereby the Robotino® travels forward.  
Explain why the direction of travel of the motors must be the reverse in the case of straight ahead travel.
3. Create a collision protection function.
4. Test the different speeds.

**Moving backwards**

5. Create and test a program which enables the Robotino® to travel backwards.
6. Answer the questions regarding the components, constants and degrees of freedom.

**Operating sequence**

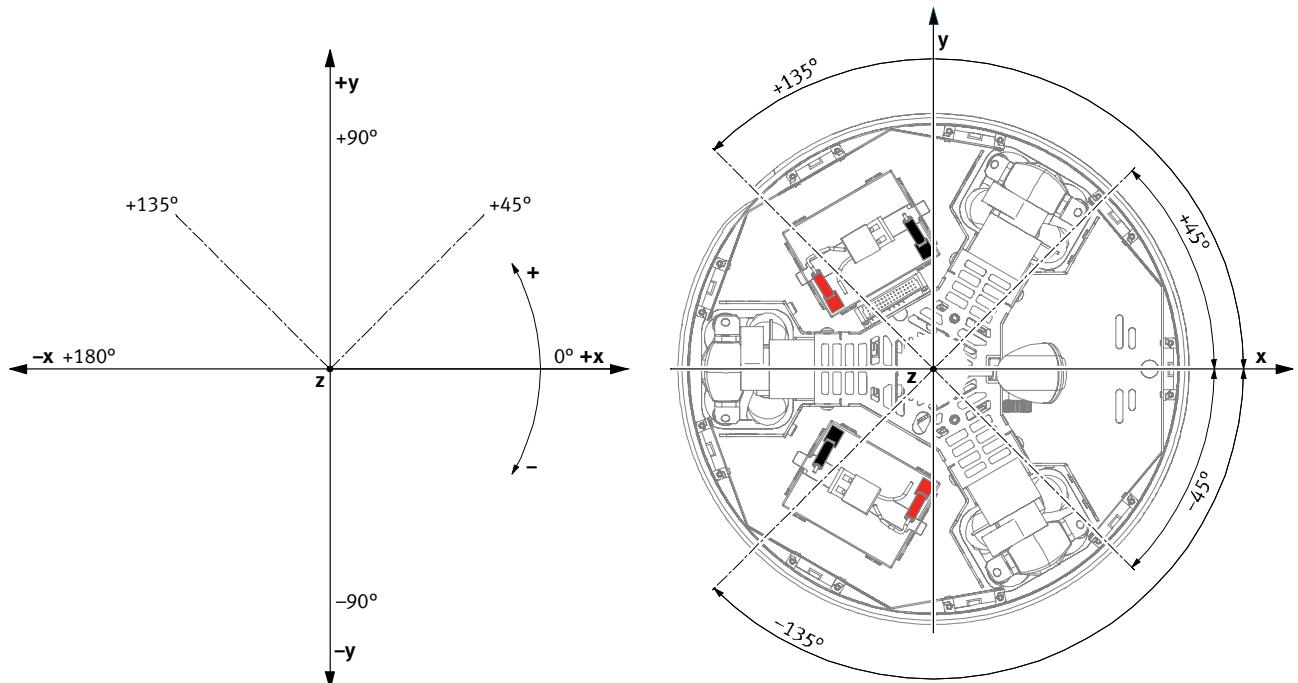
7. Create a program sequence whereby the Robotino® travels forward for 5 seconds, waits for 2 seconds and then travels backwards for 5 seconds against.
8. Test the sequence.

**Moving the Robotino® in any directions**

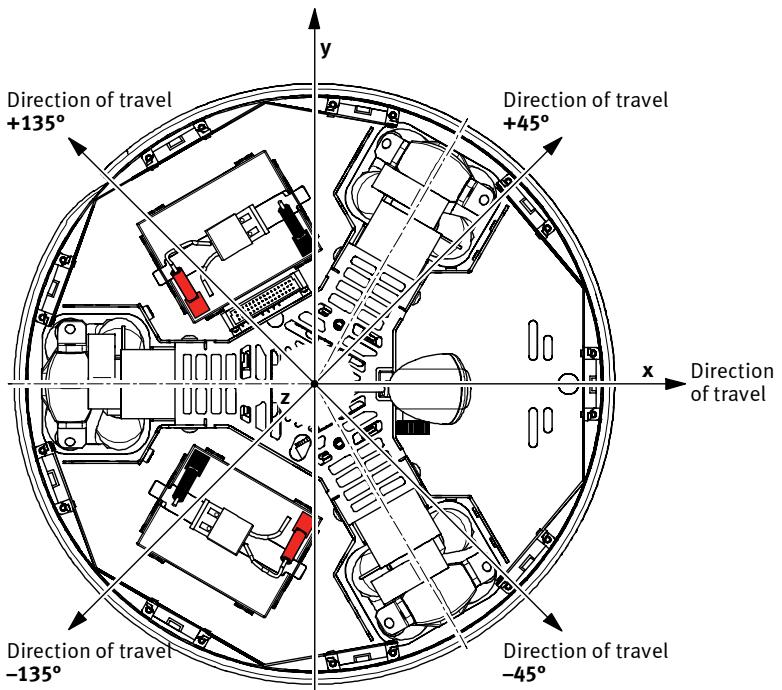
9. Let the Robotino® travel in all possible directions by setting the “omnidrive” function block in the motion program and the input device „control area“.
10. Answer the questions regarding the motion behaviour and degrees of freedom and familiarise yourself regarding the „omnidrive“ function block.
11. Create a program using the functions blocks „omnidrive“, three motors and one constant whereby the Robotino® travels forwards and backward at the same speed and same orientation.
12. Check whether the setpoint speed for the rear motor is constant = 0, see exercise of motor actuation without „omnidrive“.
13. Using experiments determine the forward speed in [mm/s] required in order for the two front motors to achieve the setpoint speed values of -1500 or 1500 [rpm].
14. Modify the program without the addition of any further function blocks so that Robotino® moves laterally to the right or left using the same speed and orientation.
15. Describe the design and the possibilities of the multidirectional wheels and the characteristics of the omnidirectional drive.
16. Modify the program without the additional of further function blocks so that the Robotino® rotates around the central axis.

17. Select a fixed orientation of the Robotino®. Modify the program by adding a further constant so that the Robotino® moves at 45° to the forward direction using the same speed and orientation
18. Modify the program without adding any further function blocks so that the Robotino® moves at 135° to the forward direction using the same speed and orientation.
19. Modify the program without adding any further function blocks so that the Robotino® moves at 45° to the forward direction using the same speed and orientation.
20. Modify the program without adding any further function blocks so that the Robotino® moves at 135° to the forward direction using the same speed and orientation.
21. Create an operating sequence whereby the Robotino® first travels forward and then back to the starting point, then travels at 45° in a forward direction and back again to the starting point, etc.

Positional sketch



## Project 2: Linear travelling of a mobile robot system in any direction



### Working aids

Technical documentation

Robotino® View Help „motor“, „operating sequence“,  
omnidirectional drive

Theory section:

Multidirectional wheels, drive and power transmission, degrees of freedom of  
multidirectional wheels, omnidirectional drive

## Project 2: Linear travelling of a mobile robot system in any direction

<b>Project 2: Linear travelling of a mobile robot system in any direction</b>	
Name:	Date:
Actuation of motor, forward movement of robot system	Sheet 1 of 4

- Move the Robotino® forward to see which motors need to be actuated. Observe the direction of rotation and rotational speed of the multidirectional wheels.
  - Create a function block diagram in Robotino® View whereby the Robotino® travels forward.
  - Explain why the direction of rotation of the motors must be the reverse in the case of straight ahead travel.
  - Create a collision protection function.
  - Test the different speeds.
- 
- Move the Robotino® forward to see which motors need to be actuated. Observe the direction of rotation of the multidirectional wheels.
- Position the Robotino® in front of you and move it forward, observing the direction of rotation and speed of the multidirectional wheels.

### Direction of rotation and rotational speed of the multidirectional wheels

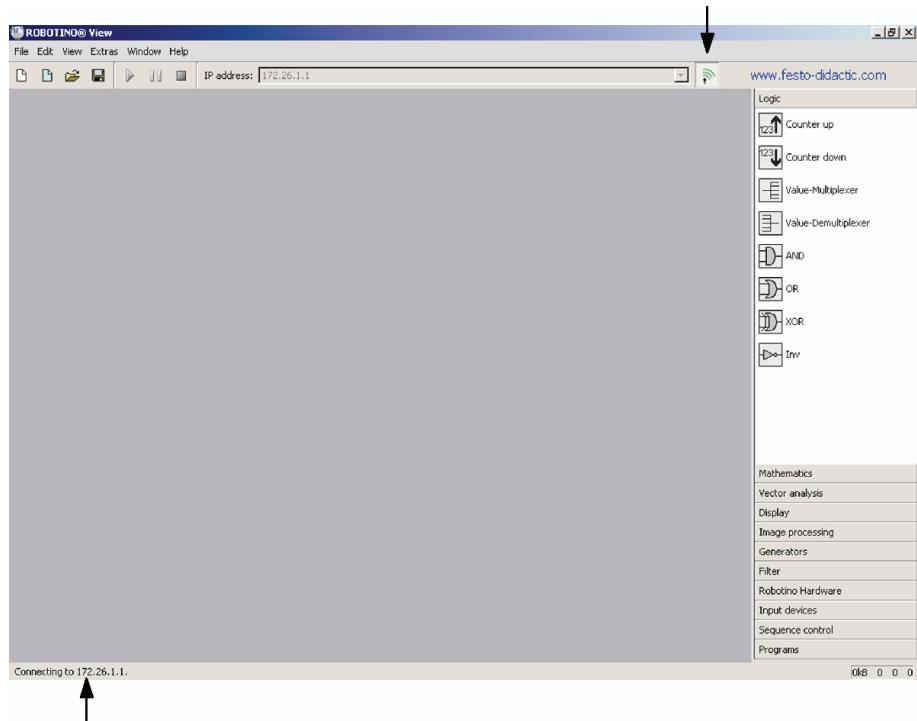
- Answer the following questions.

### Which motors must be actuated in order for the Robotino® to travel forward?

## Project 2: Linear travelling of a mobile robot system in any direction

<b>Project 2: Linear travelling of a mobile robot system in any direction</b>	
Name:	Date:
Actuation of motor, forward travel of robot system	Sheet 2 of 4

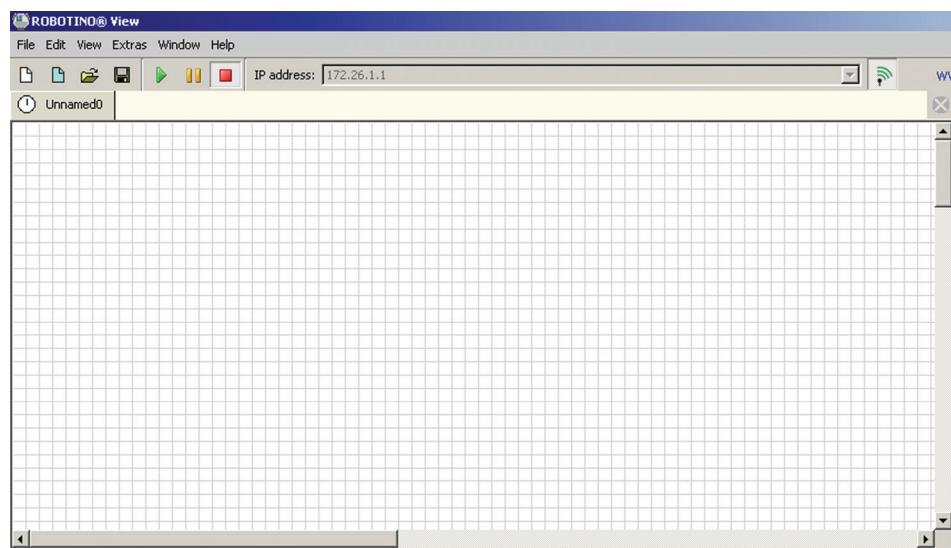
- Create a function block diagram in Robotino® View whereby the Robotino® travels forward.
- Jack up the system so that the wheels are freely movable.
- Connect the Robotino® to the power supply and switch it on.
- Start up Robotino® View and establish a connection between the Robotino® controller and Robotino® View (see technical documentation).



## Project 2: Linear travelling of a mobile robot system in any direction

Project 2: Linear travelling of a mobile robot system in any direction	
Name:	Date:
Program - actuation of motor, forward travel of robot system	Sheet 3 of 4

- Open a blank function block diagram in Robotino® View.



- Drag two motor function blocks (function block library: Robotino® hardware → motor) into the function block diagram.
- Allocate exactly one motor of the robot system to each motor function block (see technical documentation).
- In the function block diagram of the motor function block, name the motors "MotorFrontLeft" and "MotorFrontRight".
- Specify a constant speed for both motors.

### Note

Add two constants to the function block diagram (function block library → generators), and connect each of these to the input „setpoint speed“ of the two motors. Designate the constants accordingly with „SpeedLeft“ and „SpeedRight“.

## Project 2: Linear travelling of a mobile robot system in any direction

<b>Project 2: Linear travelling of a mobile robot system in any direction</b>	
Name:	Date:
Program –actuation of motor, forward travel of robot system	Sheet 4 of 4

- Start the application by clicking onto the Start symbol . Change the values of the two constants. Note that the setpoint speed is measured in rpm.
- Observe the behaviour of the robot system both in the jacked-up state and travelling state.

### Note

The technical documentation specifies that the Robotino® travels forward when moving in a linear direction in the line of vision of the camera.

- If you select the values of the constants, e.g.  
SpeedLeft = - 500 [rpm],  
SpeedRight = 1500 [rpm]  
the Robotino® travels forward.

### Note

Should the motors run erratically and this impairs the travelling of the Robotino®, then check the standard setting of the PID closed-loop motor controller in the function block dialogue of the motors:  
Kp = 0.9  
Ki = 0.01  
Kd = 0.0

- Explain why the direction of rotation of the motors must be the reverse in the case of straight ahead travel.

### Explanation of the reverse direction of rotation of motors

Project 2: Linear travelling of a mobile robot system in any direction

<b>Project 2: Linear travelling of a mobile robot system in any direction</b>	
Name:	Date:
Collision protection function	Sheet 1 of 1

- Create a collision protection function.
- Test different speeds.

Project 2: Linear travelling of a mobile robot system in any direction

<b>Project 2: Linear travelling of a mobile robot system in any direction</b>	
Name:	Date:
Description of the bumper function	Sheet 1 of 1

**Briefly describe the bumper of the Robotino®.**

**Describe the function of the bumper in Robotino® View.**

## Project 2: Linear travelling of a mobile robot system in any direction

<b>Project 2: Linear travelling of a mobile robot system in any direction</b>	
Name:	Date:
Programming of a stop function	Sheet 1 of 1

Create a function block diagram with the following function in order to explain the behaviour of the bumper:

A Robotino® motor is to be actuated such that it stops if the bumper is touched and re-start when it is released.

- Which components are required for this control program?

Quantity	Component

- Create the control program in Robotino® View and save it.

<b>Project 2: Linear travelling of a mobile robot system in any direction</b>	
Name:	Date:
Testing and evaluation of the function block diagram	Sheet 1 of 1

- Jack up the Robotino® and test your function block diagram.

<b>Test</b>	
Touch bumper, motor stops	
Release bumper, motor re-starts	

**What do you need to effect in a controller program with travel functions if you want to use this function as collision protection?**

<b>Evaluate the function of this solution in respect of the required collision protection function.</b>

## Project 2: Linear travelling of a mobile robot system in any direction

<b>Project 2: Linear travelling of a mobile robot system in any direction</b>	
Name:	Date:
Realisation of the collision function	Sheet 1 of 1

- Which function in Robotino® View fulfils the function of control program termination?
- Create a termination function based on the bumper function and save the program.

### Note

Integrate this collision protection function into all of your previously created function block diagrams in order to increase their safety.

- Test different speeds.

## Project 2: Linear travelling of a mobile robot system in any direction

<b>Project 2: Linear travelling of a mobile robot system in any direction</b>	
Name:	Date:
Backward motion program	Sheet 1 of 2

- Create and test a program which enables the Robotino® to travel backwards.
- Answer the questions regarding the components, constants and degrees of freedom.
  
- Create and test a program that enables the Robotino® to travel backwards.
  - Use the same components here as those in the program for forward travel.
  - Answer the following questions.

**Which motors must be actuated how in order for the Robotino® to travel backwards?**

**Which components are required for this?**

Project 2: Linear travelling of a mobile robot system in any direction

<b>Project 2: Linear travelling of a mobile robot system in any direction</b>	
Name:	Date:
Backward motion program	Sheet 2 of 2

SpeedLeft = \_\_\_\_\_

SpeedRight = \_\_\_\_\_

**What differences can you identify in the values of the two constants during the forward and backward motion program?**

**State which degrees of freedom are enabled in the motion behaviour of the Robotino® subsequent to the programs being executed.**

## Project 2: Linear travelling of a mobile robot system in any direction

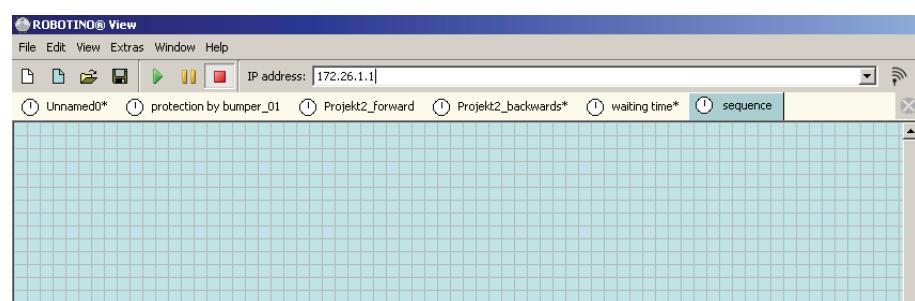
Project 2: Linear travelling of a mobile robot system in any direction	
Name:	Date:
Operating sequence	Sheet 1 of 2

- Create a program sequence whereby the Robotino® travels forwards for 5 seconds, waits for 2 seconds and then travels backwards for 5 seconds.
- Test the sequence.
  
- Create a program sequence whereby the Robotino® travels forwards for 5 seconds, waits for 2 seconds and then travels backward for 5 seconds.
  
- Create the individual programs and operating sequence.
- First create a program whereby the Robotino® travels forwards for five seconds.

### Notes

Consider which of the modules you require from the function block library to realise the timing.

- Assign the following values to the constants:  
SpeedLeft = -1500  
SpeedRight = 1500  
Travel time = 5000 [ms]
- Create a program accordingly whereby the Robotino® travels backwards for five seconds and name it.
- Now create a program that generates a wait time of 2 seconds and name it **waitingtime.rvm**.
- **Waiting time** constant = 2000
- Download the three programs **Project2\_forwards.rvm**, **project2\_backwards.rvm** and **waitingtime.rvm**.
- Start a new sequence control program and name it **project2\_sequence**.



<b>Project 2: Linear travelling of a mobile robot system in any direction</b>	
Name:	Date:
Operating sequence	Sheet 2 of 2

- First enter the three programs from the library **programs**.
- Connect output A of the forward motion program to the input of the waiting time program. Then connect output A of the waiting time program to the backward motion program.

These connections effect the following:

When the forward program is completed, the waiting time program is started and, after 2 seconds have expired, the backward program is started.

- You then connect output A of the backward program to the input of the forward program.

You now still need to establish which program is to be started when the sequence program is started.

- To do so, select the start module from the sequence control library and connect it to the input of the forward program.
  - Test the sequence.

## Project 2: Linear travelling of a mobile robot system in any direction

<b>Project 2: Linear travelling of a mobile robot system in any direction</b>	
Name:	Date:
Omnidirectional drive, all directions using “control field”	Sheet 1 of 1

- Let the Robotino® travel in all possible directions by using the “omnidrive” function block in the motion program and the input device „control field“.
- Answer the questions regarding the motion behaviour, degrees of freedom and familiarise yourself regarding the „omnidrive“ function block.
  
- Let the Robotino® travel in all possible direction by using the „omnidrive“ function block in the motion program and the „control area“ input device.
  
- Drag a control field from the list of input devices into a new function block diagram.
- Connect the outputs of the control field to the inputs (x,y,Omega) of the „omnidrive“ function block.
- Connect the outputs of the „omnidrive“ to the inputs for the setpoint speed of the three motors.
- Find out via Help about the operation of the control field and then start the program.
- Display the data (Ctrl-D or “Display Data“ under View) and note the motor values displayed.

**Describe the motion behaviour and state the possible degrees of freedom you have observed  
(flexibility of movement of bodies)**

- Familiarise yourself regarding the omnidirectional drive in the Theory section and the „omnidrive“ function block in the Robotino® View help.

## Project 2: Linear travelling of a mobile robot system in any direction

<b>Project 2: Linear travelling of a mobile robot system in any direction</b>	
Name:	Date:
Omnidrive, forward travel	Sheet 1 of 2

- Create a program using the function blocks „omnidrive“, three motors and one constant whereby the Robotino® travels forwards and backwards using the same speed and orientation.
- Check whether the setpoint speed for the rear motor is constant = 0, see exercise on motor actuation without „omnidrive“
- Using experiments, determine the forward speed in [mm/s] required to obtain the setpoint values of -1500 or 1500 {rpm} for the two front motors
- Using the function blocks „omnidrive“, three „motors“ and one „constant“ create a program whereby the Robotino® travels forwards and backwards using the same speed and orientation

### Note

The „omnidrive“ function block is contained in the hardware functions library and describes a kinematic model of the Robotino®. The inputs on the „lefthand“ side are

- Setpoint speed in x-direction [mm/s]
- Setpoint speed in y-direction [mm/s]
- Setpoint speed in [degree/s]

The module supplies as outputs the setpoint speeds in rpm “revolutions per minute” for the three motors.

The coordinate system is selected such that the positive x-axis corresponds to the forward direction for the Robotino®.

- Jack up the system again and open a new function block diagram in Robotino® View.
- Create a new program using the components omnidrive, three motors and one constant.
- Connect and define the elements.

### Note

If the value of the constant is 100, this results in a setpoint speed of 100 [mm/s], i.e. 0.36 [km/h].

- Switch on the data display (Ctrl-D or „Display Data“ under View “). Note the motor values in particular.

## Project 2: Linear travelling of a mobile robot system in any direction

<b>Project 2: Linear travelling of a mobile robot system in any direction</b>	
Name:	Date:
Omnidrive, forward travel	Sheet 2 of 2

- Check whether the setpoint speed for the rear motor is constant = 0, see exercise on motor actuation without omnidrive.

### Conclusion

- Using experiments, determine the forward speed in [mm/s] required to obtain the speed values of -1500 or 1500 rpm for the two front motors.

Answer: \_\_\_\_\_

## Project 2: Linear travelling of a mobile robot system in any direction

<b>Project 2: Linear travelling of a mobile robot system in any direction</b>	
Name:	Date:
Omnidrive, backward travel	Sheet 1 of 1

- A backward movement is obtained if you specify a negative setpoint speed in the x-direction. Put this statement to the test.

**What needs to be changed to enable the Robotino® to travel backwards using the „omnidrive“?**

- Test the „backward travel“ program

## Project 2: Linear travelling of a mobile robot system in any direction

<b>Project 2: Linear travelling of a mobile robot system in any direction</b>	
Name:	Date:
Omnidrive, lateral travel	Sheet 1 of 1

- Modify the program without adding any further function blocks so that the Robotino® travels laterally to right or left using the same speed and orientation.
- Observe the behaviour of the multidirectional wheels.
- Describe the design and the options of the multidirectional wheels and characteristics of the omnidirectional drive.
  
- Modify the program without adding any further function blocks so that the Robotino® travels laterally to the right or left using the same speed and orientation.
  
- Jack up the system again.
- Connect the components that ensure that the Robotino® travels laterally.
- Start-up the program, once in the jacked-up state and once in the mobile state.
- Answer the following questions.

### How many degrees of freedom are required for lateral travel?

- Observe the behaviour of the multidirectional wheels.

### Observation

## Project 2: Linear travelling of a mobile robot system in any direction

<b>Project 2: Linear travelling of a mobile robot system in any direction</b>	
Name:	Date:
Omnidrive, circular travel	Sheet 1 of 1

- Modify the program without adding any further function blocks so that the Robotino® rotates around the central axis.
- Connect the constant to the Omega setpoint connection of the „omnidrive“ and observe the behaviour.
- Answer the questions regarding the possible degrees of freedom.

### Behaviour

- Answer the following questions regarding the degrees of freedom.

### How many degrees of freedom were required and why?

## Project 2: Linear travelling of a mobile robot system in any direction

<b>Project 2: Linear travelling of a mobile robot system in any direction</b>	
Name:	Date:
Omnidirectional drive: Multidirectional wheels	Sheet 1 of 1

- Describe the design and the possibilities of the multidirectional wheels and characteristics of an omnidirectional drive.

Description

- Characteristics of an omnidirectional drive.

Advantage	Disadvantage

## Project 2: Linear travelling of a mobile robot system in any direction

<b>Project 2: Linear travelling of a mobile robot system in any direction</b>	
Name:	Date:
Travel at 45° to forward direction	Sheet 1 of 1

- Select a fixed Robotino® orientation. Modify the program by adding a further constant so that the Robotino® travels at 45° to the forward direction using the same speed and orientation.
- Consider how you can realise travel at 45° to the forward direction taking into account the direction of travel and speed.

Note

Please see the positional sketch in the problem description.

### Initial considerations: Speed

## Project 2: Linear travelling of a mobile robot system in any direction

<b>Project 2: Linear travelling of a mobile robot system in any direction</b>	
Name:	Date:
Travel at 135 ° to the forward direction	Sheet 1 of 1

- Modify the program without adding further function blocks so that the Robotino® travels at 135° to the forward direction using the same speed and orientation.

## Project 2: Linear travelling of a mobile robot system in any direction

<b>Project 2: Linear travelling of a mobile robot system in any direction</b>	
Name:	Date:
Travel at -45 ° to the forward direction	Sheet 1 of 1

- Modify the program without adding further function blocks so that the Robotino® travels at -45° to the forward direction using the same speed and orientation.

## Project 2: Linear travelling of a mobile robot system in any direction

<b>Project 2: Linear travelling of a mobile robot system in any direction</b>	
Name:	Date:
Travel at -135 ° to the forward direction	Sheet 1 of 1

- Modify the program without adding further function blocks so that the Robotino® travels at -135° to the forward direction using the same speed and orientation.

## Project 2: Linear travelling of a mobile robot system in any direction

<b>Project 2: Linear travelling of a mobile robot system in any direction</b>	
Name:	Date:
Operating sequence: Star-shaped travel in 45° sections in anti-clockwise direction	Sheet 1 of 1

- Create an operating sequence so that the Robotino® initially travels forward and then back to the starting point and subsequently at 45° in forward direction and back to the starting point, etc.

Note - sequence program      Please ensure that the „start module“ is integrated into the sequence program.

To continue star-shaped travelling you still only need to change the constants or the constant prefix for the correct direction of travel.

Project 2: Linear travelling of a mobile robot system in any direction

## Project 3

### Linear travelling and positioning of a mobile robot system

#### Training aims

##### Trainees

- are able to travel a multiaxis system a defined distance (with and without the use of the „omnidrive“ function block).
- are able to apply their knowledge of trigonometry and vector analysis.
- are familiarised with the main aspects of drive technology and are able to apply these.
- are able to carry out and evaluate measured value analyses for positioning accuracy.
- are able to optimise programs so that the deviation from the setpoint distance lies within the mm range.

#### Problem description

A robot system is to carry out feed tasks in storage systems and therefore needs to be able to approach defined positions on a level plane.

#### Project assignment

First create a program without and then with the „omnidrive“ function block so that the Robotino® travels forward a distance of 1 m. Two different methods are to be used to determine the travel distance:

- Calculate the distance by the number of revolutions (without the use of the „omnidrive“ function block).
- Travel time (by using the „omnidrive“ function block).
- Integration (return path of the actual distance travelled).
- Compare the results and explain the deviations.

#### General conditions

- Robotino® View is installed on the PC and the software started.
- A W-LAN connection is established to the Robotino® (technical documentation).

#### Work assignments

Travelling a distance in forward direction without the use of the „omnidrive“ function block.

1. Travel a distance of 1 m with the Robotino® without the use of the „omnidrive“ function block.
2. Determine the number of revolutions to be carried out by the two wheels in order for the Robotino® to travel forward the distance of 1 m, and calculate the number of motor increments.
3. Test a program which enables the Robotino® to travel forward a distance of 1 m.
4. Measure the distance travelled.
5. What problems arise at increased speeds?
6. Explain why deviations occur from the target value of 1 m.

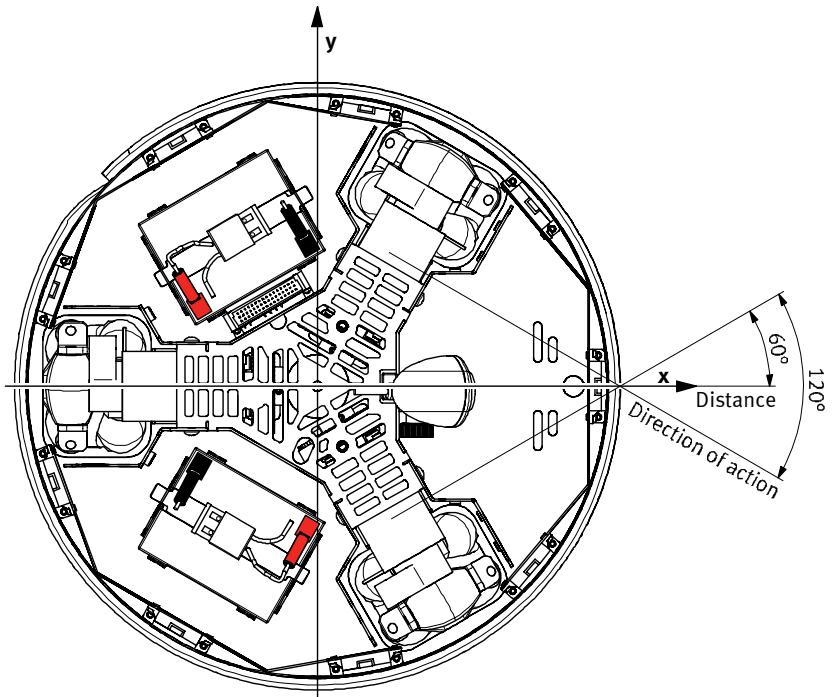
**Repetition and absolute accuracy**

1. Carry out 10 measurements by travelling the distance from the same starting point in the same direction and at the same speed. Determine the average deviation from the defined distance of 1 m.
2. Carry out these experiments for different speeds and explain the results obtained.

**Travelling forward a defined distance using the omnidrive function module**

1. Using the omnidrive function block, the Robotino® is to travel forward 1 metre and then stop.
2. Calculate the time the Robotino® needs to travel at a speed of [100 mm/s] in order to cover a distance of 1 m.
3. Create and test the program by adding a time module to your „forward travel“ program.
4. Carry out the test for different speeds and compare the results with the results obtained without the use of the omnidrive function block.
5. Explain the possible reasons for a deviation. During operation, monitor the actual values for speeds of revolution and speeds in x-, y-direction displayed by the „omnidrive“ function block.
6. Describe a concept for the optimisation of the program.
7. How can you calculate the distance travelled from a constant display of the actual speed in x-direction occurring at a clock pulse of approx. 23 ms?
8. Optimise your program so that the deviation from the setpoint distance is within the mm range.
9. Start the „odometry“ program on the same surface and evaluate the results obtained

Positional sketch for  
„forward travel of a distance  
without the use of the  
omnidrive function block“



Working aids

Technical documentation,  
Robotino® View Help - „omnidrive“,  
Theory section: Omnidirectional drive

<b>Project 3: Linear travelling and positioning of a mobile robot system</b>	
Name:	Date:
Travelling a distance of 1 m without the use of the „omnidrive“ function block	Sheet 1 of 4

- Travel the Robotino® a distance of 1 m without the use of the „omnidrive“ function block.
  - Determine the number of revolutions to be carried out by both wheels in order for the Robotino® to travel forward a distance of 1 m, and calculate the number of motor increments.
  - Test the program created which enables the Robotino® to travel forward a distance of 1 m.
  - Measure the distance travelled.
  - What problems arise at higher speeds?
  - Explain why deviations occur from the target value of 1 m.
- 
- Determine the number of revolutions to be carried out by both wheels in order for the Robotino® to travel forward a distance of 1 m, and calculate the number of motor increments.

Note

Please note the following data:

Path: 1000 mm, 2 wheels are driven

Wheel circumference = distance travelled in one revolution

Please also note the drawing in the positional sketch of the problem definition and move the diagram of the Robotino® in forward direction.

**Result – number of revolutions**

<b>Project 3: Linear travelling and positioning of a mobile robot system</b>	
Name:	Date:
Travelling a distance of 1 m – without the use of the „omnidrive“ function block	Sheet 2 of 4

- Calculation of motor increments for the distance:

Note

The gear ratio is 1:16, see technical documentation.

1 motor revolution corresponds to 2048 increments, see technical documentation

**Motor increments**

- Test the program created on this basis which enables the Robotino® to travel forward a distance of 1 m.

Note

Please note that the actual position on the motor is output in increments whereby the current position can be compared with the target position of 1 m = x increments.

- The motor function block supplies three output values:
  - Actual speed in [rpm]
  - Actual position [number of travelled increments]
  - Motor current [mA]
- In order to obtain a correct indication of the actual position it is necessary to initially reset the incremental counter to 0 when starting the movement. This is effected via the reset constant which is connected to the reset input of the motor function block.

<b>Project 3: Linear travelling and positioning of a mobile robot system</b>	
Name:	Date:
Travelling a distance of 1 m – without the use of the „omnidrive“ function block	Sheet 3 of 4

- |                  |   |
|------------------|---|
| Program sequence | <ul style="list-style-type: none"> <li>First set <b>Reset</b> = 1 and the speed = 0.</li> <li>Start the program. The incremental counter is reset.</li> <li>Stop the program and set the speeds to -1500 or 1500 and <b>Reset</b> = 0. Re-start the program.</li> </ul> |
|------------------|---|

Note Please note that the incremental counter counts down in the case of positive speed.

Should you re-start the problem, the program may stop immediately since old input values are still being evaluated in the comparison operator IncrementComparison. You can avoid this problem by selecting a different calculation mode for the function block diagram:

- Select the menu Extras → Options → Step mode → Fast. In this mode, all function blocks are fully calculated serially in sequence. The serial allocation is obtained from left to right from the geometric sequence of the function blocks in the diagram.
- What problems arise with higher speeds?
- Explain why deviations occur from the target value of 1 m.
- Test the entire program using different speeds
- Display the data (Ctrl-D or “Display Data“ under View) and observe the displayed values of the motors
- Measure the distance travelled
- Compare the results using two different calculation methods for the function block diagram.

**Result of comparison – step mode**

Project 3: Linear travelling and positioning of a mobile robot system

<b>Project 3: Linear travelling and positioning of a mobile robot system</b>	
Name:	Date:
Travelling a distance of 1 m – without the use of the „omnidrive“ function block	Sheet 4 of 4

**Explanation and behaviour at higher speeds**

## Project 3: Linear travelling and positioning of a mobile robot system

<b>Project 3: Linear travelling and positioning of a mobile robot system</b>	
Name:	Date:
Repetition and absolute accuracy	Sheet 1 of 7

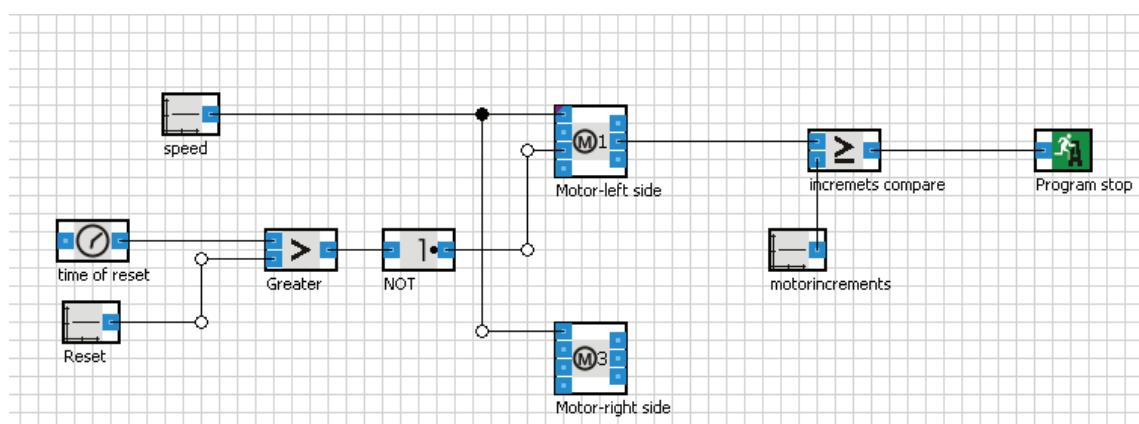
- Carry out 10 measurements by travelling the distance each time from the same starting point in the same direction and at the same speed. Determine the average deviation from the defined distance of 1 m.
- Carry out these experiments for different speeds and explain the results.

Please note

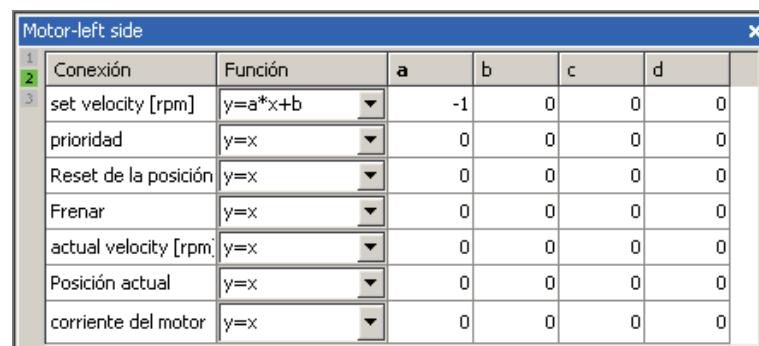
The rechargeable batteries should be fully charged.

Note

For the purpose of optimisation please use the following solution



- Only one constant is given for the motors, whereby it should be noted that the setpoint speed of the front-left motor is to be multiplied by -1 (see note on page A-48).
- Use the function block dialogue of the motor and apply the appropriate formula for the setpoint speed.



## Project 3: Linear travelling and positioning of a mobile robot system

<b>Project 3: Linear travelling and positioning of a mobile robot system</b>	
Name:	Date:
Repetition and absolute accuracy	Sheet 2 of 7

Note                    Calculation of average deviation or of the arithmetic mean value of the distances travelled:

$$\bar{x}_{\text{arithm}} = \frac{1}{n} \sum_{i=1}^n x_i = \frac{x_1 + x_2 + \dots + x_n}{n}$$

Analyse your measurements and the results as a form of quality control of your system, for example the quality of the motors, wheels and your assembly.

By means of the measurements you are testing the following: How does the system cope with the requirements; does it meet the necessary requirements and the requirements posed?

Measurement results for distances travelled on different surfaces:

Tiled floor                    Speed: 800 rpm

Distance travelled in m	Number of increments	Deviation from setpoint value of 1 m
$\bar{x}$	$\bar{x}$	$\bar{x}$

Project 3: Linear travelling and positioning of a mobile robot system

<b>Project 3: Linear travelling and positioning of a mobile robot system</b>	
Name:	Date:
Repetition and absolute accuracy	Sheet 3 of 7

Speed: 1500 rpm

<b>Distance travelled in m</b>	<b>Number of increments</b>	<b>Deviations from setpoint value of 1 m</b>
$\bar{x}$	$\bar{x}$	$\bar{x}$

Speed: 2500 rpm

<b>Distance travelled in m</b>	<b>Number of increments</b>	<b>Deviations from setpoint value of 1 m</b>
$\bar{x}$	$\bar{x}$	$\bar{x}$

Project 3: Linear travelling and positioning of a mobile robot system

<b>Project 3: Linear travelling and positioning of a mobile robot system</b>	
Name:	Date:
Repetition and absolute accuracy	Sheet 4 of 7

Laminated wood flooring      Speed: 800 rpm  
 (Robotino® operating area)

<b>Distance travelled in m</b>	<b>Number of increments</b>	<b>Deviation from setpoint value of 1 m</b>
$\bar{x}$	$\bar{x}$	$\bar{x}$

Speed: 1500 rpm

<b>Distance travelled in m</b>	<b>Number of increments</b>	<b>Deviation from setpoint value of 1 m</b>
$\bar{x}$	$\bar{x}$	$\bar{x}$

Project 3: Linear travelling and positioning of a mobile robot system

<b>Project 3: Linear travelling and positioning of a mobile robot system</b>	
Name:	Date:
Repetition and absolute accuracy	Sheet 5 of 7

Speed: 2500 rpm

<b>Distance travelled in m</b>	<b>Number of increments</b>	<b>Deviation from setpoint value of 1 m</b>
$\bar{x}$	$\bar{x}$	$\bar{x}$

Velour carpeted floor

Speed: 800 rpm

<b>Distance travelled in m</b>	<b>Number of increments</b>	<b>Deviation from setpoint value of 1 m</b>
$\bar{x}$	$\bar{x}$	$\bar{x}$

Project 3: Linear travelling and positioning of a mobile robot system

<b>Project 3: Linear travelling and positioning of a mobile robot system</b>	
Name:	Date:
Repetition and absolute accuracy	Sheet 6 of 7

Speed: 1500 rpm

<b>Distance travelled in m</b>	<b>Number of increments</b>	<b>Deviation from setpoint value of 1 m</b>
$\bar{x}$	$\bar{x}$	$\bar{x}$

Speed: 2500 rpm

<b>Distance travelled in m</b>	<b>Number of increments</b>	<b>Deviation from setpoint value of 1 m</b>
$\bar{x}$	$\bar{x}$	$\bar{x}$

Project 3: Linear travelling and positioning of a mobile robot system

<b>Project 3: Linear travelling and positioning of a mobile robot system</b>	
Name:	Date:
Repetition and absolute accuracy	Sheet 7 of 7

**Explanation of results obtained**

<b>Project 3: Linear travelling and positioning of a mobile robot system</b>	
Name:	Date:
Travelling a distance of 1 m, using the „omnidrive“ function block	Sheet 1 of 10

- Using the omnidrive function block, the Robotino® is to travel forward 1 metre and then stop.
- Calculate the time the Robotino® has to travel using a speed of [100 mm/s] in order to cover a distance of 1 m.
- Create and test the program by adding a time module to your „forward travel“ program.
- Carry out the test for different speeds and compare the results with the results obtained without the use of the omnidrive function block.
- Explain the possible reasons for a deviation. During operation, observe the actual values for rotational speed and speeds in x-, y-direction displayed by the „omnidrive“ function block.
- Describe a concept for the optimisation of the program.
- How can you calculate the distance travelled from a continuous display of the actual speed in x-direction realised at a clock cycle of approx. 23 ms?
- Optimise your program so that the deviations of the setpoint distance are within the mm range.
- Start the „odometry“ program using the same surface and evaluate the results obtained.
  
- Calculate the time the Robotino® has to travel at a speed of [100 mm/s] in order to cover a distance of 1 m.

**Calculation**

## Project 3: Linear travelling and positioning of a mobile robot system

<b>Project 3: Linear travelling and positioning of a mobile robot system</b>	
Name:	Date:
Travelling a distance of 1 m, using the „omnidrive“ function block	Sheet 2 of 10

<b>Result</b>

- Create and test the program by adding a time module to your „forward travel“ program.
  - Speed = 100 [mm/s]
  - Test the program and measure the actual distanced travelled

<b>Actual distance travelled</b>

Project 3: Linear travelling and positioning of a mobile robot system

<b>Project 3: Linear travelling and positioning of a mobile robot system</b>	
Name:	Date:
Travelling a distance of 1 m, using the „omnidrive“ function block	Sheet 3 of 10

- Carry out the test for different speeds and compare the results with the results obtained without the use of the omnidrive function block.

Carpeted floor

Distance:  $s = 1 \text{ m}$  / Speed:  $v = 100 \text{ mm/s}$  / Time:  $t = 10\text{s} = 10000 \text{ ms}$

Distance travelled in m	Deviation from setpoint value of 1 m
$\bar{x} =$	$\bar{x} =$

Project 3: Linear travelling and positioning of a mobile robot system

<b>Project 3: Linear travelling and positioning of a mobile robot system</b>	
Name:	Date:
Travelling a distance of 1 m, using the „omnidrive“ function block	Sheet 4 of 10

Distance:  $s = 1 \text{ m}$  / Speed:  $v = 50 \text{ mm/s}$  / Time:  $t = 20\text{s} = 20000 \text{ ms}$

<b>Distance travelled in m</b>	<b>Deviation from setpoint value of 1 m</b>
$\bar{x} =$	$\bar{x} =$

Distance:  $s = 1 \text{ m}$  / Speed:  $v = 200 \text{ mm/s}$  / Time:  $t = 5 \text{ s} = 5000 \text{ ms}$

<b>Distance travelled in m</b>	<b>Deviation from setpoint value of 1 m</b>
$\bar{x} =$	$\bar{x} =$

Project 3: Linear travelling and positioning of a mobile robot system

<b>Project 3: Linear travelling and positioning of a mobile robot system</b>	
Name:	Date:
Travelling a distance of 1 m, using the „omnidrive“ function block	Sheet 5 of 10

Velour carpeted floor

Distance:  $s = 1 \text{ m}$  / Speed:  $v = 100 \text{ mm/s}$  / Time:  $t = 10\text{s} = 10,000 \text{ ms}$

Distance travelled in m	Deviation from setpoint value of 1 m
$\bar{x} =$	$\bar{x} =$

Distance:  $s = 1 \text{ m}$  / Speed:  $v = 200 \text{ mm/s}$  / Time:  $t = 5 \text{ s} = 5000 \text{ ms}$

Distance travelled in m	Deviation from setpoint value of 1 m
$\bar{x} =$	$\bar{x} =$

Project 3: Linear travelling and positioning of a mobile robot system

<b>Project 3: Linear travelling and positioning of a mobile robot system</b>	
Name:	Date:
Travelling a distance of 1 m, using the „omnidrive“ function block	Sheet 6 of 10

Distance:  $s = 1 \text{ m}$  / Speed:  $v = 400 \text{ mm/s}$  / Time:  $t = 2.5 \text{ s} = 2,500 \text{ ms}$

Distance travelled in m	Deviation from setpoint value of 1 m
$\bar{x} =$	$\bar{x} =$

Tiled floor

Distance:  $s = 1 \text{ m}$  / Speed:  $v = 100 \text{ mm/s}$  / Time:  $t = 10\text{s} = 10,000 \text{ ms}$

Distance travelled in m	Deviation from setpoint value of 1 m
$\bar{x} =$	$\bar{x} =$

## Project 3: Linear travelling and positioning of a mobile robot system

<b>Project 3: Linear travelling and positioning of a mobile robot system</b>	
Name:	Date:
Travelling a distance of 1 m, using the „omnidrive“ function block	Sheet 7 of 10

Distance:  $s = 1 \text{ m}$  / Speed:  $v = 400 \text{ mm/s}$  / Time:  $t = 2.5 \text{ s} = 2,500 \text{ ms}$

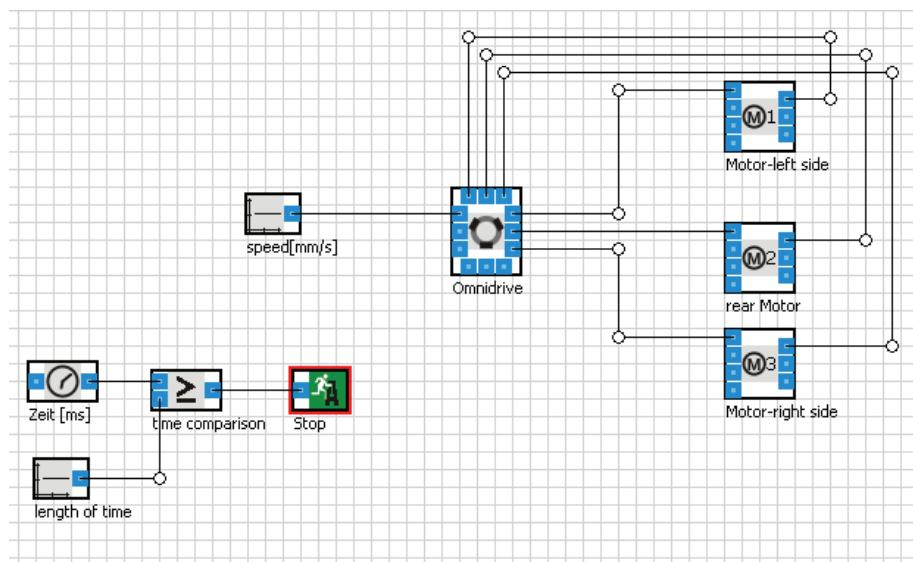
<b>Distance travelled in m</b>	<b>Deviation from setpoint value of 1 m</b>
$\bar{x} =$	$\bar{x} =$

<b>Comparison of results obtained with and without the use of the „omnidrive“</b>

## Project 3: Linear travelling and positioning of a mobile robot system

Project 3: Linear travelling and positioning of a mobile robot system	
Name:	Date:
Travelling a distance of 1 m, using the „omnidrive“ function block	Sheet 8 of 10

- Explain the possible reasons for a deviation. During operation, monitor the actual values for the rotational speed and speeds in x-, y-direction displayed by the omnidrive function block.
- Modify the program as follows to display the actual values for the speed in x- and y-direction:



### Observation

- |                                       |                 |
|---------------------------------------|-----------------|
| Actual speed in x-direction           | = approx. _____ |
| Actual speed in y-direction           | = approx. _____ |
| Actual speed in direction of rotation | = approx. _____ |
| Estimated value of distance travelled | = approx. _____ |

<b>Project 3: Linear travelling and positioning of a mobile robot system</b>	
Name:	Date:
Travelling a distance of 1 m, using the „omnidrive“ function block	Sheet 9 of 10

**Reasons for the deviations**

**Describe a concept for the optimisation of the program.**

- How can you calculate the distance travelled from a continuous display of the actual speed in x-direction occurring at a clock pulse of approx. 23 ms?

Note

If „VXactual“ designates the speed displayed, the Robotino® travels a distance of  $s = 0.023 \times VXactual [mm]$  until the next display.

- Optimise your program so that the deviation of the setpoint distance lies within the mm range.
  - Speed = 100 [mm/s]
  - Clock pulse = 0.023 [s]
- Test the program and evaluate the results obtained.

Project 3: Linear travelling and positioning of a mobile robot system

<b>Project 3: Linear travelling and positioning of a mobile robot system</b>	
Name:	Date:
Travelling a distance of 1 m, using the „omnidrive“ function block	Sheet 10 of 10

Evaluation of various measurements

Program: Project3\_Odometry.rvw  
Distance =  $s = 1 \text{ m} = 1000 \text{ mm}$  / Speed = 100 mm/s

Clock pulse of measurements = 0.023 s = 23 ms

Measured distance in mm	Vector length in mm (from Robotino® View)
$\bar{x} =$	$\bar{x} =$

Clock pulse of measurements: 0.0225 s = 22.5 ms

Measured distance in mm	Vector length in mm (from Robotino® View)
$\bar{x} =$	$\bar{x} =$

**Best results – timing**

## Project 4

### Path tracking of an automated guided vehicle system using two diffuse sensors

#### Training aims

##### Trainees

- are able to mount the diffuse sensors on the Robotino® and connect these to the I/O interface.
- are able to access the diffuse sensor signals by means of Robotino® View and evaluate these.
- are able to adjust the diffuse sensors.
- are familiarised with the switching method of the diffuse sensor.
- are able to use the diffuse sensors to control the Robotino®.
- are able to develop a strategy for path tracking.
- are able to create a simple sequence program which controls the required functions.
- are able to combine all the required functions into one closed-loop control program.

#### Problem description

The task is to travel an automated guided vehicle system on a preset path towards a loading station and to stop for the loading process when the station is reached. The path involves curves and is defined by means of coloured marking of the surface. The Robotino® acts as an automated guided vehicle system.

#### Project assignment

Mount the diffuse sensors on the Robotino®, connect these to the control unit and carry out a functional test.  
Develop a strategy for travelling along the marking and create function block diagrams for individual subfunctions of this strategy.  
Combine these into a sequence program and optimise your function block diagrams overall with regard to the travel time.

#### General conditions

The width of the marking must be narrower than the distance between the fibre optic cables.  
The colour of the marking must be of sufficient contrast to the floor surface.  
The marked strips must be darker than the floor surface.

## Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors

Work assignments	<ol style="list-style-type: none"><li>1. Mount the diffuse sensors on the Robotino® and connect these to the I/O interface.</li><li>2. Adjust the sensors and carry out a functional test.</li><li>3. Develop a strategy and corresponding function block diagrams for the functions required.</li><li>4. Develop a sequence program that comprises all the required functions and executes the path tracking automatically.</li><li>5. Carry out a test of your control program and optimise this with regard to the time required.</li><li>6. Develop a closed-loop control program which combines all the required functions into one function block diagram.</li></ol>
Working aids	<ul style="list-style-type: none"><li>• Ready-made programs</li><li>• Technical documentation</li><li>• Data sheets</li><li>• Theory section: Diffuse sensors</li><li>• Robotino® View Help</li></ul>
Possible additional exercises	<p>Carry out necessary program changes if the marking is lighter than the floor surface. Carry out any necessary program changes if the marking is wider than the sensor distance.</p>

**Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors**

<b>Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors</b>	
Name:	Date:
Creating a work plan	Sheet 1 of 1

- Create your work plan for this project. Determine all the necessary work steps as detailed as possible. Enter the work steps in the table below. Use these also as a check list for project documentation when working on the project.

<b>Activity</b>	<b>Completed</b>

Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors

<b>Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors</b>	
Name:	Date:
Mounting of diffuse sensors	Sheet 1 of 3

- Mount the two diffuse sensors at the points on the chassis provided for this. Describe how you proceed or record this when mounting the diffuse sensors.

Note            Mount the two fibre-optic cable heads within the closest possible distance of one another.

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors

<b>Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors</b>	
Name:	Date:
Mounting of diffuse sensors	Sheet 2 of 3

1. Enter the cables on the drawing shown below and label these with their characteristics and colour.



2. Connect the sensors to the power supply in accordance with your drawing.
3. Adjust the diffuse sensors and describe how you proceed.

---

---

---

---

---

---

---

---

---

---

Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors

<b>Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors</b>	
Name:	Date:
Mounting of diffuse sensors	Sheet 3 of 3

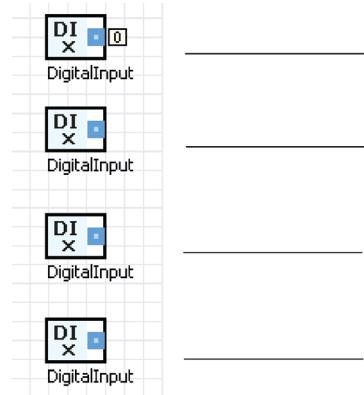
- Connect the diffuse sensors to the I/O interface. Connect the cables of the lefthand diffuse sensor to inputs DI0 and DI1 and those of the righthand diffuse sensor to inputs DI2 and DI3. Connect the black signal cables to inputs DI0 and DI2 and the white signal cables to DI1 and DI3.
- Enter the cables on the drawing shown below and label these according to their colour and the relevant sensor. In addition enter the voltage supply of the two sensors in the drawing.



Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors

<b>Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors</b>	
Name:	Date:
Selection of the inputs in Robotino® View	Sheet 1 of 2

- Enter all inputs of the two diffuse sensors in the diagram below. Label these according to the relevant sensor and the colour of the cables connected.



Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors

<b>Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors</b>	
Name:	Date:
Selection of the inputs in Robotino® View	Sheet 2 of 2

- In Robotino® View, assign the inputs to the individual input function blocks in accordance with your specification. Label the input function blocks according to your specifications.

## Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors

<b>Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors</b>	
Name:	Date:
Determining the sensor values of the individual inputs	Sheet 1 of 2

- To test the efficient functioning of the sensors, determine the sensor values for the different inputs in Robotino® View. The prerequisite is that the Robotino® stands on the intended operating surface and is adjusted accordingly. Enter the sensor values in the table below.

<b>Input</b>	<b>Signal</b>
Input DIO (sensor right)	
Input DI1 (sensor right)	
Input DI2 (sensor left)	
Input DI3 (sensor left)	

Evaluate the determined sensor signal of the inputs with regard to their switching function.

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors

<b>Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors</b>	
Name:	Date:
Determining the sensor values of the individual inputs	Sheet 2 of 2

- Consider which three possible situations may arise during the required path tracking and enter these in the appropriate column of the table below.
- Re-enact the different travel situations with the Robotino® and enter the input values occurring in the table below.

Note

To do so, use the sample program „Aufg-P4-01.rvw“

Use this table at a later stage when programming the control program.

<b>Travel situation</b>	<b>DIO</b>	<b>DI1</b>	<b>DI2</b>	<b>DI3</b>

Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors

<b>Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors</b>	
Name:	Date:
Development of the control system strategy	Sheet 1 of 1

- Use the table from the previous exercise and describe the start condition for each of the 3 travel situations, the function of a corresponding subprogram and a termination condition.

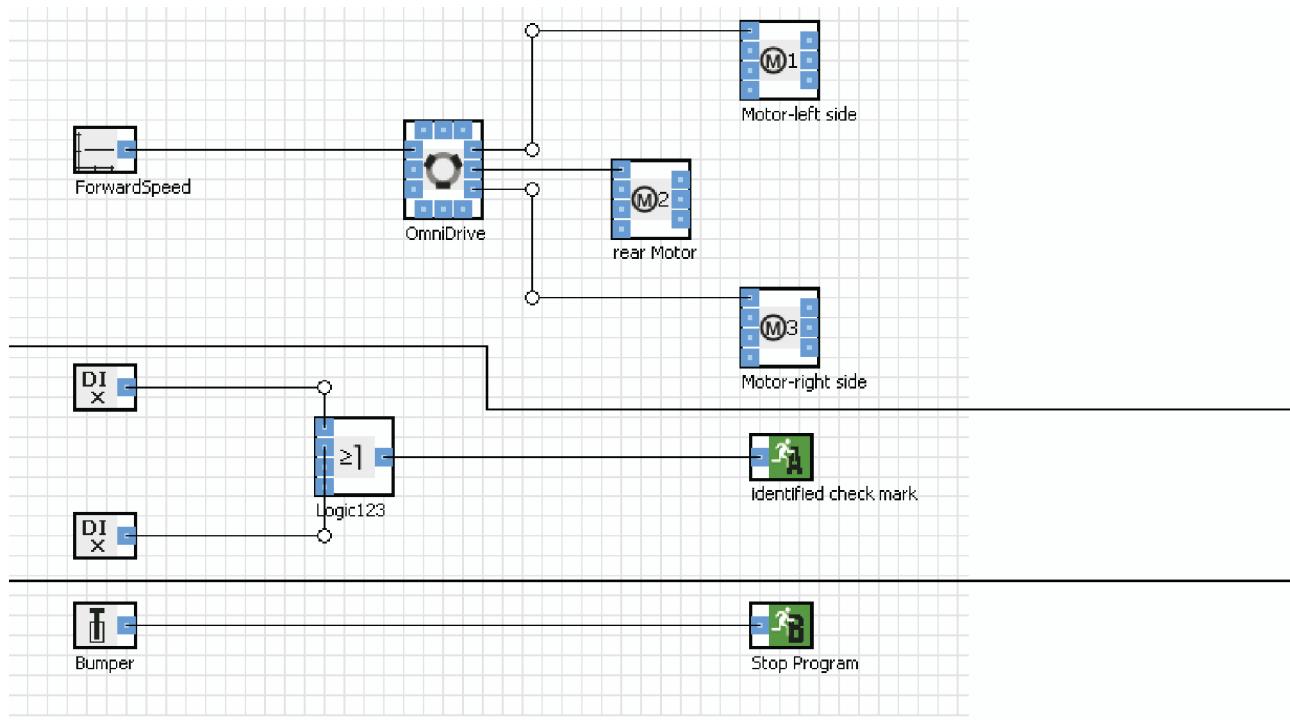
<b>Travel situations</b>

Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors

<b>Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors</b>	
Name:	Date:
Development of the control program	Sheet 1 of 7

Three sample programs are available for the development of the Robotino® control system.

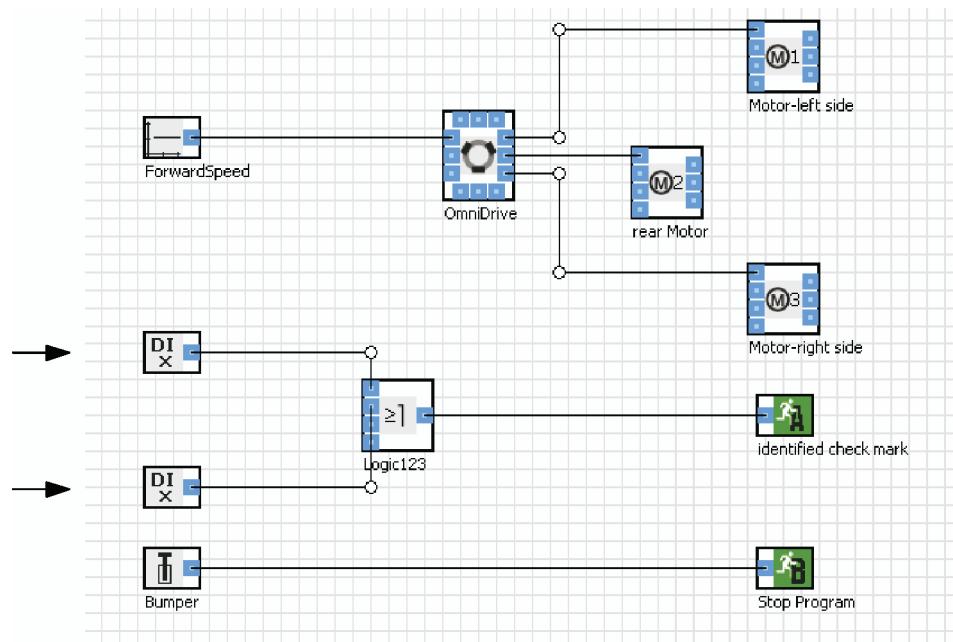
The first subprogram is called Aufg-P4-02.rvw. The task is to complete this program. To make it easier to develop the program it is useful to divide it into individual groups of functions. Assign the pertaining functions to the three groups of function blocks by briefly describing these. Enter this description in the diagram below.



Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors

Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors	
Name:	Date:
Development of the control program	Sheet 2 of 7

- Assign the appropriate input to the input function blocks. (DI0 to DI3). Enter these in the diagram below.



Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors

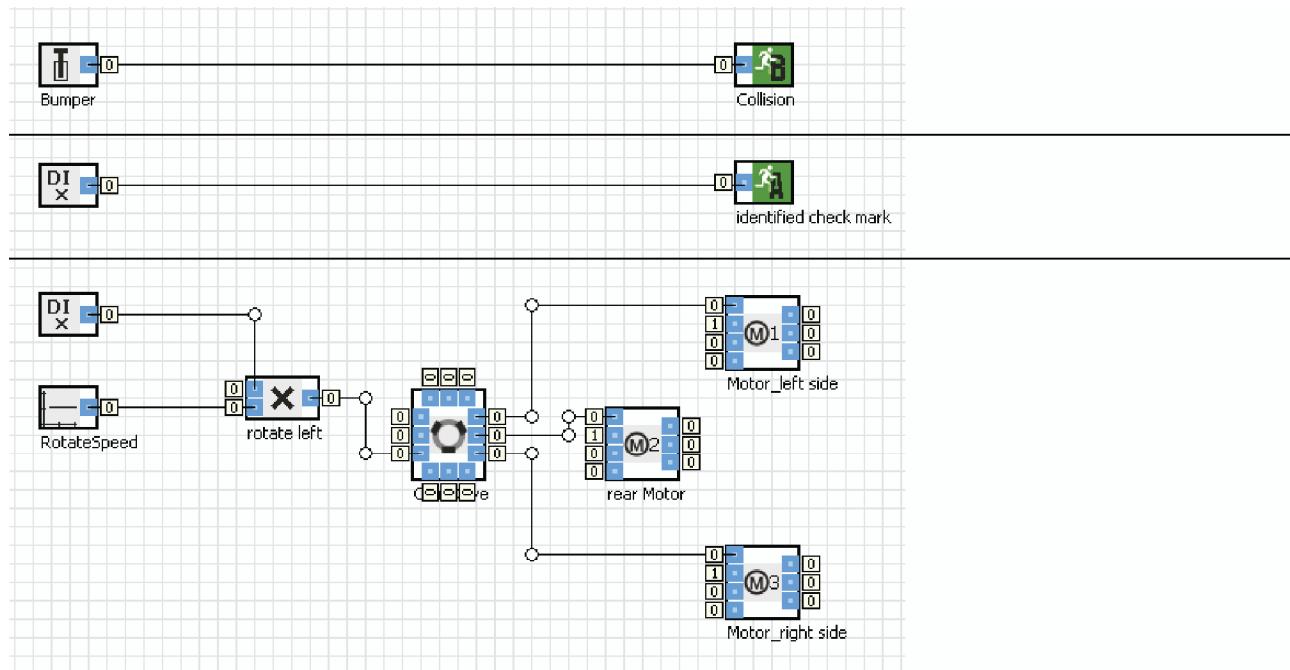
<b>Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors</b>	
Name:	Date:
Development of the control program	Sheet 3 of 7

- Amend the sample program Aufg-P4-02.rvw according to your specification.

Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors

Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors	
Name:	Date:
Development of the control program	Sheet 4 of 7

- Proceed in exactly the same way with the sample program Aufg-P4-03.RVW as with program Aufg-P4-02.RVW. First designate the individual function groups in order to understand the program behaviour.  
Enter this description in the diagram below.



- Describe by what means you identify the direction of rotation:

---



---



---



---



---



---



---

Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors

Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors	
Name:	Date:
Development of the control program	Sheet 5 of 7

- Assign the appropriate input to the input function blocks. (DI0 to DI3).
1. Consider which of the travel situations is applicable if the Robotino® is to execute a rotation in anti-clockwise direction. Which sensor needs to be interrogated in this case?

Note      Use your table with the sensor signals of the individual travel situations.

---

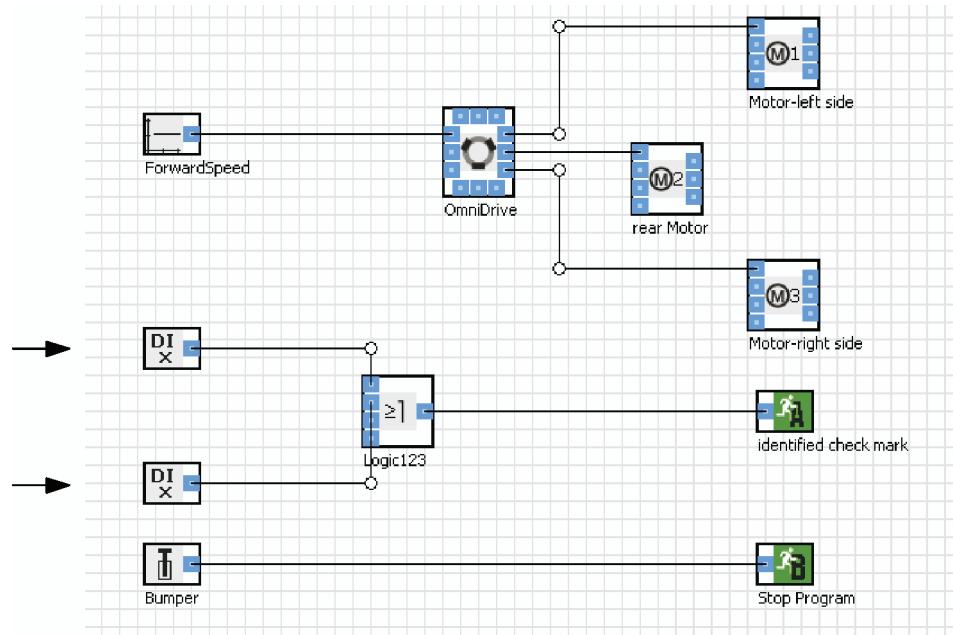


---



---

2. Enter the relevant inputs in the diagram below, taking into account which input behaviour is bright-switching and which dark-switching.



Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors

<b>Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors</b>	
Name:	Date:
Development of the control program	Sheet 6 of 7

- Amend the sample program Aufg-P4-03.rvw according to your specification.

## Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors

<b>Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors</b>	
Name:	Date:
Development of the control program	Sheet 7 of 7

- Now realise a control program for one rotation of the Robotino® in clockwise direction.
- Amend the sample program Aufg-P4-04.rvw accordingly.  
Note also the rotational speed.

Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors

<b>Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors</b>	
Name:	Date:
Testing of the subprogram	Sheet 1 of 1

- Travel the length of the marking by starting the three subprograms according to the situation. Try to travel the path as fast as possible and note your best time.

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors

<b>Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors</b>	
Name:	Date:
Combination of the individual subprograms into a sequence control	Sheet 1 of 1

- To improve the reaction time of the individual subprograms to the respective travel situations, the individual subprograms are to be combined into a sequence control. Establish the relevant information in the theory section and in the Robotino® View help.

## Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors

<b>Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors</b>	
Name:	Date:
Realisation of a closed-loop control program	Sheet 1 of 1

- Integrate all functions into a single function block diagram to convert the sequence control into a closed-loop control program.

General conditions

The closed-loop control program must be terminated only when the loading station is reached or an obstacle touched.

Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors

## **Project 5**

### Accurately positioned approach of a loading station

#### Training aims

#### Trainees

- are familiarised with the behaviour, position and mode of operation of the infrared distance sensors of the Robotino®.
- are able to record the characteristic curve of the Robotino® distance sensors and use these to measure the distance.
- are able to use and optimise the results in a control program for the Robotino®.

#### Problem description

The task is to approach a loading station using an automated guided vehicle system (AGV). The AGV is to stop 8 cm before the station to load a workpiece.

#### Project assignment

Carry out a functional test of the Robotino® distance sensors and amend one of the control programs so that Robotino® stops 8 cm before an obstacle. The Robotino® acts as an automated guided vehicle system in this case.

#### General conditions

The Robotino® is to face the station and approach the loading station from a distance greater than 8 cm.

#### Work assignments

1. Determine the position of all the Robotino® distance sensors and carry out a functional test of the sensors.
2. Record the characteristic curve of sensor 1 and check the manufacturer's data.
3. Linearise this characteristic curve and convert the determined values into millimetres.
4. Amend one of the control programs for the Robotino® so that it will stop 8cm before an obstacle.
5. Carry out a test of this control program.

## Project 5: Accurately positioned approach of a loading station

Working aids	<ul style="list-style-type: none"><li>• Programs</li><li>• Technical documentation</li><li>• Data sheets</li><li>• Theory section: Infrared sensors, recording of a characteristic curve and linearisation of characteristic curve</li><li>• Robotino® View: Configuring distance sensors, adapting measured values</li><li>• Robotino® View help</li></ul>
Possible additional exercises	<p>Competition for several participants or groups: Who positions how quickly and accurately. Consider what needs to be done to detect obstacles located laterally to the line to be travelled.</p>

<b>Project 5: Accurately positioned approach of a loading station</b>	
Name:	Date:
Position detection of distance sensors	Sheet 1 of 4

In order to respond to the individual distance sensors, you need to determine which sensor is mounted at what position of the Robotino®.

- Describe how you proceed to determine the position of the individual distance sensors of the Robotino®.

Procedure:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

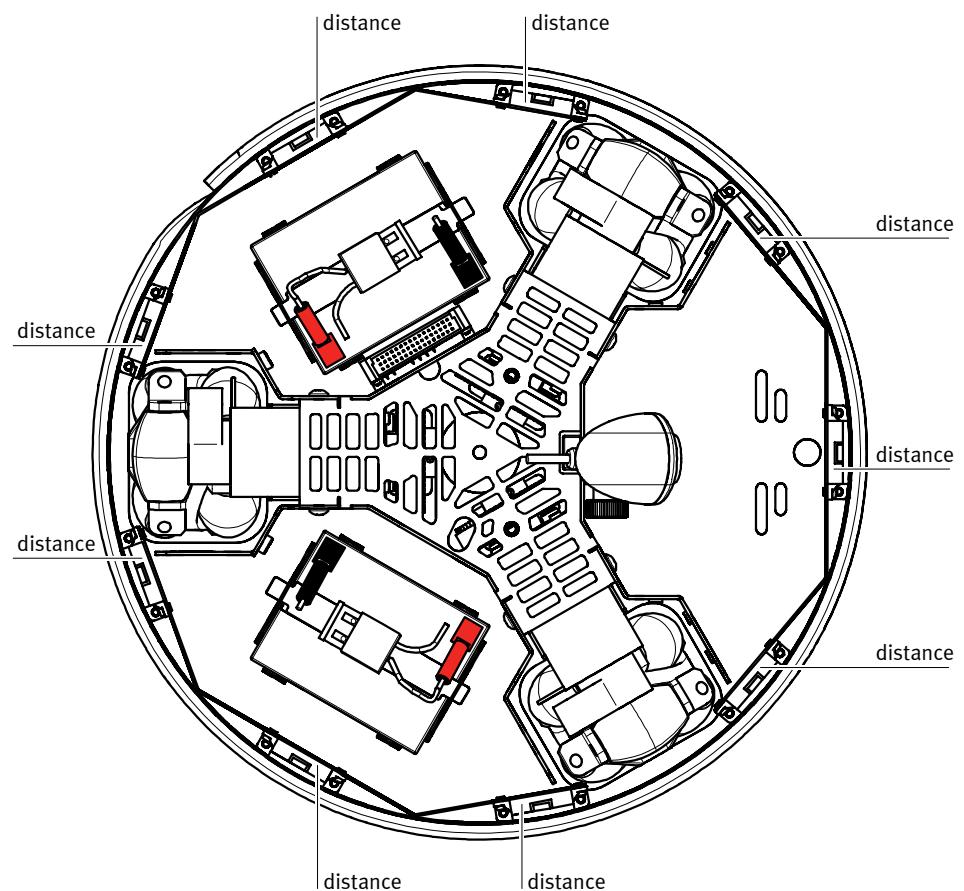
## Project 5: Accurately positioned approach of a loading station

<b>Project 5: Accurately positioned approach of a loading station</b>	
Name:	Date:
Determining the position of the distance sensors	Sheet 2 of 4

- Create a program in Robotino® View to determine the position of the distance sensors "distance 1" to "distance 9" and save the program.

<b>Project 5: Accurately positioned approach of a loading station</b>	
Name:	Date:
Determining the position of the distance sensors	Sheet 3 of 4

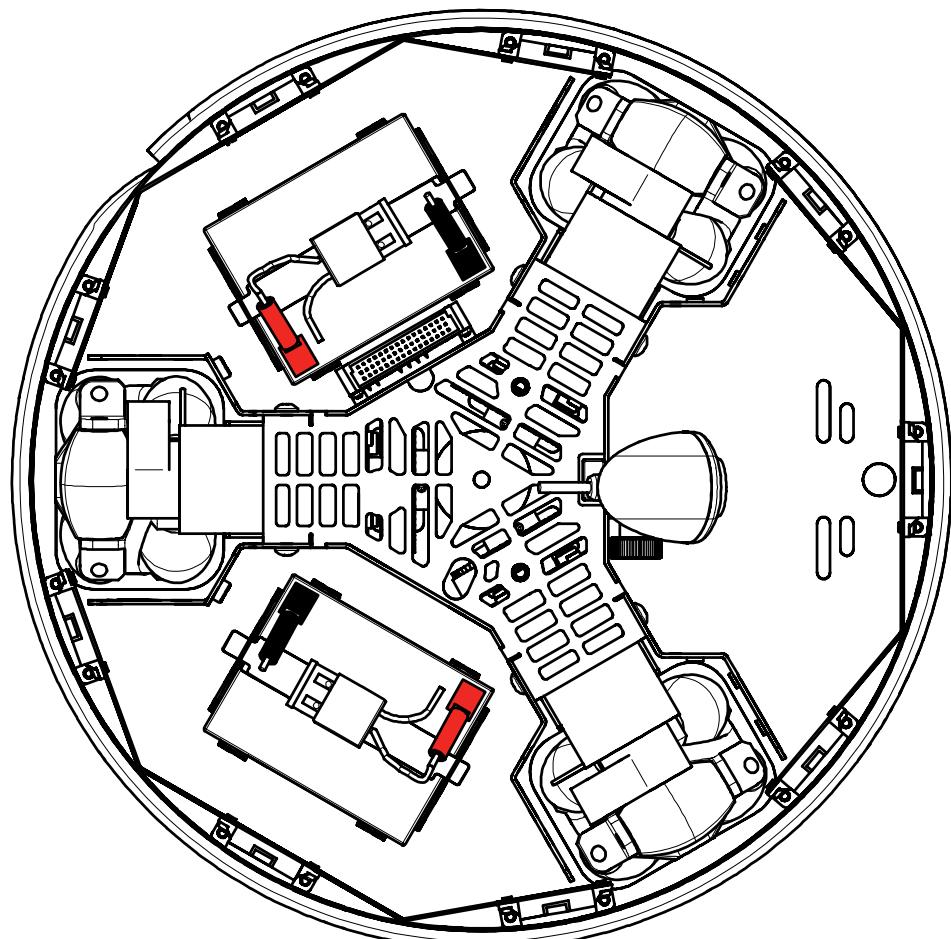
- Start your control program, determine the position of the sensors and enter the number of the respective distance sensor in the diagram below.
- Complete the check list for the functional test.



Sensor	Function O.K.	Sensor	Function O.K.
Distance 1	<input type="checkbox"/>	Distance 6	<input type="checkbox"/>
Distance 2	<input type="checkbox"/>	Distance 7	<input type="checkbox"/>
Distance 3	<input type="checkbox"/>	Distance 8	<input type="checkbox"/>
Distance 4	<input type="checkbox"/>	Distance 9	<input type="checkbox"/>
Distance 5	<input type="checkbox"/>		

<b>Project 5: Accurately positioned approach of a loading station</b>	
Name:	Date:
Determining the position of the distance sensors	Sheet 4 of 4

- The infrared sensors used in the Robotino® emit their light beam vertically. Determine or calculate the beam angle of the individual sensors in relation to the centre of the Robotino®. The beam direction of the "distance 1" sensor is 0°.
- Enter the sensing direction of the individual sensors in the diagram below and enter the respective degree value.



<b>Project 5: Accurately positioned approach of a loading station</b>	
Name:	Date:
Initial considerations regarding distance measurement	Sheet 1 of 1

The Robotino® is to stop at a predetermined distance from the loading station. Consider how this can be achieved.

1. First, determine which distance sensors are required for the distance measurement. Explain your choice.

---

---

---

---

---

---

2. How do you need to proceed to detect the required distance from the loading station in a control program? Take into consideration that possibly different distances may be required. Describe how you proceed.

---

---

---

---

---

---

<b>Project 5: Accurately positioned approach of a loading station</b>	
Name:	Date:
Recording of the characteristic curve	Sheet 1 of 3

- Describe your procedure for the recording of the characteristic curve of the distance sensor.

Procedure:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

Project 5: Accurately positioned approach of a loading station

<b>Project 5: Accurately positioned approach of a loading station</b>	
Name:	Date:
Recording the characteristic curve	Sheet 2 of 3

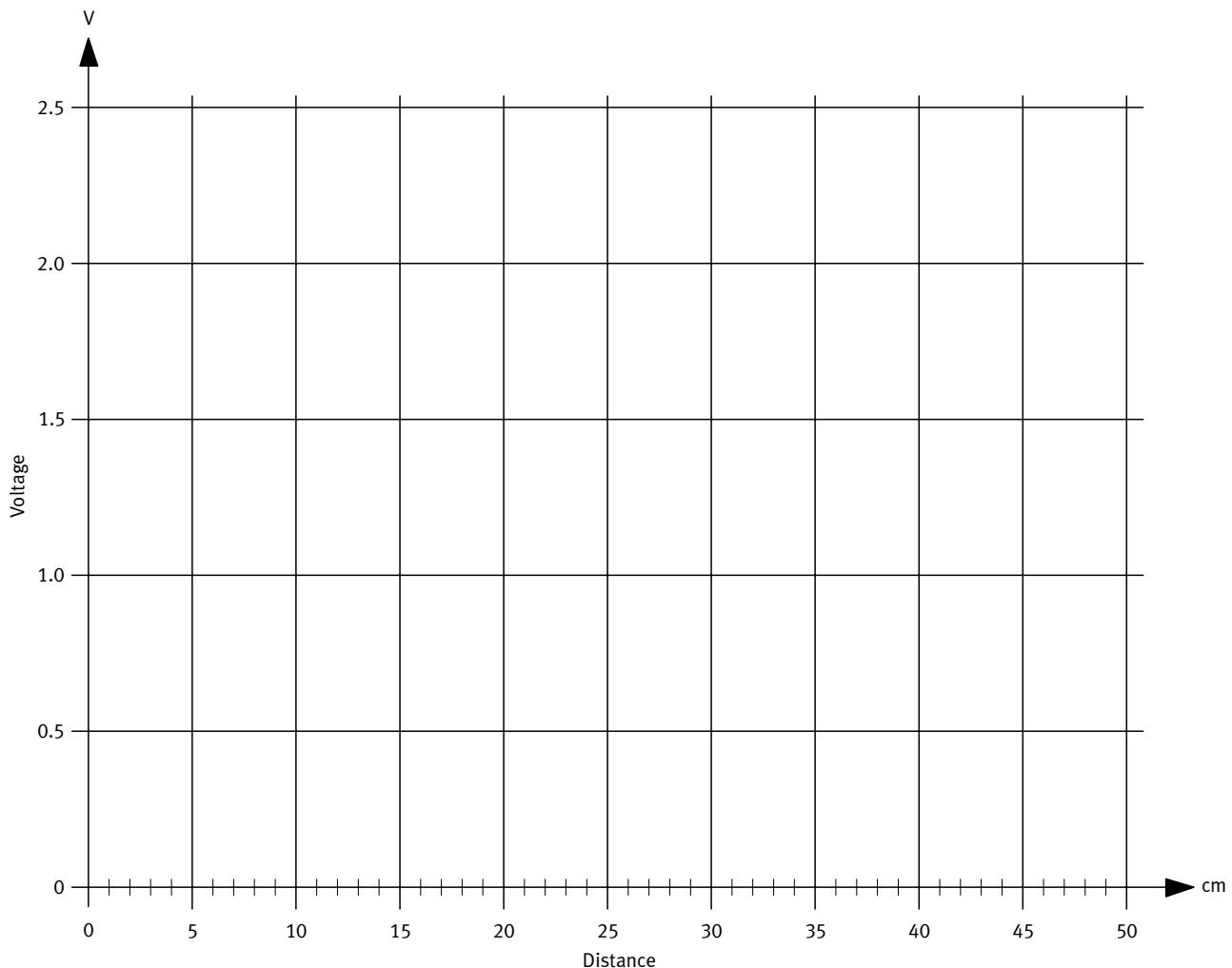
- Record the characteristic curve.

<b>Distance (cm)</b>	<b>Voltage (V)</b>	<b>Distance (cm)</b>	<b>Voltage (V)</b>
1		21	
2		22	
3		23	
4		24	
5		25	
6		26	
7		27	
8		28	
9		29	
10		30	
11		31	
12		32	
13		33	
14		34	
15		35	
16		36	
17		37	
18		38	
19		39	
20		40	

Project 5: Accurately positioned approach of a loading station

<b>Project 5: Accurately positioned approach of a loading station</b>	
Name:	Date:
Recording the characteristic curve	Sheet 3 of 3

- Enter the characteristic curve into the coordinate system.



## Project 5: Accurately positioned approach of a loading station

<b>Project 5: Accurately positioned approach of a loading station</b>	
Name:	Date:
Representation of the characteristic curve using MS Excel	Sheet 1 of 1

- Represent the characteristics in the form of a curve using MS-Excel.

Enter the determined pairs of values in an Excel table and represent these in the form of a diagram.

## Project 5: Accurately positioned approach of a loading station

<b>Project 5: Accurately positioned approach of a loading station</b>	
Name:	Date:
Linearisation of the characteristic curve	Sheet 1 of 1

- Determine the characteristic curve area required for the exercise given.
- Linearise the characteristic curve within this area and enter the linearised characteristic curve in your drawing or represent it in an Excel diagram. The distance should be expressed in cm.
- Document the calculation in individual steps on the worksheet.

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Project 5: Accurately positioned approach of a loading station

<b>Project 5: Accurately positioned approach of a loading station</b>	
Name:	Date:
Adaptation of the control program	Sheet 1 of 1

- Adapt the program Aufg\_P5\_01.rvw so that the Robotino® stops 8 cm in front of an obstacle. Integrate the calculated correlation between the output values and the distance as parameters into the function block dialogue of the distance sensor. Define the desired distance.

Project 5: Accurately positioned approach of a loading station

<b>Project 5: Accurately positioned approach of a loading station</b>	
Name:	Date:
Creating a test plan	Sheet 1 of 1

- Create a test plan for your control program and describe your test procedure.

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Project 5: Accurately positioned approach of a loading station

<b>Project 5: Accurately positioned approach of a loading station</b>	
Name:	Date:
Testing the control program	Sheet 1 of 1

- Test the function of your control program in accordance with your test plan.  
Document your findings.

Note                   The step mode in Robotino® View must be set to fast.  
                         Extras → Options → Step Mode → Fast

<b>Test protocol</b>	
Function in jacked-up state	
Accuracy of distance	
Function in travel mode	
Accuracy of distance in travel mode	
Comments	

Date, signature

---

## Project 5: Accurately positioned approach of a loading station

<b>Project 5: Accurately positioned approach of a loading station</b>	
Name:	Date:
Testing the accuracy of the stop process	Sheet 1 of 1

- Travel towards the obstacle using different speeds and measure the accuracy of the distance kept from the obstacle. Compare the distances with regard to the different speeds.
- Describe the causes for the different distances.

### Note

You can change the travel speed of the Robotino® by assigning a higher value to the "speed (mm/s)" constant.

The step mode in Robotino® View must be set to fast.

Extras → Options → Step Mode → Fast

<b>Speed</b>	<b>Measured distance from the obstacle</b>
20	
50	
100	
200	

## Project 5: Accurately positioned approach of a loading station

<b>Project 5: Accurately positioned approach of a loading station</b>	
Name:	Date:
Project documentation	Sheet 1 of 1

- Compile the project documentation.

<b>Exercises</b>	<b>Required documents</b>
Determining the position of the distance sensors	
Recording the characteristic curve	
Representation of characteristic curve using MS Excel	
Linearisation of the characteristic curve	
Adaptation of the control program	
Creating a test plan	
Test	

Project 5: Accurately positioned approach of a loading station

## Project 6

### Approaching an obstacle and maintaining a defined distance

#### Training aims

#### Trainees

- are able to design a closed-loop control system using a distance sensor of the Robotino® so that the Robotino® approaches an obstacle at a defined distance.
- are familiarised with the consequences of processes overlapping in the case of a forward and lateral travel sequence.
- are able to design sensor-guided path control so that the Robotino® travels along a wall at a defined distance.
- are able to test and explain a sequence program using different travel processes.

#### Problem description

A mobile robot system is to approach different transfer positions on a conveyor. In the first step it is therefore necessary for the system to maintain a defined distance when travelling along the conveyer.

#### Project assignment

In the laboratory environment, the conveyor is replaced by a band. Create and test a program whereby the Robotino® approaches the band (obstacle), and then travels along the band at a distance of 60 mm. The line of vision of the Robotino® is to be directed towards the band during motion.

#### Step 1

Your first step is to ensure that the Robotino® maintains the distance of 60 mm by means of readjustment.

Assignment: First, design and test a program which

- approaches an obstacles, e.g. a square object, up to a distance of 60 mm
- and maintains the 60 mm distance even if the obstacles moves.

#### Step 2

Your second step is to ensure that the Robotino® travels laterally along a band (wall), maintaining the distance of 60 mm with its line of vision facing the band.

#### General conditions

- Robotino® View is installed on the PC and the software is started
- A W-LAN connection is established with the Robotino® (technical documentation)
- An obstacle is positioned within the working area
- The Robotino® is positioned near the obstacle and orientated such that the camera points in the direction of the obstacle (step 1)
- The Robotino® is positioned near a wall and orientated such that the camera points in the direction of the wall (step 2)

## Project 6: Approaching an obstacle and maintaining a defined distance

### Work assignments

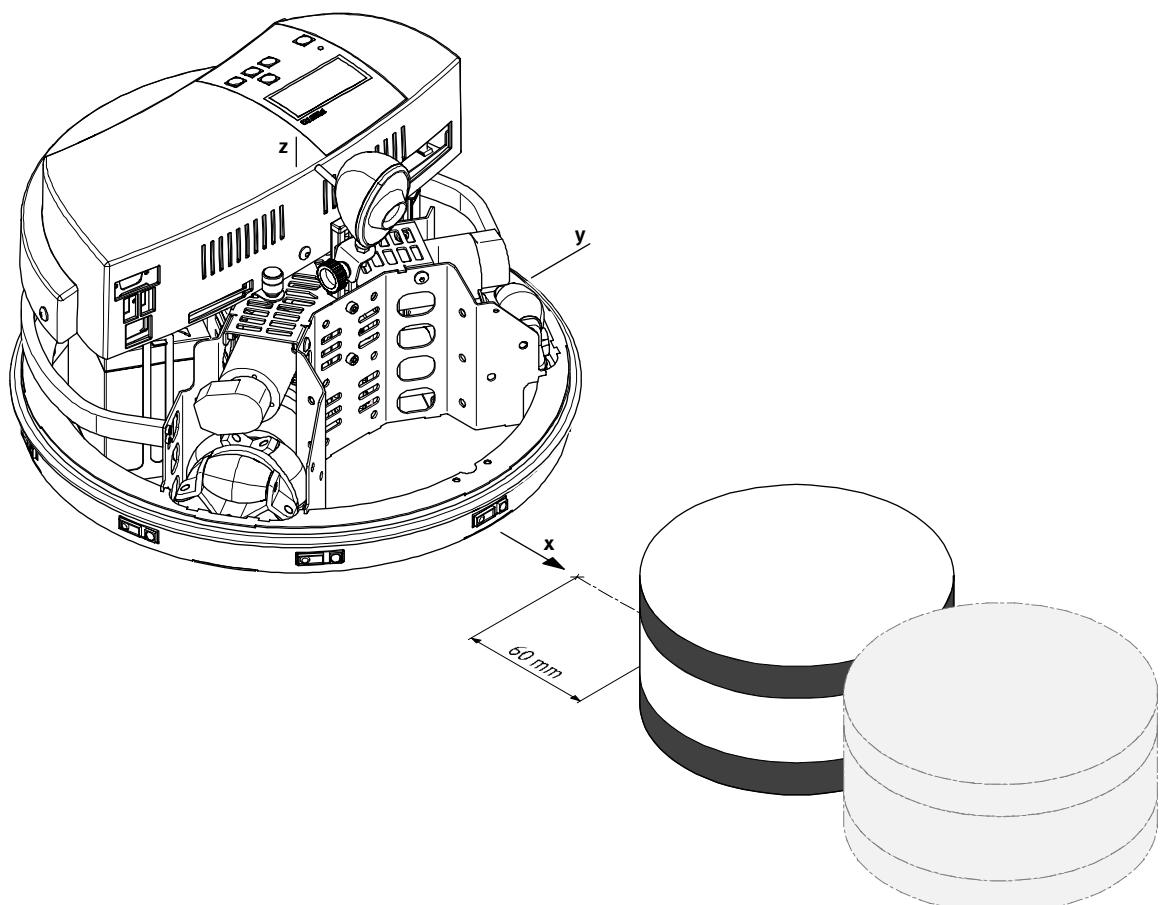
#### Step 1

1. Create and test a program in Robotino® View whereby the Robotino® stops at a distance of 60 mm in front of an obstacle.
2. The Robotino® is to automatically adjust the distance to the setpoint value of 60 mm if the obstacle is moved.
3. Move the obstacle whilst running the program and observe what happens.  
Explain the possibilities of optimising the program.
4. Explain the difference between the **distance sensor** program from project 5.

#### Step 2

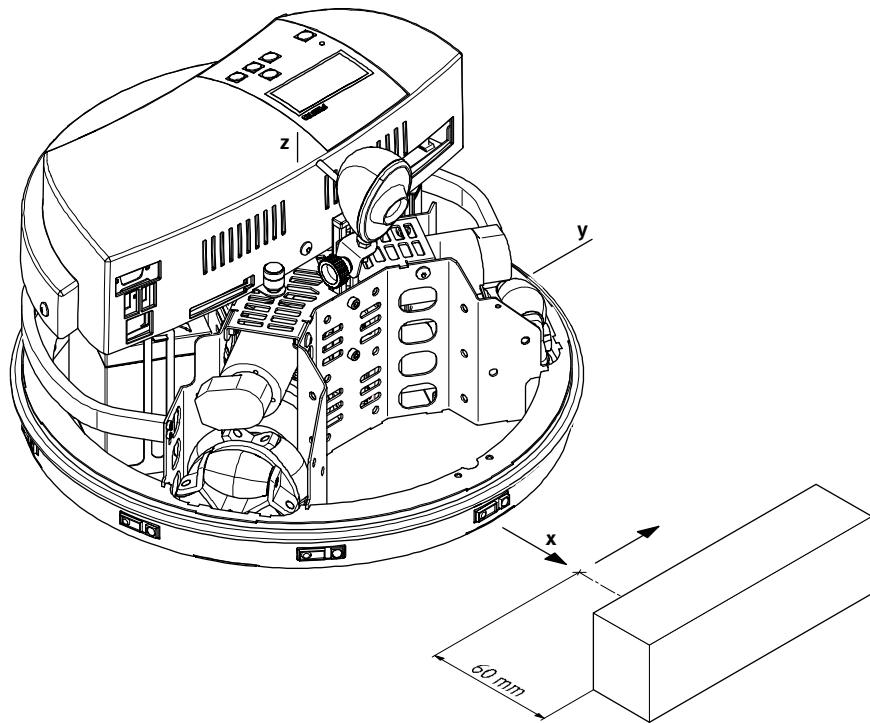
1. Create a program whereby the Robotino® travels sensor-guided along a wall at a distance of 60 mm, with its line of vision facing towards the wall.
2. Test the program using the following cases:
  - The start position of the Robotino® is 60 mm away from the wall
  - The start position of the Robotino® is more than 100 mm in front of the wall
3. Create a sequence program so that the Robotino® approaches the wall up to 60 mm facing it and then along it at a constant distance of 60 mm.
4. Test and explain the sequence program.

### Positional sketch of step 1



Project 6: Approaching an obstacle and maintaining a defined distance

Positional sketch of step 2



Working aids

Technical documentation,  
Robotino® View Help: Multiplication module  
Theory section: Omnidirectional drive, closed-loop control technology

## Project 6: Approaching an obstacle and maintaining a defined distance

<b>Project 6: Approaching an obstacle and maintaining a defined distance</b>	
Name:	Date:
Approaching and readjusting to a distance of 60 mm	Sheet 1 of 4

- Create and test a program in Robotino® View whereby the Robotino® stops in front of an obstacle at a distance of 60 mm.
- The Robotino® is to automatically adjust the distance to the setpoint value of 60 mm if the obstacle is moved.
- Move the obstacle whilst running the program and observe what happens. Explain the possibilities of optimising the program.
- Explain the difference between the distance sensor program from project 5.
  
- Create and test a program in Robotino® View whereby the Robotino® stops 60 mm in front of an obstacle.
  
- Create the program from project 5 (distance sensor) under another name
- Determine the direct sensor value from the distance sensor diagram (project 5), and enter this as a minimum distance.

### Direct sensor value

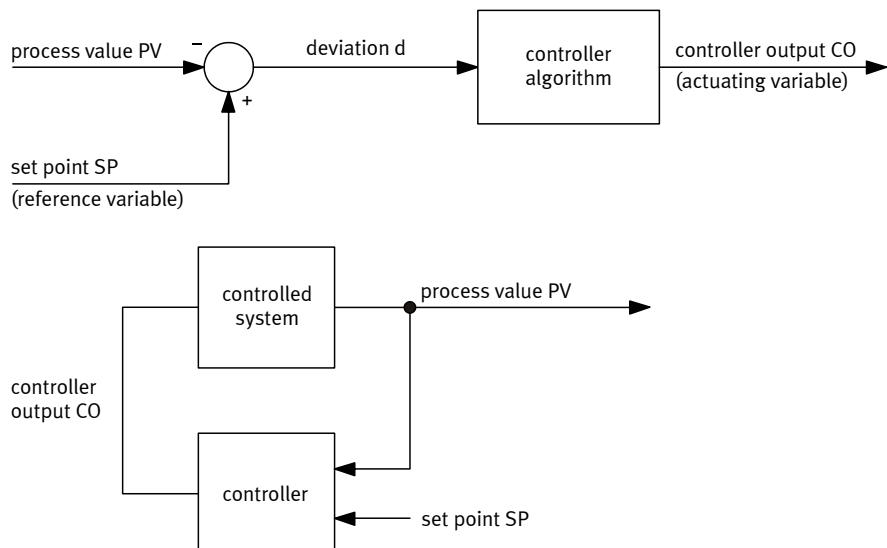
#### Note

Make sure that the correct value is entered in the function block diagram of the distance sensor.

## Project 6: Approaching an obstacle and maintaining a defined distance

<b>Project 6: Approaching an obstacle and maintaining a defined distance</b>	
Name:	Date:
Approaching and readjusting to a distance of 60 mm	Sheet 2 of 4

- If the obstacle is moved, the Robotino® is to automatically readjust the distance to the setpoint value of 60 mm.
- Answer the following questions using the block diagram:



Questions	Answer
What is the controlled variable?	
How is the actual value of the controlled variable measured?	
What is the setpoint value of the controlled variable?	
What is the controlled system?	
What is the disturbance variable?	
Determine the system deviation. Use the functionality of a P-controller for the solution.	
Determine the manipulated variable.	

## Project 6: Approaching an obstacle and maintaining a defined distance

<b>Project 6: Approaching an obstacle and maintaining a defined distance</b>	
Name:	Date:
Approaching and readjusting to a distance of 60 mm	Sheet 3 of 4

- Create and test the program

### Note

Reference variable = 2 V / velocity factor = 75

The manipulated variable is calculated from the system deviation via multiplication by a velocity factor. For example, if the Robotino® is positioned 10 cm in front of an obstacle, the Robotino® approaches at a speed of 90 [mm/s]. However, if the Robotino® is only 7 cm from the obstacle, it only travels forward at a speed of 30 [mm/s].

## Project 6: Approaching an obstacle and maintaining a defined distance

<b>Project 6: Approaching an obstacle and maintaining a defined distance</b>	
Name:	Date:
Approaching and readjusting to a distance of 60 mm	Sheet 4 of 4

- Move the obstacle whilst running the program and observe what happens. How can you improve the closed-loop control?

**Change if the obstacle is moved, if the closed-loop control is improved**

- Explain the difference between the distance sensor program from project 5.

**Difference compared to the distance sensor program in project 5**

## Project 6: Approaching an obstacle and maintaining a defined distance

<b>Project 6: Approaching an obstacle and maintaining a defined distance</b>	
Name:	Date:
Travelling along a wall	Sheet 1 of 2

- Create a program whereby the Robotino® travels sensor-guided along a wall at a distance of 60 mm with its line of vision facing towards the wall.
  - Test the program using the following cases:
    - The Robotino®'s start position is 60 mm away from the wall
    - The Robotino®'s start position is more than 100 mm away from the wall
  - Create a sequence program so that the Robotino® approaches the wall facing it up to a distance of 60 mm and then along it at constant distance of 60 mm.
  - Test and explain the sequence program
- 
- Add lateral travel to the existing program and describe what happens.
- 
- Test the program using the following cases:
    - The Robotino® start position is 60 mm away from the wall
    - The Robotino® start position is more than 100 mm away from the wall

### Note

Make sure that the line of vision of the Robotino® is towards the wall when starting.  
Sample value for lateral travel = 100 [mm/s].

#### What happens? Case 1

#### What happens? Case 2

## Project 6: Approaching an obstacle and maintaining a defined distance

<b>Project 6: Approaching an obstacle and maintaining a defined distance</b>	
Name:	Date:
Travelling along a wall	Sheet 2 of 2

- Create a sequence program so that the Robotino® approaches the wall facing it up to a distance of 60 mm and then along it at a constant distance of 60 mm.
- Open the programs **Distance.rvm** and **WallTravel.rvm** and open a new sequence program.
- First start the Distance program, followed by the WallTravel program.

### Explanation of the program

Project 6: Approaching an obstacle and maintaining a defined distance

## Project 7

### Circling a station and approaching various transfer positions

#### Training aims

#### Trainees

- are familiarised with the degrees of freedom of a driven multi-axis system.
- are able to use distance sensors for closed-loop controlled path control.
- are able to execute the programming of sensor-guided path control of a mobile robot along a circular band using Robotino® View.
- are able to identify the main parameters of closed-loop path control and change these to optimise the solution.

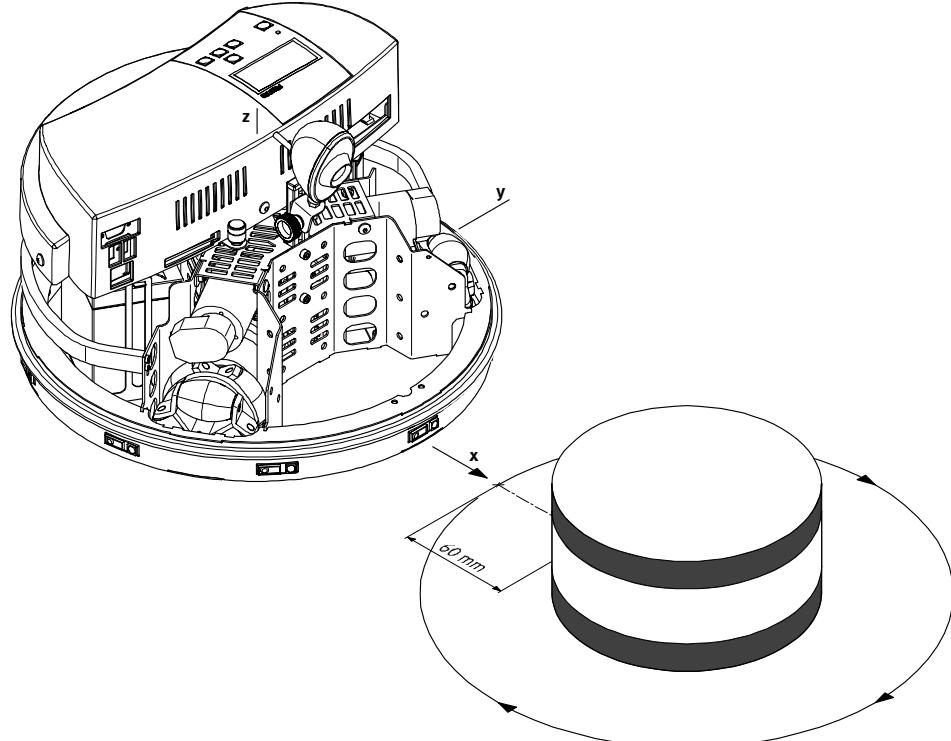
#### Problem description

A mobile robot system is to operate a rotary indexing table with several transfer positions. The system is to approach the various transfer positions as efficiently as possible. This requires the system to circle the rotary indexing table whilst maintaining a specific distance.

#### Project assignment

Create and test a program whereby, facing the obstacle, the Robotino® circles this at a distance of 60 mm. To do so, use the findings from project 6 “Approaching an obstacle and maintaining a defined distance”.

#### Positional sketch



General conditions	<ul style="list-style-type: none"><li>• Robotino® View is installed on the PC and the software started</li><li>• A WLAN is established to the Robotino® (technical documentation)</li><li>• A cylindrical object is positioned in the working area (diameter approx. 250 mm, height approx. 150 mm)</li><li>• Robotino® is positioned near the object and orientated so that the camera points in the direction of the object.</li></ul>
Work assignments	<ul style="list-style-type: none"><li>• Consider:<ul style="list-style-type: none"><li>– how many degrees of freedom are required to approach and circle the obstacle. Use the positional sketch for this</li><li>– how, using the omnidrive function block, you can generate a circular path and create the appropriate program.</li><li>– how, and by using which distance sensors, you can monitor the distance to the obstacle whilst circling it. Use the positional sketch for this.</li></ul></li></ul>
Initial considerations	<ul style="list-style-type: none"><li>• What should the value be of the two distance sensors 2 and 9 in order for the camera to be aligned with the obstacle? How can you use the sensor values for closed-loop control of the orientation towards the obstacle?</li></ul>
Program	<ul style="list-style-type: none"><li>• Position the Robotino® at a distance of 6 cm from the obstacle so that the camera faces in the direction of the obstacle. Complete the program <b>WallTravel</b> from project 6 for circular motion, which maintains the orientation of the camera vertical to the obstacle throughout.</li><li>• Test and optimise your program.</li><li>• Create a sequence program so that the Robotino® first approaches the station up to a distance of 6 cm and then circles it at a distance of 6 cm. Test your program.</li></ul>
Working aids	<p>Technical documentation Help software of Robotino® View: Multiplications, additions and subtraction module Theory section: Omnidirectional drive, degrees of freedom, closed-loop control technology</p>

## Project 7: Circling a stationn and approaching various transfer positions

<b>Project 7: Circling a stationn and approaching various transfer positions</b>	
Name:	Date:
Initial considerations	Sheet 1 of 3

- Consider:
  - how many degrees of freedom are required to approach and circle the obstacle. Use the positional sketch for this
  - how, using the omnidrive function block, you can generate a circular path and create the appropriate program.
  - how and, by using which distance sensors, can you monitor the distance to the obstacle whilst circling it. Use the positional sketch for this.
- What should the value be of the two distance sensors 2 and 9 in order for the camera to be aligned with the obstacle? How can you use the sensor values for closed-loop control of the orientation towards the obstacle?
- Consider how many degrees of freedom are required to approach and circle the obstacle (circling of the station).

### Note

Use the positional sketch from the problem definition and the program from project 5.

**How many degrees of freedom are required to circle a station?**

## Project 7: Circling a stationn and approaching various transfer positions

<b>Project 7: Circling a station and approaching various transfer positions</b>	
Name:	Date:
Initial considerations	Sheet 2 of 3

- Consider how, using the omnidrive function block, you can generate a circular path and create an appropriate program.

### **What happens and why?**

- Consider how, and with which distance sensors, you can monitor the distance to the obstacle whilst circling it. Use the positional sketch for this.

### **Monitoring of the distance – which sensors?**

## Project 7: Circling a stationn and approaching various transfer positions

<b>Project 7: Circling a station and approaching various transfer positions</b>	
Name:	Date:
Initial considerations	Sheet 3 of 3

- What should the value be of the two distance sensors 2 and 9 in order for the camera to be aligned with the obstacle? How can you use the sensor values for closed-loop control of the orientation towards the obstacle?

**Values of distance sensors 2 and 9 – use of sensor values for closed-loop control of the orientation towards the obstacle**

<b>Project 7: Circling a station and approaching various transfer positions</b>	
Name:	Date:
Program	Sheet 1 of 1

- Position the Robotino® at a distance of 6 cm from the obstacle so that the camera faces in the direction of the obstacle. Add a rotational movement to the program **WallTravel** from project 6, which maintains the orientation of the camera vertical to the obstacle throughout.
- Test and optimise your program.
- Create a sequence program so that the Robotino® approaches the station up to a distance of 6 cm and then circles it at a distance of 6 cm. Test your program.

Note

The solution consists of a path control using two P-controllers, which on the one hand control the distance to the obstacle and on the other hand the alignment with the obstacle.

Using the three constants      velocity factor = 75  
    lateral = 100  
    rotational factor = 30

you can significantly influence the motion behaviour.

- Create a sequence program so that the Robotino® fist approaches the station up to a distance of 6 cm and then circles it at a distance of 6 cm. Test your program.
- Select the program **Distance** from project 5 and combine it with the above program.

## **Project 8**

### Path tracking of an automated guided vehicle system using an analogue inductive sensor

#### Training aims

##### Trainees

- are able to mount the inductive analogue sensor on the Robotino® and commission it.
- are familiarised with the mode of operation and signal behaviour of an inductive sensor.
- are able to analyse the behaviour of a control program and adapt the program accordingly.
- are able to combine several function block programs into a sequence control program.
- are able to convert a sequence program into a closed-loop control program.

#### Problem description

The task is to travel an automated guided vehicle system towards a loading station moving along a guide line consisting of a metal tape embedded in the surface. When it reaches the loading station, the loading process is triggered upon impact with a switch.

The Robotino® represents the automated guided vehicle system in this case. The guide line is represented by an aluminium adhesive tape.

#### Project assignment

Mount an analogue inductive sensor on the Robotino® and carry out a functional test. Modify an existing sequence program and develop it into a closed-loop control program with the appropriate functions.

#### General conditions

The guide line must be 50 mm wide and lead to an obstacle in a straight line.

#### Possible additional exercise

The automated guide vehicle is to travel up to the loading station and actuate a switch which starts the loading process. It then is to move into the loading position for loading and remain there for 20 seconds until the loading process is completed, turn around its own axis by 180°, and return to the starting point of the guide line in order to deliver the load. The distance of the AGV to the loading station should be 6 cm. You can use existing programs from previous projects for this.

Project assignment

1. Mount the inductive analogue sensor, connect it to the I/O interface of the Robotino® and test the sensor function.
2. Analyse the signal behaviour of the sensor.
3. Record the value pattern of the sensor when travelling along the metal strip and determine the relevant range for travel.
4. Develop a strategy for the task required.
5. Analyse the motion behaviour of the Robotino® when executing a function block diagram for linear tracking.
6. Improve an existing sequence program and convert this into a closed-loop control program.

Working aids

- Sample programs  
Technical documentation  
Data sheet  
Theory section: Inductive distance sensors. IF function, sequence control, sign reversal, characteristic curve  
Robotino®View: Addressing analogue input  
Robotino®View help

Project 8: Path tracking of an automated guided vehicle system using an analogue inductive sensor

<b>Project 8:</b> <b>Path tracking of an automated guided vehicle system using an analogue inductive sensor</b>	
Name	Date:
Preparatory work	Sheet 1 of 3

- To make it easier for you to carry out your task, you should first of all prepare a work plan for the assembly and commissioning of the sensor. Draw up a list of tools and materials prior to starting the assembly.
- Subsequently use your work plan as a check list for project documentation.

Work plan	Work step	Time required	Completed

Total time required: \_\_\_\_\_

Materials & tools required	Quantity	Tool / Material

<b>Project 8: Path tracking of an automated guided vehicle system using an analogue inductive sensors</b>	
Name:	Date:
Preparatory work	Sheet 2 of 3

- Determine the necessary sensor connections in accordance with the Robotino® manual, operating instructions and data sheets of the sensor.

---

---

---

---

---

---

- Determine the colours of the required cables and their designation in the operating instructions. Explain your choice.

Explanation:

---

---

---

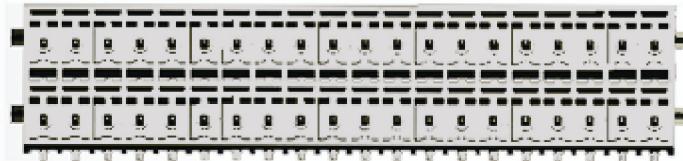
---

---

---

<b>Project 8:</b> <b>Path tracking of an automated guided vehicle system using an analogue inductive sensor</b>	
Name:	Date:
Preparatory work	Sheet 3 of 3

- The sensor is to be connected to the analogue input AI0. Create a connection diagram for the I/O interface.
- Enter the cables on the drawing shown below and indicate their characteristics and colour.



<b>Project 8: Path tracking of an automated guided vehicle system using an analogue inductive sensor</b>	
Name:	Date:
Mounting of sensor and connection to the I/O interface	Sheet 1 of 2

- Attach the sensor in accordance with the Robotino® manual. What should be noted here?

---

---

---

---

---

---

- Connect the cables shown in your drawing to the contact strip accordingly.

Notes

Attach the cables to the contact strip provided as follows: Plug the cables into the appropriate socket and press them in. The cables are thus clamped. Check that the cables are securely attached. You can release the cables again by pressing in the orange coloured locking mechanism of the respective socket with a screwdriver and pulling out the cable.

Run and attach the sensor cable such that it cannot be pulled off or become entangled with the drive unit during travel.

To exchange the batteries, the command bridge needs to be removed. You should therefore allow sufficient play for the cable to be able to remove the contact strip or command bridge without having to release the cable connection.

Make sure that the camera lens is not obscured by the cable.

<b>Project 8: Path tracking of an automated guided vehicle system using an analogue inductive sensor</b>	
Name:	Date:
Mounting of sensor and connection to the I/O interface	Sheet 2 of 2

- Carry out a functional test of the sensor. To do so, establish access to the sensor data in Robotino®View and represent the input values using the oscilloscope. Document how you proceed.

Procedure

---

---

---

- Carry out a functional test of the sensor. Describe how you proceed.

---

---

---

- Save the function block diagram.

Project 8: Path tracking of an automated guided vehicle system using an analogue inductive sensor

<b>Project 8:</b> <b>Path tracking of an automated guided vehicle system using an analogue inductive sensor</b>	
Name:	Date:
Experiment: Sensing behaviour of the sensor	Sheet 1 of 3

- Check the reaction behaviour of the inductive sensor in respect of different materials (alloys) and determine the correlation between distance and surface of the object and output voltage.  
Use Euro coins of 1 Cent to 2 Euros as test objects.  
Place one of the coins on the surface within the sensing range of the sensor and enter the output voltage measured in the table below. In order to determine the values for different distances, you should place a non-metallic base (paper, cardboard or plastic material) underneath the coin used.

Value	Diameter (mm)	Thickness (mm)	Material	Magnetic	Output voltage (V)	Output voltage (V) with base
1 Cent	16.25	1.67	Steel with copper coating (Fe, Cu)	Yes		
2 Cent	18.75	1.67	Steel with copper coating	Yes		
5 Cent	21.25	1.67	Steel with copper coating	Yes		
10 Cent	19.75	1.93	Nordic gold (Cu89 Al5 Zn5 Sn1)	No		
20 Cent	22.25	2.14	Nordic gold (Cu89 Al5 Zn5 Sn1)	No		
50 Cent	24.25	2.38	Nordic gold (Cu89 Al5 Zn5 Sn1)	No		
1 Euro	23.25	2.33	External: Brass-Ni (Cu75 Zn20 Ni5) Internal: Cu-Ni, Ni, Cu-Ni coated	Weak		
2 Euro	25.75	2.20	External: Cu-Ni (Cu75 Ni25), Internal: Brass-Ni, Ni, Brass-Ni coated	Weak		

<b>Project 8: Path tracking of an automated guided vehicle system using an analogue inductive sensor</b>	
Name:	Date:
Experiment: Sensing behaviour of the sensor	Sheet 2 of 3

- Analyse the values determined in respect of alloy, surface and distance to the sensor and make a note of these.
- The following can be deduced from the values determined:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

Project 8: Path tracking of an automated guided vehicle system using an analogue inductive sensor

<b>Project 8:</b> <b>Path tracking of an automated guided vehicle systems using an analogue inductive sensor</b>	
Name:	Date:
Experiment: Sensing behaviour of the sensor	Sheet 3 of 3

- What conclusions can be drawn from the two series of measurements in the table below with regard to the sensor position if the objects of both series of measurements were placed on the same base?

Value	Diameter (mm)	Thickness (mm)	Material	Magnetic	Output voltage (V) Measurement 1	Output voltage (V) Measurement 2
1 Cent	16.25	1.67	Steel with copper coating (Fe, Cu)	Yes	8.88	5.08
2 Cent	18.75	1.67	Steel with copper coating	Yes	7.92	5.16
5 Cent	21.25	1.67	Steel with copper coating	Yes	6.40	5.36

---



---



---



---



---



---



---



---

<b>Project 8:</b> <b>Path tracking of an automated guided vehicle system using an analogue inductive sensor</b>	
Name:	Date:
Recording the characteristic curve	Sheet 1 of 2

- Determine the parameters for tracking the guide line in order to develop a strategy. Record the characteristic curve of the sensor when travelling along the aluminium strip.  
For the object to be measured, use a 5 cm wide aluminium strip, which is glued to a level surface. This aluminium strip should correspond in width and material to the one used subsequently as guide line for the Robotino®.
- 
- 
- 

Notes

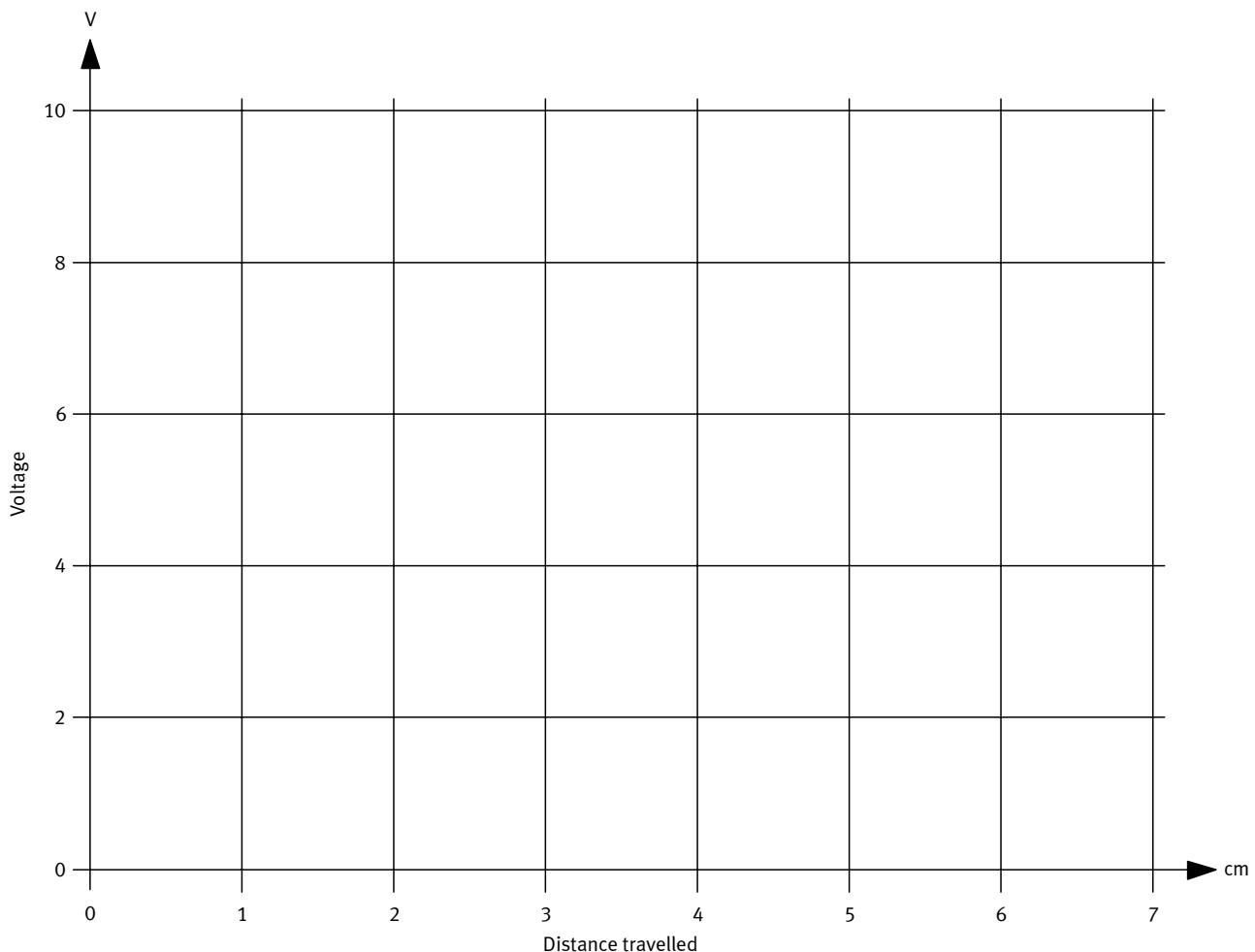
Use the function block diagram 'Aufg-P8-02.rvw' provided for this. Adapt the program to your requirements accordingly.  
Document your changes in the table below and save the program together with your changes.

Function block	Value

Project 8: Path tracking of an automated guided vehicle system using an analogue inductive sensor

<b>Project 8: Path tracking of an automated guided vehicle system using an analogue inductive sensor</b>	
Name:	Date:
Recording the characteristic curve	Sheet 2 of 2

- Record the characteristic curve and enter the values in the diagram below.



Note

The voltage values determined may vary depending on the installation height above the base. The curve pattern is, however, always similar.

<b>Project 8: Path tracking of an automated guided vehicle system using an analogue inductive sensor</b>	
Name:	Date:
Developing a control strategy	Sheet 1 of 1

- Examine the value pattern determined and develop possible strategies to track the line. Make a note of your strategies. Select a strategy and explain your choice.

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

<b>Project 8: Path tracking of an automated guided vehicle system using an analogue inductive sensor</b>	
Name:	Date:
Analysis of the enclosed control program	Sheet 1 of 1

- Analyse the enclosed sequence program 'Aufg-P8-04.rvw'. Place the Robotino® on the aluminium strip and execute the sequence program. Describe the Robotino® behaviour.

Note

The limit values of the sensor signal used in the sample programs may vary depending on the installation height of the sensor and the type of aluminium strip in your configuration. If necessary, adapt the subprograms to these conditions.

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

<b>Project 8: Path tracking of an automated guided vehicle system using an analogue inductive sensor</b>	
Name:	Date:
Optimisation of the control program	Sheet 1 of 1

- You are to optimise the sequence program so that a smoother overall sequence is achieved and for the defined rotation used in the sample program to be replaced by a rotation dependent on the sensor values.
  - Determine the subprograms to be modified and record your solution; then amend the respective subprograms.
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
-

<b>Project 8: Path tracking of an automated guided vehicle system using an analogue inductive sensor</b>	
Name:	Date:
Programming of a closed-loop control program	Sheet 1 of 1

The sequence program optimised by you is to be converted into a closed-loop control program.

- Make a note of the advantages of a closed-loop control program and realise the required closed-loop control program.

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Project 9

### Determining the optimal motion behaviour

#### Training aims

##### Trainees

- are familiarised with the fundamentals of closed-loop control technology for electric motors
- are familiarised with the functionality of remote control and are able to use the basic functions of graphic programming software
- are able to set and optimise the parameters of a PID controller with the help of software and describe and analyse the effects on motion behaviour

#### Problem description

A mobile robot system must exhibit optimal motion behaviour in any situation. This is why, by means of setting the PID controller, you need to determine the range within which the motor controller enables acceptable motion behaviour. In order to ensure acceptable motion behaviour the wheels must move smoothly at the specified speed. This is achieved by a correct PID controller setting for the motor speed.

#### Project assignment

- With the help of Robotino® View set the motor controller so as to ensure optimal motion behaviour within a range.
- Document the range within which the motion behaviour is acceptable.
- Examine what influence the PID controller parameters have on the motion behaviour and the controlled system if you set parameter values which are not within the permissible range.

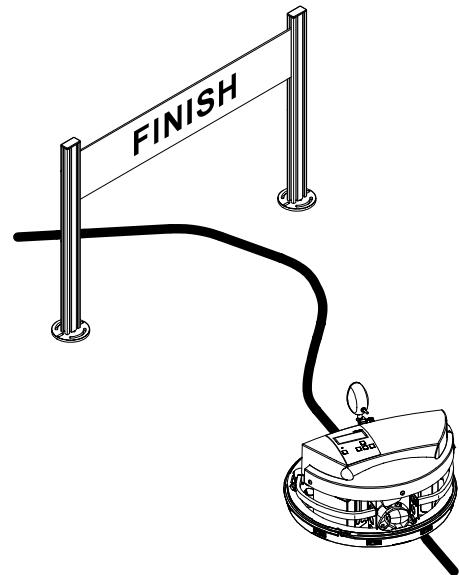
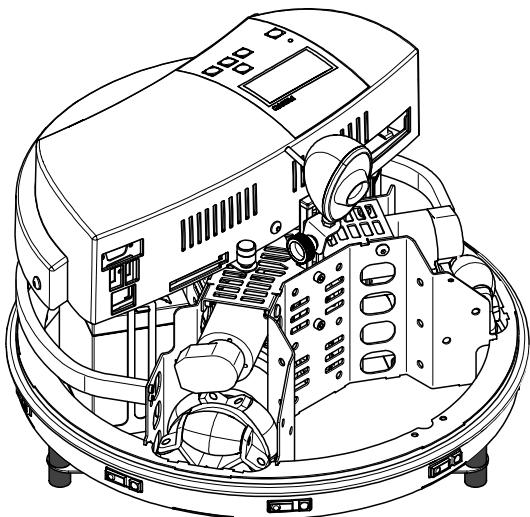
#### General conditions

- Robotino® View is started
- A WLAN connection to the Robotino® is established (technical documentation)

Work assignments

1. Create a program for motor actuation in Robotino® View and visualise and observe the setpoint/actual behaviour of the speed
2. Set the PID controller parameters so that the setpoint and actual signal behaviour optimally match and describe the effects on the motion behaviour. Explain and document your findings.
3. Describe the effects on the movement of the multidirectional casters, if major fluctuations occur as a result of deviations in setpoint/actual behaviour. Explain the effects of major deviations on the motion behaviour described.
4. Answer the questions regarding the functions of the PID controller parameters.
5. Create a program for the actuation of three motors in Robotino® View (with one motor only is also possible).
6. Visualise and observe the setpoint/actual behaviour of a motor in jacked-up state and during motion using the same controller setting.
7. Explain the deviations in the curve progression.

Positional sketch



Working aids

- Technical documentation
- Theory section: Closed-loop control technology sizing by means of testing, oscilloscope and generator
- Robotino® View: Help - Getting Started

<b>Project 9: Determining the optimal motion behaviour</b>	
Name:	Date:
Motor actuation program	Sheet 1 of 1

- Create a program for the motor actuation in Robotino® View.
  - Visualise and observe the setpoint/actual behaviour.
- 
- Jack up the Robotino®.
  - Start Robotino® View and establish a connection between the Robotino® control and Robotino® View.
  - Open a blank function block diagram in Robotino® View.
  - Create a function block diagram from the function blocks „motor“, „square-wave generator“ and “oscilloscope”.
  - Connect the elements so that the setpoint and actual signal is displayed.
  - Set the oscilloscope.

Note

If you want to compare signals it is useful to set the V positions and the scales to the same value on the oscilloscope.

- Set the amplitude of the square-wave generator to 100.
- Start the program by clicking onto the start symbol.



- Observe the behaviour of the multidirectional caster and describe and explain it (square-wave generator).

**Description: Behaviour of the multidirectional caster**

**Explanation: Behaviour of multidirectional caster**

<b>Project 9: Determining the optimal motion behaviour</b>	
Name:	Date:
Setting the PID controller	Sheet 1 of 1

- Set the PID controller parameter so that the setpoint and actual signal behaviour match optimally and describe the effects on the motion behaviour. Explain and document your findings.
- Set the PID controller parameters on the motor. Change kp, ki, kd in succession.
- Observe the setpoint/actual behaviour displayed by the virtual oscilloscope.
- Optimise the transient response so that the setpoint and actual curves virtually coincide.
- Determine a range of values for the control parameters within which the motion behaviour is acceptable.
- Document the parameter values determined.

<b>Range of values of PID controller parameters</b>	
kp	
ki	
kd	

<b>Project 9: Determining the optimal motion behaviour</b>	
Name:	Date:
Oscillations	Sheet 1 of 1

- Describe the effect on the movement of the multidirectional casters if high oscillations occur as a result of deviations in the setpoint/actual behaviour.
- Explain the effects of high oscillations on the motion behaviour.
  
- Change the parameters  $k_i$  and  $k_d$  so that high overshoot/undershoot occur within short periods.
- Observe the setpoint/actual behaviour and the movement of the multidirectional casters.

**Description: How does the multidirectional caster move?**

**Why do high oscillations influence the motion behaviour?**

<b>Project 9: Determining the optimal motion behaviour</b>	
Name:	Date:
Functions of the PID controller parameters	Sheet 1 of 1

- Answer the questions regarding the functions of the PID controller parameters.
- Refer to the theory section to find out what the parameters p, i, and d mean and cause.

**Meaning of p-action and its effect**

**Meaning of i-action and its effect**

**Meaning of d-action and its effect**

<b>Project 9: Determining the optimal motion behaviour</b>	
Name:	Date:
Program for the actuation of three motors	Sheet 1 of 1

- Create a program for the actuation of three motors in Robotino® View.
- Visualise and observe the setpoint/actual behaviour in the jacked-up state and during motion using the same controller setting.
- Explain the deviations in the curve progression.
  
- Create a motion program using three motors with Robotino® View.
- Proceed in the same way as that used to create the program using one motor.
- Connect one generator and one oscilloscope in each case.
- Set the controller to the value you have determined.
- Observe the setpoint/ actual curve progression in the jacked-up state and during motion.
- Describe the behaviour of the curves both in the jacked-up state and during motion.

<b>Description: Behaviour of curve</b>	
Jacked-up Behaviour of curve	
Moving Behaviour of curve	

- Explain why the curve during motion deviates from the curve in the jacked-up state.

<b>Explanation: Different behaviour of curves</b>	

Project 9: Determining the optimal motion behaviour

## **Project 10**

Path tracking of an automated guided vehicle system with the help of a webcam

### Training aims

#### Trainees

- are familiarised with the webcam of the Robotino® and are able to access this
- are able to configure and use the line detection function in Robotino® View
- are familiar with the limitations and parameters for line detection
- are able to realise path tracking via the webcam.

### Problem description

An automated guided vehicle system (AGV) is to follow a visible guide line on the surface with the help of a camera system. The AGV is to stop once it has reached the end of the guide line.

### Project assignment

Commission the camera of the Robotino® and create a program which enables you to access the camera image. Expand this program by adding a function for line recognition. Establish the parameters which enable line recognition and tracking. Realise a program for path tracking.

### Work assignment

Commission the camera system and set the camera.

Test the camera and line recognition functions in Robotino® View.

Parameterise the camera and line recognition.

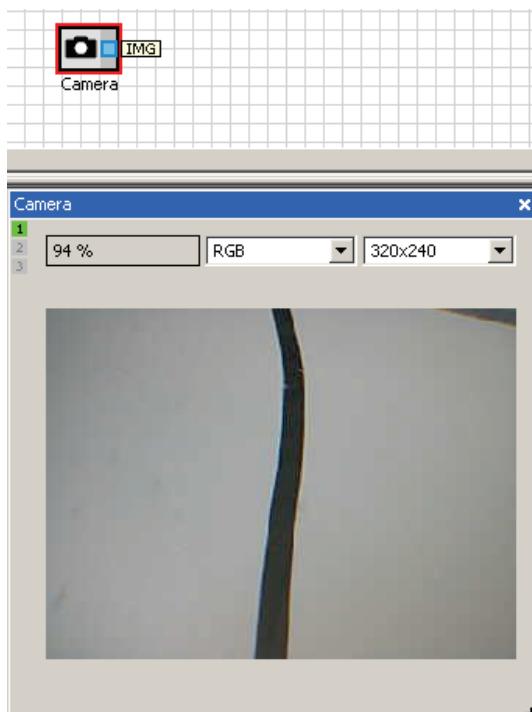
Program and optimise the required functions.

<b>Project 10: Path tracking of an automated guided vehicle system with the help of a webcam</b>	
Name:	Date:
Commissioning of the Robotino® webcam	Sheet 1 of 1

To realise this project you first of all will need to commission and test the Robotino® webcam. Describe how you proceed and create a function block diagram whereby you can test the functioning of the camera.

- Describe your procedure in individual steps:

Note By turning the lens, the camera can be accurately focussed for the desired image acquisition range.



<b>Project 10: Path tracking of an automated guided vehicle system with the help of a webcam</b>	
Name:	Date:
Alignment of the camera	Sheet 1 of 1

To evaluate the camera image, the camera must be set accordingly.

How does the camera need to be aligned to be able to detect a line in front of the Robotino®?

- Carry out your setting.

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

Project 10: Path tracking of an automated guided vehicle system with the help of a webcam

<b>Project 10: Path tracking of an automated guided vehicle system with the help of a webcam</b>	
Name:	Date:
Programming the line detection function	Sheet 1 of 1

- Expand the program to test the line detection function of the camera.
- Describe in a few words what you need to do so.

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Project 10: Path tracking of an automated guided vehicle system with the help of a webcam

<b>Project 10: Path tracking of an automated guided vehicle system with the help of a webcam</b>	
Name:	Date:
Testing the functions of the line detection function block	Sheet 1 of 1

- Describe the functions of the line detection function block in Robotino® View.
- Test the different parameters (inputs/outputs) of the module.

Parameter	Function

## Project 10: Path tracking of an automated guided vehicle system with the help of a webcam

<b>Project 10: Path tracking of an automated guided vehicle system with the help of a webcam</b>	
Name:	Date:
Determining the optimal colour spaces for the line detection	Sheet 1 of 2

- Examine the effect of the different colour spaces of the camera in respect of line detection.
- Enter your findings in the table below.

Solution	Colour space	Findings
	RGB	
	YCbCr	
	HSV	
	HLS	

<b>Project 10: Path tracking of an automated guided vehicle system with the help of a webcam</b>	
Name:	Date:
Expansion of the test program with the addition of line detection	Sheet 2 of 2

- Test the individual inputs and outputs of the line detection module and describe the possible settings. Place a DIN A3 sheet with a black line (width > 5mm) in front of the Robotino®. Align the Robotino® and the camera so that the line is within the detection range of the camera.
- By means of experiments, determine the colour model of the camera best suited for line detection and test the input setting options of the line detection module.

How is the detected line displayed?

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

<b>Project 10: Path tracking of an automated guided vehicle system using webcam</b>	
Name:	Date:
Determining the line detection value pattern	Sheet 1 of 1

To realise path tracking you need to determine the output values supplied by the line detection. This will enable you to decide what strategy to consider for the realisation of path tracking.

- What information is supplied by output X of the line detection module?
- Enter the values determined in the table below.

Note

Determine the value patterns as follows:

Make sure that a line is detected. You can establish this by the fact that the input "line found" has the value 1.

First, position the Robotino® so that the line detected is located in the bottom righthand corner of the camera image.

Then rotate the Robotino until the line detected is directly in front of the Robotino.  
Now position the Robotino so that the line detected is in the bottom lefthand corner of the camera image.

---

---

---

---

---

---

---

---

Position	Resolution 640 x 480	Resolution 320 x 240
Line is located outside to the right		
Line is located directly in front of the Robotino		
Line is located outside to the left		

<b>Project 10: Path tracking of an automated guided vehicle system with the help of a webcam</b>	
Name:	Date:
Determining the strategy for path tracking	Sheet 1 of 1

- Describe the program functions required for path tracking. Use the higher resolution of the camera (640 x 480).
  - Realise the necessary functions in a closed-loop control program.
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 

Note

Use sequence or closed-loop control programs you have already developed and modify these accordingly.

Use the sensor values to adapt the rotational speed of the deviation from the zero position. Parameterise the sensor value using an appropriate parameter dialogue function.

Select a low threshold value, such as 20, in order to accelerate the line detection. Remember to integrate collision protection into the program. Adapt the output values of the line detection module such as to achieve constant motion and reasonably jerk-free rotational movement. Move forward slowly.

<b>Project 10: Path tracking of an automated guided vehicle system using webcam</b>	
Name:	Date:
Expansion of the closed-loop control program: Detection of the end of the guide line	Sheet 1 of 2

The desired destination is reached when the guide line ends. The Robotino® must then be stopped.

- Expand your closed-loop control program with the addition of a function which stops the Robotino® if the line is not detected.

<b>Project 10: Path tracking of an automated guided vehicle system with the help of a webcam</b>	
Name:	Date:
Expansion of the closed-loop control program: Optimisation of the line detection	Sheet 2 of 2

It is quite possible that in addition to the line to be followed, other lines not visible to the naked eye will be detected on the camera image. These may interfere with the detection of the desired line and result in the guide line to be abandoned, particularly if these lines are close to the image border.

- Consider how you can minimise these influences on line detection.

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

Project 10: Path tracking of an automated guided vehicle system with the help of a webcam

## **Project 11**

### Searching and approaching a coloured object with the help of a webcam

#### Training aims

#### Trainees

- are familiarised with the webcam of the Robotino® and are able to access it
- are able to configure and use the colour recognition functions of Robotino® View
- are familiarised with the limitations and parameters of colour recognition
- are able to realise object detection using the webcam.

#### Problem description

An object is to be examined for corrosion in an area which is difficult to access. A mobile diagnostic system is to search for the object which can be identified by colour and approach it sufficiently closely so that it can be examined for corrosion with the help of the camera image. The Robotino® represents the diagnostic system. The object is to be represented by a coloured card.

#### Project assignment

If you have not yet commissioned the camera of the Robotino®, you should first commission the camera and create a program which enables you to access the camera image.

Expand this program with the addition of colour recognition.

Establish the parameters which facilitate colour recognition.

Realise a program for the detection of coloured objects and move the Robotino® towards the objects to the extent where these are transmitted as detailed as possible to the PC.

Make sure that the object is not damaged as a result of coming into contact with the Robotino®. Maintain a minimum distance from the object.

#### Work assignments

Establish access to the camera image of the Robotino®.

Determine the function blocks for colour recognition and examine their potential for the evaluation of colour information. Parameterise the function blocks.

Program the required subprograms and combine these into a sequence program.

#### Working aids

Theory section, Robotino® manual

Project 11: Searching and approaching a coloured object with the help of a webcam

<b>Project 11: Searching and approaching a coloured object with the help of a webcam</b>	
Name:	Date:
Evaluation of the camera image	Sheet 1 of 3

- Determine the function blocks required for colour recognition and create a function block diagram for colour recognition.
- Note down the function blocks required and create the necessary function block diagram.

Function blocks required:

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Project 11: Searching and approaching a coloured object with the help of a webcam

<b>Project 11: Searching and approaching a coloured object with the help of a webcam</b>	
Name:	Date:
Evaluation of the camera image	Sheet 2 of 3

The camera image must be evaluated for the motion behaviour of the Robotino®.

- Which function block do you need to use to do so and which function block outputs can you thus evaluate?
- Describe their function and how these can be used for the exercise given.

Output	Functional description
X	
Y	
Area	
Found	

Output	Use within the function block diagram
X	
Y	
Area	

### Note

A higher tolerance can be achieved for colour recognition by adjusting the pixel intensity for individual channels. By increasing the pixel intensity (+ key), it is to some extent possible to compensate colour deviations which can occur, for example, when approaching the object or if the lighting conditions are changed. This means that the coloured object will still be detected in this case. Determine the optimal setting by means of experimenting.

**Check these settings whenever you reload the program.**

Project 11: Searching and approaching a coloured object with the help of a webcam

<b>Project 11: Searching and approaching a coloured object with the help of a webcam</b>	
Name:	Date:
Evaluation of the camera image	Sheet 3 of 3

How can you ensure that the object is shown in full image and optimal quality?

- Describe all the possibilities you are considering and select one of these for your program.

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

Project 11: Searching and approaching a coloured object with the help of a webcam

<b>Project 11: Searching and approaching a colour object with the help of a webcam</b>	
Name:	Date:
Definition of the strategy and its subfunctions	Sheet 1 of 2

- Define your strategy for the solution of the exercise. Note down the necessary subfunctions.

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

## Project 11: Searching and approaching a coloured object with the help of a webcam

<b>Project 11: Searching and approaching a coloured object with the help of a webcam</b>	
Name:	Date:
Definition of the strategy and its subfunctions	Sheet 2 of 2

- Describe the step enabling conditions for the program functions required for your strategy and make a note of the necessary function blocks and the output required.

Function	Step enabling condition
Finding object	
Approaching object	

Project 11: Searching and approaching a coloured object with the help of a webcam

<b>Project 11: Searching and approaching a coloured object with the help of a webcam</b>	
Name:	Date:
Realisation of the sequence program	Sheet 1 of 1

- Realise the individual program functions and combine these into a sequence program.

Project 11: Searching and approaching a coloured object with the help of a webcam

## **Part B – Theory**

### **Drive**

1	Closed-loop control/PID controller	B-3
2	Robot subsystems: Drive	B-21

### **Sensors**

3	Characteristic curve	B-35
4	Infrared distance sensor	B-37
5	Optical proximity sensor	B-41
6	Inductive sensor	B-51
7	Sensitive edge, collision detection	B-53
8	Webcam	B-55

### **Robotino® View**

9	Generators	B-57
10	Oscilloscope	B-61
11	Segmenter	B-65
12	Segment extractor	B-69
13	IF function	B-71
14	Sign reversal	B-73
15	Sequence control	B-75



# 1. Closed-loop control/PID controller

What is closed-loop control?

On machines or within systems, variables such as pressure, temperature or flow often need to be set to predefined values. Furthermore these set values should not change even in the event of any disturbances occurring. These functions are assumed by closed-loop control.

Closed-loop control deals with any problems occurring in conjunction with this task. In order for a variable to be controlled, and to be available to a closed-loop controller in the form of an electrical signal, it first has to be measured and correspondingly converted.

This variable needs to be compared with the specified value or the value pattern in the controller. From this comparison it is then necessary to derive the response required within the system.

Finally, a suitable point must be found within the system, via which the variable can be controlled (e.g. the actuator of a heater). To be able to do so, it is important to know how the system behaves.

Closed-loop control technology tries to establish the generally applicable relationships which universally occur in different technologies. Most textbooks explain this with the help of higher mathematics. This chapter is intended to explain basic terminology and information regarding closed-loop control technology so as to largely dispense with mathematics.

Open-loop control/  
closed-loop control  
technology

## **Open-loop control**

The German standard DIN 19226 defines this as follows: Open-loop control is a process occurring within a system where one or several input variables exert an influence on other variables in the form of output variables according to the specific rules of the system.

The characteristic feature of open-loop control is the open action flow, i.e. the output variable does not have any retroactive influence on the input variable.

## **Closed-loop control**

The German standard DIN 19226 defines this as follows: Closed-loop control is a process within a system whereby the variable to be controlled (controlled variable) is continuously monitored and compared with the specified value (reference variable). Depending on the result of this comparison, the input variable of the system is influenced in such a manner as to bring about the adaptation of the output variable to the specified value despite any disturbances. This response brings about closed action flow.

Basic terms of closed-loop control technology

### **Reference variable**

The reference variable W is also referred to as the setpoint value of the controlled variable. It specifies the desired value of the controlled variable. The reference variable can remain constant over time; but can also change over time. The desired value of the reference variable is known as the actual value.

In closed-loop control the task is to keep the controlled variable at a desired value or to follow the desired value curve. This desired value is known as the reference variable.

### **Controlled variable**

Definition: The aim of closed-loop control is to keep a variable at a specified value or value curve. This variable to be controlled is referred to as controlled variable x.

This problem occurs in systems or on machines of most widely diverging technologies, the variable to be controlled is known as the controlled variable.

Example: Speed of a DC motor

See project 2

The setpoint and actual value of the speed should be set virtually the same in order to obtain optimal motion behaviour.

Examples of controlled variables are:

- The pressure in an air reservoir
- The pressure in a hydraulic press
- The temperature in an electroplating bath
- The flow of coolants in a heat exchanger
- The concentration of a chemical in a stirring vessel
- The speed of a feed motion in machine tool using an electric drive
- The speed of a motor

### **Manipulated variable**

Closed-loop control can only be effected if there is a possibility of intervening in a machine or system in order to change the controlled variable.

The controlled variable can be influenced in any system by means of intervention. This intervention alone allows the controlled variable to be set such that it corresponds to the specified value. The variable which effects such intervention is known as the manipulated variable y.

## 1. Closed-loop control/PID controller

Examples of manipulated variables are:

- The setting of an exhaust air restrictor on an air reservoir
- The setting of a hydraulic pressure regulating valve
- The voltage applied to the electric heating element of an electroplating bath
- The setting of a flow control valve in a coolant feed line
- The setting of a valve in a chemicals feed line
- The voltage on the armature of a DC motor

### Disturbance variable z

Disturbances occur in any controlled system and these are what make closed-loop control a matter of necessity. The effects of such disturbances are known as disturbance variables  $z$ .

The controlled system is the part of a machine or system where the controlled variable is to be adjusted to the specified value and where the manipulated variables adjust the disturbance variables. A controlled system consists not only of the manipulated variable as an input variable, since disturbance variables also occur as input variables.

### System deviation $x_d$

The comparison of reference and controlled variable is known as system deviation  $x_d$ . It is calculated from the difference:

$$x_d = e = W - x$$

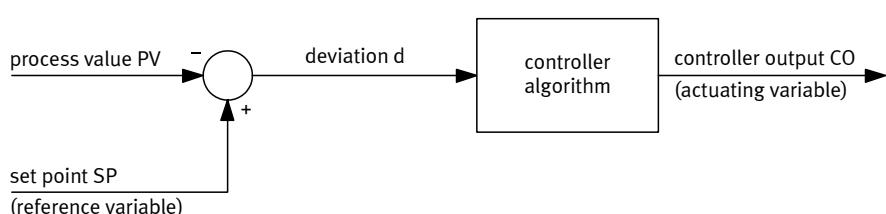
### Control response

The control response indicates how the controlled system responds to changes to the input variable. Determination of the control response is the aim of closed-loop control technology.

### Closed-loop controller

The task of the closed-loop controller is to keep the controller variable as near as possible to the reference variable. The controller constantly compares the value of the controlled variable with the value of the reference variable.

From this comparison and the control response, the controller determines and outputs the value of the manipulating variable.



## 1. Closed-loop control/PID controller

### Final control element and actuator

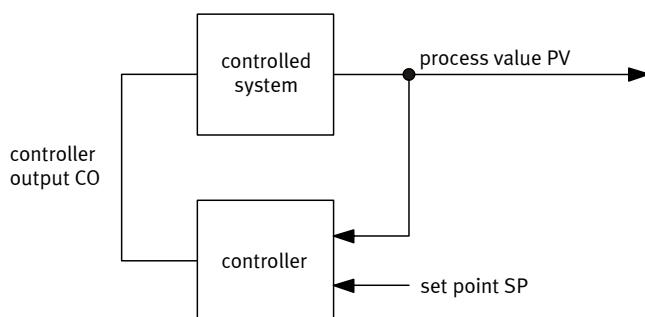
The final control element adjusts the controlled variable. The final control element is normally actuated by a special actuator. An actuator is always required in cases where it is not possible for the closed-loop controller to actuate the final control element directly.

### Measuring element

In order to make the controlled variable accessible to the controller, it must be measured by a measuring element (sensor, transducer) and converted into a physical variable that can be processed by the controller as an input.

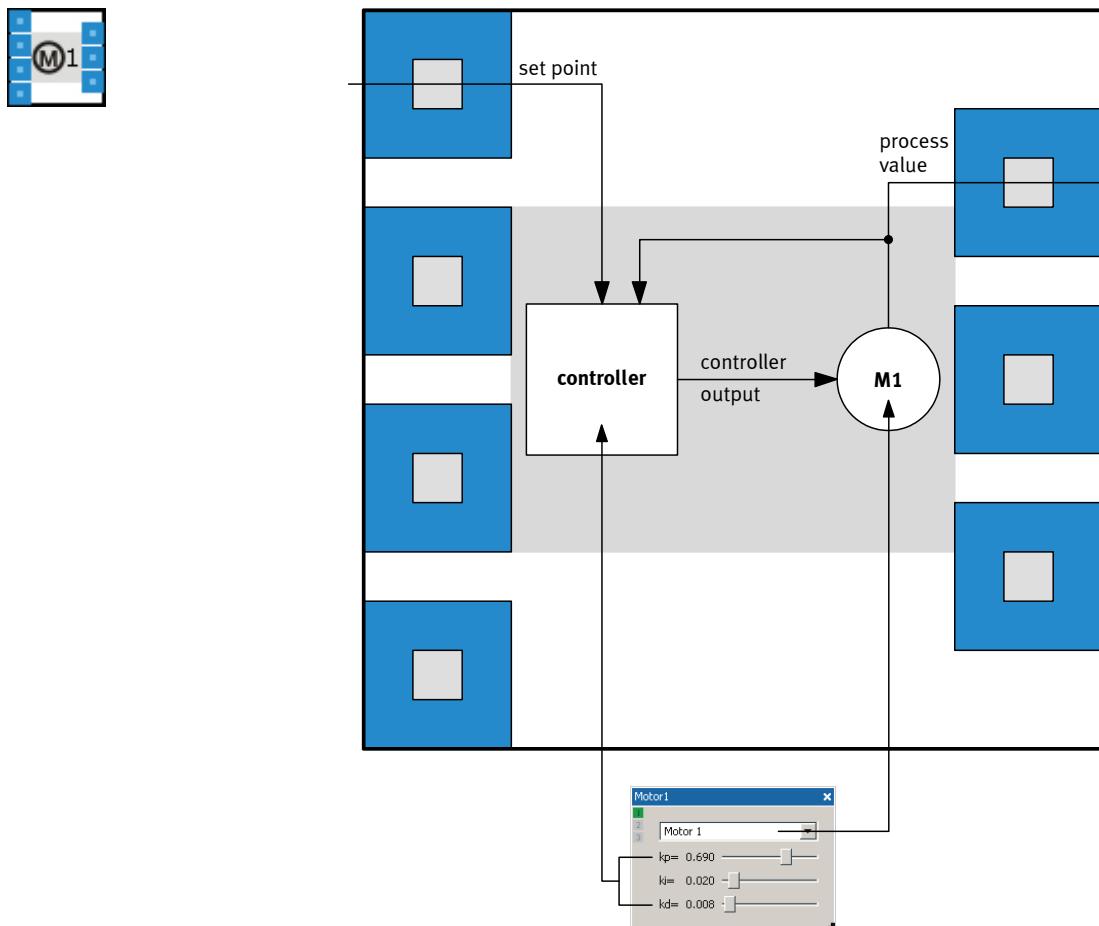
### Closed loop

The closed loop contains all components necessary for automatic closed-loop control.



## 1. Closed-loop control/PID controller

Example -Robotino®



The function block „motor“ comprises a software controller to adjust the speed of the motor. The reference variable  $W$  of the closed-loop controller is thus identical to the setpoint speed  $x$  of the motor.

- Controlled variable = Actual value of motor speed  
Measurement is effected via the motor encoder.  
The task of a closed-loop controller is to minimise the system deviation, i.e. the deviation of the actual value from the reference variable.
- The solution is effected via a PID controller:  
 $Ax(t) + P(t) + I(t) + D(t)$ , whereby  $Ax(t) = x(t)$  refers to the current speed of motor at the time  $t$ .  
Also the following applies:  
 $P(t) = \text{System deviation}(t) * K_p$ , whereby the following scaling was carried out:  
 $\text{System deviation}(t) = (W(t) - x(t)) * 255$

## 1. Closed-loop control/PID controller

- In order to calculate the integral-action component, the last 32 system deviations are added:

$$I(t) = \text{Total } (R(t) - R(t_j)) * K_i * 100, j=0,..,31.$$

The total should be interpreted as follows:  $R(t)$  refers to the system deviation at time  $t$ . The values  $R(t_j)$  refer to the measured system deviation at previous times  $t_j, j = 0,..,31$ .

- The differential component is calculated via:

$$D(t) = (R(t) - R(t-1)) * K_d$$

Numerous experiments have shown that the motor controller provides excellent overall response using the values

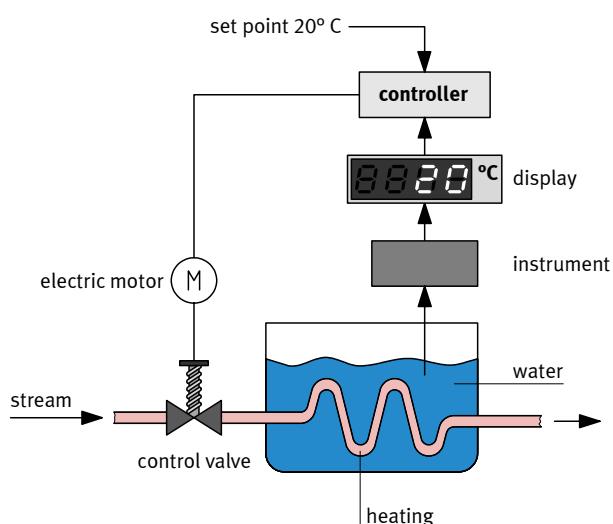
$$K_p = 0.9$$

$$K_i = 0.020$$

$$K_d = 0$$

### Example

Dependent on the system deviation, the closed-loop controller supplies a signal to the final control element. If the system deviation is mainly in the negative direction, i.e. the measured value of the volumetric flow rate is greater than the preset value (the reference variable), the valve is further closed. If the system deviation is mainly in the positive direction, i.e. the measured value is lower than the preset value, the valve is further opened.

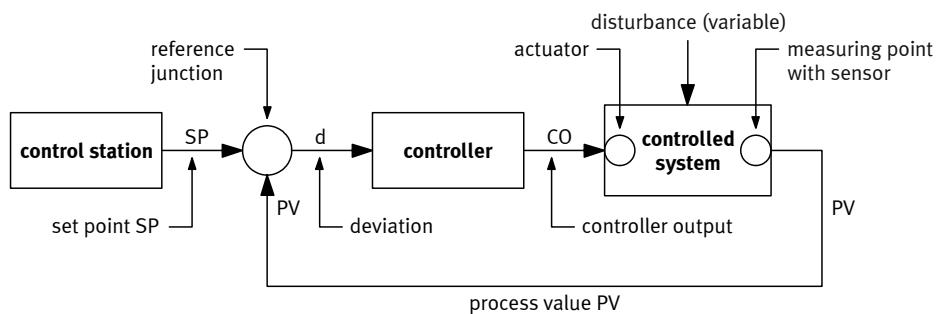


Generally it is not possible to optimally follow-up the output variable:

- If intervention is too quick or sudden, the system is too heavily activated at the input. The consequence is a fluctuating response at the output.
- If intervention is slow or weak, the output variable will only approximately follow the desired response.

## 1. Closed-loop control/PID controller

Moreover different systems, i.e. different controlled systems, also require different control strategies. Systems which involve long delays need to be controlled with care and foresight. This briefly outlines the problems of closed-loop control technology and the task facing the control engineer.



The following steps are required if closed-loop control is to be designed for a variable within a system:

- Defining the manipulated variable (this defines the controlled system),
- Determining the response of the controlled system,
- To establish the control strategy for the controlled system (response of the "controller" system),
- Selecting appropriate measuring and final control elements.

### Controlled systems

Definition: Complex relationships exist between the manipulated variable and the controlled variable. This relationship results from the physical interdependence of the two variables. The part of the control that describes these physical processes is called the controlled system.

The controlled system is the part of the machine or system in which the controlled variable is to be adjusted to the specified value and where the manipulated variables adjust the disturbance variables. A controlled system not only comprises the manipulated variable as input variable, since disturbance variables also occur as input variables.

## 1. Closed-loop control/PID controller

Before a controller can be defined for a controlled system, the behaviour of the controlled system must be known. The control engineer is not interested in the technical processes within the controlled system, but only in the system behaviour.

### Time response of a system

Of particular importance in closed-loop control technology is the time response of a system (also known as dynamic response). This is the time characteristic of the output variable (controlled variable) for changes in the input variable. Particularly important is the response when the manipulated variable is changed.

The control engineer must understand that nearly every system has a characteristic dynamic response.

Description of the time response of control systems

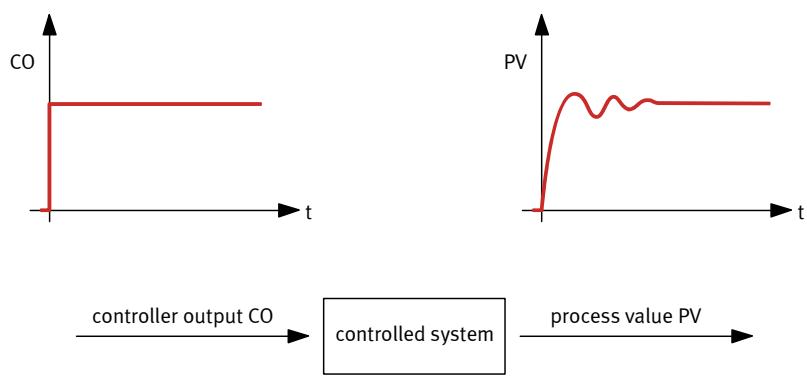
### Step response or transient function

The response of a system to a sudden change of the input variable is called the step response or transient function. Every system can be characterised by its step response. The step response also allows a system to be described using mathematical formulas.

### Dynamic response

This description of a system is also known as dynamic response. The illustration below demonstrates this correlation. Here the manipulated variable  $y$  is suddenly increased (see diagram below).

The step response of the controlled variable  $x$  is a settling process with transient overshoot.



### Steady state

Another description of a system is the behaviour in the steady state of the system, the static behaviour.

### **Static behaviour**

The static behaviour of a system is reached when none of the variables change with time. The steady state is therefore reached only when the system has settled. This state can be maintained for an unlimited time.

The output variable is still equally dependent on the input variable. This dependence is shown by the characteristic of a system.

### **Closed-loop controllers**

The previous section dealt with the controlled system – the part of the system which is to be controlled by a controller. This section looks at the closed-loop controller.

The controller is the device in a closed-loop system that compares the measured value (actual value) with the desired value, the setpoint value, and then calculates and outputs the manipulated variable.

The above section showed that controlled systems can have very different responses. There are systems which respond quickly, systems that respond very slowly and systems with storage properties.

In the case of each of these controlled systems, changes to the manipulated variable must take place in a different way. For this reason there are various types of controller each with its own control response. The task of the control engineer is to select the controller with the optimal control response for the controlled system.

### **Control response**

Control response is the way in which the closed-loop controller derives the manipulated variable from the system deviation.

PID control for motor control Standard linear controllers are most commonly used in industry. The transmission ratio of these controllers can be ascribed to the P-, I- and D components which represent the three basic linear forms.

The PID controller is the most important standard controller since it combines the good characteristics of other controller types and is very fast and accurate. This controller combines proportional, integral and differential behaviour. If a step occurs in a signal, the controlled variable initially exhibits PD behaviour, the D-action then drops off while the I-action increases as a function of time. The characteristics are those of the individual control units:

- K<sub>p</sub> - the proportional-action component of the PID controller upstream of the motor
- K<sub>i</sub> – the integral-action component of the PID controller upstream of the motor
- K<sub>d</sub> – the differential-action component of the PID controller upstream of the motor

## 1. Closed-loop control/PID controller

### Proportional controller

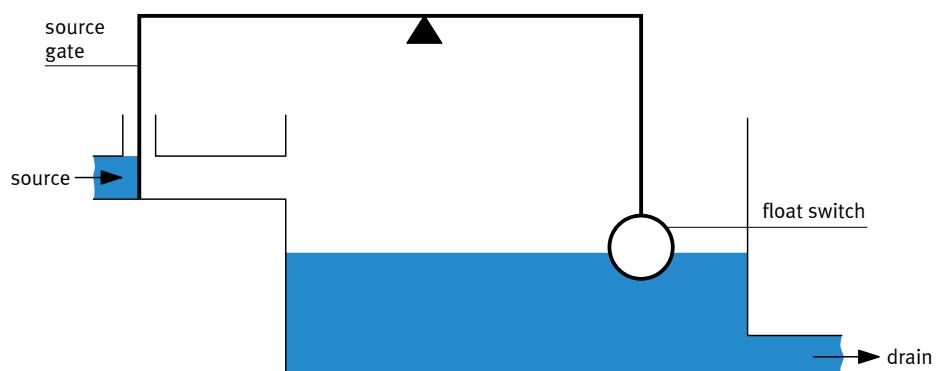
In the case of the proportional controller, the control signal is calculated proportional to the system deviation. If the system deviation is large, the value of the manipulated variable is also large. If the system deviation is small, the value of the manipulated variable is small. The time response of the P controller in the ideal state is exactly the same that of the input variable. The advantage is that the controller intervention is very fast and without delay.

### Example- level control

Water flows into a container via an inlet and forces the float upwards. The float acts on the gate valve via a lever. If water consumption is high, the gate valve has to be correspondingly opened. If consumption is low, the gate valve opens only very slightly.

This means that, if water consumption is high, the level of water in the tank is also lower than if consumption is low. This is the disadvantage of a proportional controller: Depending on the disturbance variable Z, the level of water in the tank varies.

These controllers are generally realised electronically.



### Area of application

Proportional-action controllers are used wherever the requirements for control precision are minimal. The P controller converts a step in the input signal directly into a step in the output signal. It has a fast response behaviour.

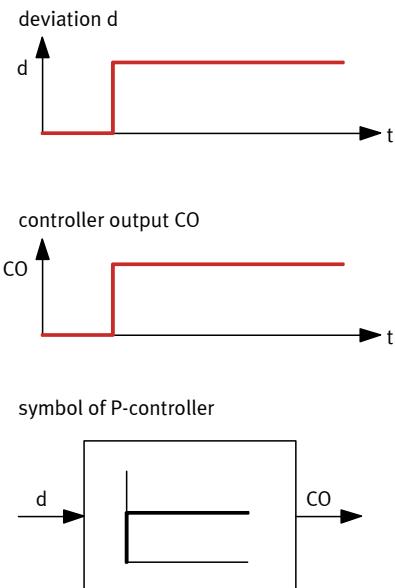
- Advantages

The advantages of a proportional controller are its speed and simple design.

- Disadvantage

The disadvantage is that control loops using proportional controllers exhibit residual system deviation. The controlled variable (actual value) never reaches the reference variable (setpoint value).

## 1. Closed-loop control/PID controller



Time response of a P controller:

With a P controller the response of the manipulated variable  $y$  is proportional to the system deviation  $e$

Integral-action controller

The effectiveness of an I controller increases over time. Even a slight system deviation results in a high output signal if it exists for a sufficiently long period. This controller converts input signal jumps into ramp-type output signals by means of continuous integration

This means that changes in manipulated variables are continuous and considerably slower than in the case of a proportional-action controller.

If a constant signal is applied at the input of an integral-action controller, the output changes continuously until the system deviation is compensated. The manipulated variable of an integral-action controller is proportional to the system deviation-time-area.

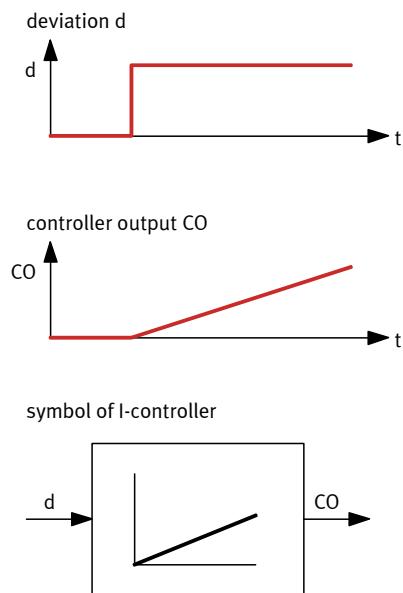
The greater the system deviation and system deviation over time, the steeper is the increase in the manipulated variable. In the case of an I controller, the system deviation and manipulating speed of the manipulated variable are proportional, i.e. the greater the system deviation, the faster the final control element is changed.

Pure integral-action controllers are rarely used since they tend towards instability and respond too slowly to fast changes.

## 1. Closed-loop control/PID controller

### Area of application

I controllers are frequently used to eliminate the disadvantage of a proportional controller of not being able to fully compensate the system deviation. This is why they are often used in combination with proportional controllers in practice.



Time response of an I controller:

With an I controller the manipulated variable responds proportional to the area of the system deviation and time

Differential-action controller In some controlled system major disturbance variables can rapidly become apparent. The controlled variable deviates greatly from the reference variable within a short time. Deviations such as these can be compensated with a D controller.

The output variable of a D controller is proportional to the temporal change in system deviation. A sudden change in system deviation therefore creates an infinitely large manipulated variable at the controller output.

### Area of application

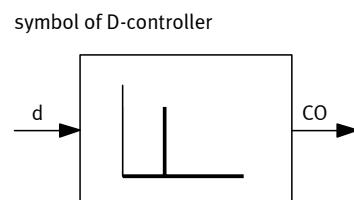
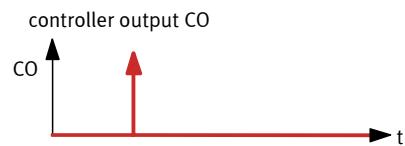
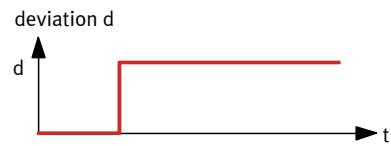
Since a D controller responds solely to the change in system deviation, it is not used on its own. It is therefore always used in combination with a P or PI controller.

A differential-action controller cannot adjust a residual system deviation and is therefore rarely used in industry.

Differential-action controllers are used in combination with a proportional-action or integral-action controller.

## 1. Closed-loop control/PID controller

The faster the change in system deviation occurs, the more effective a differential-action controller becomes.



Time response of a differential-action controller:

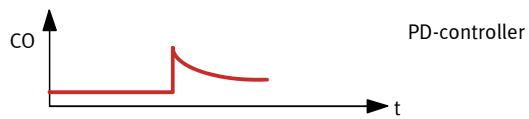
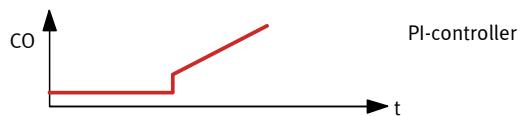
With a D controller, the manipulated variable is proportional to the change in system deviation

## 1. Closed-loop control/PID controller

### Combined controllers

Since the various types of closed-loop controller often do not exhibit the desired response for a particular control task, these are often combined. However, not all combinations of the three controller types are practical. The most frequently used combinations are:

- PI controller
- PD controller
- PID controller

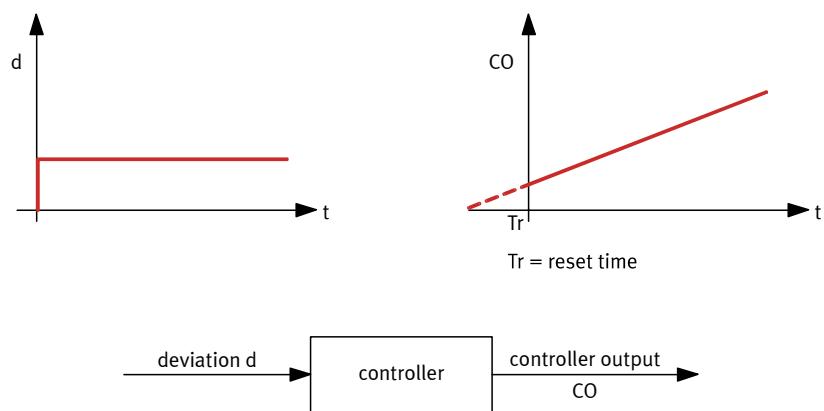


### PI controller

A PI controller combines the behaviour of the I controller and P controller whereby the advantages of both controller types are to be used: fast response of the integral-action controller and compensation of the residual system deviation of the proportional-action controller. A PI controller can therefore be used in a large number of controlled systems.

## 1. Closed-loop control/PID controller

In addition to proportional gain, a PI controller has a further characteristic that indicates the behaviour of the I component: the integral-action time which provides a measure of how fast the controller resets the manipulated variable in addition to the manipulated variable generated by the P-action to compensate a residual system deviation. The reset time is the period by which the PI controller is faster than the integral-action controller.



### PID controller

In addition to the properties mentioned of a PI controller, a PID controller also includes the derivative-action component. This takes into account the rate of change of the system deviation.

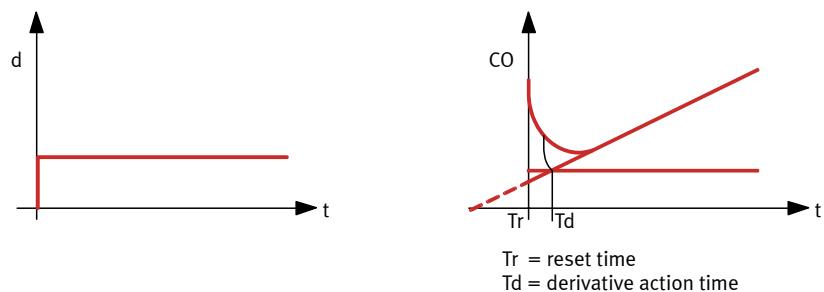
If the system deviation is large, the D-component ensures a momentary extremely high change in the manipulated variable. While the influence of the D-component drops off immediately, the I component increases slowly. If the system deviation is slight, the behaviour of the D-component is negligible.

- Advantage  
This behaviour has the advantage of faster response in the event of changes or disturbance variables and system deviations are therefore compensated more rapidly.
- Disadvantage  
The disadvantage is that the control loop is much more prone to oscillation and the correct setting of the controller is therefore more difficult.

## 1. Closed-loop control/PID controller

Derivative-action time

As a result of the D-action, this controller type is faster than a P or a PI controller. This manifests itself in the derivative-action time  $T_d$ . The derivative-action time is the period by which the PID controller is faster than the PI controller.



Summary	Controller type	Time response	Characteristics
P controller	P controller		For minimal requirements regarding reference variable. It is fast, but not able to fully compensate a system deviation.
I controller	I controller		Slow response; a system deviation can be fully compensated. In the event of large changes in the disturbance variable, the integral-action tends to oscillate.
D controller	D controller		Responds only to changes in system deviation. Is not used on its own.
PI controller	PI controller		Proportional-action controllers are often provided with a small integral-action component, which allows the system deviation to be fully compensated. This is a frequently used combination.
PD controller	PD controller		This combination is rarely used. It is suitable for closed-loop control where fast response is required to large changes in the disturbance variable.
PID controller	PID controller		Used for high requirements of closed-loop control systems. The P component effects fast closed-loop control, the I-component ensures high accuracy and the D component increase the speed of closed-loop control.

## 1. Closed-loop control/PID controller

Structuring and parameterisation of controllers

Closed-loop control forms a components part of automated systems whose main function consists of process stabilisation. They are used with the aim of

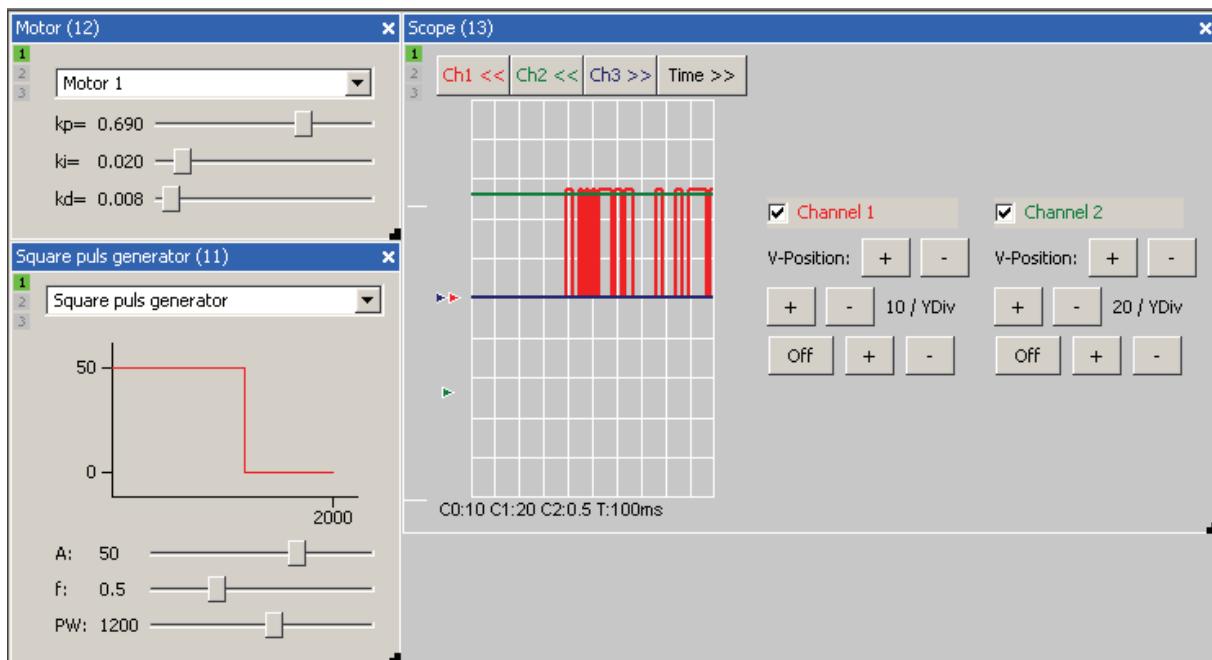
- bringing about and automatically maintaining certain process states (modes of operation)
- eliminating disturbances in process sequences and
- preventing the unwanted linking of subprocesses within the technical process.

Sizing by means of trial

This method of sizing the individual components is particularly suitable in this instance since we are dealing with a simple system. First the Kp, Ki and Kd components need to be roughly tuned. This is achieved by selecting the smallest possible Kp component and tuning the other two components to zero (Kp small, Kd=0 and Ki=0).

The Kp component (gain) is now slowly increased until poor damping is obtained.

Example - Robotino® View



Tuning of the PID controller

In Robotino® View, the adjustment of the speed of a DC motor can be easily realised. The sizing of the components kp, ki, kd can be effected by means of adjusting the slides.

## 1. Closed-loop control/PID controller

Poor damping

Poor damping is obtained if oscillation is reduced

→ Displacement of oscillation.

However if a tendency to oscillate is to occur, the K<sub>p</sub> component must be adjusted slightly lower. The I component is then added and increased and tested in steps until the result virtually coincides. If the result is still not satisfactory, the D component can also be added. This component enables closed-loop control to become more stable. If this is the case, the K<sub>p</sub> and K<sub>i</sub> components can be increased once more. This is repeated until the result is absolutely satisfactory.

This very practice-oriented and commonly used method of determining controller parameters does not always provide optimum results; this result is however adequate in the case of this system.

Other methods of tuning controller parameters are,

Sizing according to:

- Tuning rules
- The oscillation method
- Sep response
- Bode diagram

## 2. Robot subsystems: Drive

Industrial robots consist of various subsystems that fulfil different functions. The table below provides a schematic representation of these subsystems and subfunctions. The table also provides you with the relevant references as to where these subsystems and subfunctions are described in the technical documentation, data sheets or in this training package.

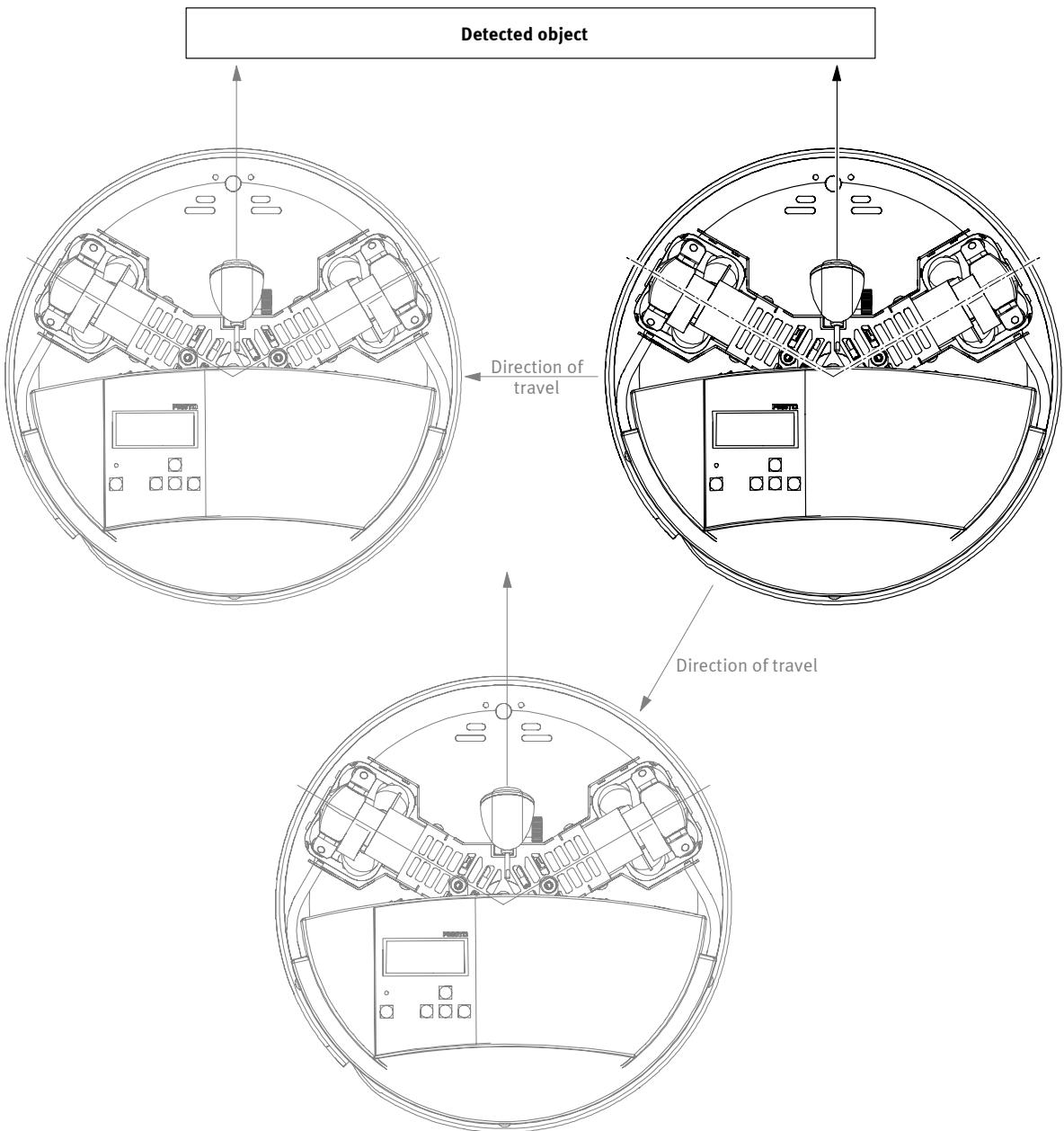
Subsystems	Subfunctions	Description in documentation regarding Robotino®	
Kinematics	Establishing the spatial relationship between workpiece/tool and production device	Part B	Freedom of movement of a system in the plane and within a space
		Part A	Project 6 and 7
Drive	Conversion and transfer of the required energy to all axes of motion	Part B	Omnidirectional robot, multidirectional wheels, omnidirectional drive
		Part A	Project 2 and 3 Technical documentation and data sheets regarding the topics: „DC motor, gear units, transmission, toothed-belt drive and multidirectional wheels“
Displacement encoder	Measurement of the position and speed of the individual axes of motion	Part A	Project 2 and 3 Technical documentation, data sheets: Incremental encoders
Open-loop/ closed-loop control	Storing, controlling and monitoring of the program sequence. Adjustment and control of rotational speeds and velocities	Part B	Closed-loop control technology
		Part A	Projects 2,3,4, 8,9
Sensors	Measurement of physical variables sample and position detection	Part B	Sensors
		Part A	Projects 4 to 8 Data sheets, technical documentation
Grippers	Gripping of workpiece, securing workpiece position during travel	Planned as expansion of Robotino®	

## 2. Robot subsystems: Drive

General information  
regarding omnidirectional  
robots

The advantage of omnidirectionally driven vehicles is that they can move in any direction without having to rotate.  
(= omnidirectional). The core element of an omnidirectional drive is a so-called omnidirectional wheel or multidirectional wheel or caster, also known as omniwheel. Usually barrel-shaped, these wheels are attached to the revolving surface of the main wheel, whose axes of rotation are at a right angle to the axis of rotation of the main wheel (see illustration).  
Multidirectional wheels can be actively driven via the motor and also passively roll laterally via the casters integrated in the wheel.  
A robot does not need to turn away from the object monitored; it retains its line of vision

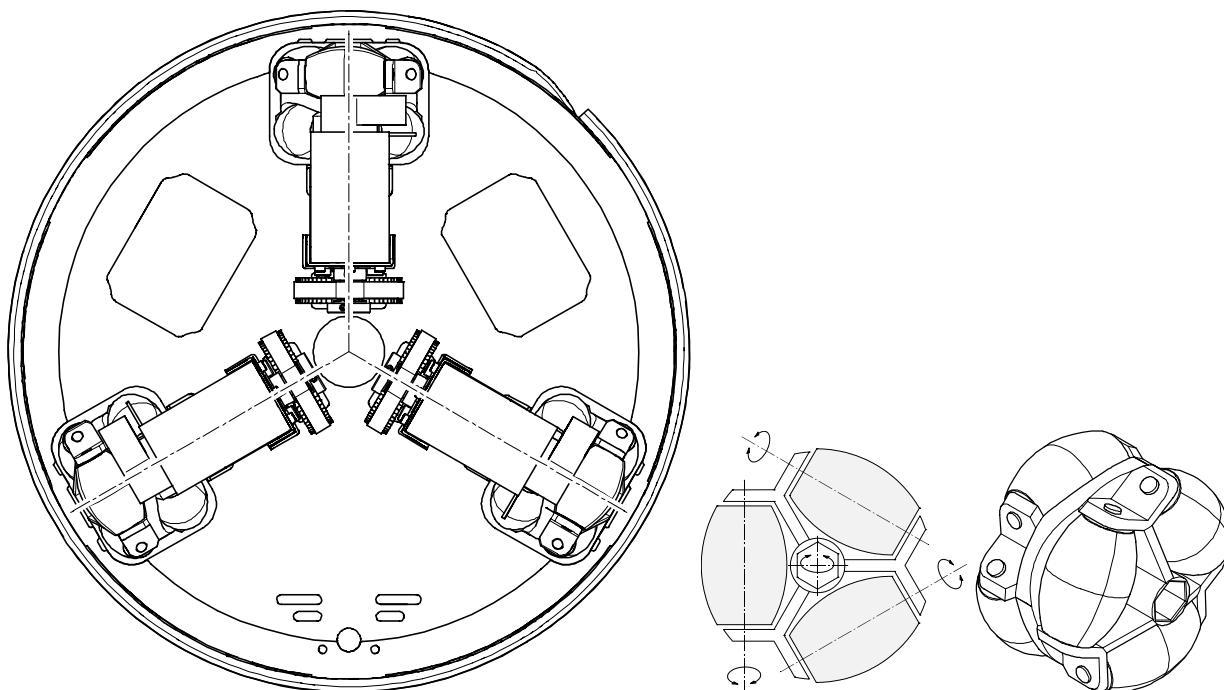
## 2. Robot subsystems: Drive



Moving with rotating (see also project 2, positional sketch)

### Multidirectional wheels

A multidirectional caster is moved in one direction via its drive axis and in addition can roll in any direction by means of the other casters. Through interaction with the other two drive units, it is therefore possible to generate a direction of motion which deviates from that of the direction of actuation.

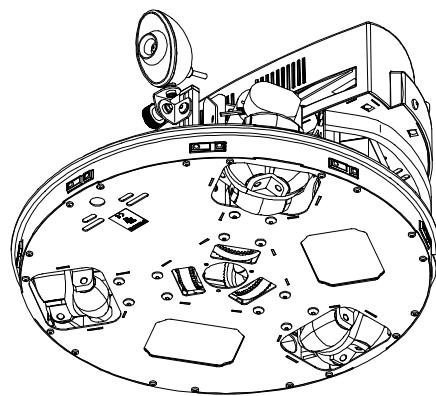


Omnidirectional drive

If the main wheel is driven, the two auxiliary wheels lock and act as the running surface of the main wheel. If the drive is stopped and the vehicle moved into another direction, for example via a second omniwheel attached at a right angle, the auxiliary wheels rotate and thus minimise the frictional resistance of the wheel. This type of design allows fast locomotion at virtually any angle to the direction of travel of the main wheel.

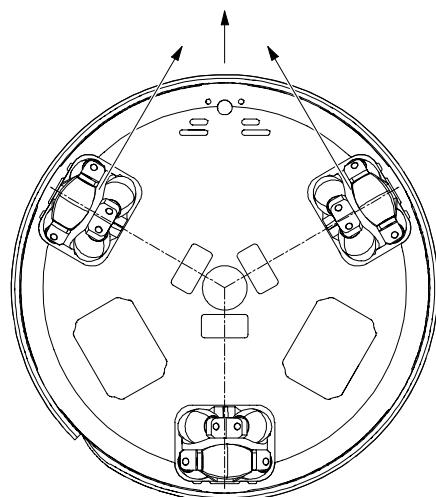
## 2. Robot subsystems: Drive

On a symmetrical drive, the wheels (multidirectional wheels or casters) are attached 120° apart.



Wheels 120° apart

The wheels on an omnidirectional drive are actuated such as to rotate together.



Direction of motion

## 2. Robot subsystems: Drive

### Multidirectional wheel – advantages and disadvantages

The advantage of this system is that, owing to the different speed control of the motors, the robot is able to travel in any direction without the need for turning. Additional advantages are:

- An omnidirectional vehicle is able to turn when stationary
- The turning circle on these vehicles is equal to zero, no need for shunting
- The weight of the entire robot is distributed across three wheels and system balance is therefore improved
- No steering mechanism required
- Mechanically simple and sturdy.

One disadvantage is that power consumption is relatively high in the case of forward travel, since there is a large number of rolling surfaces and greater rolling resistance is generated due to increased frictional resistance.

### Different wheel types

Different wheel types are used in the field of omnidirectional robots. This is always dependent on the surface on which the robot is to travel. Transverse casters made of soft plastic or polyurethane, are particularly suitable for hard and smooth surfaces such as glass or tiles. Hard transverse casters are more suitable for soft surfaces such as carpets or cardboard. A further difference with multidirectional wheels is the number of transverse casters used. As can be seen in the illustration on page 24, multidirectional wheels have three transverse casters per wheel. Multidirectional casters of this type originate from the materials handling sector where three transverse casters are adequate.

### **Freedom of movement of a system in the plane and space**

#### Degrees of freedom

The freedom of movement of a system in a plane or spatial direction is known as degrees of freedom. Degrees of freedom are the flexibility of movement of bodies in a plane or space.

Various possibilities are available to move a body within a space:

#### Translatory movements

x-coordinate

y-coordinate

z-coordinate

These degrees of freedom relating to such spatial displacement within a space are known as translatory degrees of freedom.

In addition, one two or three rotational degrees of freedom apply in the case of rotating, rigid bodies, depending on whether the body moves around one, two or three of its rotational axes.

#### Rotatory movements:

Rotation around x-axis

y-axis

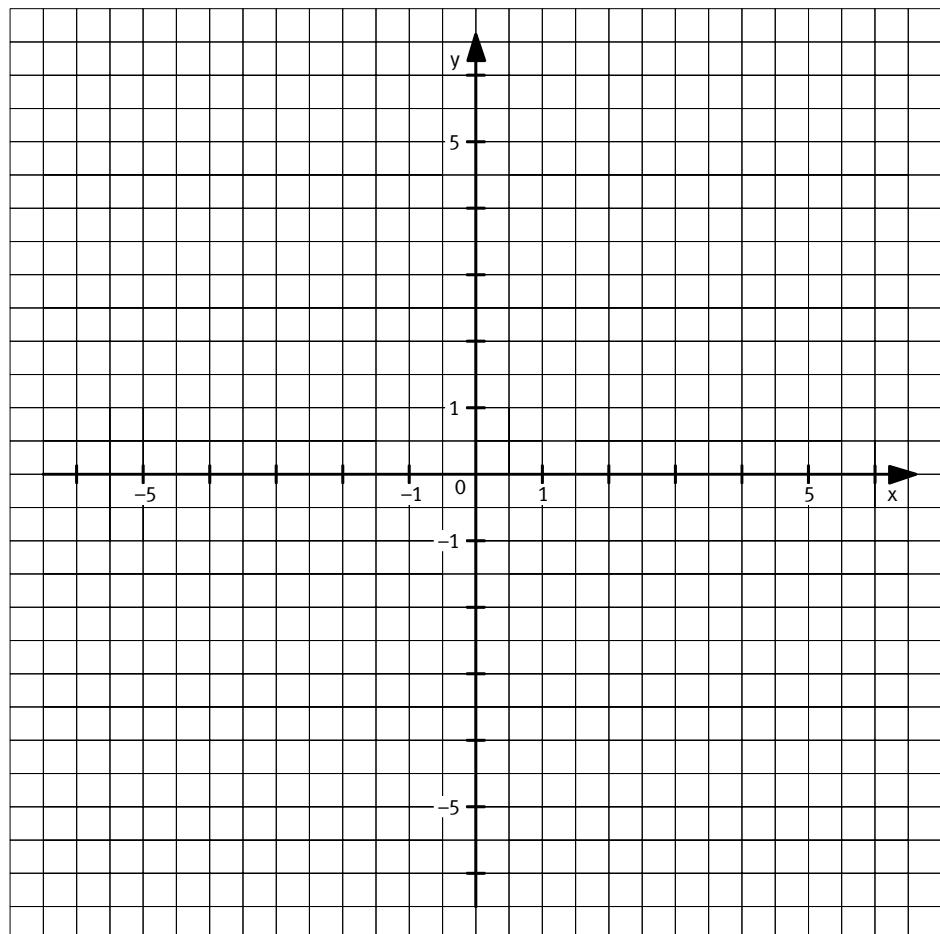
z-axis

#### Coordinate system

In order to represent the position and degrees of freedom of a body, the rectangular cartesian coordinate system is used (cartesian according to R. Descartes, 1596 to 1650).

The purpose of a coordinate system is to indicate the position of points of bodies within a space.

The position in the space is uniquely determined using the selected coordinate system by indicating numerical values, i.e. the coordinates. Specific objects (lines, curves, distances, areas, bodies), can then be specified by means of individual points.



Cartesian coordinate system, two-dimensional

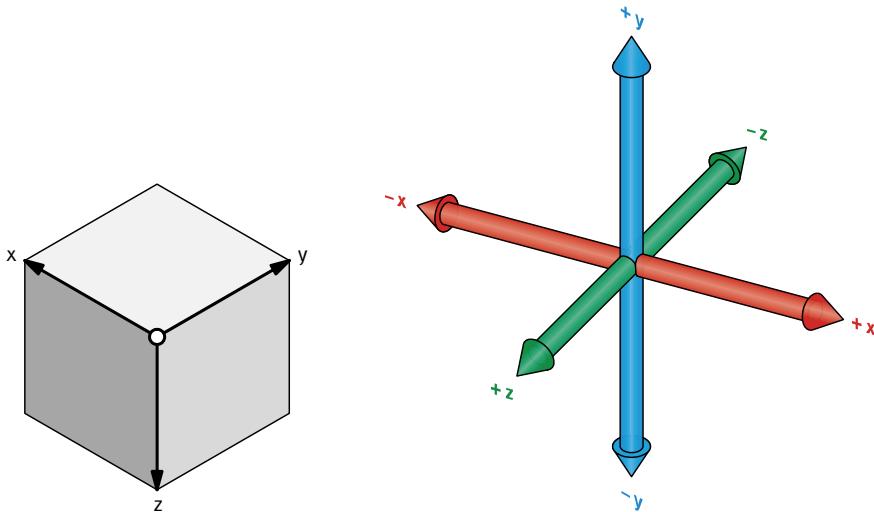
The rectangular planar Cartesian coordinate system consists of two perpendicular real axes. The horizontal line is the x-axis and vertical line the y-axis.

Any point in the plane can be denoted by the associated negative or positive x- and y-values.

Sections along the x-axis are known as the abissa, and along the y-axis as the ordinates. Measurement starts from the intersection (origin).

The three-dimensional spatial Cartesian coordinate system consists of three perpendicular real axes (the vertical axis is the z-axis). These have a common coordinate origin. The system or body is able to rotate freely around the coordinate origin. The x- and y axes are horizontal and the z-axis is vertical.

## 2. Robot subsystems: Drive



Cartesian coordinate system, three-dimensional

### Movement of bodies

Any movement of a body is possible in positive or negative direction. A body moving forward or backward along the x-axis, for example a rail vehicle. It can move forward or backward along a predefined path and has only one degree of movement.

A robot which is able to move straight ahead in forward or back direction moves along the x-axis in positive or negative direction (one degree of freedom, translation). A robot moving laterally moves along the y-axis in positive or negative direction. A robot moving in a straight line (forward, backward, positive or negative within the coordinate system) and laterally (to the right and left, positive and negative within the coordinate system) therefore has two translatory degrees of freedom.)

A robot travelling in a circle rotates around the central axis, the z-axis and therefore has an additional degree of freedom, the rotary degree of freedom. Rotation around the z-axis is possible in two directions.

The Robotino® can move in any direction:

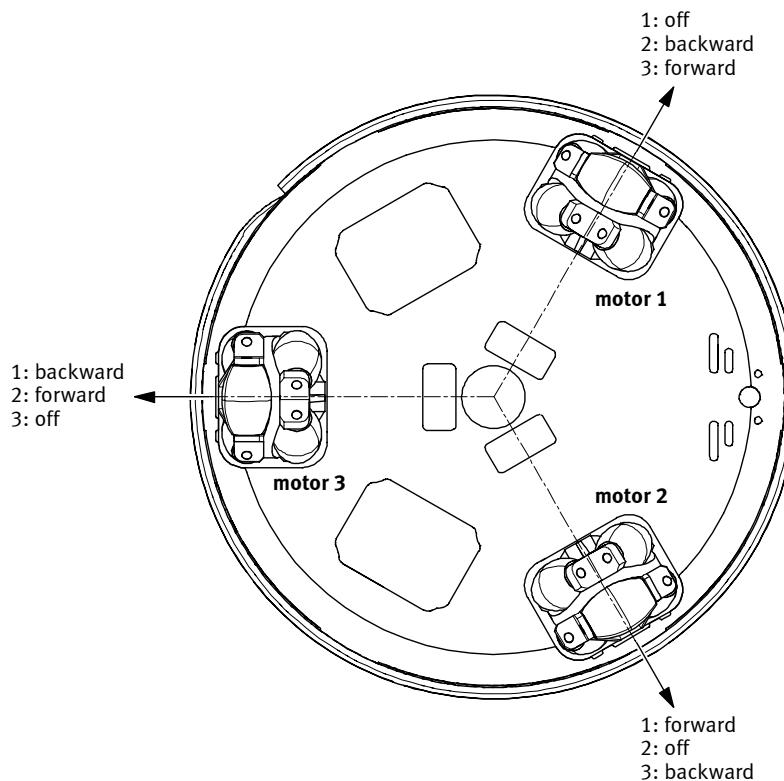
- Forward
- Backward
- Laterally
- In a circle with or without retaining the line of vision

and therefore has three degrees of freedom, two translatory and one rotary degree of freedom.

### Actuation of an omnidirectional drive

The actuation of an omnidirectional drive is fairly complex. With an omnidirectional drive it is only possible to move along by means of three non-steerable wheels. Three drive motors are actuated in the case of an omnidirectional drive. On the assumption that there are three control commands for the actuation of a motor (forward, backward, off), these three possibilities alone result in 27 options for three motors. These 27 control options allow the robot to be controlled in different directions.

A lateral control command, as is possible with the Robotino®, is not dealt with here at this stage.



### Actuation of the motors

The table below lists the 27 options possible via the following „commands“

- forwards,
- off,
- backwards

## 2. Robot subsystems: Drive

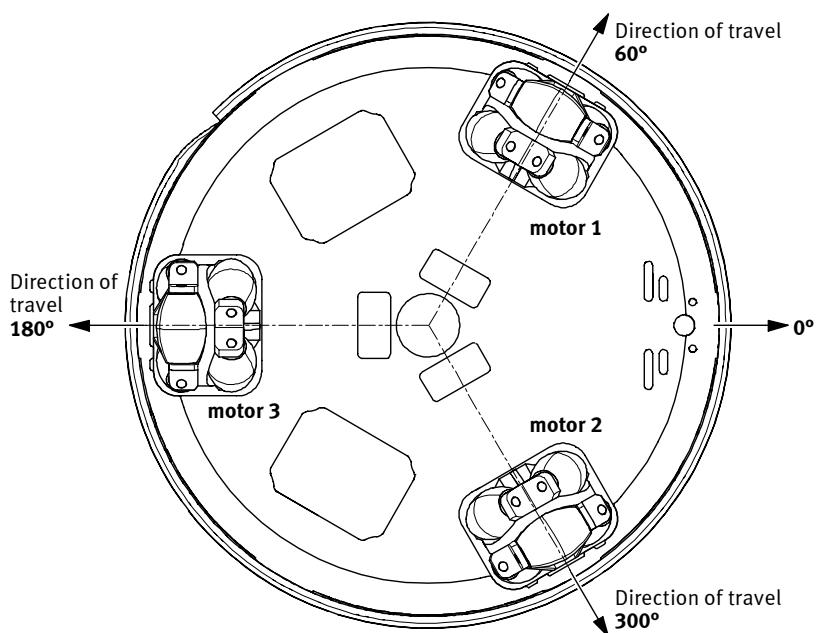
However the various speed levels have not been taken into consideration. Different speed levels would also enable the direction of travel to be changed. Each motor travels at the same speed level.

<b>Motor 1</b>	<b>Motor 1</b>	<b>Motor 1</b>	<b>Direction of travel</b>
Forward	Forward	Forward	Rotation in clockwise direction, while stationary
Forward	Forward	Off	Rotation in clockwise direction with small radius
Forward	Forward	Backward	Rotation in clockwise direction with large radius
Forward	Off	Forward	Rotation in clockwise direction with small radius
Forward	Off	Off	Rotation in clockwise direction with mean radius
Forward	Off	Backward	Travel to 300°
Forward	Backward	Forward	Rotation in -clockwise direction with large radius
Forward	Backward	Off	Travel to 0°
Forward	Backward	Backward	Rotation in anti-clockwise direction with large radius
Off	Forward	Forward	Rotation in clockwise direction with small radius
Off	Forward	Off	Rotation in clockwise direction with mean radius
Off	Forward	Backward	Travel to 240°
Off	Off	Forward	Rotation in clockwise direction with mean radius
Off	Off	Off	Stationary
Off	Off	Backward	Rotation in anti-clockwise direction with mean radius
Off	Backward	Forward	Travel to 60°
Off	Backward	Off	Rotation in anti-clockwise direction with mean radius
Off	Backward	Backward	Rotation in anti-clockwise direction with small radius
Backward	Forward	Forward	Rotation in clockwise direction with large radius
Backward	Forward	Off	Travel to 180°
Backward	Forward	Backward	Rotation in anti clockwise direction with large radius
Backward	Off	Forward	Travel to 120°
Backward	Off	Off	Rotation in anti clockwise direction with mean radius
Backward	Off	Backward	Rotation in anti clockwise direction with small radius
Backward	Backward	Forward	Rotation in anti clockwise direction with large radius
Backward	Backward	Off	Rotation in anti clockwise direction with small radius
Backward	Backward	Backward	Rotation in anti clockwise direction, while stationary

Actuation and direction of travel

## 2. Robot subsystems: Drive

The table above demonstrates how the motors need to be actuated to enable the robot to travel in 300 degree direction as shown in the illustration overleaf.



### Division into degrees

#### Actuation of the three Robotino® motors

The Robotino® has three drive motors which drive three multidirectional wheels. The rotational speed of an individual wheel can be set irrespective of the rotational speed of the other wheels. In order to move the Robotino® in one direction, the rotation of the individual wheels must be coordinated, since the correct interaction alone generates the desired Robotino® movement.

The motors can be actuated either directly with the help of Robotino View or with the help of the function block „omnidrive“. However, the use of the function block „Omnidrive“ (Robototino® View) is recommended for all applications.

#### Function block „omniantrieb“

The „omnidrive“ calculates the setpoint rotational speeds of motors 1,2 and 3 in x- and -y direction on the basis of a specified setpoint speed. In addition, the actual speed in x- and y-direction and the actual rotational speed is calculated from the actual speed.

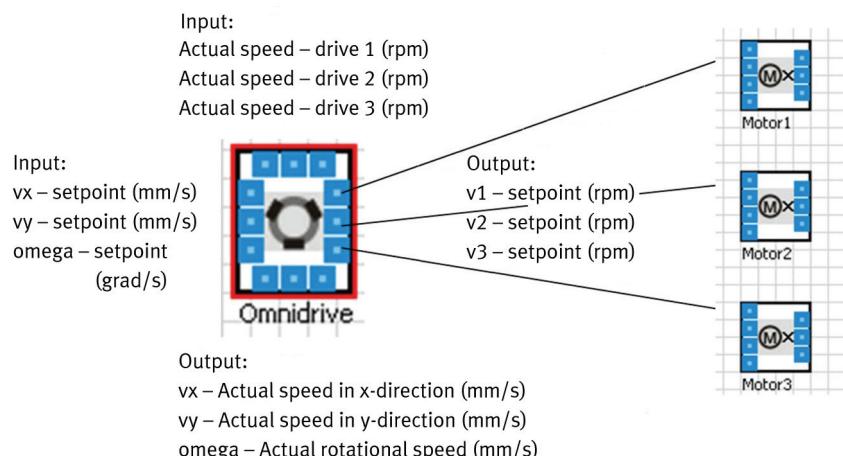
The Robotino® has two translatory degrees of freedom, the movement along the x- and y-axis for straight ahead and lateral travel. In addition rotation is possible around the z-axis.

The inputs and outputs of the omni module are correspondingly occupied:

## 2. Robot subsystems: Drive

<b>Inputs</b>	<b>Description</b>
vx_set	Setpoint speed in x-direction in mm/s
vy_set	Setpoint speed in y-direction in mm/s
omega_set	Setpoint rotational speed in deg/s
v1	Actual speed of motor 1 in rpm
v2	Actual speed of motor 2 in rpm
v3	Actual speed of motor 3 in rpm

<b>Outputs</b>	<b>Description</b>
v1_set	Setpoint speed of motor 1 in rpm
v2_set	Setpoint speed of motor 2 in rpm
v3_set	Setpoint speed of motor 3 in rpm
vx	Actual speed in x-direction (forwards, backwards) in mm/s
vy	Actual speed in y-direction (lateral to both sides) in mm/s
omega	Actual rotational speed (rotation) in deg/s



### Omnidrive

The direction (coordinates) and setpoint speed are defined, input and calculated at the input of the function block „Omnidrive“ and converted into the setpoint values for the respective motor speed (see fig. above).

The setpoint speeds for the three drives are output at the output of the function block „omnidrive“. These setpoint speeds realise the corresponding actual speed in x-, y-direction or the rotational speed.

## 2. Robot subsystems: Drive

Under the keyword „omnidrive“ in Robotino® View Help, you can find out how the conversion of the setpoint speed vector and the setpoint rotational speed is derived into the setpoint speed of the individual motors.

### Note

A small example of applied vector analysis can be found in project 3, linear travelling and positioning of a robot system „travelling a defined distance“.

### 3. Characteristic curve

General	A characteristic curve represents the relationship of two values in the form of a line within a two-dimensional coordinate system. In the case of the distance sensors of the Robotino® this is the relationship between the distance of the obstacles and the output values generated in Robotino® View.
Recording of a characteristic curve	As a rule, a characteristic curve is recorded by means of measuring the two variables and entering these in a corresponding coordinate system or table, whereby the values of the one variable are uniquely allocated to the other. The characteristic curve of the component is obtained if these point-pairs are connected by a line/curve. The accuracy of the curve is improved by increasing the number of measuring points. For a particularly important range of values, it is advisable to record the characteristic curve more precisely.
Linearisation of a characteristic curve	<p>Characteristic curves are mainly non-linear, but can at least be linearised in sections whereby a simple conversion of sensor values into standard units of measurement can be carried out in those sections. Characteristic curves can be easily determined by means of linearisation via a linear equation.</p> <p>The formula for a straight line in a coordinate system is (for the distance D and the sensor values X) <math>D = MX + B</math></p> <p>whereby</p> <p>M = Gradient of curve B = Offset of curve D = Distance of object X = Output value of sensor</p> <p>If the linear equation of the linearisation of a section is to represent the distance characteristic curve of the sensor, then M and B is to be determined such that the sensor value X supplies the distance value D.</p> <p>The constants M and B are determined via two trial measurements: D1 and X1 are the distance and the output value of a measurement (for example at 5 cm). D2 and X2 are the distance and output value of a second measurement (for example at 10 cm).</p> <p>The gradient M of a curve is determined by establishing the relationship between the difference of the distance values and sensor values and calculating a conversion factor from this.</p> $M = \Delta D / \Delta X$

### 3. Characteristic curve

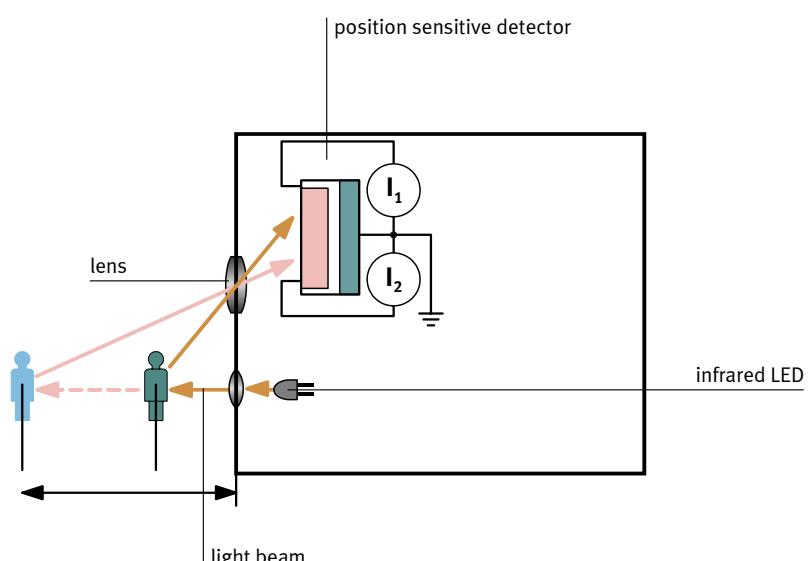
Inaccuracies occur with this method which can however be disregarded in most cases. It may be advisable to carry out different linearisation for different sections in order to eliminate inaccuracies. A reduction of the linearised section also serves this purpose.

## 4.Infrared distance sensors

### Functional principle

Infrared distance sensors consist of an emitter which emits an infrared light beam, a corresponding receiver and an electronic evaluation unit.

The emitter emits an infrared beam. If this does not impinge on an object, it is not reflected and the receiver therefore does not receive a light beam. However if the light is reflected by an object, the light beam is detected within a certain range of the receiver. Since the phototransmitter and the receiver are located a small distance from one another within the sensor, the emitted and received light beams form a triangle.



Infrared distance sensor: Triangulation method

Depending on the distance, the reflected light beam impinges at a different point on the receiver. The receiver consists of a position-sensitive detector (PSD), which detects the different points of incidence. A signal processing unit converts these into an analogue voltage value.

A PSD is a photo diode of lamellar form. It consists of a light-sensitive and a metallic layer. Metal electrodes are located at the ends of these layers. If a light beam impinges at a point on this light-sensitive layer, this releases charge carriers which generate a current flow towards the two electrodes. The unlit parts of the layer act as resistance. The relationship between the currents on the electrodes is dependent on the position of the point of incidence. The relationship determined between the currents is processed by the evaluation electronics.

#### 4.Infrared distance sensors

The relationship between the currents is independent of the impinging quantity of light; the distance measurement is therefore not dependent on the reflectivity and material of the object.

Diffused light and daylight are eliminated by means of pulsing the emitted light beam. Only signals received via such pulsing are used for evaluation. Continuous light is virtually "ignored".

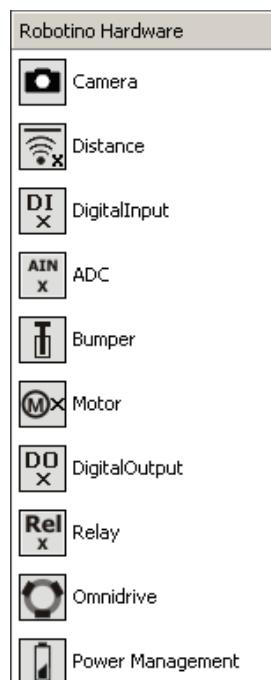
In the majority of cases, emitter, receiver and signal processing are combined into one unit.

Examples of typical areas of application for infrared sensors are parking distance control systems on cars, toilet flushing, door openers or alarm systems.

#### Infrared sensors in Robotino® View

The distance sensors in Robotino® View have one output A1, which supplies a voltage value of 0 – 2.55 V. The scaling and conversion of these values within a distance sensor must be carried out by the user.

The function block for distance sensors is in the function block library "Robotino® Hardware".



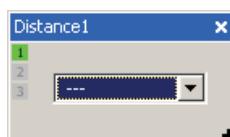
Create a new function block diagram.

#### 4.Infrared distance sensors

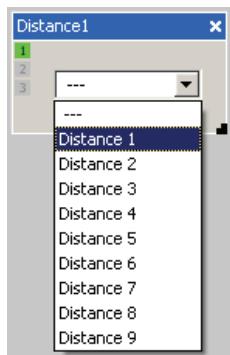
Drag the symbol for the function block "distance" to the desktop.



The respective internal parameter dialogue is opened by double clicking the function block.



Now assign one of the 9 distance sensors to the function block by selecting this from the list box.

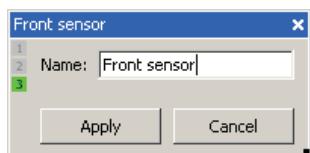


The appropriate sensor is then assigned to the function block. You can identify this by the corresponding sensor number which is displayed within the symbol.



#### 4.Infrared distance sensors

For easier identification you can also allocate a name to the sensor. To do so open the third window of the function block and enter the desired name and click onto the "Accept" button



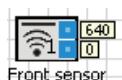
The name of the symbol is changed.



The second window lists all the output data of the function block with the corresponding designations which can be smoothed, scaled and adjusted.

Slot	Function	a	b	c	d
Value	y=x	0	0	0	0
Heading	y=x	0	0	0	0

The connection "Value" indicates the current voltage value of the sensor and the connection „Direction“ the direction of detection of the sensor, starting with 0° for the sensor "Distance 1".



## 5. Optical proximity sensors

### General characteristics

Optical proximity sensors use optical and electronic means for object detection whereby a red or infrared light is used. Particularly reliable sources for red and infrared light are semiconductor light emitting diodes (LEDs). They are small and sturdy, have a long service life and can be easily modulated. Photodiodes or phototransistors are used as receiver elements. The advantage of red light is that it can be detected by the naked eye when adjusting the optical axes of the proximity sensor used. Also polymer fibre-optic cables can be easily used thanks to their minimal subduing of light in this wave length range.

Infrared (non visible) light is used in applications where increased luminous power is required, for example to bridge greater distances. Furthermore, it is less susceptible to interference (ambient light).

In the case of both types of optical proximity sensors additional suppression of external light influences is achieved by means of modulation (pulsing) of the optical signal. The receiver is tuned to the pulse of the emitter. Particularly in the case of infrared light, the use of day-light filters further improves insensitivity to ambient light.

In the case of optical diffuse sensors the switching function works as follows:

- Brightness switching  
The output closes if an object to be sensed enters into the light beam. (normally open output = NO)
- Dark switching  
The output opens if an object to be sensed enters into the light beam. (normally closed output = NC)

### Design of optical proximity sensors

Optical proximity sensors basically consist of two main groups: the emitter and the receiver. Depending on the design and application, reflectors and fibre-optic cables will be required.

The emitter and receiver are either incorporated into one housing (diffuse sensors and retro-reflective sensors), or accommodated in separate housings (through-beam sensors).

The emitter contains a radiant source for red or infrared light which spreads in a straight line and can be diverted, focussed, interrupted, reflected and directed. It is received by the receiver and electronically evaluated.

Usually proximity sensors contain a light emitting diode (LED), which is illuminated when the output switches. The LED display serves as an adjustment aid and can be used for functional testing.

### **Operational reserve of optical proximity sensors**

Optical proximity sensors may be exposed to contamination such as dust, swarf or lubricants during operation and therefore function may be impaired as a result of contamination. Both contamination of the lenses of the proximity sensor optics or the object to be sensed may be the cause of this.

Heavy contamination within the light beam of through-beam sensors or retro-reflective sensors may cause the light beam to be interrupted. This will then simulate the presence of an object. In the case of diffuse sensors, heavy contamination of the lens system can be evaluated as an object being present, if the optical radiation reflects the contamination on the lens back to the receiver. Heavy contamination of the object itself can result in the evaluation of an object which is not present if less radiation is reflected as a result of contamination.

Optical proximity sensors have a certain degree of operational reserve (also known as functional reserve). A flashing indicator on the proximity sensor is a worthwhile means of checking the operational reserve, which becomes active if the minimum operational reserve is inadequate. Designs are for example available which start flashing if the operational reserve factor of  $\beta = 1.5$ . This signals that an operational reserve of 50 % is still available.

The flashing indicator can be used as an adjustment aid for the assembly and adjustment of the proximity sensor configuration. It also acts as a contamination display during subsequent operation the functional reserve is gradually reduced due to the effects of contamination.

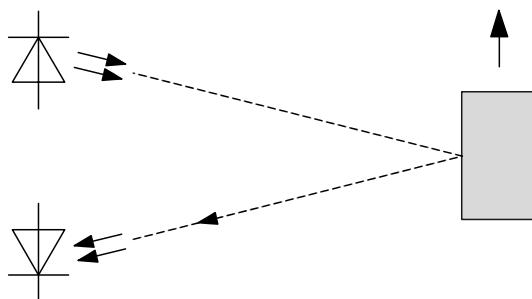
Inadequate functional reserve may also have causes other than contamination, e.g.:

- Exceeding of reliable operating range or sensing range
- Changes in the material surface of objects sensed
- Incorrect assembly (misalignment)
- Ageing of transmitter diode
- Fracture of fibre-optic cables

## 5. Optical proximity sensors

### Mode of operation

The emitter and receiver are accommodated in the same housing. The object reflects part of the emitted radiation and activates the receiver as a result of this. Depending on the receiver design, the output is then switched through (NO contact function) or switched off (NC contact function). The switching distance is heavily dependent on the radiance factor of the object. The size, surface, shape, density and colour of the object as well as the angle of incidence determine the intensity of the reflected radiation so that only small distances within a range of a few decimetres can generally be monitored. The background must absorb or reflect radiation; in other words, if no object is present, the reflected radiation must be clearly below the triggering level of the receiver circuit.



### Diffuse sensor (principle)

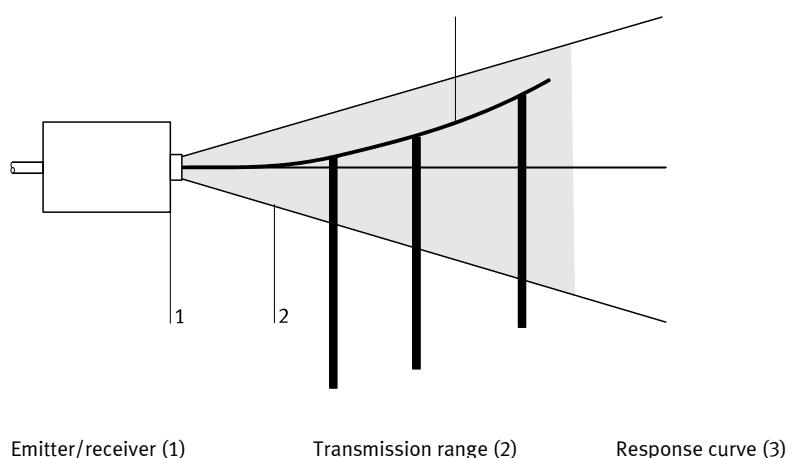
### Technical characteristics

The tables below list the most important technical characteristics of diffuse sensors. These tabular compilations provide typical data and are merely intended as a general overview.

Parameter	Value
Object material	Any
Operating voltage	10 – 30 V DC or 20 – 250 V AC
Sensing range	50 mm – 2 m (usually adjustable)
Maximum switching current (transistor output)	100 – 500 mA
Sensitivity to contamination	Sensitive
Service life	Long (approx. 100.000 h)
Switching frequency	20 – 2000 Hz
Designs	Cylindrical or block-shaped
Protection class (DIN 40050)	Up to IP67
Ambient operating temperature	0 – 60 °C or -25 – +80 °C

## 5. Optical proximity sensors

The sensing range data in the data sheets is usually in relation to white cardboard in that the white rear side of the Kodak grey card CAT 152 7795 from Eastman Kodak is generally used. The white side of this test card has a constant reflection of 90° within the spectral range of approx. 450 – 700 nm. The grey side reflects 18 %.



Small distance: No reflecting surface required.

Large distance: Large reflecting surface required.

### Response curves of diffuse sensors

#### Notes regarding use

##### Advantages of diffuse sensors

- Since the reflection of the light on the object activates the receiver, no additional reflector is required.
- The object can be diffuse reflecting, specular or diaphanous to transparent provided that a sufficiently high proportion of radiation is reflected by the object.
- Whereas through-beam sensors merely allow the detection of objects at a right angle to the light beam, diffuse sensors allow frontal detection, i.e. in the direction of the light beam.
- Depending on the diffuse sensor setting, objects can be detected selectively against a background.

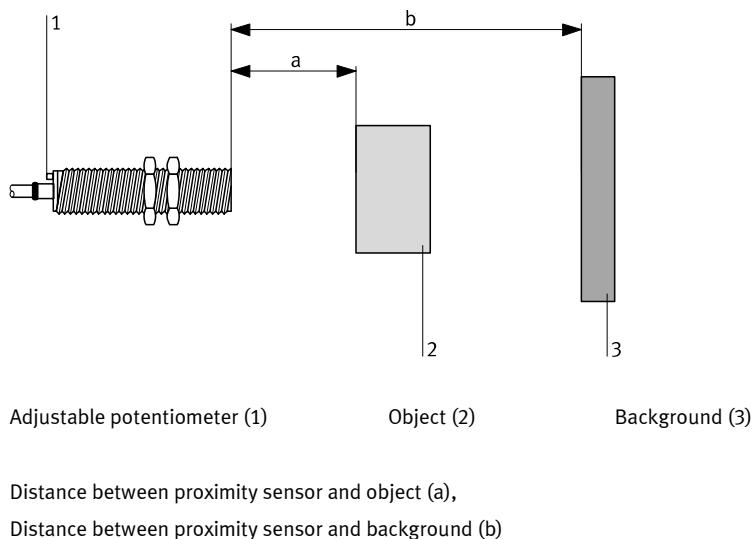
##### Disadvantages of diffuse sensors

- The response curves at a right angle to the beam spread direction are not exactly linear. Diffuse sensors are therefore not quite as effective as through-beam sensors if precision lateral response is important.

Notes

- The size, surface, shape, density and colour of the object as well as the angle of incidence determines the intensity of the radiation reflected so that, as a rule, only small distances within a range of a few decimetres can be monitored. The background needs to absorb or reflect radiation, i.e. in the absence of an object the reflected radiation must be clearly below the triggering level of the receiver circuit.
- Failure of the emitter is evaluated as "no object present".

**Background suppression with a diffuse sensor**



Background suppression with a diffuse sensor

Adjustable sensitivity

The action of a diffuse sensor is based on the difference in reflection from the object and background. If contrast is only minimal, the triggering level of the circuit can be selected by means of the sensitivity adjustment on the proximity sensor (single-turn potentiometer or multi-turn potentiometer), so that the object is reliably detected even under these less favourable conditions.

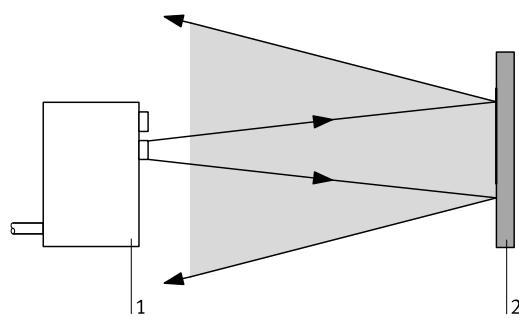
However a tolerance range must be allowed in respect of ageing, voltage and temperature fluctuations and contamination. The setting range should therefore not be fully exhausted when making the adjustment.

The careful adjustment of a diffuse sensor using a potentiometer must allow a certain reserve in respect of changes in the quality of the object, contamination of the proximity sensor, dust in the atmosphere, etc. Narrow, only just functioning settings may cause problems.

## 5. Optical proximity sensors

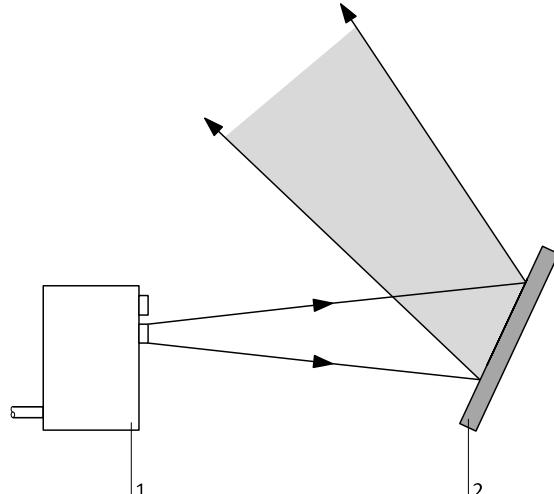
Some diffuse sensors are equipped with an adjustment aid in the form of a flashing LED to ensure reliable setting. The LED display flashes in the range of uncertainty. The setting is to be made such that, in the case of a proximity sensor with normally open contact, the LED is illuminated without flashing in the active switching status.

### Behaviour of a diffuse sensor in the case of a specular object



Emitter/receiver (1)      Specular surface (2)

Object is detected



Emitter/receiver (1)      Specular surface (2)

Object is not detected

## 5. Optical proximity sensors

### Transparent objects

- Light glass
- Light Plexiglass
- Transparent plastic foil

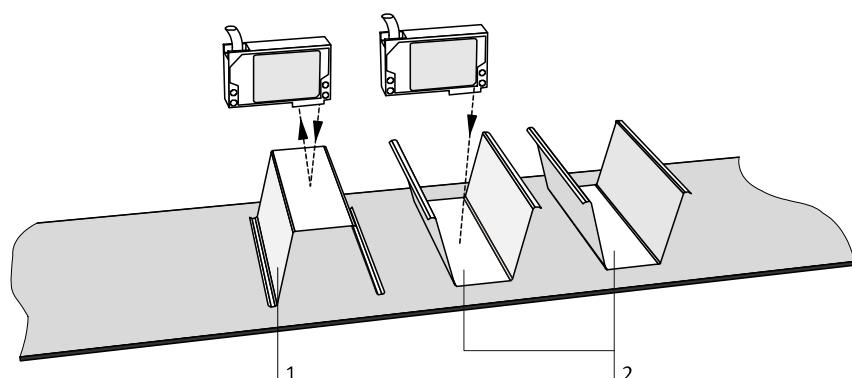
These materials generally have smooth, reflecting surfaces so that a diffuse sensor can be used. The prerequisite is that the object must be vertical to the beam direction.

### Objects of minimal reflection

- Matt black plastic
- Black rubber
- Dark materials with rough surface
- Dark textiles
- Burnished steel

Diffuse sensors do not react to the above materials or only at a very close distance.

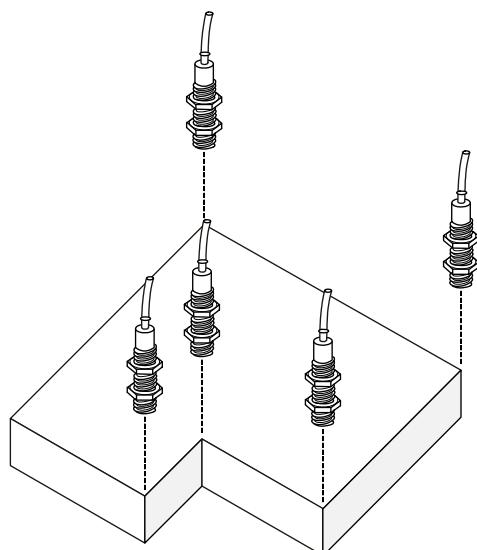
### Application examples



Monitoring of workpiece position via diffuse sensor

## 5. Optical proximity sensors

Careful adjustment of the sensitivity on the potentiometer is necessary, whereby tolerances owing to differences in material, contamination, etc. must be taken into consideration.



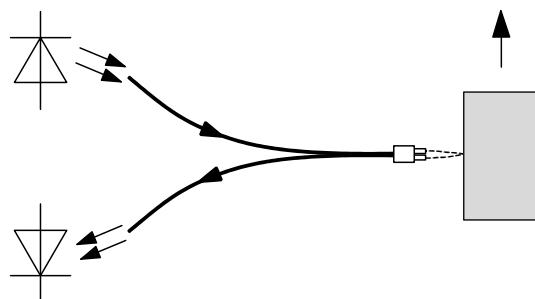
Identity and position checking with diffuse sensors

A connected controller checks whether all sensors respond (the proximity sensor outputs are linked via a logic "AND-operation). Diffuse sensors with fibre-optic cables are required for high accuracy and small distances from the object.

### Optical proximity sensors with fibre-optic cables

#### Mode of operation

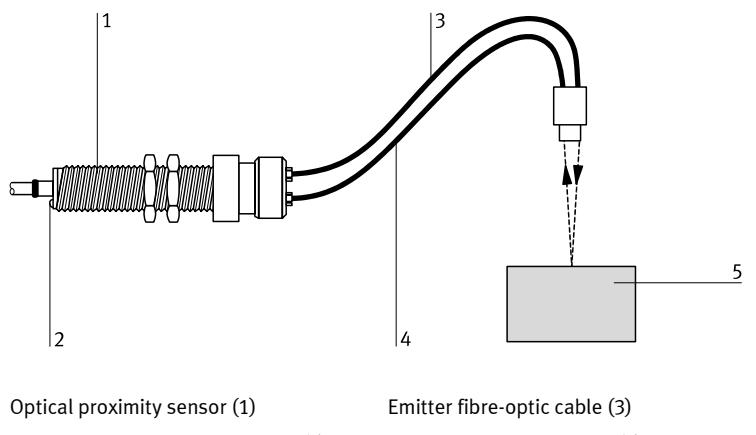
Optical proximity sensors with fibre-optic cable attachments are used if conventional devices require too much space. Similarly, the use of fibre-optic cables is of advantage in areas subject to explosion hazards. The position of small objects can be very precisely detected using fibre-optic cables.



Diffuse sensor with fibre-optic cable (principle)

## 5. Optical proximity sensors

Emitter and receiver fibre-optic cable combined into one unit.



Diffuse sensor with fibre-optic cables (sample design)

### Notes regarding use

Advantages of fibre-optic attachments on optical proximity sensors

- The detection of objects in difficult to access areas such as through holes.
- Possibilities of offset mounting of the proximity sensor housing (e.g. in hazardous environment: heat, water, interference radiation, explosion hazard).
- Precision detection of small objects.
- Possibility of movable configuration of sensing elements.

Advantages of polymer-fibre-optic cables

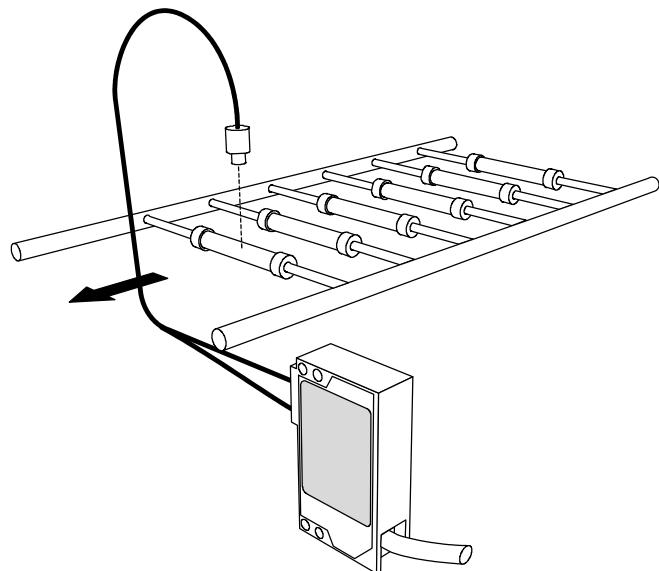
- Mechanically sturdier than glass fibre.
- Possibility of simply cutting to length the sensor-side end using a sharp knife.
- Cost effective.

Advantages of glass fibre fibre-optic cables

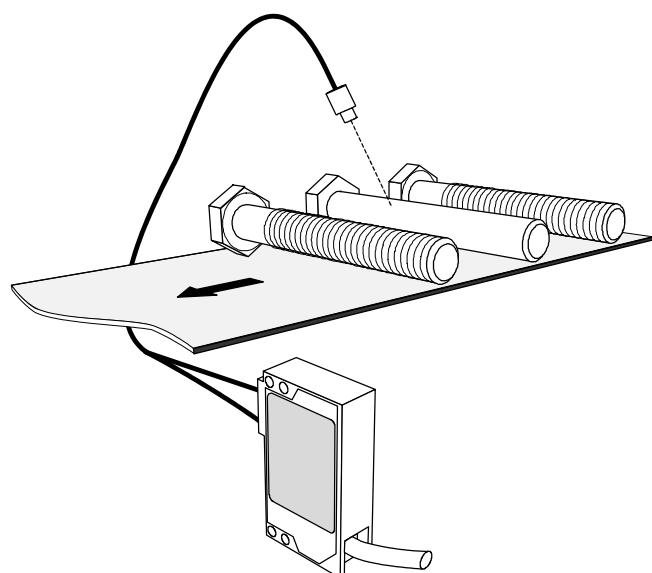
- Suitable for high temperature range.
- Minimal optical damping with long lengths and within close infrared range.
- High resistance to ageing.

## 5. Optical proximity sensors

### Application examples



Detection of small objects via diffuse sensor with fibre-optic cables



Thread checking

The threaded screws (diffusely) reflect sufficient light to cause the receiver to switch. The light emitted by the emitter is reflected away by the smooth surface.

## 6. Inductive sensor

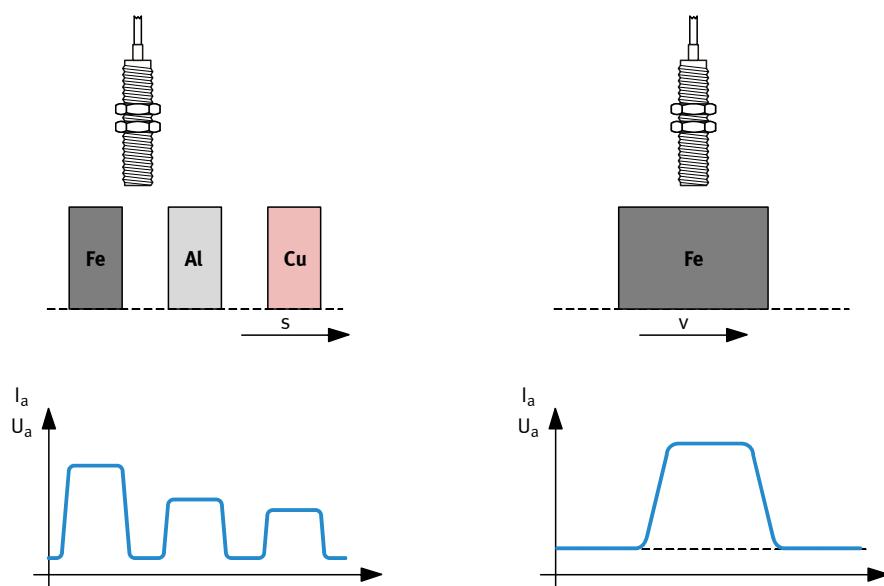
### Function

An inductive sensor consists of a coil with ferrite core. Together with a capacitor, the coil forms a resonant circuit thereby creating an electromagnetic field in front of the coil. If an electrically conductive object is present in this field, a voltage is generated within this, which in turn causes so-called eddy currents within the material. The required energy is drawn from the resonant circuit. This is also known as attenuation of the resonant circuit. The output stage is switched via the evaluation stage in accordance with programming. Depending on material or distance, analogue inductive sensors supply an analogue voltage or current intensity signal.

### Use

Inductive sensors are for example used

- to differentiation between various metallic materials
- to measure distances
- for monitoring assembly

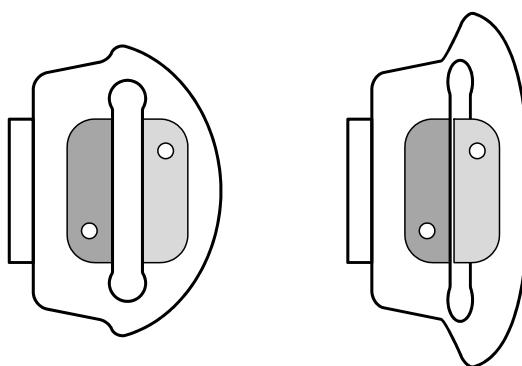


## 6. Inductive sensor

## 7. Sensitive edge, collision detection

### Mode of operation

The collision protection sensor of the Robotino® consists of a so-called sensitive edge. This sensitive edge consists of a polymer profile of different shape with integrated switching chamber. Two separate conductive areas are located within the chamber that short-circuit if pressure is applied to the sensitive edge thereby generating a signal for the evaluation unit. The sensitive edge on the Robotino® operates according to the quiescent current principle so that a potential cable fracture can be detected and the Robotino® stopped. Quiescent current is an electrical current which continuously flows within a circuit even if this is not active. If the current is not flowing then this is due to a cable fracture or a damaged sensitive edge.

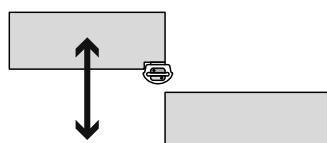


### Function of collision protection sensor

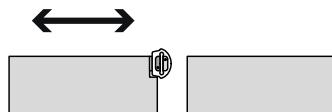
#### Areas of application

Sensitive edges are used predominantly in safety technology in order to eliminate injury to persons or damage to machinery or materials as a result of crushing or shearing. In medical engineering, they are used on diagnostic equipment, radiation devices and movable protective covers in order to protect patients and personnel. They are also in use on lift doors, bus doors and electrical skylights to protect fingers.

**Protection shear point**



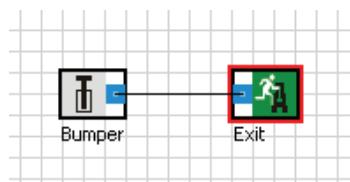
**Protection - crushing point**



## 7. Sensitive edge, collision detection

The bumper in  
Robotino® View

The bumper is available in Robotino® View in the Robotino® hardware folder and does not need to be parameterised. It supplies a 1-signal upon contact and is used mainly to stop the Robotino® in the event of a collision. For this purpose it is connected to an output from the Sequence Control folder. The program sequence is then interrupted in the event of a collision.

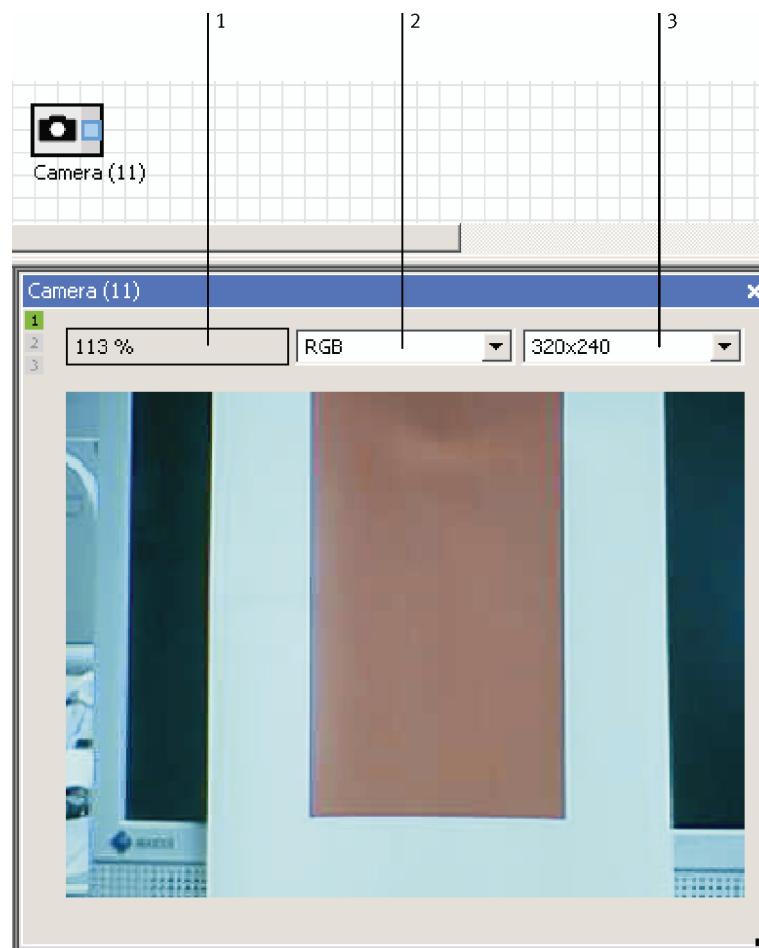


Function diagram for collision protection

## 8. Webcam

### Description

The function module Camera enables access to the images in the Robotino® webcam. The camera is connected to the command bridge via one of the two USB plugs. The resolution of the image can be adjusted by turning the lens.



(1) Picture size

(2) Colour space selection

(3) Resolution

### Function

The camera image is displayed in the function block dialogue. The size of the picture is changed by enlarging the dialogue window. The size of the display is shown in percentage (1) of the original size of the image.

You can select different colour spaces via the middle list box (2). The colour space corresponds to the natural representation of colours.

## 8. Webcam

The YcbCr colour space differentiates between the brightness and shades of the individual colours and represents these correspondingly. These are not represented in the customary way, although this colour model is less sensitive to changes in lighting.

The HSV colour space differentiates between the individual colours according to hue, saturation and value. Representation in relation to the RGB representation is much less defined. All this equally applies in the case of the HLS colour space, where the representation in terms of colour is however different. Instead of the value, the HLS colour space uses the luminance of a colour in order to determine it.

These different colour spaces can, with certain tasks or light conditions, produce better results. This must however be determined by means of experimenting.

The list box on the right (3) enables the selection of two possible resolutions (320 x 240, 640 x 480). In the case of colour recognition, the lower resolution is generally more useful, since the picture contains less different colour information in this case. This facilitates the recognition of associated colour areas for the software.

## 9. Generators

### Waveshape generator

A waveshape generator generates signal waveforms; different frequencies, voltages, etc. can be set.

### Generators in Robotino® View

This category contains numerous function blocks for the generation of signals.

Different types of generator are available in Robotino® View:

- Sine-wave generator
- Square-wave generator
- Saw-tooth voltage generator

Type selection	Outputs	Function (brief description)
Sine-wave generator	Sine curve	Sinusoidal waveform of a setpoint value, e.g. for speed controlling devices, voltage.
Square-wave generator	Square-wave pulse	Generates square-wave pulses of variable pulse width. The pulse width is set in ms and is independent of frequency.
Saw-tooth voltage generator	Saw-tooth curve	A setpoint value of saw-tooth waveform is generated.

The selected amplitude and duration (ms) of a pulse are represented.

### Symbol

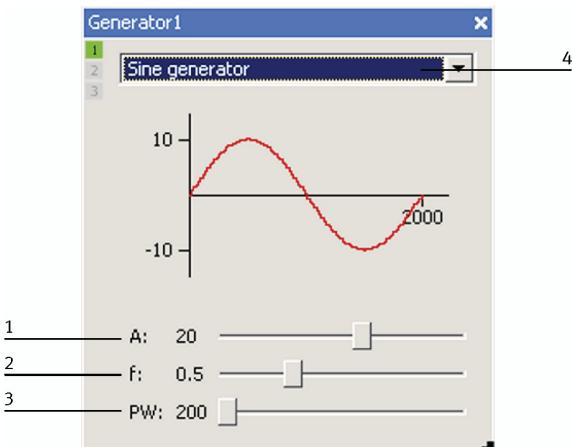
A double click onto the symbol opens the function block dialogue, which enables access to the internal parameters of the individual function blocks, in this case the generator, and also the setting of parameters.



	Output	Function (brief description)
1	The output depends of the type of generator.	Generates a setpoint value and acts as a source for additional blocks, e.g. motor.

## 9. Generators

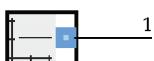
Generator dialogue box



	Element	Function (brief description)
1	A:	Setting of amplitude, range of values from 0.1 - 1000
2	F:	Frequency (=Hz). Max. 25 Hz. The frequency can be set in 8 steps, i.e.. 25 Hz, 10 Hz, 5 Hz, 2 Hz, 1 Hz, 0.5 Hz, 0.2 Hz and 0.1 Hz.
3	PW:	Signal width (ms) range of values can only be changed with the square-way generator and is dependent on the frequency.
4	Type selection	The generator type can be quickly changed by clicking onto the generator used in the function block diagram and then selecting the appropriate generator in the dialogue box.

Constant

Generation of a constant value.



	Output	Function (brief description)
1	Constant value	The output is a constant value. Several function blocks can be connected to this output, e.g. a connection to the motor and a connection to the oscilloscope (to display the setpoint value for motor control and also numerical constants for mathematical functions).

## 9. Generators

### Settings



	Parameter	Function (brief description)
1	Name	The name of the constant can be changed here.
2	Value	Setting of the output value. A value for the constant must be entered here. Several function blocks can be connected to this output, a connection to the motor and a connection to the oscilloscope (to display the setpoint value).

Meaning of generators and effect on the rotational speed of the motor			
Type	Symbol	Meaning	Rotational speed behaviour
Sine wave generator		The sine wave generator generates a speed setpoint value with sinusoidal waveform.	The wheel accelerates slowly up to the positive or negative maximum speed.
Square-wave generator		The square-wave generator generates a signal waveform which enables a fast increase or drop in amplitude.	The wheel accelerates quickly to the maximum speed. Then remains in this state. The duration of the wheel movement depends on the pulse width.
Saw-tooth voltage generator		The saw-tooth voltage generator generates a ramp-shaped output signal which is quickly reset to zero after a certain time.	The speed of the wheel increases linearly to the maximum value and stops suddenly after a time.
Constant		Supplies an input constant value.	The wheel rotates at a constant speed in one direction.

## 9. Generators

## 10. Oscilloscope

The oscilloscope enables you to represent the time characteristics of signals of one or several input channels. Signals characteristics are portrayed as a function of time.

An oscilloscope (memory oscilloscope) enables the representation and visual depiction of rapid electrical processes and is also used in the case of periodically recurring signals.

Voltage measurement using a general oscilloscope

Voltage can be measured via the inputs (channels). The measuring range can be adjusted via the switches.

Triggering

Triggering means to release a process.  
Correct triggering is necessary in order to obtain a fixed pattern for the measurement of the signal.  
Triggering is to be selected via a changeover switch for the respective channels.  
There is also an oscilloscope with an additional external triggering input.  
Once the process of representing the signal to be measured is completed, there is a pause until the measured signal is in the same direction and at the same level again.  
Triggering does not occur until then when the new signal is thus displayed again.  
This is effected with the help of the time-base generator.

The oscilloscope in Robotino® View

Three input values (channels) can be visualised with the oscilloscope. This function block has been modelled on an actual digital memory oscilloscope.

Measurement using the oscilloscope in Robotino® View

However, apart from only voltage, different variables in relation to time can be displayed and measured using the oscilloscope in Robotino® View. This is also effected via the inputs (channels). In Robotino® View the channels are designated with Ch1 – Ch3.

This enables you to read the cycle duration and the respective variable (speed, current intensity, voltage,...), depending on the actual or setpoint value applied at the oscilloscope inputs.

## 10. Oscilloscope

Parameter

Settings of channels (Ch 1 to 3):

Scaling of y-axis

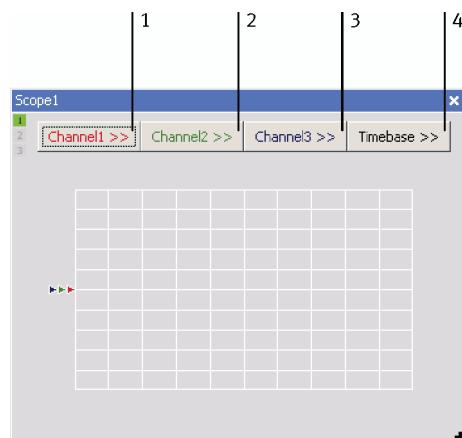
Zero position on y-axis

Definition of trigger condition: falling, rising edge

Triggering position

Time base

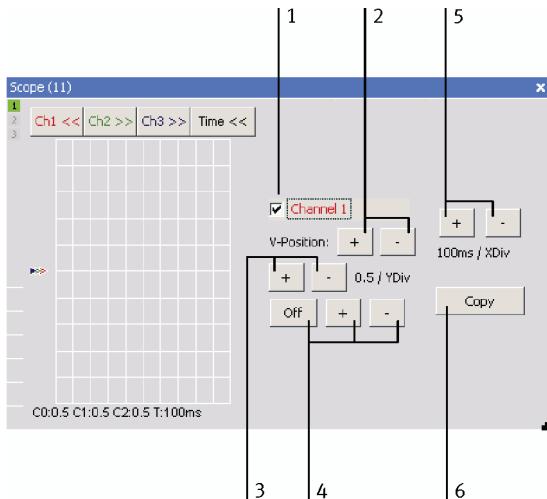
Oscilloscope dialogue box



	Element	Function (brief description)
1	Channel 1	Opens dialogue for settings of channels 1 to 3
2	Channel 2	
3	Channel 3	
4	Time base	Opens dialogue for setting of time base and copying of display
5	Display of zero position	Displays the zero position of the individual channels
6	Scaling	Scaling of the individual channels in y-direction (C0 to C2) and time resolution

Channel settings

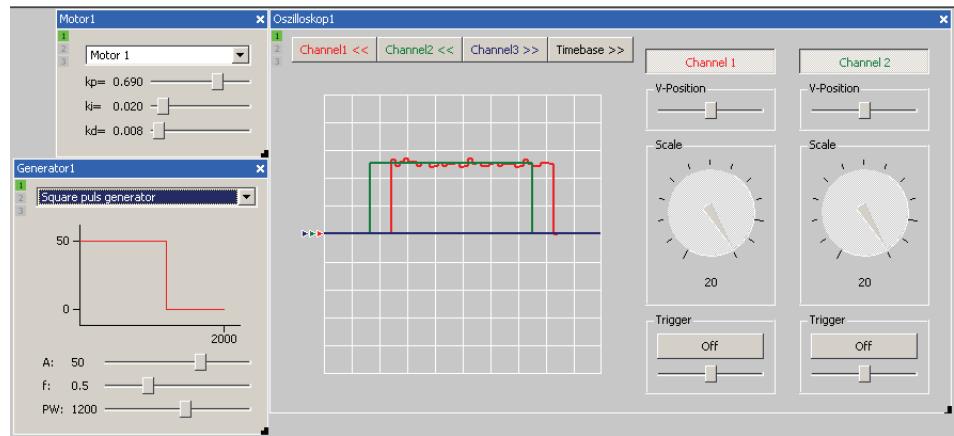
## 10. Oscilloscope



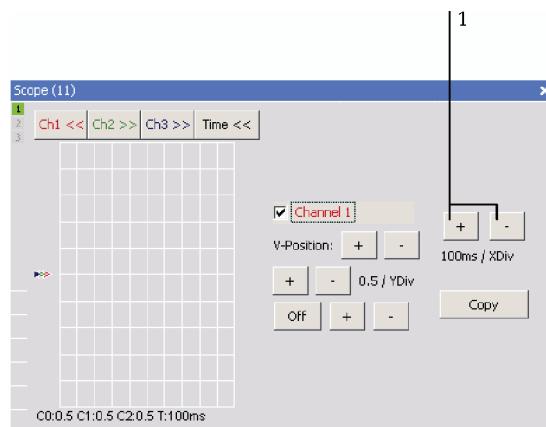
	<b>Element</b>	<b>Function (brief description)</b>
1	Channel (settings)	Channels can be activated or deactivated by clicking the symbols.
2	V-position	Sets the y-zero position of the respective channel.
3	Scale	Sets the Y-scaling (amplitude) of the respective channel. The displayed value corresponds to the numerical gap between the horizontal grid lines.  The unit of measurement depends of the variables to be measured, e.g. if speed is applied at input 1 of the oscilloscope, this means that the 1 box in the diagram corresponds to 0.5 s.
4	Trigger	Trigger position of the respective channel.  Definition of trigger conditions: off, falling, rising edge.  Displacement of the trigger position (display of trigger position to the right of the display)
5	Time base	Sets the time base for all channels. The displayed value indicates the time gap between the vertical grid lines.
6	Clipboard	Copies the display to the clipboard in the form of a bitmap for use in other programs.

## 10. Oscilloscope

Example for setting of channels



Time base settings



	Element	Function (brief description)
1	Scaling	Definition of time base for all channels. One box in x-direction corresponds to the set value. Minimum: 20 ms, Maximum: 500 s

## 11. Segmenter

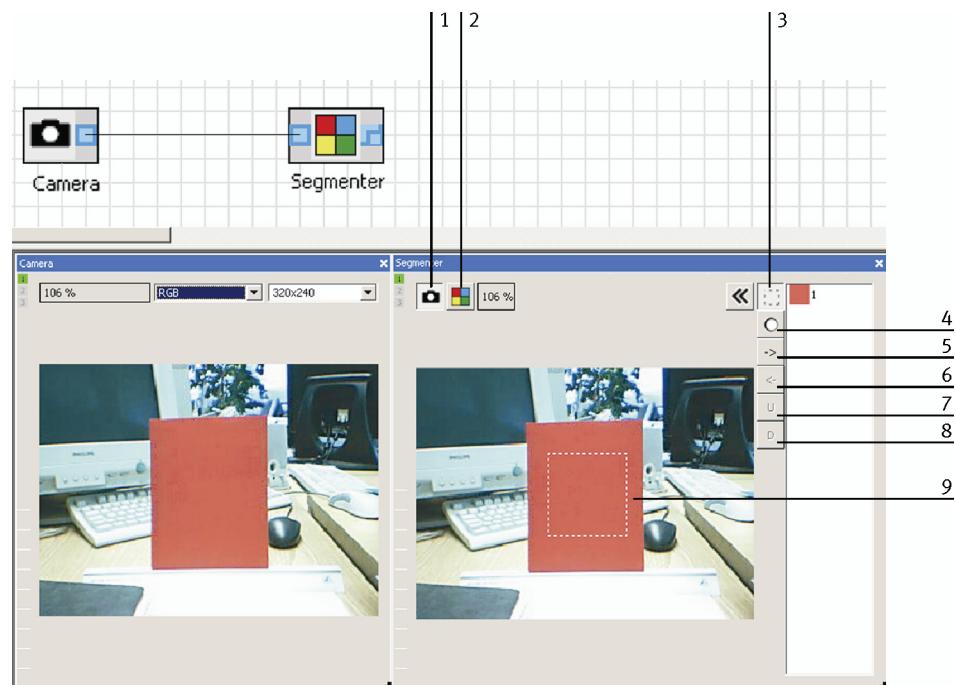
### Description

The function block Segmenter identifies areas of the same colour within an image and provides the possibility of defining required colours.

### Function

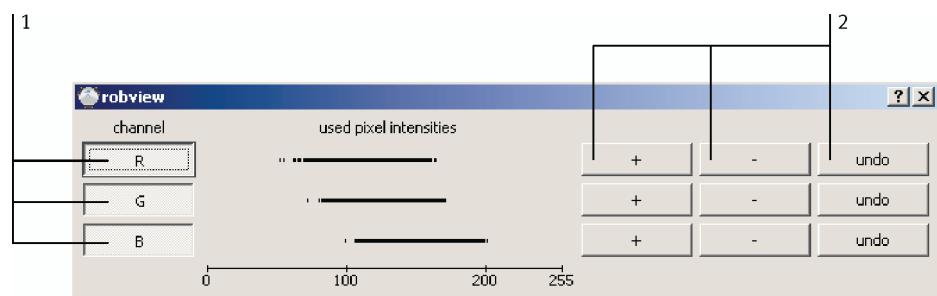
The segmenter processes an image supplied by the webcam. A fixed image is selected from the camera image by clicking onto the camera symbol (1) of the function block dialogue. Using the square (3) or circular (4) selection tool, the desired colour range is highlighted by marking the selection range (9) via the pressed down mouse button. The selected colour is transferred into the segment list by pressing the button. Several colour segments can be transferred to list or removed (6), or the sequence changed (7,8).

Button (2) enables you to fade in the area associated with a segment.



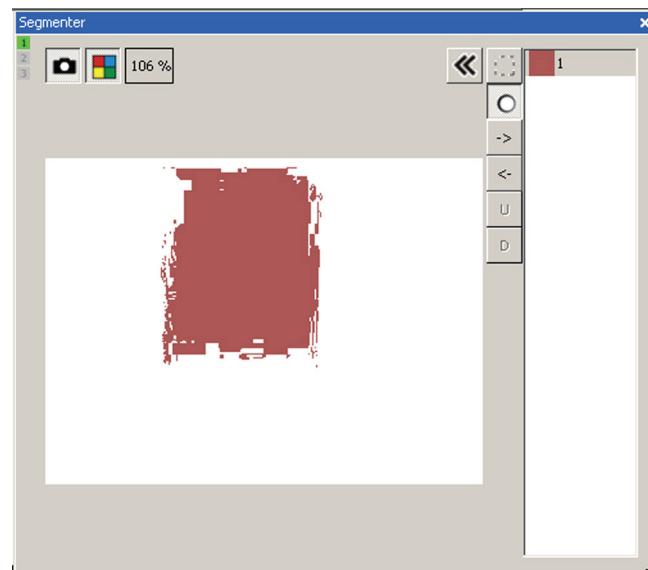
## 11. Segmenter

To optimise the captured area of a segment, it can be amended by changing the pixel intensity (2) or switching on or off individual image channels (1) by clicking onto the desired segment and using a further dialogue. The display of the captured segment must be switched on for these settings so that the outcome of the setting can be checked.



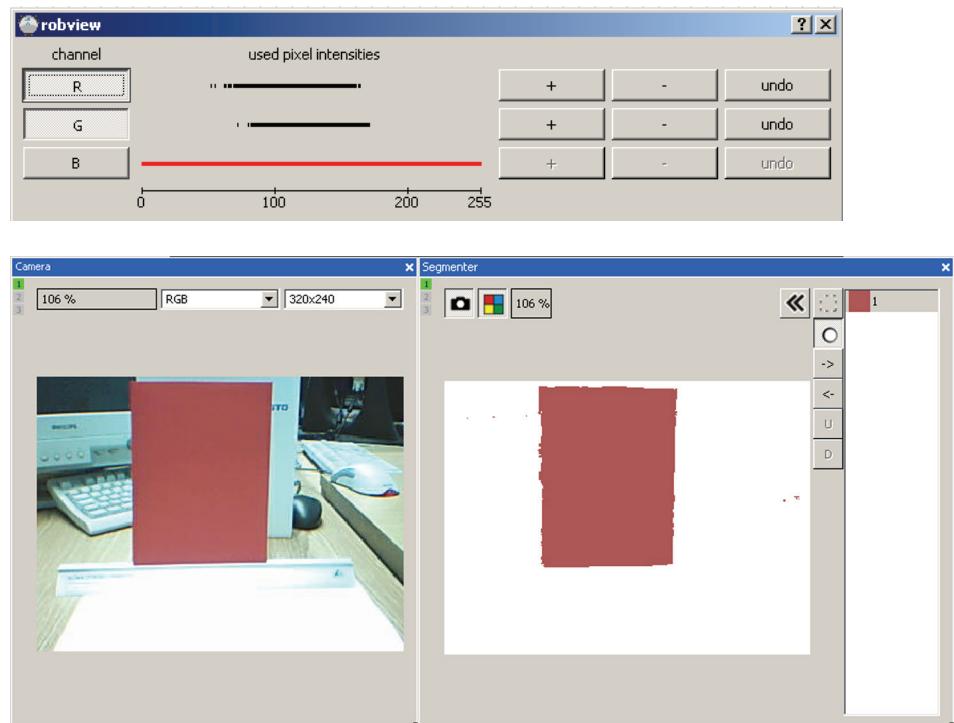
### Example

If the original setting from Robotino® View is used, a segment (object), as shown in the illustration below is identified.



## 11. Segmenter

If the channel for blue (B) is now switched off and the pixel intensities used for red (R) and green (G) are increased, the area realised is virtually identical to the object shown on the original camera image.



### Note

The selected segments and their precision adjustment can already be distorted to such an extent by minor changes in light conditions that the segments are difficult to or no longer recognised. It is therefore often necessary to realise the segmentation anew.



## 12. Segment extractor

Function	The function block Segment Extractor provides the position and size of a segment in a previously segmented image.
Inputs	<p>The Segment extractor has the following inputs:</p> <ul style="list-style-type: none"><li>• Segmented image This is where a camera image is copied with its segment information.</li><li>• Segment selection Selection of the segment for which the camera image is to be examined. This selection can also be effected in the function block dialogue. The segments are addressed via a number from 1 to x. Please note that this is based on the sequence in the segment selection box and not on the numbers allocated by the segmenter. The first segment is selected via the number 1.</li></ul>
Note	Only if this input is not occupied, can the segment be selected via the function block dialogue. The input therefore has priority over the function block dialogue.
Outputs	<p>X indicates the x-position of the focal point of the segment in relation to the centre of the image. Y indicates the y-position of the focal point of the segment in relation to the centre of the image.</p> <p>A positive value indicates a deviation to the right or upwards whereas a negative value indicates a deviation to the left or downwards. It is therefore very easy to navigate towards the centre of the image.</p> <p>The output area indicates the size of an associated segment within the image. The number output indicates the number of pixels of this area.</p> <p>The output found indicates whether an area of a minimum size has been found. The size of the minimum segment area is defined in the function block dialogue and is indicated in pixels. The input is 1 if a corresponding segment is found in the current image. If a sufficiently large associated area is not found, the value output is 0.</p>
Function block dialogue	<ul style="list-style-type: none"><li>• Segment selection Selection of the segment to be found if the input Segment Selection is not occupied.</li><li>• Minimum size Minimum number of pixels (e.g. 200) for one segment which is to be extracted.</li></ul> <p>The original image can be faded in or out by clicking onto the camera symbol.</p>



## 13. IF-Function

### Description

In most programming languages IF instructions are used. The IF instruction is available in the form of selection instruction – also known as instructions for program branching. Instructions can be executed as a function of a specified condition. This is used to be able to react to certain conditions or events.

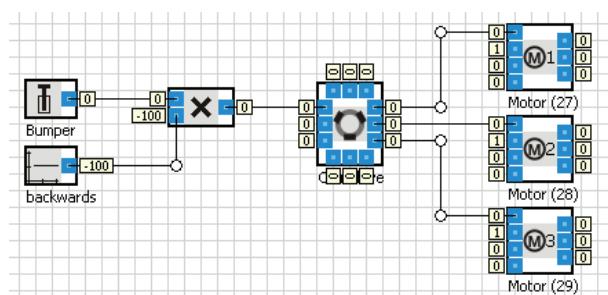
The simplest of IF instructions consists of an IF-THEN construct (simple branching). This is demonstrated on the example below

```
IF X < 0 THEN X := -X;  
END_IF;
```

If the condition following the keyword IF is true, then the instructions that follow the keyword THEN are executed. If the condition is not fulfilled, the instructions are not executed.

### IF function in Robotino® View

In Robotino® View, an IF function can be realised by means of simple mathematical multiplication.



The above function block diagram represents such an If function. If the bumper is not activated, the value of the constant Backwards is multiplied by zero, and the result is therefore also 0. However, if the bumper is activated, the value is multiplied by 1 and the result of this multiplication is -100. The omnidrive receives the value of the constant in the form of travel speed. This applies for as long as the bumper remains activated.

### Note

This form of an IF function can only be realised with binary signals since these only generate 1 or 0 signals.



## 14. Sign reversal of a value

### Description

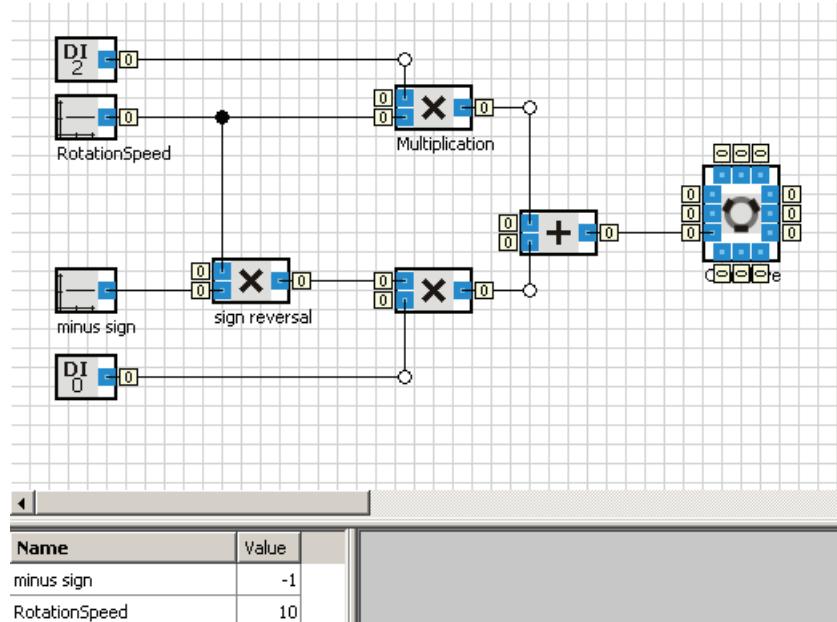
The sign reversal of a value is for instance required if a single speed or rotational direction constant is to be used simultaneously for the forward and backward movement of the Robotino® or for lefthand or righthand rotation. This is how the program change is simplified since only one constant has to be changed and incorrect entries by omitting the sign are eliminated. For example if the direction of rotation is to be exchanged –forward travel is to become backward travel - merely the sign in the associated constant needs to be changed.

### Sign reversal in Robotino® View

A sign reversal via multiplication by the value -1 is realised in Robotino® View.

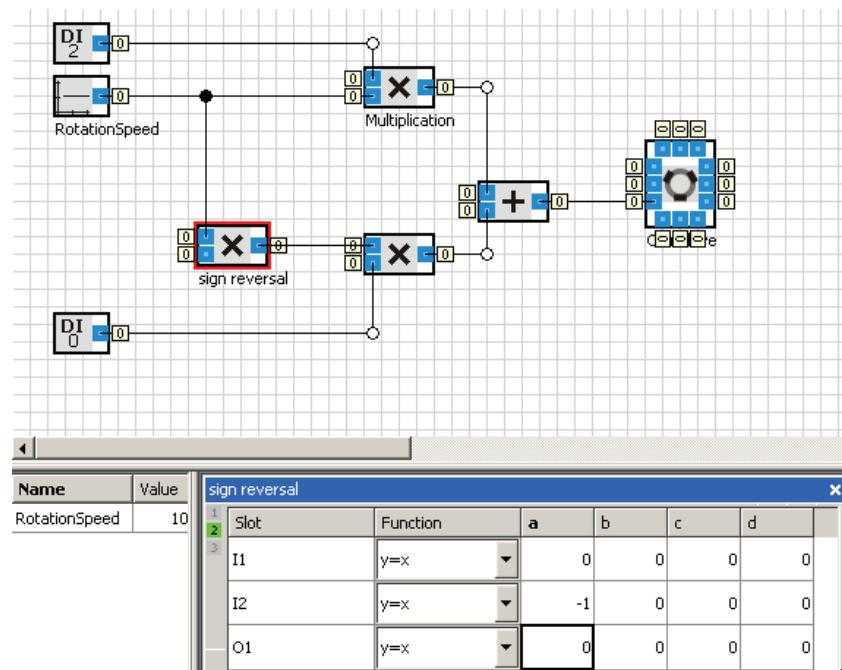
Two options are available for this.

In the case of the first variant the constants 'Rotational Speed' is multiplied by the constant 'Minus Sign', which has the value -1. If, in the example below, input DI0 supplies a 1-signal, the setpoint speed of the omnidrive is -10, which thus rotates to the right. If input DI2 supplies a 1-signal, the value of the constant 'Rotational Speed' remains unchanged; the omnidrive receives a positive value and the Robotino® rotates to the left. If the directions of rotation are to be exchanged, the value of the constant 'Rotational Speed' only has to receive a minus sign (-).



## 14. Sign reversal of a value

With the second variant, input E2 is set to -1 via the function block dialogue of the multiplication module 'Sign Reversal'. To do so, the function  $y = a$  needs to be selected in the function block dialogue for input E2 and the value a set to -1. This variant renders a constant superfluous and a program can therefore be more clearly structured.



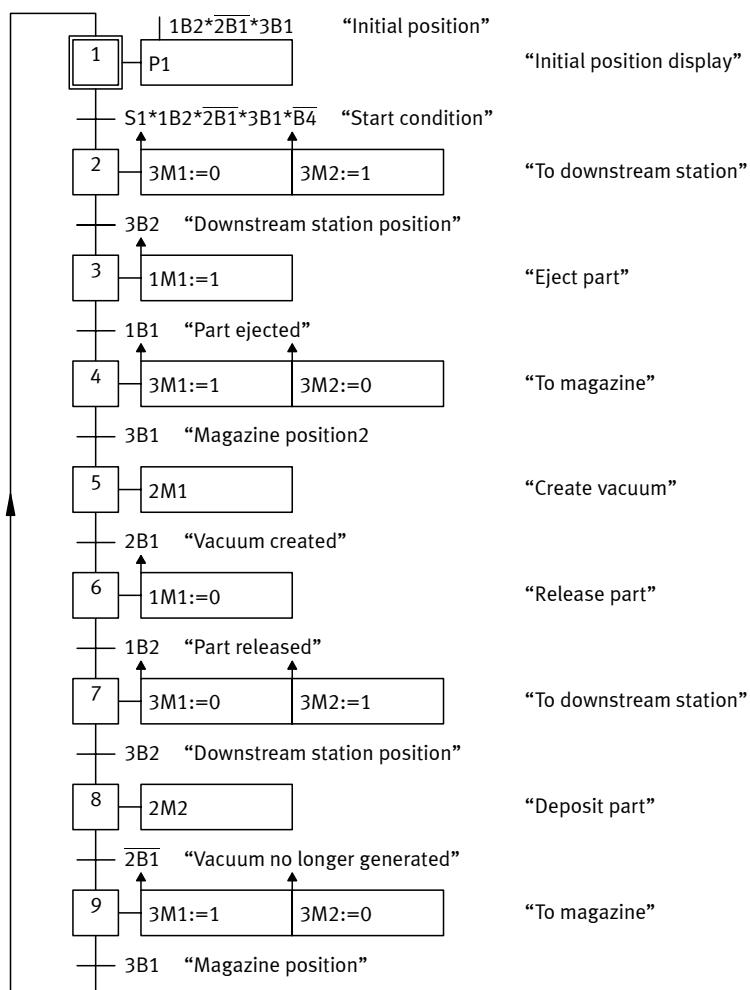
## 15. Sequence control

### General

Sequence controls are processes which are executed successively in several, clearly separate steps. The advancing from one step to the next depends on step enabling conditions. The main characteristic is that only ever one step is active.

The advantages of sequence control are as follows:

- A program divided into steps is more transparent and therefore easier to maintain and expand.
- Typical examples of sequence controls are machine control systems in the field of production technology or recipe control systems in process engineering.
- Function charts (or also flow diagrams) were introduced as tools to facilitate the analysis and representation of sequences.

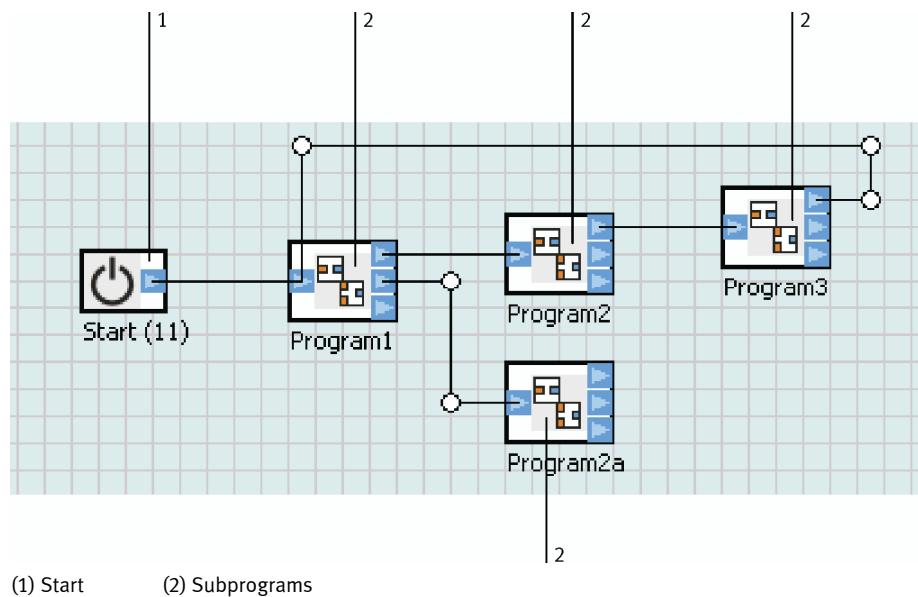


### Example of a sequence program

### Sequence control in Robotino® View

Robotino® View offers the possibility of linking different programs into a sequence program whereby the individual programs are treated as function blocks. Each subprogram must have one or several output function blocks. In addition a start function block is also required in order to start a sequence program.

Each subprogram has one input and three outputs (A, B, C). These correspond to the output function blocks used in the respective function block diagram. If the corresponding output function block is set in the subprogram, the subprogram is terminated and the corresponding output of the subprogram is set. If this output is connected to the input of a further subprogram, then this subprogram is started. In this way programs can be used as loops. Equally, branching can be realised between individual programs.



### Example of sequence control in Robotino® View

## **Part C – Solution**

### **Project 1**

Inspection of supplied components and commissioning  
of the Robotino<sup>®</sup> – solution \_\_\_\_\_ C-3

### **Project 2**

Linear travelling of a mobile robot system in any direction – solution \_\_\_\_\_ C-9

### **Project 3**

Linear travelling and positioning of a mobile robot system – solution \_\_\_\_\_ C-45

### **Project 4**

Path tracking of an automated guided vehicle system  
using two diffuse sensors – solution \_\_\_\_\_ C-69

### **Project 5**

Accurately positioned approach of a loading station – solution \_\_\_\_\_ C-91

### **Project 6**

Approaching an obstacle and maintaining a defined distance – solution \_\_\_\_\_ C-107

### **Project 7**

Circling a station and approaching various transfer positions – solution \_\_\_\_\_ C-115

### **Project 8**

Path tracking of an automated guided vehicle system  
using an analogue inductive sensor – solution \_\_\_\_\_ C-121

### **Project 9**

Determining the optimal motion behaviour – solution \_\_\_\_\_ C-137

### **Project 10**

Path tracking of an automated guided vehicle system  
with the help of a webcam – solution \_\_\_\_\_ A-149

### **Project 11**

Searching and approaching a coloured object  
with the help of a webcam – solution \_\_\_\_\_ A-161

#### Note

The exercises and solutions are based on Version 1.6 of Robotino<sup>®</sup> View.



## Project 1

### Inspection of supplied components and commissioning of the Robotino® – solution

Project 1: Inspection of supplied components and commissioning of the Robotino®	
Name:	Date:
Creating a check list	Sheet 1 of 1

- Create a check list for the visual inspection to check whether the system is complete.

Quantity	Description	ok
3	DC motor	ok
3	Gear unit with a transmission ratio of 16:1	ok
3	Toothed belt	ok
4	4* 12V storage batteries, of which 2 are enclosed	ok
1	Base plate with bumper	ok
9	Infrared distance sensor	ok
3	Incremental encoder, one per motor	ok
3	Multidirectional caster	ok
1	Collision protection sensor (bumper)	ok
1	Inductive proximity sensor, analogue	ok
2	Optical sensor, digital (diffuse sensor)	ok
1	Control unit with display, embedded controller and interfaces (=controller housing)	ok
1	Camera	ok

Date \_\_\_\_\_

Signature \_\_\_\_\_

## Project 1: Inspection of supplied components and commissioning of the Robotino® – solution

<b>Project 1: Inspection of supplied components and commissioning of the Robotino®</b>	
Name:	Date:
Commissioning of the Robotino®	Sheet 1 of 2

- Test the functionality of the components and document your findings.
- Via the control panel, check whether the system is signalling correctly whilst observing the LED on the control panel.

<b>Display</b>	<b>Description</b>
LED	Illuminated, operating status ON
ROBOTINO®	
172.26.1.1	PC104 IP address
V1.0	Software version

- Check the charge states of the storage batteries via the control panel display.

<b>Charge state of storage batteries</b>
You can read the charge states from the bar chart of the control panel. If only a few bars are displayed, the charge state is correspondingly low


<b>Idle state, no electrical malfunction</b>
Wheels are in the idle state and no electrical malfunction has occurred.

Project 1: Inspection of supplied components and commissioning of the Robotino® – solution

<b>Project 1: Inspection of supplied components and commissioning of the Robotino®</b>	
Name:	Date:
Commissioning of the Robotino®	Sheet 2 of 2

- Document your results on the worksheet.

Commissioning on the \_\_\_\_\_ (current date)

Commissioned by \_\_\_\_\_ A N Other

Power supply and status display \_\_\_\_\_ ok

Charge state of storage batteries \_\_\_\_\_ ok

Date \_\_\_\_\_ (current date)

Signature \_\_\_\_\_ A N Other

## Project 1: Inspection of supplied components and commissioning of the Robotino® – solution

<b>Project 1: Inspection of supplied components and commissioning of the Robotino®</b>	
Name:	Date:
Testing of motion behaviour	Sheet 1 of 2

- Test the motion behaviour of the Robotino® by testing the demo applications „forward“, „circle“, „rectangle“ and „roam“.
- Observe the motion behaviour in the jacked-up state and moving state.

<b>Description: Behaviour of „forward“ demo</b>	
Jacked-up Behaviour of casters	Front wheels move. Therefore M1 and M3 are active.
Travelling Motion behaviour Sensors Behaviour of casters	Moves straight ahead. Sensors: Collision protection sensor (bumper) is active upon contact with an obstacle. Front wheels move. Therefore M1 and M3 are active.
Additional observations	In order to travel forwards, M1 and M3 must move at identical speed in the “line of vision” of the Robotino®.

<b>Description: Behaviour of „circle“ demo</b>	
Jacked-up Behaviour of casters	All three wheels move. The wheels turn forwards once and backwards once at alternate intervals.
Travelling Motion behaviour Sensors Behaviour of casters	Orientation is maintained so that the Robotino® maintains its “line of vision”. Sensors: Collision protection sensor (bumper) All three wheels move.
Additional observations	All three wheels are required for circular travel. The wheels turn forward once and backwards once at alternate intervals.

# Project 1: Inspection of supplied components and commissioning of the Robotino® – solution

<b>Project 1: Inspection of supplied components and commissioning of the Robotino®</b>	
Name:	Date:
Testing motion behaviour	Sheet 2 of 2

<b>Description: Behaviour of „rectangle“ demo</b>	
Yescked-up Behaviour of casters	All wheels move. M2 and M3 turn one direction, M1 in the other. In order to move forward, M1 and M3 have to move in the line of vision of the Robotino®.
Travelling Motion behaviour Sensors Behaviour of casters	Orientation is maintained provided that the Robotino® always faces inwards. Sensors: Collision protection sensor (bumper) All wheels are rotating, M1 and M3 in the line of vision of the Robotino®.
Additional observations	Optimal: Linear travel of identical distance whereby the direction of the casters is changed thereby creating a rectangle. However it is possible that the rectangle may not be absolutely perfect and the distances are not completely identical.

<b>Description: Behaviour of „roam“ demo</b>	
Jacked-up Behaviour of casters	Front wheels move. M1 and M3 are active. In order to move forward, M1 and M3 have to move in the line of vision of the Robotino®. Upon actuation of the infrared distance sensors 1,2,9 : M1 changes the direction of movement, M3 becomes active thereby effecting evasive movement. This is effected by means of all wheels rotating more rapidly in the same direction.
Travelling Motion behaviour Sensors Behaviour of casters	Attempts to avoid approaching obstacles and effects evasive movement beforehand. Only the front infrared distance sensors are active, whereby the Robotino® detects obstacles within the range of infrared distance sensors 1,2,9. Evasive movement is effected by means of all wheels rotating more rapidly in the same direction. This results in evasive movement to the left.
Additional observations	Linear travel identical to „forward“ program. Owing to the activated infrared sensors, the Robotino® stops not just when it comes into contact with an obstacle, but avoids the obstacle beforehand. Evasive movement is effected by means of all wheels rotating more rapidly in the same direction. Evasive movement to the left.

Project 1: Inspection of supplied components and commissioning of the Robotino® – solution

## Project 2

### Linear travelling of a mobile robot system in any direction – solution

Project 2: Linear travelling of a mobile robot system in any direction	
Name:	Date:
Actuation of motor, forward movement of robot system	Sheet 1 of 5

- Move the Robotino® forward to see which motors need to be actuated. Observe the direction of rotation and rotational speed of the multidirectional wheels.
  - Create a function block diagram in Robotino® View whereby the Robotino® travels forward.
  - Explain why the direction of rotation of the motors must be the reverse in the case of straight ahead travel.
  - Create a collision protection function.
  - Test different speeds.
- 
- Move the Robotino® forward to see which motors need to be actuated. Observe the direction of rotation of the multidirectional wheels.
- Position the Robotino® in front of you and move it forward, observing the direction of rotation and rotational speed of the multidirectional wheels.

#### Direction of rotation and rotational speed of the multidirectional wheels

Direction of rotation of multidirectional wheels: Wheels on motors M1 and M3 both roll forward. (You will find the motor designations in the technical documentation)

The rotational speed is identical.

- Answer the following questions.

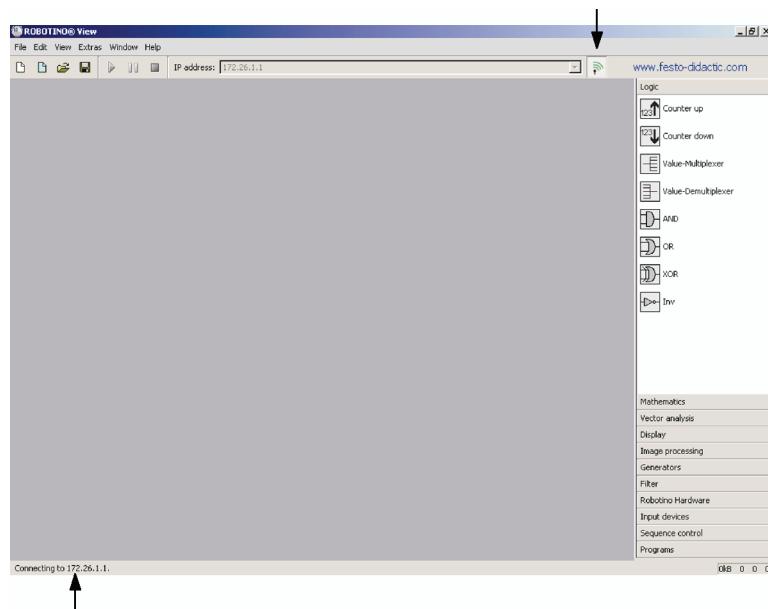
#### Which motors must be actuated in order for the Robotino® to travel forward?

M1 and M3 must be actuated.

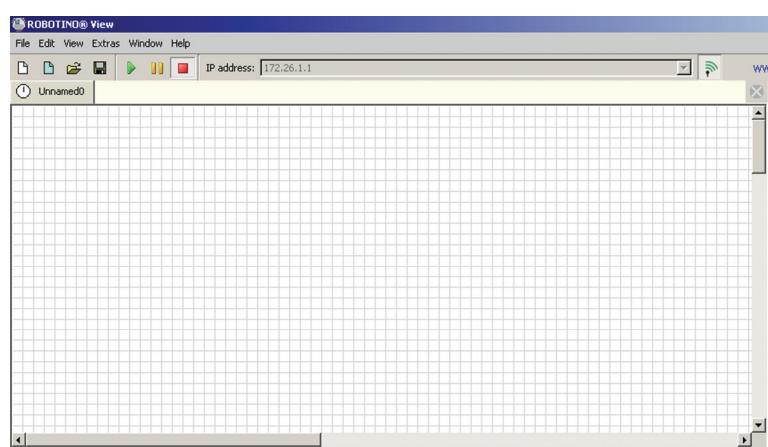
## Project 2: Linear travelling of a mobile robot system in any direction – solution

Project 2: Linear travelling of a mobile robot system in any direction	
Name:	Date:
Actuation of motor, forward travel of robot system	Sheet 2 of 5

- Create a function block diagram in Robotino® View whereby the Robotino® travels forward.
- Jacked-up the system so that the wheels are freely movable.
- Connect the Robotino® to the power supply and switch on.
- Start up Robotino® View and establish a connection between the Robotino® and Robotino® View (see technical documentation).



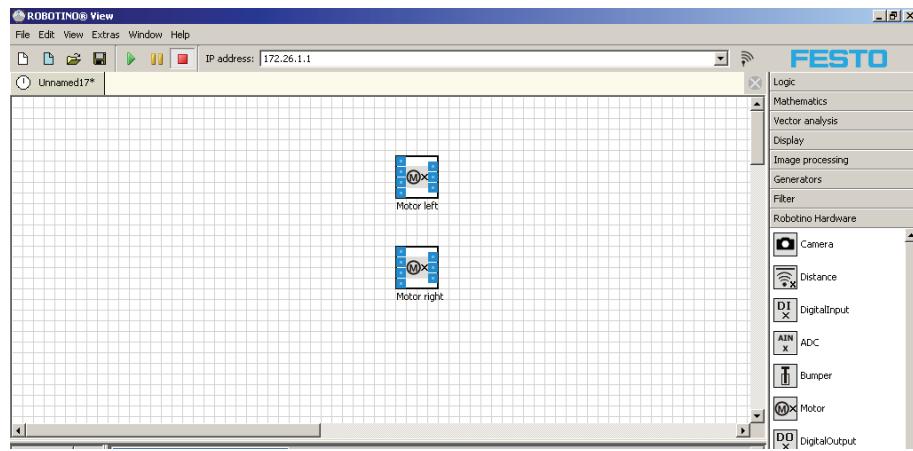
- Open a blank function block diagram in Robotino® View.



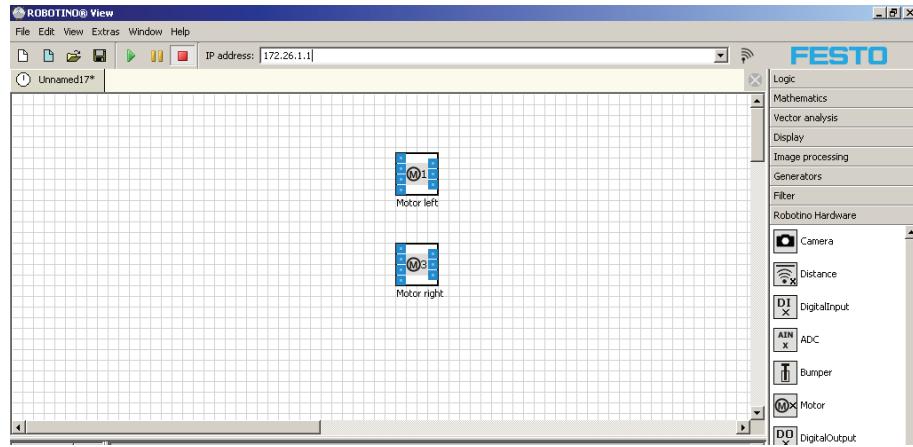
## Project 2: Linear travelling of a mobile robot system in any direction – solution

Project 2: :Linear travelling of a mobile robot system in any direction	
Name:	Date:
Program - actuation of motor, forward travel of robot system	Sheet 3 of 5

- Drag two motor function blocks (function block library: Robotino® hardware → motor) into the function block diagram.



- Allocate to each motor function block exactly one motor of the robot system (see technical documentation).
- In the function block diagram of the motor function block, name the motors “MotorFrontLeft” and “MotorFrontRight“.



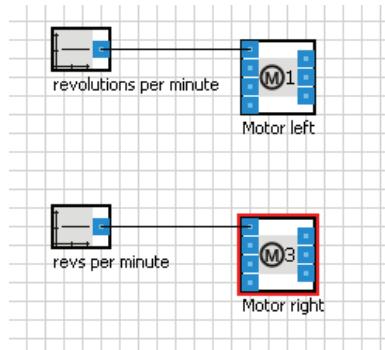
## Project 2: Linear travelling of a mobile robot system in any direction – solution

Project 2: Linear travelling of a mobile robot system in any direction	
Name:	Date:
Program – actuation of motor, forward travel of robot system	Sheet 4 of 5

- Specify a constant speed for both motors.

### Note

Add two constants to the function block diagram (function block library → generators), and connect each of these to the input „setpoint speed“ of the two motors.  
Designate the constants accordingly with “SpeedLeft” and “SpeedRight”.



- Start the applications by clicking onto the Start symbol . Change the values of the two constants. Note that the setpoint speed is measured in rpm.
- Observe the behaviour of the robot system both in the Jacked-up state and also travelling state.

### Note

Should be motors run erratically and this impairs the travelling of the Robotino®, then check the standard setting of the PID closed-loop motor controller in the function block dialogue of the motors:  
 $K_p = 0.9$   
 $K_i = 0.01$   
 $K_d = 0.0$

The technical documentation specifies that the Robotino® travels forward if it moves linearly to the line of vision of the camera.

- Select the values of the constants, e.g..  
SpeedLeft = -1500 [rpm]  
SpeedRight = 1500 [rpm]  
the Robotino® then moves forward.

<b>Project 2: Linear travelling of a mobile robot system in any direction</b>	
Name:	Date:
Program – actuation of motor, forward travel of robot system	Sheet 5 of 5

- Explain why the direction of rotation of the motors must be the reverse in the case of straight ahead travel.

#### **Explanation of reverse direction of rotation of motors**

If M1 and M3 are actuated in the same direction, the Robotino® travels in a circle, since the wheels are not in parallel with one another but configured at an angle of 120°. Therefore:

Motor M1 must be actuated to the left, i.e. a negative value is assigned to the constant.

Motor M3 must be actuated to the right, i.e. a positive value must be assigned to the constant.

Important: Both constants require the same value for the rotational speed, one positive and one negative, otherwise the Robotino® does not travel forward.

## Project 2: Linear travelling of a mobile robot system in any direction – solution

<b>Project 2: Linear travelling of a mobile robot system in any direction</b>	
Name:	Date:
Collision protection function	Sheet 1 of 1

- Create a collision protection function.
- Test different speeds.

<b>Project 2: Linear travelling of a mobile robot system in any direction</b>	
Name:	Date:
Description of the bumper function	Sheet 1 of 1

**Briefly describe the bumper of the Robotino®.**

The bumper of the Robotino® consists of a plastic body with embedded contacts which is fitted around the chassis. If the plastic body is pressed the bumper supplies a signal.

**Describe the function of the bumper in Robotino® View.**

Actuation of the bumper causes a 1-signal to be generated.

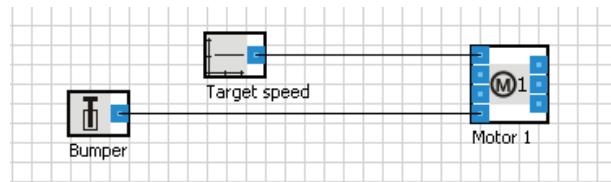
## Project 2: Linear travelling of a mobile robot system in any direction – solution

Project 2: Linear travelling of a mobile robot system in any direction	
Name:	Date:
Programming of a stop function	Sheet 1 of 1

- Create a function block diagram with the following function in order to explain the behaviour of the bumper:  
A Robotino® motor is to be actuated such that it stops if the bumper is touched and re-starts when it is released.
- Which components are required for this control program?

Quantity	Component
1	Constant
1	Motor
1	Bumper

- Create the control program in Robotino® View and save it.



<b>Project 2: Linear travelling of a mobile robot system in any direction</b>	
Name:	Date:
Testing and evaluation of the function block diagram	Sheet 1 of 1

- Jack-up the Robotino® and test your function block diagram.

<b>Test</b>	
Touch bumper, motor stops	o.k.
Release bumper, motor re-starts	o.k.

**What do you need to effect in a control program with travel functions if you want to use this function as collision protection?**

The bumper must be connected to all motor brakes used in a function block diagram.

**Evaluate the function of this solution in respect of the required collision protection function.**

In the case of flexible and lightweight obstacles, the bumper will revert to its original shape and the Robotino® will approach the obstacle again.

If the obstacle is removed, the Robotino® may continue travelling unchecked.

The required disconnection of all Robotino® functions is not guaranteed.

## Project 2: Linear travelling of a mobile robot system in any direction – solution

<b>Project 2: Linear travelling of a mobile robot system in any direction</b>	
Name:	Date:
Realisation of the collision protection function	Sheet 1 of 1

- Which function in Robotino® View fulfils the function of a control program termination?



Sequence program > Output A, B or C.

- Create a termination function based on the bumper function and save the program.

### Note

Integrate this collision protection function into all previously created function block diagrams to increase their safety.



<b>Project 2: Linear travelling of a mobile robot system in any direction</b>	
Name:	Date:
Backward motion program	Sheet 1 of 2

- Create and test a program which enables the Robotino® to travel backwards.
- Answer the questions regarding the components, constants and degrees of freedom.
  
- Create a program that enables the Robotino® to travel backwards.
  - Use the same components here as those in the program for forward travel.
  - Answer the following questions.

**Which motors must be actuated and how in order for the Robotino® to travel backwards?**

In order for M1 to move in the direction of travel of the Robotino®, the constant must be assigned a positive value (direction of rotation to the right). In order for M3 to move in the direction of travel, the constant must be assigned a negative value (direction of travel to the left). This generates the different directions of rotation.

**Which components are required for this?**

Two constants, two motors (M1 and M3)

## Project 2: Linear travelling of a mobile robot system in any direction – solution

<b>Project 2: Linear travelling of a mobile robot system in any direction</b>	
Name:	Date:
Backward motion program	Sheet 2 of 2

SpeedLeft = 1500 [rpm]

SpeedRight = -1500 [rpm]

**What differences can you identify in the values of the two constants during the forward and backward motion program?**

The programs are identical in structure; however the signs of the constant values must be changed.

**State which degrees of freedom are enabled in the motion behaviour of the Robotino® subsequent to the programs being executed:**

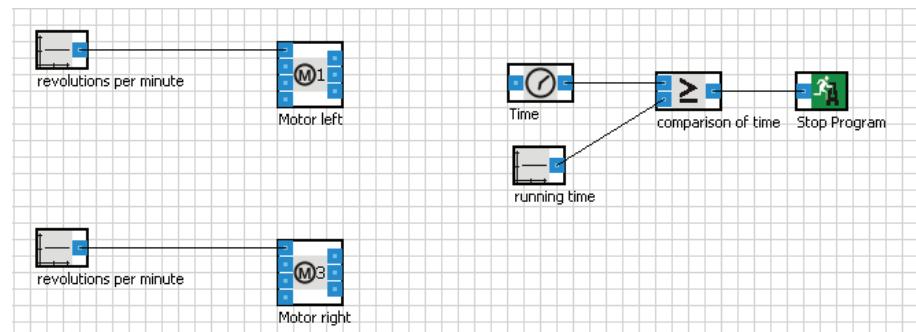
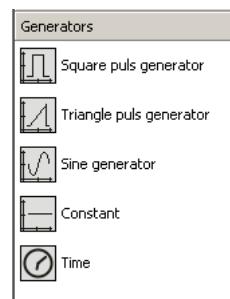
One degree of freedom, x (forward, backward motion along the x-axis)

Project 2: Linear travelling of a mobile robot system in any direction	
Name:	Date:
Operating sequence	Sheet 1 of 3

- Create a program sequence whereby the Robotino® travels forwards for 5 seconds, waits for 2 seconds and then travels backwards for 5 seconds.
- Test the sequence.
  
- Create a program sequence whereby the Robotino® travels forwards for 5 seconds, waits for 2 seconds and then travels backwards for 5 seconds.
  
- Create the individual programs and operating sequence.
- First create a program whereby the Robotino® travels forwards for five seconds.

Note

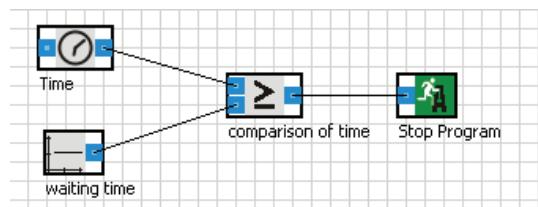
Consider which of the modules you require from the function block library to realise the timing.



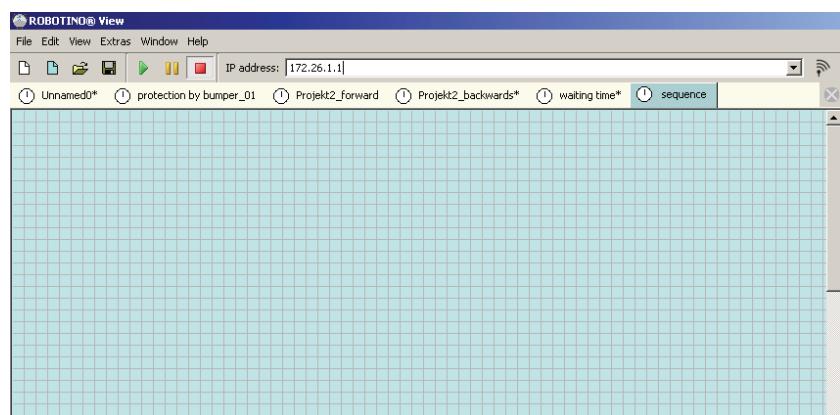
## Project 2: Linear travelling of a mobile robot system in any direction – solution

Project 2: Linear travelling of a mobile robot system in any direction	
Name:	Date:
Operating sequence	Sheet 2 of 3

- Assign the following values to the constants:  
 SpeedLeft = -1500  
 SpeedRight = 1500  
 Travel time = 5000 [ms]
- Create a program accordingly whereby the Robotino® travels backwards for five seconds.
- Create a program which generates a wait time of 2 seconds and name it **waitingtime.rvm**.



- Waiting time** constant = 2000
- Download the three programs **project2\_forwards.rvm**, **project2\_backwards.rvm** and **waitingtime.rvm**.
- Start a new sequence control program and name it **project2\_sequence**.



Project 2: Linear travelling of a mobile robot system in any direction	
Name:	Date:
Operating sequence	Sheet 3 of 3

- First enter the three programs from the library **programs**.
- Connect output A from the forward motion program to the input of the waiting time program. Then connect output A of the waiting time program to the backward motion program.

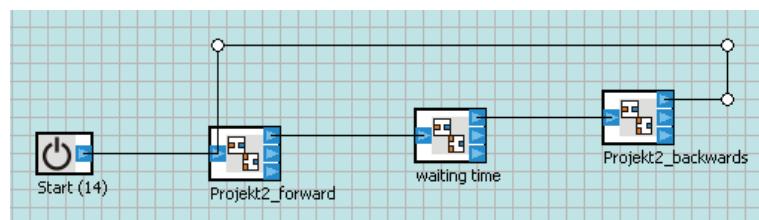
These connections effect the following:

When the forward motion program is completed, the waiting time program is started and, after 2 seconds have expired, the backward motion program is started.

- You then connect output A of the backwards program to the input of the forward program.

You now still need to establish which program is to be started when the sequence program is started.

- To do so, select the start module from the sequence control library and connect it to the input of the forward program.

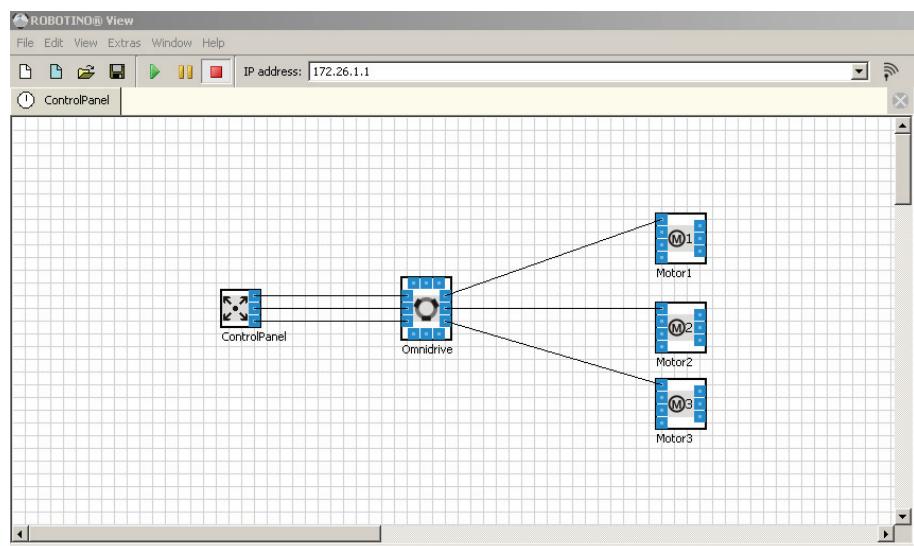


- Test the sequence.

## Project 2: Linear travelling of a mobile robot system in any direction – solution

Project 2: Linear travelling of a mobile robot system in any direction	
Name:	Date:
Omnidirectional drive, all directions using „control field“	Sheet 1 of 2

- Let the Robotino® travel in all possible directions by using the “omnidrive” function block in the motion program and the input device „control field“.
- Answer the questions regarding the motion behaviour, degrees of freedom and familiarise yourself with the “omnidrive” function block.
- Let the Robotino® travel in all possible directions by using the “omnidrive” function block in the motion program and the input device “control field”.
- Drag a control field from the list of input devices into a new function block diagram.
- Connect the outputs of the control field to the inputs (x,y,omega) of the “omnidrive” function block.
- Connect the outputs of the “omnidrive” to the inputs for the setpoint speed of the three motors.



<b>Project 2: Linear travelling of a mobile robot system in any direction</b>	
Name:	Date:
Omnidirectional drive, all directions using „control field“	Sheet 2 of 2

- Discover via Help about the operation of the control field and then start the program.
- Display the data (Ctrl-D or “Display Data“ under View) and note the displayed motor values.

**Describe the motion behaviour and state the possible degrees of freedom you have observed  
(flexibility of movement of bodies)**

Three degrees of freedom x (forward, backward) y, (lateral), rotation around the z-axis. Diagonal at an angle to the x-axis.

- Familiarise yourself regarding the omnidirectional drive in the Theory section and the “omnidrive” function block via the Robotino® View Help.

<b>Project 2: Linear travelling of a mobile robot system in any direction</b>	
Name:	Date:
Omnidrive, forward travel	Sheet 1 of 3

- Create a program using the function blocks “omnidrive”, three “motors” and one „constant“ whereby the Robotino® travels forwards and backwards using the same speed and orientation.
- Check whether the setpoint speed for the rear motor is constant = 0, see exercise on motor actuation without “omnidrive”.
- Using experiments, determine the forward speed in [mm/s] required to obtain the setpoint values of -1500 or 1500 [rpm] for the two front motors.
  
- Using the function blocks “omnidrive”, three “motors” and one “constant”, create a program whereby the Robotino® travels forwards and backwards using the same speed and orientation.

Note

The „omnidrive“ function block is contained in the hardware functions library and describes a kinematic model of the Robotino®. The inputs on the “lefthand” side are

- Setpoint speed in x-direction [mm/s]
- Setpoint speed in y-direction [mm/s]
- Setpoint speed in [degree/s]

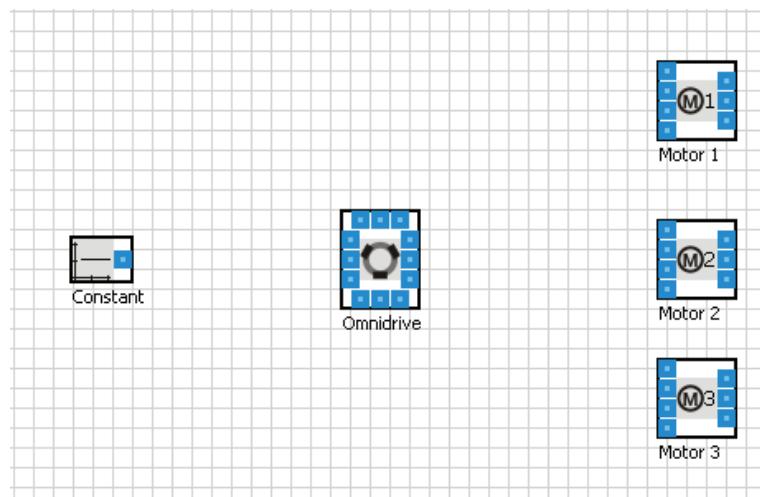
The module supplies as outputs the setpoint speeds in [revolutions per minute] for the three motors.

The coordinate system is selected such that the positive x-axis corresponds to the forward direction for the Robotino®.

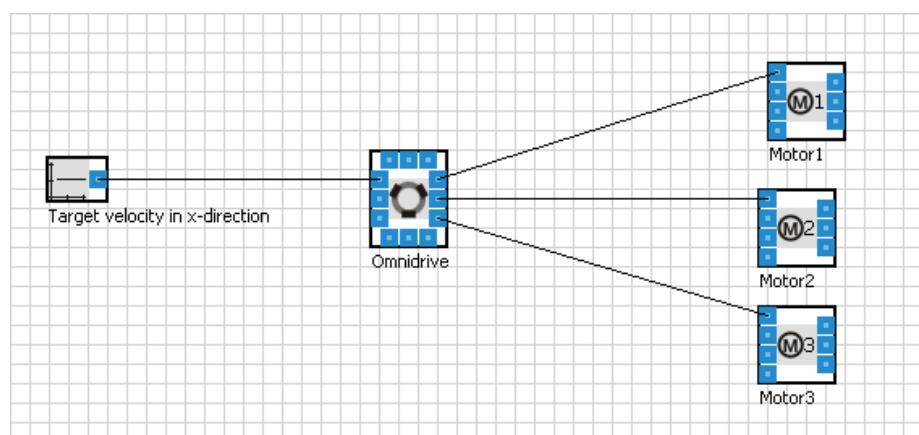
## Project 2: Linear travelling of a mobile robot system in any direction – solution

Project 2: Linear travelling of a mobile robot system in any direction	
Name:	Date:
Omnidrive, forward travel	Sheet 2 of 3

- Jack up the system again and open a new function block diagram in Robotino® View.
- Create a new program using the components omnidrive, three motors and one constant.



- Connect and define the elements.



Project 2: Linear travelling of a mobile robot system in any direction	
Name:	Date:
Omnidrive, forward travel	Sheet 3 of 3

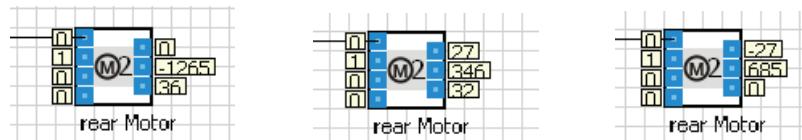
Note If the value of the constant is 100, this results in a setpoint speed of 100 [mm/s], i.e. 0.36 [km/h].

- Switch on the data display (Ctrl-D or “Display Data” under View). Note the motor values in particular.
- Check whether the setpoint speed for the rear motor is constant = 0, see exercise on motor actuation without omnidrive.

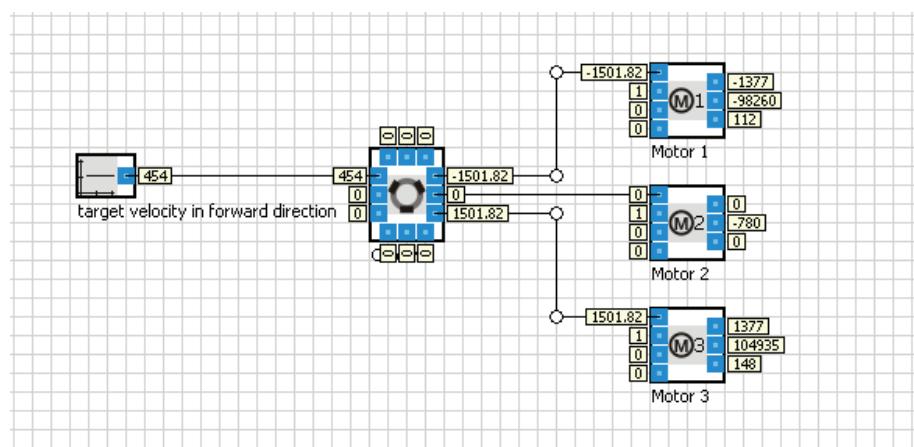
#### Conclusion

The actual speed is not equal to zero and generally fluctuates between 27 and -27. Furthermore, it is noted that the motor current also is not equal to zero and generally fluctuates between 0 and 40 mA.

#### Example



- Using experiments, determine the forward speed in [mm/s], required to obtain the setpoint speed values of -1500 or 1500 [rpm] for the two front motors.



Answer: vx setpoint 454 [mm/s]

<b>Project 2: Linear travelling of a mobile robot system in any direction</b>	
Name:	Date:
Omnidrive, backward travel	Sheet 1 of 1

- A backward movement is obtained if you specify a negative setpoint speed in the x-direction. Put this statement to the test.

**What needs to be changed and how to enable the Robotino® to travel backwards using the „omnidrive“?**

The constant must have a negative value in order for the Robotino® to travel in the negative x-direction.

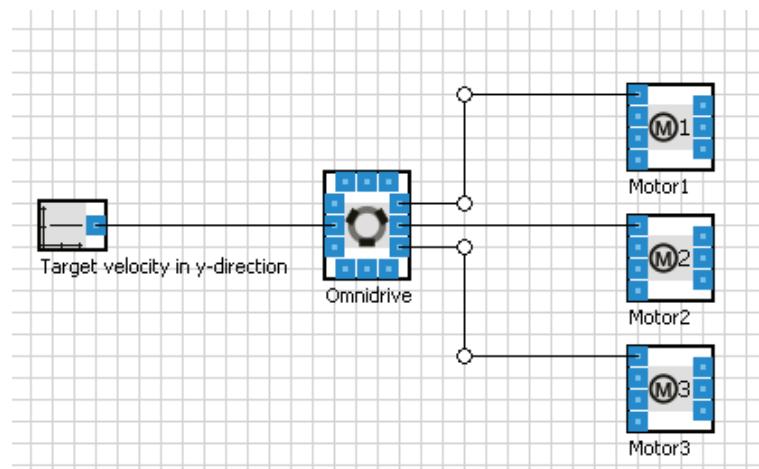
Name	Value
Target velocit	-100

- Test the „backward travel“ program.

## Project 2: Linear travelling of a mobile robot system in any direction – solution

<b>Project 2: Linear travelling of a mobile robot system in any direction</b>	
Name:	Date:
Omnidrive, lateral travel	Sheet 1 of 2

- Modify the program without adding any further function blocks so that the Robotino® travels laterally to the right or left using the same speed and orientation.
- Observe the behaviour of the multidirectional wheels.
- Describe the design and the options of the multidirectional wheels and characteristics of the omnidirectional drive.
  
- Modify the program without adding any further function blocks so that the Robotino® travels laterally to the right or left using the same speed and orientation.
  
- Jack up the system again.
- Connect the components that ensure that the Robotino® travels laterally.



<b>Project 2: Linear travelling of a mobile robot system in any direction</b>	
Name:	Date:
Omnidrive, lateral travel	Sheet 2 of 2

- Start-up the program, once in the jacked-up state and once in the mobile state.
- Answer the following questions.

**How many degrees of freedom are required for lateral travel?**

An additional degree of freedom is required for lateral travel, i.e. the movement along the y-axis.

- Observe the behaviour of the multidirectional wheels.

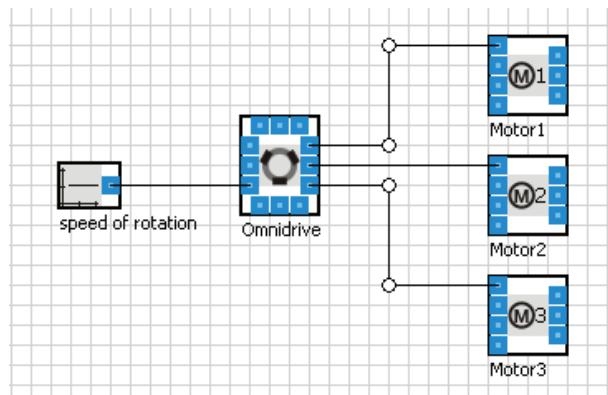
**Observation**

M1 and M3 both rotate in positive direction at identical speed. M2 rotates in negative direction and approx. twice the speed.

## Project 2: Linear travelling of a mobile robot system in any direction – solution

Project 2: Linear travelling of a mobile robot system in any direction	
Name:	Date:
Omnidrive, circular travel	Sheet 1 of 2

- Modify the program without adding any further function blocks so that the Robotino® rotates around the central axis.



<b>Project 2: Linear travelling of a mobile robot system in any direction</b>	
Name:	Date:
Omnidrive, circular travel	Sheet 2 of 2

- Connect the constant to the Omega setpoint connection of the „omnidrive“ and observe the behaviour.
- Answer the questions regarding the possible degrees of freedom.

**Behaviour**

The Robotino® rotates around its own axis.

- Answer the following questions regarding the degrees of freedom.

**How many degrees of freedom were required and why?**

The Robotino® requires one degree of freedom as it rotates around the z-axis.

## Project 2: Linear travelling of a mobile robot system in any direction – solution

<b>Project 2: Linear travelling of a mobile robot system in any direction</b>	
Name:	Date:
Omnidirectional drive, multidirectional wheels	Sheet 1 of 1

- Describe the design and the possibilities of the multidirectional wheels and the characteristics of an omnidirectional drive.

Description
In the case of a symmetrical drive, the wheels (multidirectional wheels) are each fitted at 120°. The special wheels of the Robotino® cannot generate a movement in the rolling direction (i.e. vertical to the main axis of rotation) like standard wheels but, thanks to the transverse casters, can be moved vertical to the rolling direction.  All directions of travel are possible.

- Characteristics of an omnidirectional drive.

Advantage	Disadvantage
Travel is possible in all directions in the plane. Uses the respective degrees of freedom.	Increases energy consumption in the case of forward travel
Turning circle = Zero	

## Project 2: Linear travelling of a mobile robot system in any direction – solution

Project 2: Linear travelling of a mobile robot system in any direction	
Name:	Date:
Travel at 45° to forward direction	Sheet 1 of 1

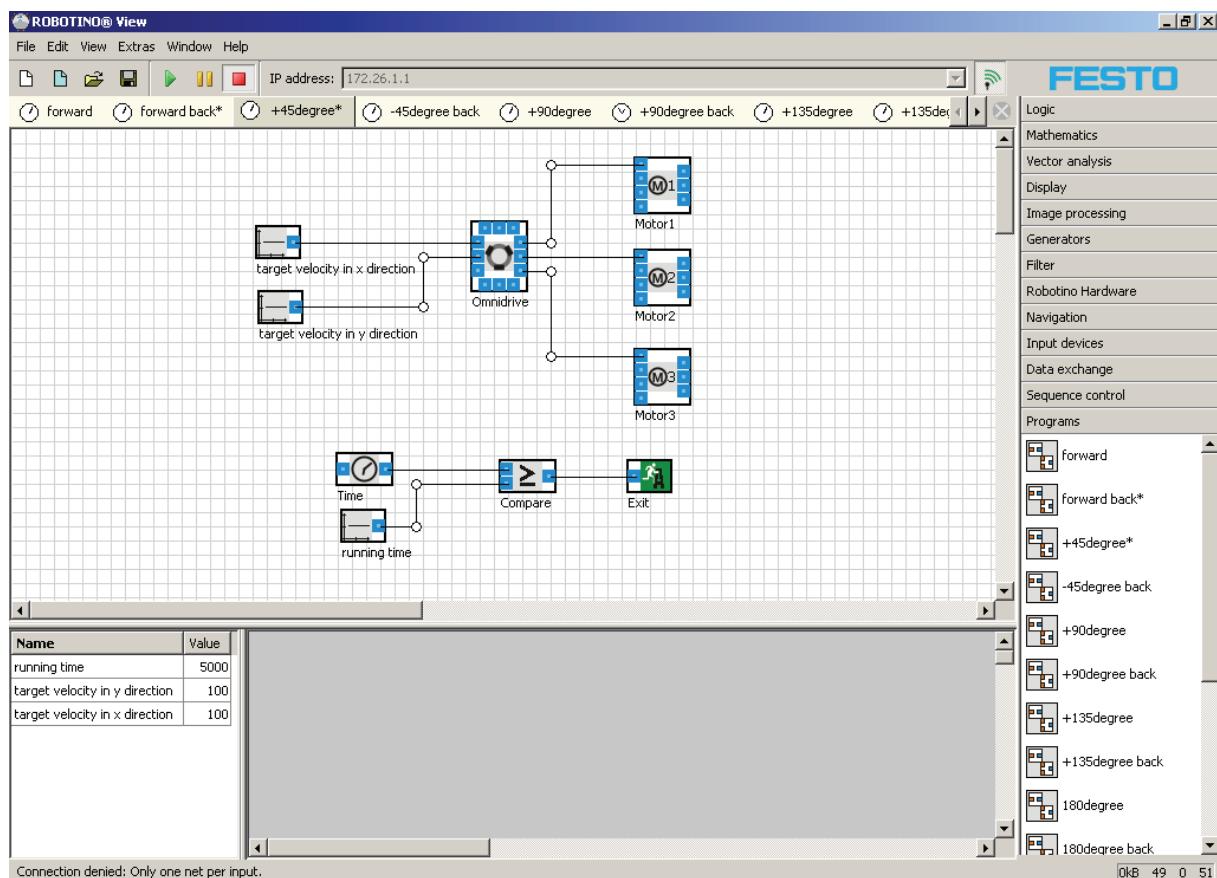
- Select a fixed Robotino® orientation. Modify the program by adding a further constant so that the Robotino® travels at 45° to the forward direction using the same speed and orientation.
- Consider how you can realise travel at 45° to the forward direction taking into account the direction of travel and speed.

### Note

Please see the positional sketch in the problem description.

#### Initial consideration: Speed

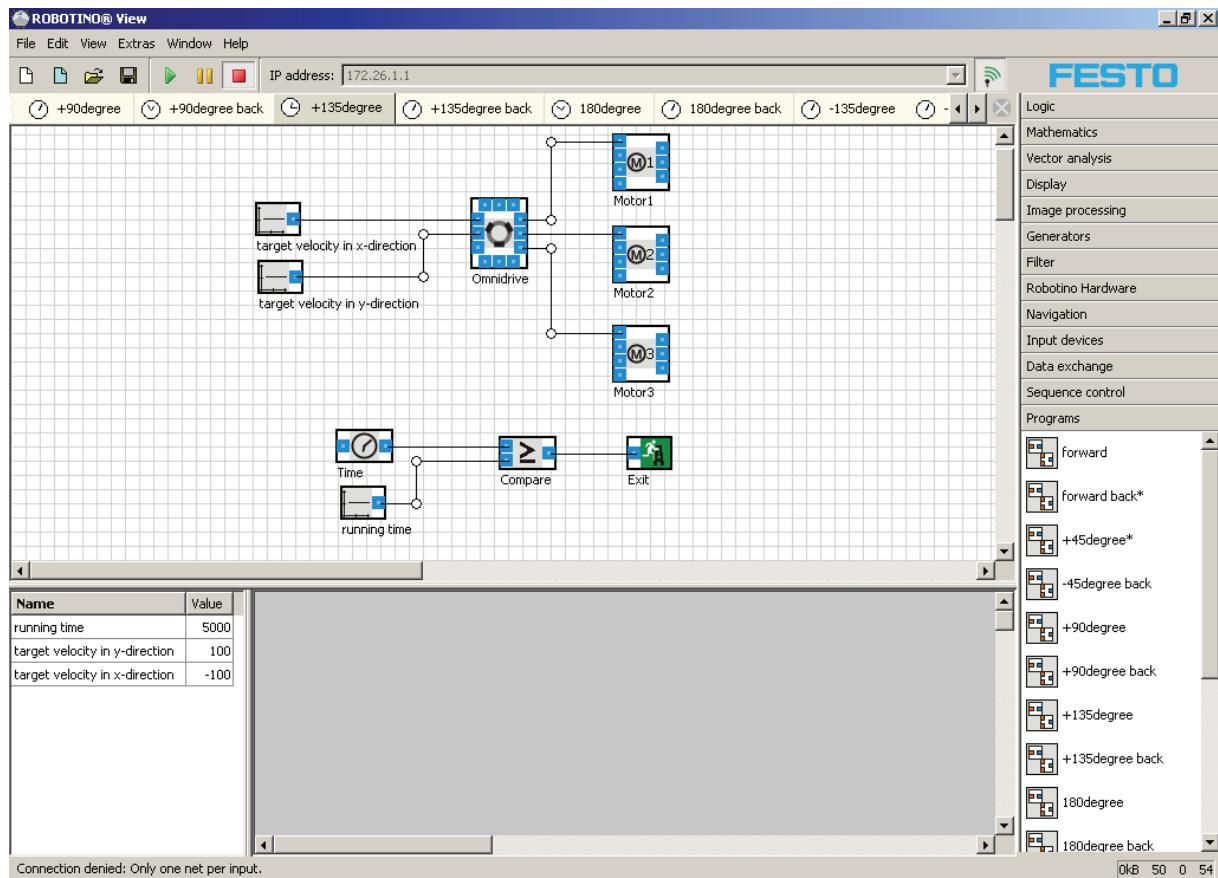
In x- and y-direction, the same speed must be applied to the “omnimodule”, whereby the sign is to be observed. The Robotino® consequently travels at identical speed straight ahead and in y-direction, thereby generating motion at 45° to the forward direction.



## Project 2: Linear travelling of a mobile robot system in any direction – solution

Project 2: Linear travelling of a mobile robot system in any direction	
Name:	Date:
Travel at 135 ° to the forward direction	Sheet 1 of 1

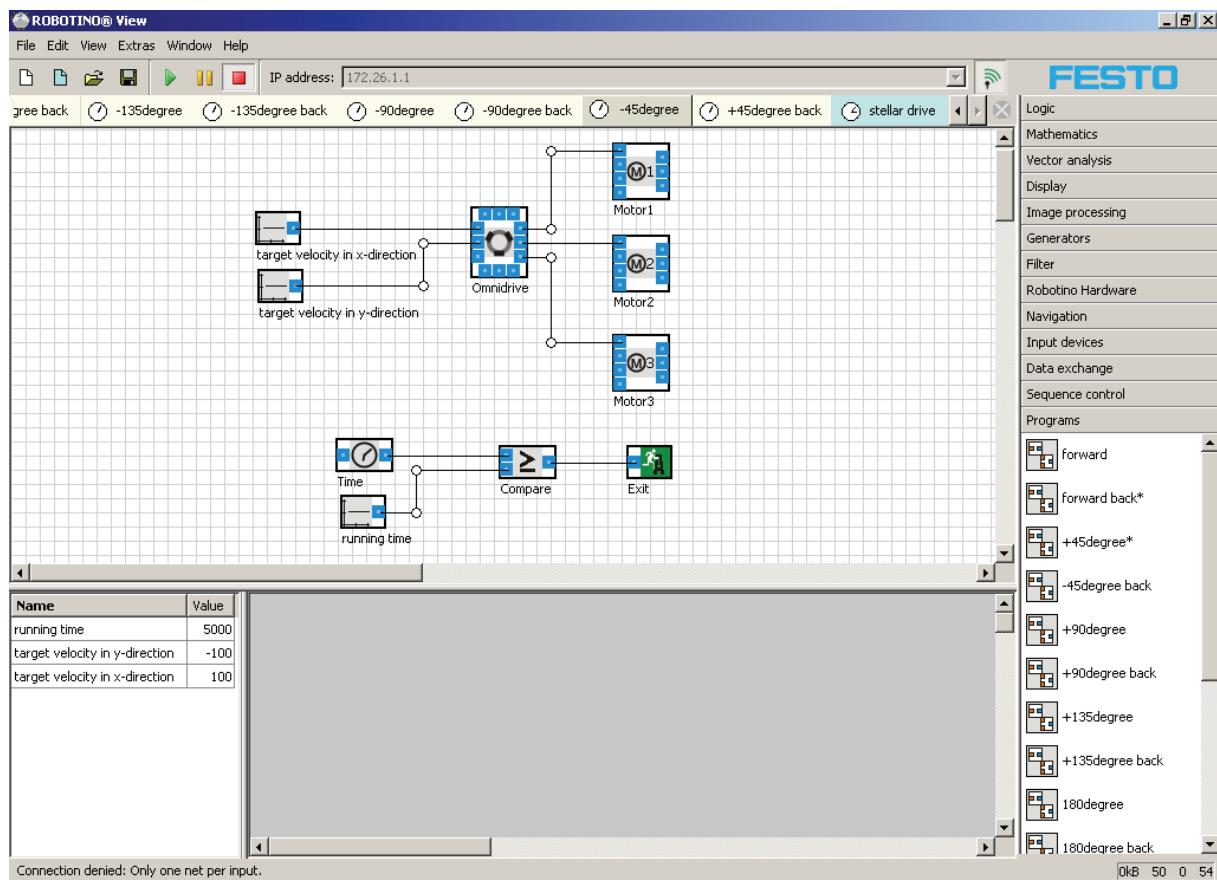
- Modify the program without adding further function blocks so that the Robotino® travels at 135° to the forward direction using the same speed and orientation.



## Project 2: Linear travelling of a mobile robot system in any direction – solution

Project 2: Linear travelling of a mobile robot system in any direction	
Name:	Date:
Travel at -45 ° to the forward direction	Sheet 1 of 1

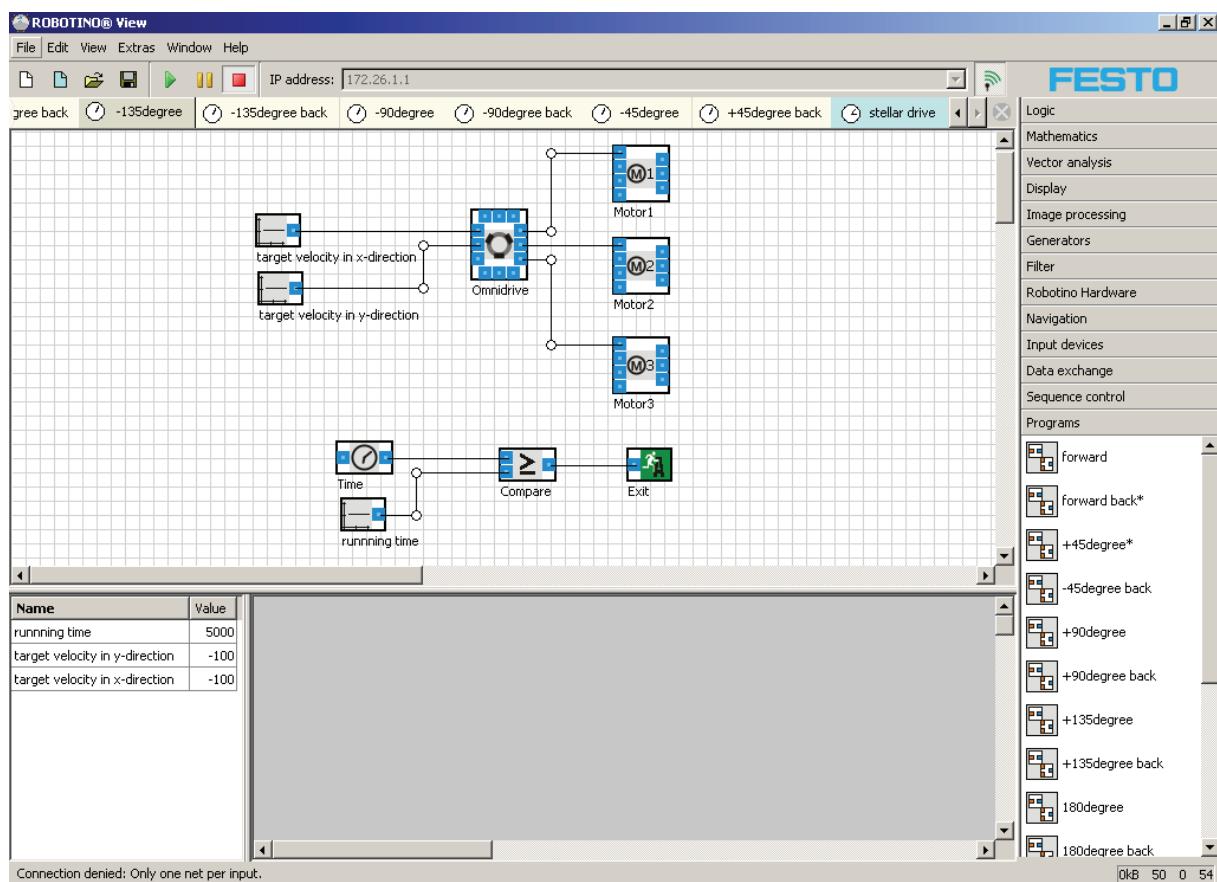
- Modify the program without adding further function blocks so that the Robotino® travels at -45° to the forward direction using the same speed and orientation.



## Project 2: Linear travelling of a mobile robot system in any direction – solution

Project 2: Linear travelling of a mobile robot system in any direction	
Name:	Date:
Travel at -135 ° to the forward direction	Sheet 1 of 1

- Modify the program without adding further function blocks so that the Robotino® travels at -135° to the forward direction using the same speed and orientation.



## Project 2: Linear travelling of a mobile robot system in any direction – solution

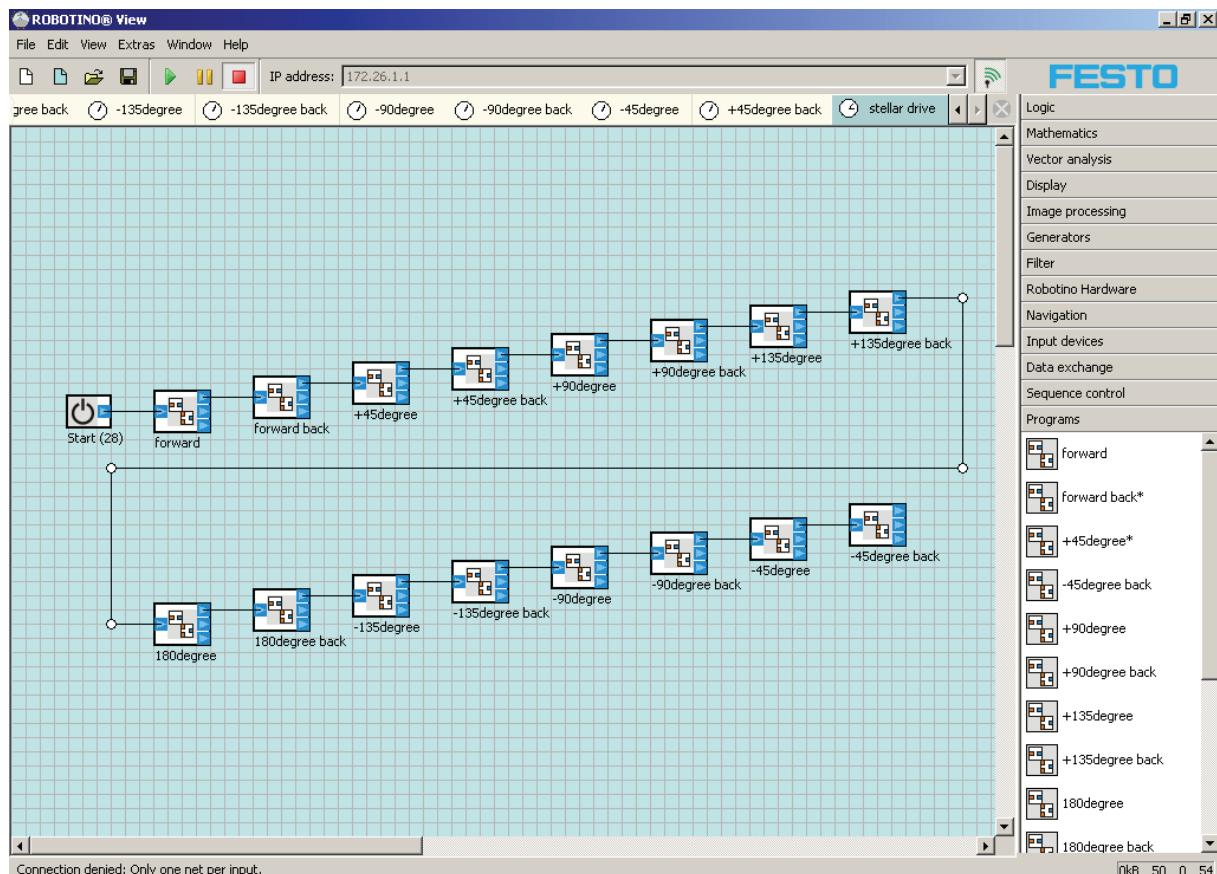
Project 2: Linear travelling of a mobile robot system in any direction	
Name:	Date:
Operating sequence: Star-shaped travel in 45° sections in anti-clockwise direction	Sheet 1 of 5

- Create an operating sequence so that the Robotino® initially travels forward and then back to the starting point and subsequently at 45° in forward direction and back to the starting point, etc.

### Sequence program

#### Note

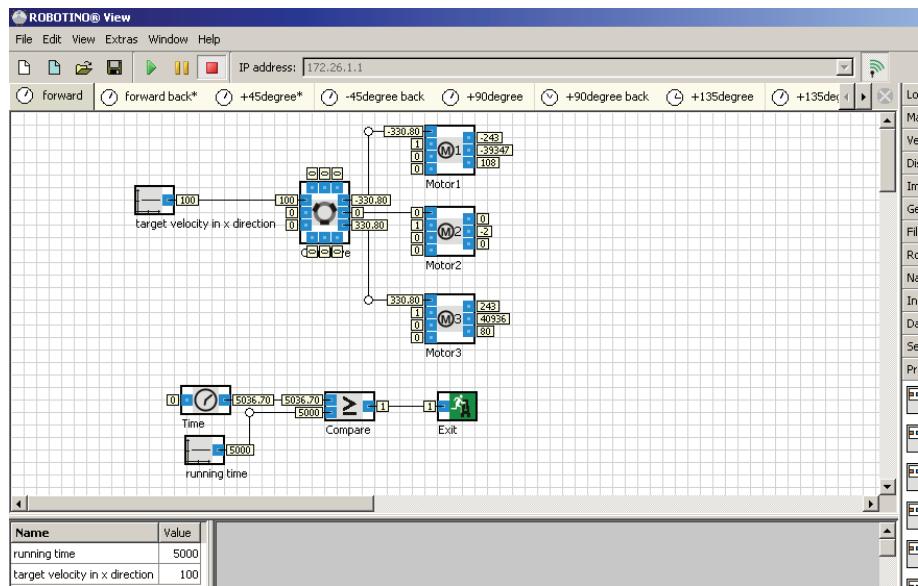
Please ensure that the “start module” is integrated into the sequence program.



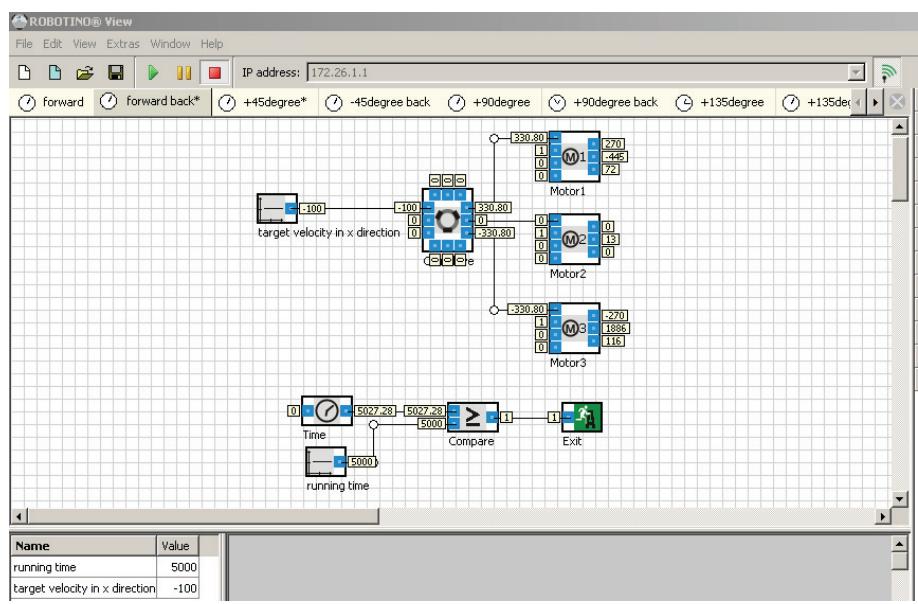
## Project 2: Linear travelling of a mobile robot system in any direction – solution

Project 2: Linear travelling of a mobile robot system in any direction	
Name:	Date:
Operating sequence: Star-shaped travel in 45° sections in anti-clockwise direction.	Sheet 2 of 5

### Forwards



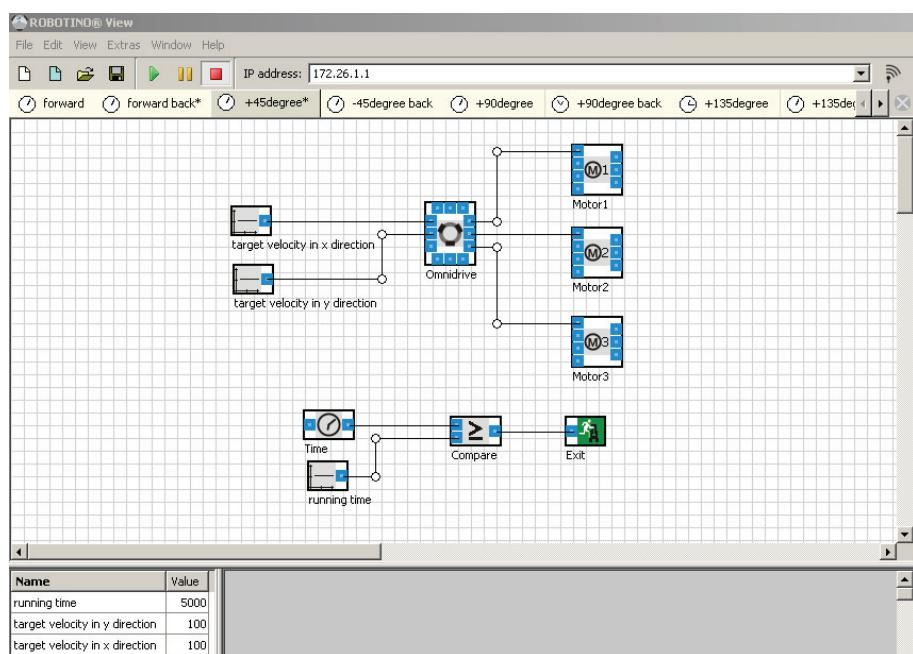
### Forwards back



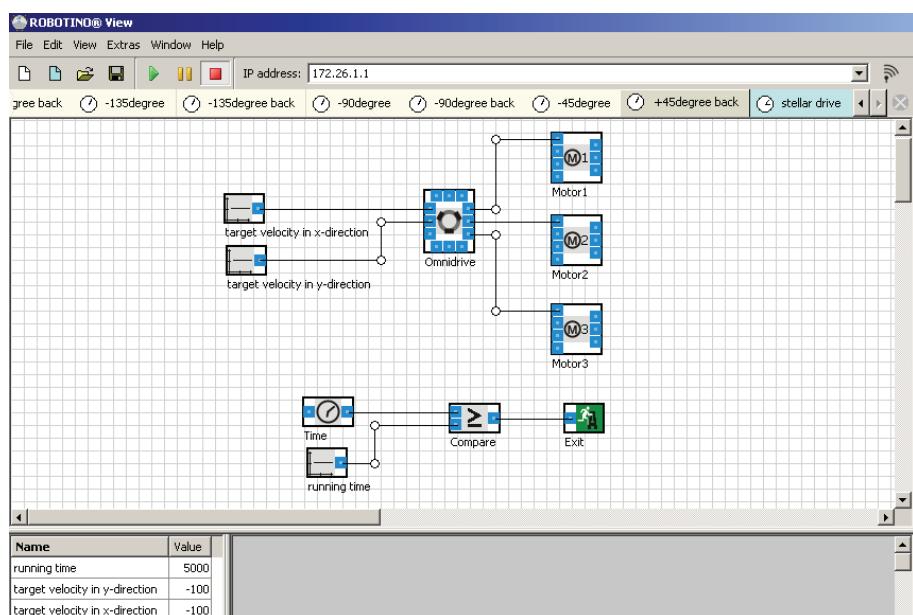
## Project 2: Linear travelling of a mobile robot system in any direction – solution

Project 2: Linear travelling of a mobile robot system in any direction	
Name:	Date:
Operating sequence: Star-shaped travel in $45^\circ$ sections in anti-clockwise direction	Sheet 3 of 5

$+45^\circ$



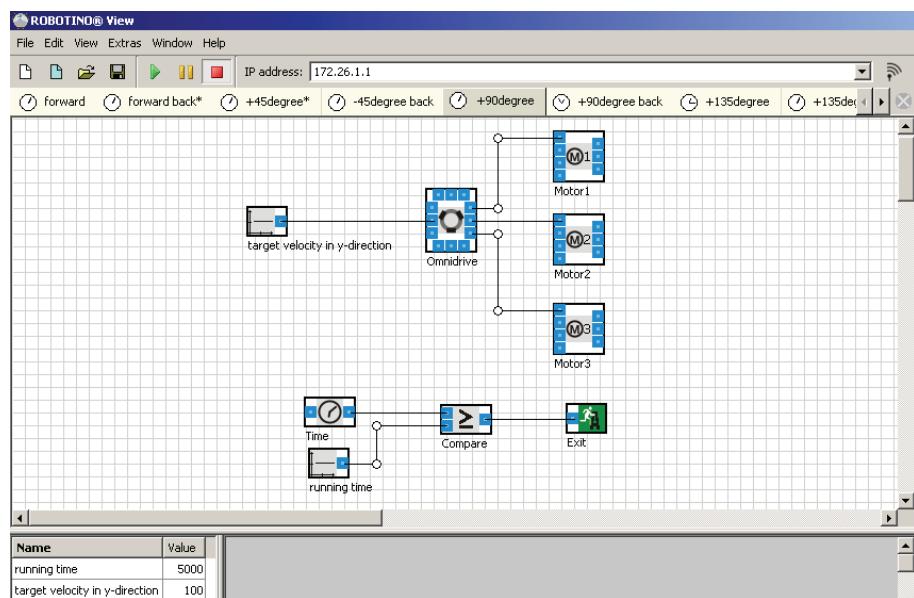
$+45^\circ$  back



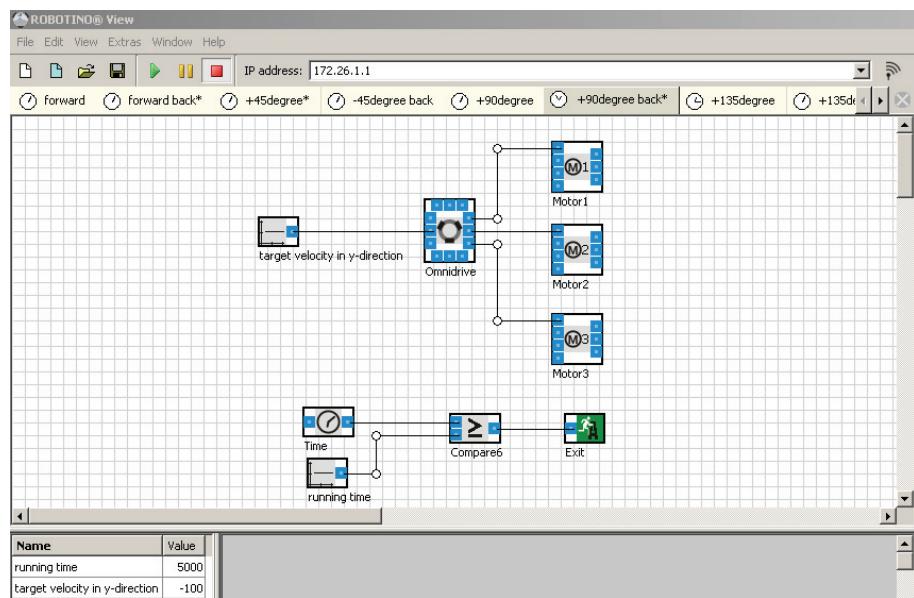
## Project 2: Linear travelling of a mobile robot system in any direction – solution

Project 2: Linear travelling of a mobile robot system in any direction	
Name:	Date:
Operating sequence: Star-shaped travel in 45° sections in anti-clockwise direction	Sheet 4 of 5

+90°



+90° back



Note

To continue star-shaped travel, you simply need to change the constants and their signs for the correct direction of travel.

Project 2: Linear travelling of a mobile robot system in any direction – solution

<b>Project 2: Linear travelling of a mobile robot system in any direction</b>	
Name:	Date:
Operating sequence: Star-shaped travel in 45° sections in anti-clockwise direction.	Sheet 5 of 5

+135°

Name	Value
running time	5000
target velocity in y-direction	100
target velocity in x-direction	-100

+135° back

Name	Value
running time	5000
target velocity in y-direction	-100
target velocity in x-direction	100

180°

Name	Value
running time	5000
target velocity in x-direction	-100

180° back

Name	Value
running time	5000
target velocity in x-direction	100

Project 2: Linear travelling of a mobile robot system in any direction – solution

## Project 3

### Linear travelling and positioning of a mobile robot system – solution

Project 3: Linear travelling and positioning of a mobile robot system	
Name:	Date:
Travelling a distance of 1 m without the use of the “omnidrive” function block	Sheet 1 of 6

- Travel the Robotino® a distance of 1 m without the use of the “omnidrive” function block.
  - Determine the number of revolutions to be carried out by both wheels in order for the Robotino® to travel forward a distance of 1 m, and calculate the number of motor increments.
  - Test the program created which enables the Robotino® to travel forward a distance of 1 m.
  - Measure the distance travelled.
  - What problems arise at higher speeds?
  - Explain why deviations occur from the target value of 1 m.
- 
- Determine the number of revolutions to be carried out by both wheels in order for the Robotino® to travel forward a distance of 1 m and calculate the number of motor increments.

#### Note

Please note the following data:

Path: 1000 mm, 2 wheels are driven

Wheel circumference = distance travelled in one revolution

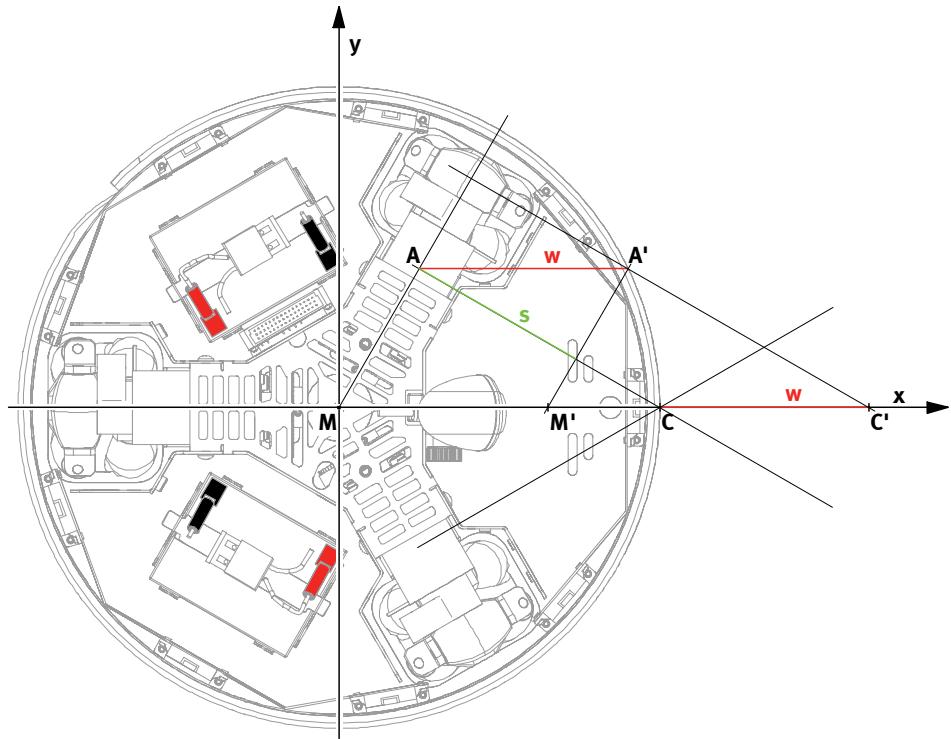
Please also note the drawing in the positional sketch of the problem definition and move the diagram of the Robotino® in forward direction.

<b>Project 3: Linear travelling and positioning of a mobile robot system</b>	
Name:	Date:
Travelling a distance of 1 m without the use of the “omnidrive” function block	Sheet 2 of 6

For the solution, take a look at the motion parallelogram  $\langle A, C, C', A' \rangle$  in illustration below.

w = Travel distance of the Robotino® with a forward motion

s = Distance travelled by wheels with a forward motion by distance w



The illustration results in: (1)  $s = w \cdot \sin(60^\circ)$

Calculation of wheel revolution for distance s:

Wheel circumference  $U=d \cdot \pi$  ( $d$  = Diameter of wheels = 80 mm)

$$\text{Number of revolutions} = \frac{s}{U}$$

<b>Project 3: Linear travelling and positioning of a mobile robot system</b>	
Name:	Date:
Travelling a distance of 1 m without the use of the “omnidrive” function block	Sheet 3 of 6

If we use the formula (1) here and note that  $w = 1000 \text{ mm}$ , then the number of revolutions is calculated as follows:

$$(2) \text{ Number of revolutions } \frac{1000 \cdot \sin(60^\circ)}{80\pi} = \frac{25 \cdot \sin(60^\circ)}{2\pi} = 3.446$$

#### **Result – number of revolutions**

The driven wheels of the Robotino® require 3.446 revolutions in order to cover a distance of 1000 mm.

- Calculation of motor increments for the distance:

#### Note

The gear ratio is 1:16, see technical documentation.

1 motor revolution corresponds to 2048 increments, see technical documentation

From the formula (2) the following calculation formula is obtained for the number of increments for the distance  $s$  travelled by the wheels:

$$s = \frac{2048 \cdot 16 \cdot 25 \cdot \sin(60^\circ)}{2\pi}$$

$$(3) \text{ Number of motor increments} = \frac{409600 \cdot \sin(60^\circ)}{\pi} = 112912.16$$

#### **Motor increments in order to travel a distance of 1m**

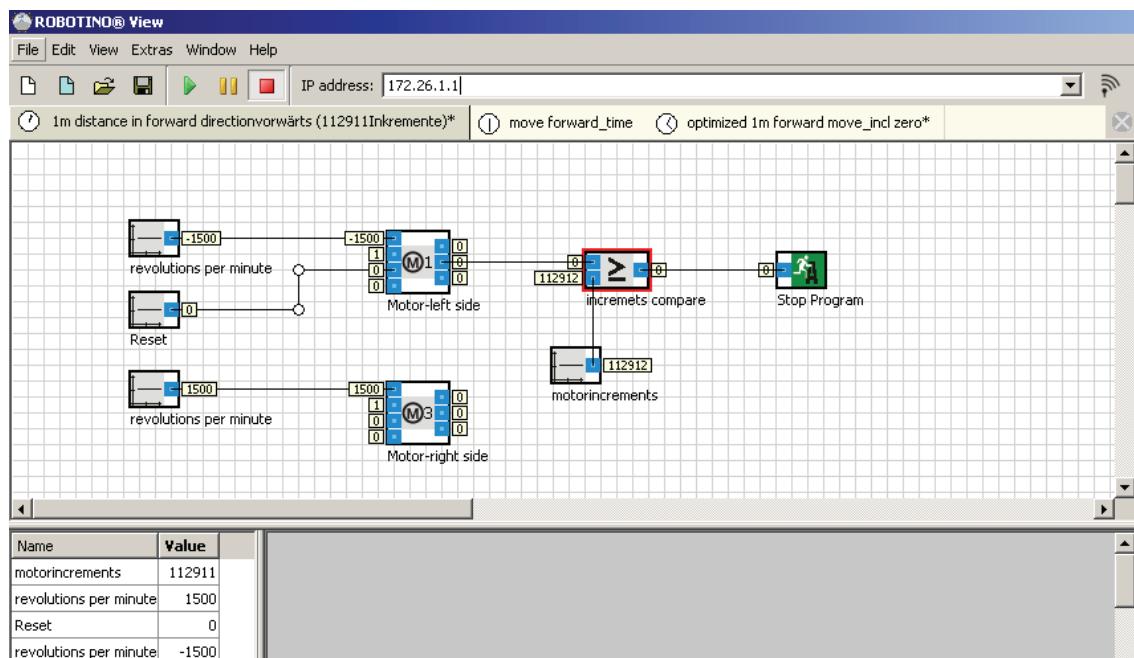
112.912 motor increments required for the two front motors to enable the Robotino® to travel a distance of 1 m.

## Project 3: Linear travelling and positioning of a mobile robot system – solution

Project 3: Linear travelling and positioning of a mobile robot system	
Name:	Date:
Travelling a distance of 1 m without the use of the “omnidrive” function block	Sheet 4 of 6

- Test the program created which enables the Robotino® to travel forward a distance of 1 m.

**Note** Note that the actual position on the motor output is supplied in increments whereby the current position can be compared with the target position of 1 m = 112912 increments.



- The motor function block supplies three output values:
  - Actual speed in [rpm]
  - Actual position [number of travelled increments]
  - Motor current [mA]
- In order to obtain a correct indication of the actual position it is necessary to initially reset the incremental counter to 0 when starting the movement. This is effected via the reset constant which is connected to the reset input of the motor function block.

<b>Project 3: Linear travelling and positioning of a mobile robot system</b>	
Name:	Date:
Travelling a distance of 1 m without the use of the “omnidrive” function block	Sheet 5 of 6

- |                  |  |
|------------------|--|
| Program sequence | <ul style="list-style-type: none"> <li>First set <b>Reset</b> = 1 and the speeds = 0.</li> <li>Start the program. The incremental counter is reset.</li> <li>Stop the program and set the speeds to -50 or 50 and <b>Reset</b> = 0. Re-start the program.</li> </ul> |
|------------------|--|

Note	Please note that the incremental counter counts down in the case of positive speeds.
------	--

Should you re-start the program, the program may stop immediately since the old input value is still being evaluated in the comparison operator IncrementComparison. You can avoid this problem by selecting a different calculation mode for the function block diagram:

- Select the menu Extras → Options → Step mode → Fast. In this mode, all function blocks are fully calculated serially in sequence. The serial allocation is obtained from left to right from the geometric sequence of the function blocks in the diagram.
  - What problems arise at higher speeds?
  - Explain why deviations occur from the target value of 1 m.
- Test the entire program using different speeds.
- Display the data (Ctrl-D or “Display Data“ under View) and observe the displayed values of the motors.
- Measure the distance travelled.
- Compare the results using two different calculation methods for the function block diagram.

#### Result of comparison – step mode

A more precise result is obtained in fast step mode as the result of the comparison is evaluated immediately.

<b>Project 3: Linear travelling and positioning of a mobile robot system</b>	
Name:	Date:
Travelling a distance of 1 m without the use of the “omnidrive” function block	Sheet 6 of 6

**Explanation and behaviour at higher speeds**

- Loss due to calculation time
- Time lost due to data transfer
- Frictional losses at base depending on state of floor
- Frictional losses on rear caster that is dragged along
- A DC motor cannot stop immediately at any point as there are fixed latching points
- Problems at higher speeds: longer braking time, inaccuracies

## Project 3: Linear travelling and positioning of a mobile robot system – solution

Project 3: Linear travelling and positioning of a mobile robot system	
Name:	Date:
Repetition and absolute accuracy	Sheet 1 of 7

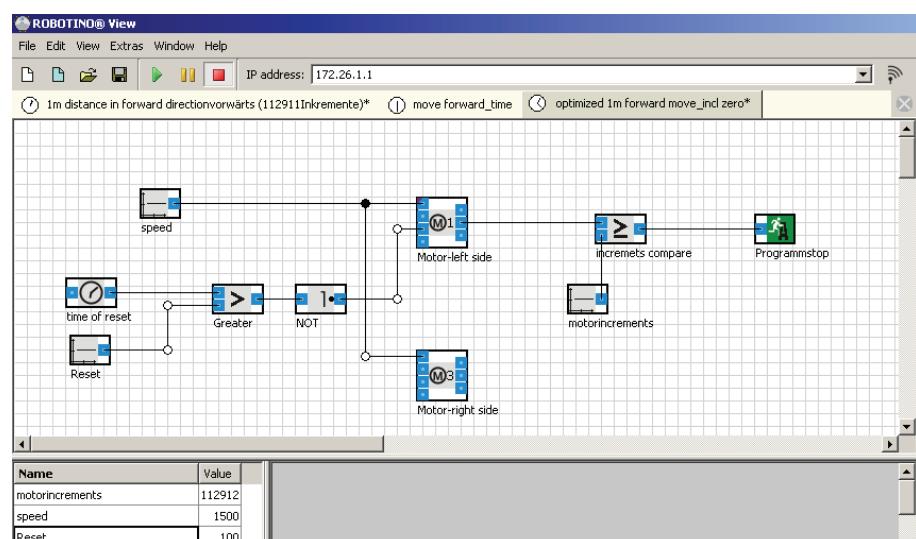
- Carry out 10 measurements by travelling the distance each time from the same starting point in the same direction and at the same speed. Determine the average deviation from the defined distance of 1 m.
- Carry out these experiments for different speeds and explain the results obtained.

Please note

The rechargeable batteries should be fully charged.

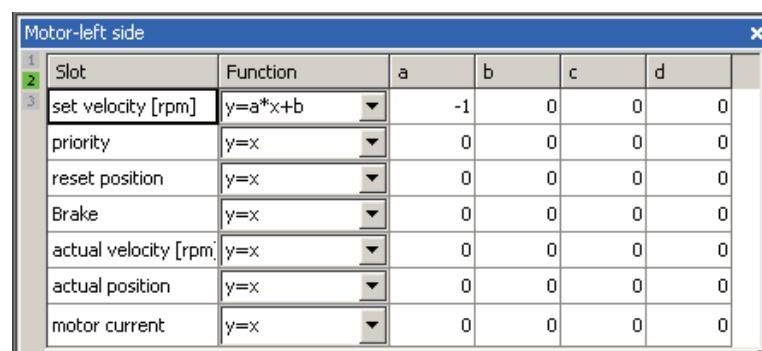
Note

For the purpose of optimisation, please us the following solution.



Only one constant is given for the motors, whereby it should be noted that the setpoint speed of the front-left motor is to be multiplied by -1.

- Use the function block dialogue of the motor and apply the appropriate formula for the setpoint speed.



<b>Project 3: Linear travelling and positioning of a mobile robot system</b>	
Name:	Date:
Repetition and absolute accuracy	Sheet 2 of 7

Calculation of average deviation or of the arithmetic mean value of the distances travelled:

$$\bar{x}_{\text{arithm}} = \frac{1}{n} \sum_{i=1}^n x_i = \frac{x_1 + x_2 + \dots + x_n}{n}$$

Note

Analyse your measurements and the results as a form of quality control of your system, for example the quality of the motors, wheels and your assembly. By means of the measurements you are testing the following: How does the system cope with the requirements, does it meet the necessary requirements?

Measurement results for distances travelled on different surfaces (possible deviations depending on the condition of the surface):

Tiled floor      Speed: 800 rpm

<b>Distance travelled in m</b>	<b>Number of increments</b>	<b>Deviation from setpoint value of 1 m</b>
1.02	113528	0.02
1.025	113739	0.025
1.015	114768	0.015
1.02	113354	0.02
1.075	117729	0.075
1.035	113452	0.035
1.035	113596	0.035
1.045	113472	0.045
1.03	113604	0.03
1.03	113605	0.03
$\bar{x} = 1.033$	$\bar{x} = 114084.7$	$\bar{x} = 0.033$

Project 3: Linear travelling and positioning of a mobile robot system – solution

<b>Project 3: Linear travelling and positioning of a mobile robot system</b>	
Name:	Date:
Repetition and absolute accuracy	Sheet 3 of 7

Speed: 1500 rpm

<b>Distance travelled in m</b>	<b>Number of increments</b>	<b>Deviation from setpoint value of 1 m</b>
1.045	113912	0.045
1.12	113774	0.12
1.05	115026	0.05
1.075	116253	0.075
1.045	114966	0.045
1.105	120895	0.105
1.05	113932	0.05
1.04	114462	0.04
1.02	113813	0.02
1.032	113872	0.032
$\bar{X} = 1.0582$	$\bar{X} = 115090,5$	$\bar{X} = 0.0582$

Speed: 2500 rpm

<b>Distance travelled in m</b>	<b>Number of increments</b>	<b>Deviation from setpoint value of 1 m</b>
1.045	115515	0.045
1.055	114398	0.055
1.025	115490	0.025
1.031	114317	0.031
1.025	114398	0.025
1.045	114375	0.045
1.045	115134	0.045
1.02	114869	0.02
1.11	120241	0.11
1.065	114297	0.065
$\bar{X} = 1.0466$	$\bar{X} = 115303,4$	$\bar{X} = 0.0466$

Project 3: Linear travelling and positioning of a mobile robot system – solution

<b>Project 3: Linear travelling and positioning of a mobile robot system</b>	
Name:	Date:
Repetition and absolute accuracy t	Sheet 4 of 7

Laminated wood flooring      Speed: 800 rpm  
 (Robotino® operating area)

<b>Distance travelled in m</b>	<b>Number of increments</b>	<b>Deviation from setpoint value of 1 m</b>
1.06	115762	0.06
1.045	114622	0.045
1.113	115841	0.113
1.027	113587	0.027
1.05	115761	0.05
1.027	113572	0.027
1.03	114725	0.03
1.04	113979	0.04
1.065	115903	0.065
1.01	113475	0.01
$\bar{X} = 1.0467$	$\bar{X} = 114722.7$	$\bar{X} = 0.0467$

Speed: 1500 rpm

<b>Distance travelled in m</b>	<b>Number of increments</b>	<b>Deviation from setpoint value of 1 m</b>
1.015	113915	0.015
1.06	114638	0.06
1.06	113956	0.06
1.03	113865	0.03
1.125	114096	0.125
1.025	114274	0.025
1.05	118789	0.05
1.078	125845	0.078
1.05	113776	0.05
1.045	113723	0.045
$\bar{X} = 1.0538$	$\bar{X} = 115687.7$	$\bar{X} = 0.0538$

Project 3: Linear travelling and positioning of a mobile robot system – solution

<b>Project 3: Linear travelling and positioning of a mobile robot system</b>	
Name:	Date:
Repetition and absolute accuracy	Sheet 5 of 7

Speed: 2500 rpm

<b>Distance travelled in m</b>	<b>Number of increments</b>	<b>Deviation from setpoint value of 1 m</b>
1.085	118598	0.085
0.945	113960	-0.055
0.96	116397	-0.04
0.98	114621	-0.02
1.08	129114	0.08
1.09	114412	0.09
0.98	115074	-0.02
1.03	114934	0.03
0.992	116078	-0.008
1.082	1245733	0.082
$\bar{X} = 1.0224$	$\bar{X} = 117892.1$	$\bar{X} = 0.0224$

Velour carpeted floor

Speed: 800 rpm

<b>Distance travelled in m</b>	<b>Number of increments</b>	<b>Deviation from setpoint value of 1 m</b>
1.01	113087	0.01
0.99	113262	-0.01
1.015	114398	0.015
1.01	113324	0.01
1.003	113202	0.003
0.983	113344	-0.017
0.99	113243	-0.01
1.03	114484	0.03
0.97	113266	-0.03
1.005	115053	0.005
$\bar{X} = 1.0006$	$\bar{X} = 113666.3$	$\bar{X} = 0.0006$

Project 3: Linear travelling and positioning of a mobile robot system – solution

<b>Project 3: Linear travelling and positioning of a mobile robot system</b>	
Name:	Date:
Repetition and absolute accuracy	Sheet 6 of 7

Speed: 1500 rpm

<b>Distance travelled in m</b>	<b>Number of increments</b>	<b>Deviation from setpoint value of 1 m</b>
1.015	113648	0.015
0.98	113772	-0.02
1.05	113933	0.05
1.06	114039	0.06
1.085	113734	0.085
0.99	114502	-0.01
1.04	117499	0.04
0.994	114939	0.006
1.02	114074	0.02
0.995	113709	0.005
$\bar{X} = 1.0229$	$\bar{X} = 114384.9$	$\bar{X} = 0.0251$

Speed: 2500 rpm

<b>Distance travelled in m</b>	<b>Number of increments</b>	<b>Deviation from setpoint value of 1 m</b>
1.009	113818	0.009
1.03	113638	0.03
1.037	113450	0.037
1.055	112917	0.055
1.055	118394	0.055
0.98	113565	-0.02
1.01	113692	0.01
1.03	113462	0.03
1.017	118860	0.017
1.097	114282	0.097
$\bar{X} = 1.032$	$\bar{X} = 114607.8$	$\bar{X} = 0.032$

<b>Project 3:Linear travelling and positioning of a mobile robot system</b>	
Name:	Date:
Repetition and absolute accuracy	Sheet 7 of 7

<b>Explanation of results obtained</b>
Loss due increment evaluation
Loss due to calculation time
Time lost due to data transfer
Frictional losses at base depending on the condition of floor
Frictional losses on rear caster that is moved along
A DC motor cannot stop immediately at any point as there are fixed latching points
Problems at higher speeds: longer braking time, inaccuracies

<b>Project 3: Linear travelling and positioning of a mobile robot system</b>	
Name:	Date:
Travelling a distance of 1 m using the “omnidrive” function block	Sheet 1 of 11

- Using the omnidrive function block, the Robotino® is to travel forward 2 metre and then stop.
- Calculate the time the Robotino® has to travel using a speed of [100 mm/s] in order to cover a distance of 1 m.
- Create and test the program by adding a time module to your “forward travel” program.
- Carry out the test for different speeds and compare the results with the results obtained without the use of the omnidrive function block.
- Explain the possible reasons for a deviation. During operation, observe the actual values for rotational speed and speeds in x-, y-direction displayed by the “omnidrive” function block.
- Describe a concept for the optimisation of the program.
- How can you calculate the distance travelled from a continuous display of the actual speed in x-direction realised at a clock cycle of approx. 23 ms?
- Optimise your program so that the deviations of the setpoint distance are within the mm range.
- Start the “odometry” program using the same surface and evaluate the results obtained.
  
- Calculate the time the Robotino® has to travel at a speed of [100 mm/s] in order to cover a distance of 1 m

#### Calculation: Solution

The following applies for the calculation  $s = v \cdot t$

$$v = 100 \text{ mm/s}$$

$$s = 1000 \text{ mm}$$

$$t = \frac{s}{v} = \frac{1000}{100} = 10$$

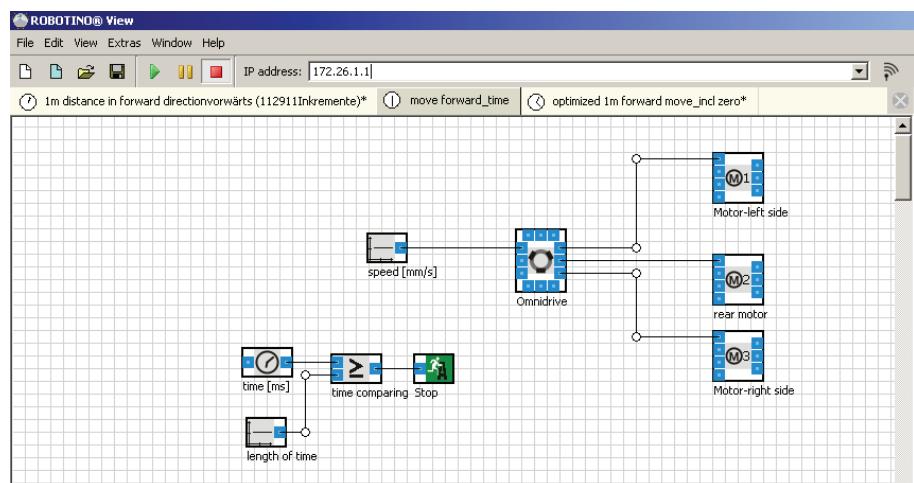
## Project 3: Linear travelling and positioning of a mobile robot system – solution

Project 3: Linear travelling and positioning of a mobile robot system	
Name:	Date:
Travelling a distance of 1 m using the “omnidrive” function block	Sheet 2 of 11

### Result

At a speed of 100 [mm/s], the Robotino® needs to travel for 10 seconds in order to cover a distance of 1 m.

- Create and test the program by adding a time module to your “forward travel” program



- Speed = 100 [mm/s]
- Duration = 10 000 [ms]
- Test the program and measure the actual distance travelled

### Actual distance travelled

Depending on the condition of the floor 70 to 85 cm

<b>Project 3: Linear travelling and positioning of a mobile robot system</b>	
Name:	Date:
Travelling a distance of 1 m, using the “omnidrive” function block	Sheet 3 of 11

- Carry out the test for different speeds and compare the results with the results obtained without the use of the omnidrive function block.

Note Deviations in the measurement results may occur depending on the condition of the surface.

Carpeted floor Distance:  $s = 1 \text{ m}$  / Speed:  $v = 100 \text{ mm/s}$  / Time:  $t = 10\text{s} = 10000 \text{ ms}$

<b>Distance travelled in m</b>	<b>Deviation from setpoint value of 1 m</b>
0.77	0.23
0.70	0.3
0.75	0.25
0.69	0.305
0.69	0.31
0.73	0.27
0.69	0.31
0.75	0.25
0.7	0.3
0.71	0.29
$\bar{x} = 0.7185$	$\bar{x} = -0.2815$

Project 3: Linear travelling and positioning of a mobile robot system – solution

<b>Project 3:Linear travelling and positioning of a mobile robot system</b>	
Name:	Date:
Travelling a distance of 1 m, using the “omnidrive” function block	Sheet 4 of 11

Distance:  $s = 1 \text{ m}$  / Speed:  $v = 50 \text{ mm/s}$  / Time:  $t = 20\text{s} = 20,000 \text{ ms}$

<b>Distance travelled in m</b>	<b>Deviation from setpoint value of 1 m</b>
0.635	-0.365
0.56	-0.46
0.6	-0.4
0.635	-0.365
0.62	-0.38
0.645	-0.355
0.55	-0.45
0.58	-0.42
0.595	-0.405
0.58	-0.42
$\bar{x} = 0.6$	$\bar{x} = -0.402$

Distance:  $s = 1 \text{ m}$  / Speed:  $v = 200 \text{ mm/s}$  / Time:  $t = 5 \text{ s} = 5000 \text{ ms}$

<b>Distance travelled in m</b>	<b>Deviation from setpoint value of 1 m</b>
0.85	- 0.15
0.83	- 0.17
0.865	- 0.135
0.8	- 0.2
0.84	- 0.16
0.84	- 0.16
0.82	- 0.18
0.805	- 0.195
0.85	- 0.15
0.88	- 0.12
$\bar{x} = 0.838$	$\bar{x} = -0.162$

Project 3: Linear travelling and positioning of a mobile robot system – solution

<b>Project 3: Linear travelling and positioning of a mobile robot system</b>	
Name:	Date:
Travelling a distance of 1 m, using the “omnidrive” function block	Sheet 5 of 11

Velour carpeted floor

Distance:  $s = 1 \text{ m}$  / Speed:  $v = 100 \text{ mm/s}$  / Time:  $t = 10\text{s} = 10000 \text{ ms}$

<b>Distance travelled in m</b>	<b>Deviation from setpoint value of 1 m</b>
0.81	-0.19
0.79	-0.21
0.74	-0.26
0.73	-0.27
0.7	-0.3
0.79	-0.21
0.75	-0.25
0.74	-0.26
0.76	-0.24
0.7	-0.3
$\bar{x} = 0.751$	$\bar{x} = -0.249$

Distance:  $s = 1 \text{ m}$  / Speed:  $v = 200 \text{ mm/s}$  / Time:  $t = 5 \text{ s} = 5000 \text{ ms}$

<b>Distance travelled in m</b>	<b>Deviation from setpoint value of 1 m</b>
0.78	-0.22
0.86	-0.14
0.88	-0.12
0.86	-0.14
0.88	-0.12
0.885	-0.115
0.84	-0.16
0.81	-0.19
0.85	-0.15
0.83	-0.17
$\bar{x} = 0.8475$	$\bar{x} = -0.1525$

Project 3: Linear travelling and positioning of a mobile robot system – solution

<b>Project 3: Linear travelling and positioning of a mobile robot system</b>	
Name:	Date:
Travelling a distance of 1 m, using the “omnidrive” function block	Sheet 6 of 11

Distance:  $s = 1 \text{ m}$  / Speed:  $v = 400 \text{ mm/s}$  / Time:  $t = 2.5 \text{ s} = 2500 \text{ ms}$

<b>Distance travelled in m</b>	<b>Deviation from setpoint value of 1 m</b>
0.93	-0.07
0.99	-0.01
0.8	-0.2
0.8	-0.2
0.95	-0.05
0.93	-0.07
0.91	-0.09
0.93	-0.07
0.97	-0.03
0.9	-0.1
$\bar{x} = 0.911$	$\bar{x} = -0.089$

Tiled floor

Distance:  $s = 1 \text{ m}$  / Speed:  $v = 100 \text{ mm/s}$  / Time:  $t = 10\text{s} = 10000 \text{ ms}$

<b>Distance travelled in m</b>	<b>Deviation from setpoint value of 1 m</b>
0.85	-0.15
0.83	-0.17
0.83	-0.17
0.825	-0.175
0.81	-0.19
0.79	-0.21
0.855	-0.145
0.81	-0.19
0.84	-0.16
0.835	-0.165
$\bar{x} = 0.8275$	$\bar{x} = -0.1725$

<b>Project 3: Linear travelling and positioning of a mobile robot system</b>	
Name:	Date:
Travelling a distance of 1 m, using the “omnidrive” function block	Sheet 7 of 11

Distance:  $s = 1 \text{ m}$  / Speed:  $v = 400 \text{ mm/s}$  / Time:  $t = 2.5 \text{ s} = 2500 \text{ ms}$

Distance travelled in m	Deviation from setpoint value of 1 m
0.95	-0.05
0.96	-0.04
0.94	-0.06
0.95	-0.05
0.95	-0.05
0.96	-0.04
0.94	-0.06
0.945	-0.055
0.98	-0.02
0.96	-0.04
$\bar{x} = 0.9535$	$\bar{x} = -0.0465$

#### Comparison of results obtained with and without the use of the „omnidrive“

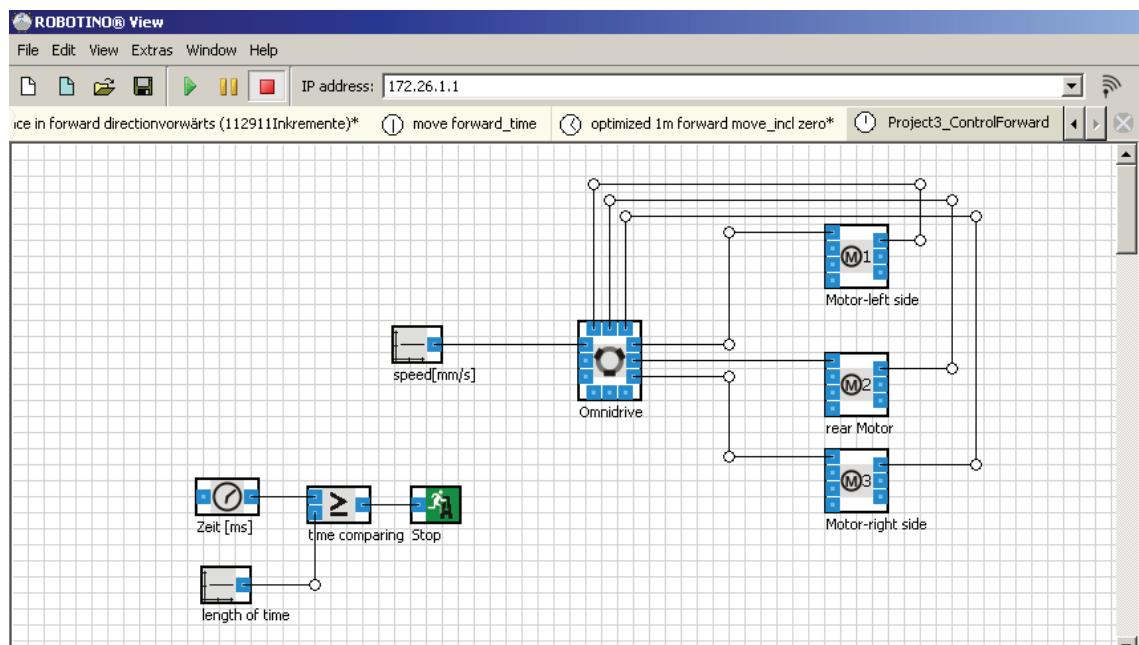
The program without the omnidrive function block produces distinctly better results.

Particularly noticeable is that deviations at higher speed are reduced. In order to better understand this, a closer analysis of the reasons for the deviations is necessary.

## Project 3: Linear travelling and positioning of a mobile robot system – solution

Project 3: Linear travelling and positioning of a mobile robot system	
Name:	Date:
Travelling a distance of 1 m, using the “omnidrive” function block	Sheet 8 of 11

- Explain the possible reasons for a deviation. During operation, monitor the actual values for the rotational speed and speeds in x-, y-direction displayed by the omnidrive function block.
- Modify the program as follows to display the actual values for the speeds in x- and y-direction:



Observation
Actual speed in x-direction = approx. 83
Actual speed in y-direction = approx. + 2.3
Actual speed in direction of rotation = approx. + 1.05
Estimated value of distance travelled = approx. 85 cm

<b>Project 3: Linear travelling and positioning of a mobile robot system</b>	
Name:	Date:
Travelling a distance of 1 m, using the “omnidrive” function block	Sheet 9 of 11

#### **Reasons for the deviations**

The kinematic hardware model does not correspond to reality. The kinematic model assumes that the wheels are arranged in symmetrical configuration of 120°. In reality this is not possible for reasons of assembly.

A further assumption is that the wheels are precisely configured at an angle of 90 ° to the axis of rotation. This again is only possible to a limited extent.

The lesser deviation at higher speed can be explained as follows: The effect of the faulty model only briefly manifests itself for a shorted period and consequently results in a smaller error overall.

Note: It would be interesting to calculate a correction factor.

#### **Describe a concept for the optimisation of the program.**

For optimisation, the program needs to be designed such that it takes into account the actual data of speeds and the associated deviation from the movement in x-direction.

- How can you calculate the distance travelled from a continuous display of the actual speed in x-direction occurring at a clock pulse of approx. 23 ms?

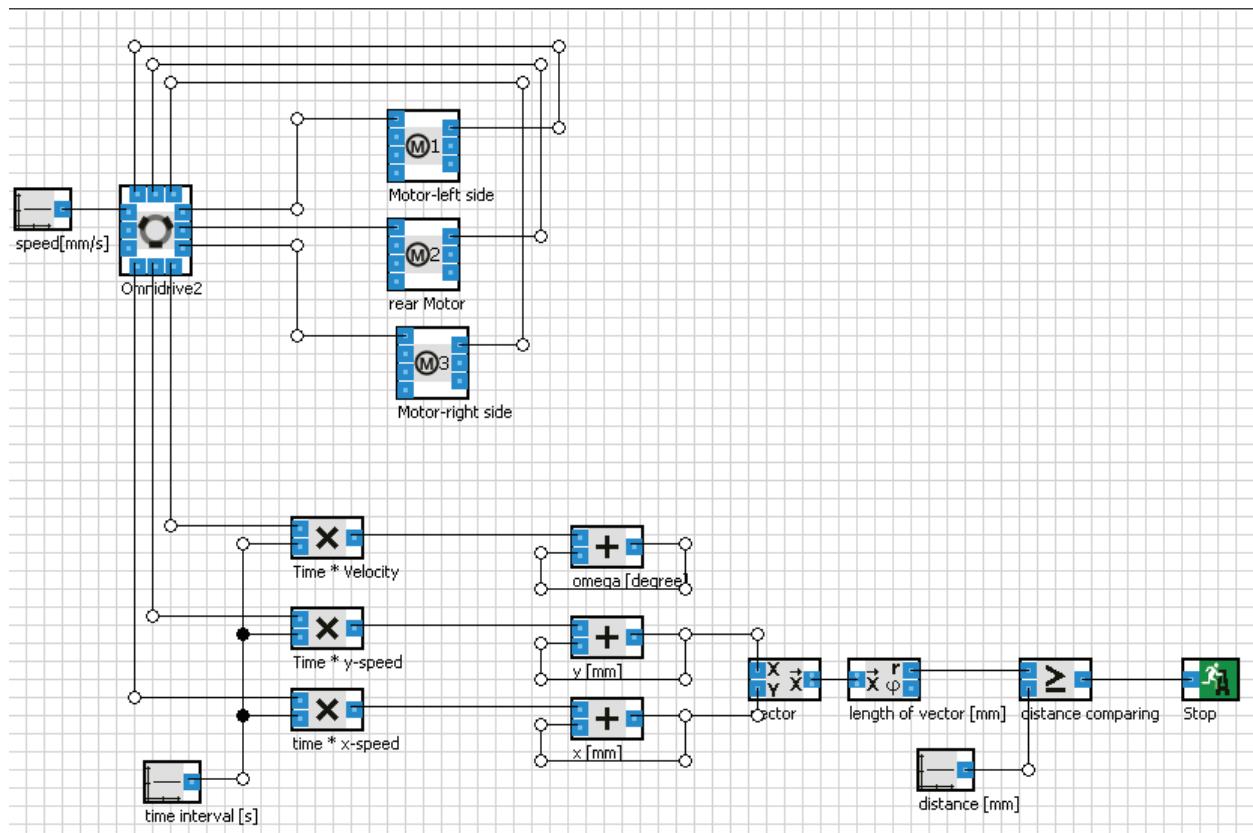
#### **Note**

If „VXactual“ designates the speed displayed, the Robotino® travels a distance of  $s = 0.023 \times VXactual [mm]$  until the next display.

## Project 3: Linear travelling and positioning of a mobile robot system – solution

Project 3: Linear travelling and positioning of a mobile robot system	
Name:	Date:
Travelling a distance of 1 m, using the “omnidrive” function block	Sheet 10 of 11

- Optimise your program so that the deviation from the setpoint distance lies within the mm range.



- Speed = 100 [mm/s]
- Clock pulse = 0.023 [s]
- With the “looped” addition, the distances travelled are added per clock pulse. A vector is from the x- and y-values whose length is the distance actually travelled. Addition is stopped as soon as the setpoint value of 1 [m] is reached.

## Project 3: Linear travelling and positioning of a mobile robot system – solution

<b>Project 3: Linear travelling and positioning of a mobile robot system</b>	
Name:	Date:
Travelling a distance of 1 m, using the “omnidrive” function block	Sheet 11 of 11

- Test the program and evaluate the results obtained.

Evaluation of various measurements

Program: Project3\_Odemetry.rvw

Distance =  $s = 1 \text{ m} = 1000 \text{ mm}$  / Speed = 100 mm/s

Clock pulse of measurements = 0.023 s = 23 ms

Measured distance in mm	Vector length in mm (from Robotino® View)
1025	1001.95
985	1003.05
990	1003.07
965	1003.7
995	1002.86
$X = 992$	$X = 1002.962$

Clock pulse of measurements: 0.0225 s = 22.5 ms

Measured distance in mm	Vector length in mm (from Robotino® View)
980	1002.26
990	1002.45
995	1002.88
995	1002.86
990	1002.61
$X = 990$	$X = 1002.612$

### Best results - timing

With this project, the best results are obtained using a clock pulse between 0.02 s and 0.025 s.

The stability of the W-LAN connection and charge state of the storage batteries further affect the results.

## Project 4

### Path tracking of an automated guided vehicle system using two diffuse sensors – solution

<b>Project 4: Path tracking of an automated guided vehicle using two diffuse sensors</b>	
Name:	Date:
Creating a work plan	Sheet 1 of 1

- Create your work plan for this project. Determine all the necessary work steps as detailed as possible. Enter the work steps in the table below. Use these also as a check list for project documentation when working on the project.

Solution	<b>Activity</b>	<b>Completed</b>
	Commission diffuse sensors	
	Mount diffuse sensors	
	Adjust diffuse sensors	
	Connect diffuse sensors to the Robotino®	
	Test the inputs with the help of Robotino® View	
	Define the control system strategy	
	Programming of open-loop control program	
	Define subprograms of open-loop control program	
	Realise subprograms	
	Test subprograms	
	Execute path tracking using subprograms	
	Realise sequence control	
	Connect subprograms	
	Test sequence programs	
	Realise closed-loop control program	
	Optimise closed-loop control program	
	Compile documentation	

<b>Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors</b>	
Name:	Date:
Mounting of diffuse sensors	Sheet 1 of 3

- Mount the two diffuse sensors at the points on the chasses provided for this. Describe how you proceed or record this when mounting the diffuse sensors.

Note                    Mount the two fibre-optic cable heads as close as possible to one another.

Solution              Required tools and components:

Fibre-optic cable cutter

3 mm hexagon

The following components are required for each sensor:

- 1 fibre-optic unit
- 1 fibre-optic cable
- 1 bracket for fibre-optic cable head

1. Preassemble diffuse sensor.

Shorten fibre-optic cable to required length.

Screw fibre-optic cable head into the bracket.

Secure fibre-optic cable head.

Connect loose end of the fibre-optic cable to the fibre-optic unit.

2. Mount bracket with fibre-optic cable head to chassis.

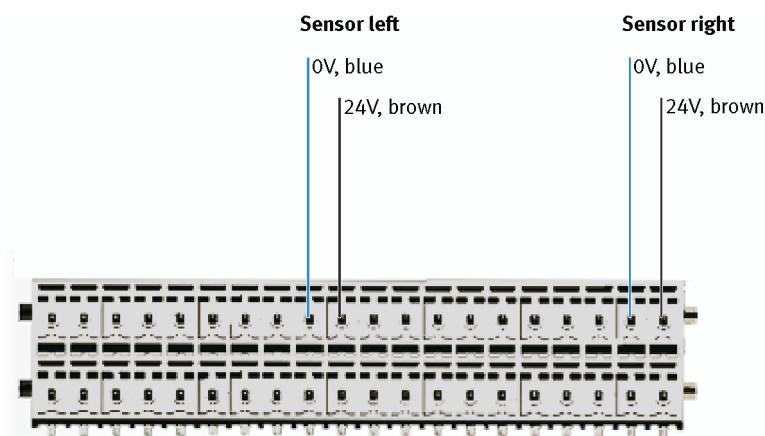
3. Attach fibre-optic unit to the assembly panel of the Robotino®.

## Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors – solution

Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors	
Name:	Date:
Mounting of diffuse sensors	Sheet 2 of 3

1. Enter the cables on the drawing shown below and label these with their characteristics and colour.

Solution



2. Connect the diffuse sensors to the power supply in accordance with your drawing.

3. Adjust the diffuse sensors and describe how you proceed.

Solution

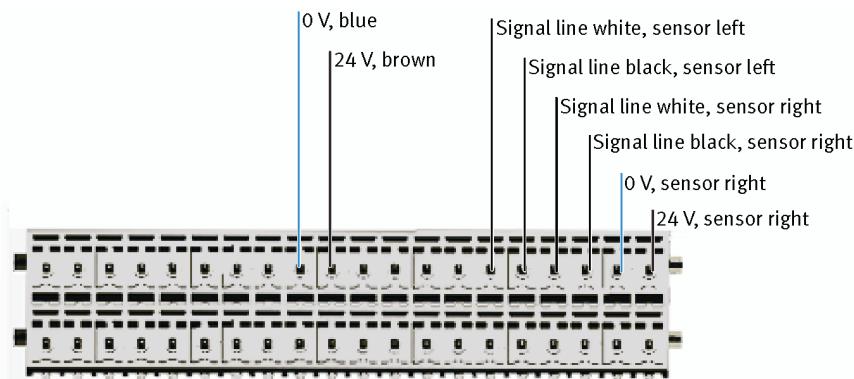
Connect sensors to the power supply of the I/O interface.  
Put the Robotino® onto the surface to be travelled.  
Using the enclosed small screwdriver, turn the adjusting screw until the operating status display (LED) switches on.  
Place an object underneath the Robotino® in accordance with the marking to be travelled. The operating status display must switch off. Re-adjustment may be necessary.

## Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors – solution

Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors	
Name:	Date:
Mounting of diffuse sensors	Sheet 3 of 3

- Connect the diffuse sensors to the I/O interface. Connect the cable of the lefthand diffuse sensor to inputs DI0 and DI1 and those of the righthand diffuse sensor to inputs DI2 and DI3. Connect the black signal cables to inputs DI0 and DI2 and the white signal cables to DI1 and DI3.
- Enter the cables on the drawing shown below and label these according to their colour and the relevant sensor. In addition enter the voltage supply of the two sensors in the drawing.

### Solution

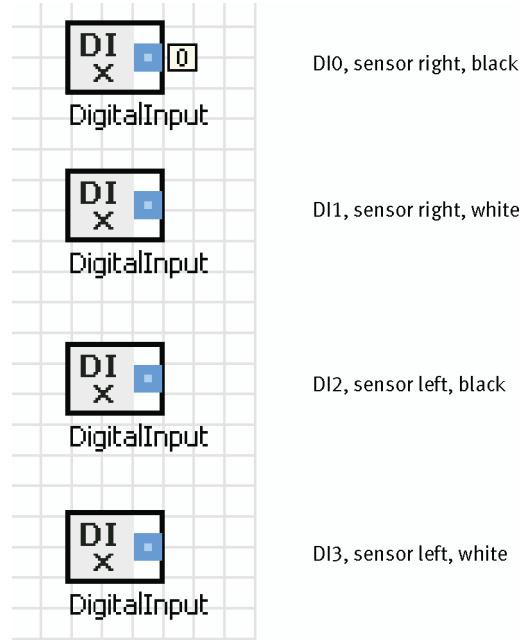


## Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors – solution

<b>Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors</b>	
Name:	Date:
Selection of the inputs in Robotino® View	Sheet 1 of 2

- Enter all inputs of the two diffuse sensors in the diagram below. Label these according to the relevant sensor and the colour of the cables connected.

Solution

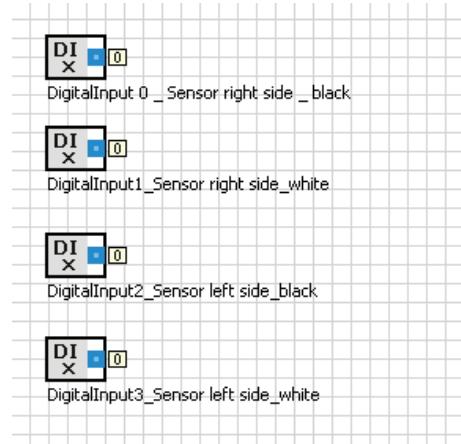


## Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors – solution

<b>Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors</b>	
Name:	Date:
Selection of the inputs in Robotino®View	Sheet 2 of 2

- In Robotino®View, assign the inputs to the individual input function blocks in accordance with your specification. Label the input function blocks according to your specification.

Solution



## Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors – solution

<b>Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors</b>	
Name:	Date:
Determining the sensor values of the individual inputs	Sheet 1 of 2

- To test the efficient functioning of the sensors, determine the sensor values for the different inputs in Robotino® View. The prerequisite is that the Robotino® stands on the intended operating surface and is adjusted accordingly. Enter the sensor values in the table below.

Solution	<b>Input</b>	<b>Signal</b>
	Input DI0 (sensor right)	0
	Input DI1 (sensor right)	1
	Input DI2 (sensor left)	0
	Input DI3 (sensor left)	1

- Evaluate the sensor input signals thus determined with regard to their switching function.

Solution  
 Each sensor has to signal cables with two different functions.  
 The white signal cable is for bright switching  
 The black signal cable is for the dark switching function

## Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors – solution

<b>Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors</b>	
Name:	Date:
Determining the sensor values of the individual inputs	Sheet 2 of 2

Consider which three possible situations may arise during the required path tracking and enter these in the appropriate column in the table below.

- Re-enact the different travel situations with the Robotino® and enter the input values occurring in the table below.

**Note** To do so, use the sample program „Aufg-P4-01.rvw“  
Use this table at a later stage when programming the open-loop control program.

Solution	<b>Travel situation</b>	<b>DIO</b>	<b>DI1</b>	<b>DI2</b>	<b>DI3</b>
		0	1	0	1
	The Robotino® is in the correct position with regard to the marking. Both sensors are above the surface	0	1	0	1
	The Robotino® deviates to the left of the correct positioning relation to the marking. Sensor right is located above the marking	1	0	0	1
	The Robotino® deviates to the right of the correct position in relation to the marking. Sensor left is located above the marking	0	1	1	0

Use this table at a later stage when programming the open-loop control program.

Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors	
Name:	Date:
Development of the control system strategy	Sheet 1 of 1

- Use the table from sheet 3 of the previous exercise and describe the start condition for each of the 3 travel situations, the function of a corresponding subprogram and a termination condition.

Solution

If the Robotino® is in the correct position:

**Start condition:** Both sensors detect the surface

**Function:** The Robotino® moves forward.

**Termination condition:** If one of the two diffuse sensors detects the marking, forward travel is terminated.

The Robotino® deviates to the left of the correct position in relation to the marking.

**Start condition:** The righthand diffuse sensor has detected the marking

**Function:** The Robotino® is to rotate in a clockwise direction.

**Termination condition:** If the righthand diffuse sensor detects the surface, the rotational movement is terminated.

The Robotino® deviates to the right of the correct position as regards the marking.

**Start condition:** The lefthand diffuse sensor has detected the marking

**Function:** The Robotino® is to rotate in an anti-clockwise direction.

**Termination condition:** If the lefthand diffuse sensor detects the surface, the rotational movement is terminated.

## Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors – solution

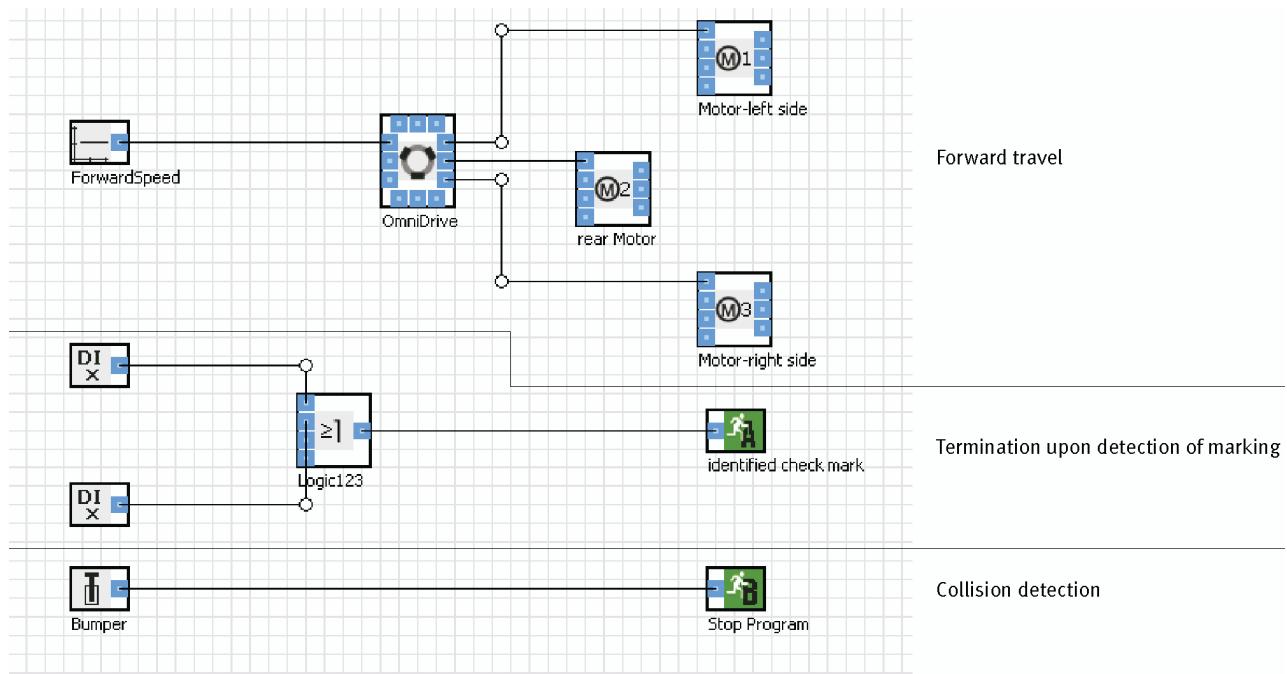
Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors	
Name:	Date:
Development of the open-loop control program	Sheet 1 of 7

Three sample programs are available for the development of the Robotino® control system.

The first subprogram is called Aufg-P4-02.rvw and is shown on this page, although it needs to be completed. Your task is to complete this program.

- To make it easier to prepare the program it is useful to divide it into individual groups of functions. Assign the pertaining functions to the groups of functions by briefly describing these. Enter this description in the diagram below.

### Solution

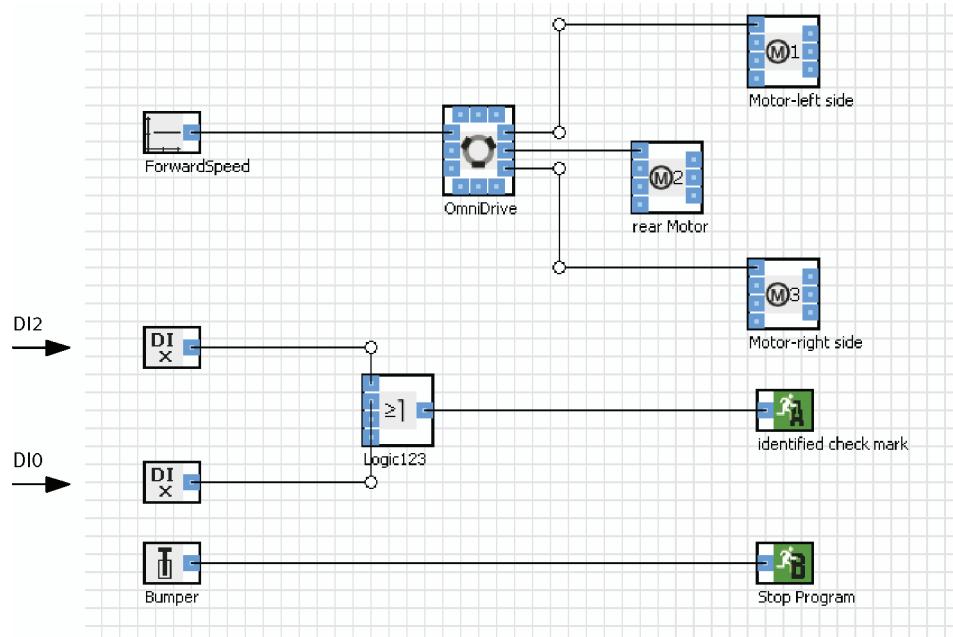


Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors – solution

Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors	
Name:	Date:
Development of the open-loop control program	Sheet 2 of 7

- Assign the appropriate inputs to the input function blocks. (DIO to DI3). Enter these in the diagram below.

Solution

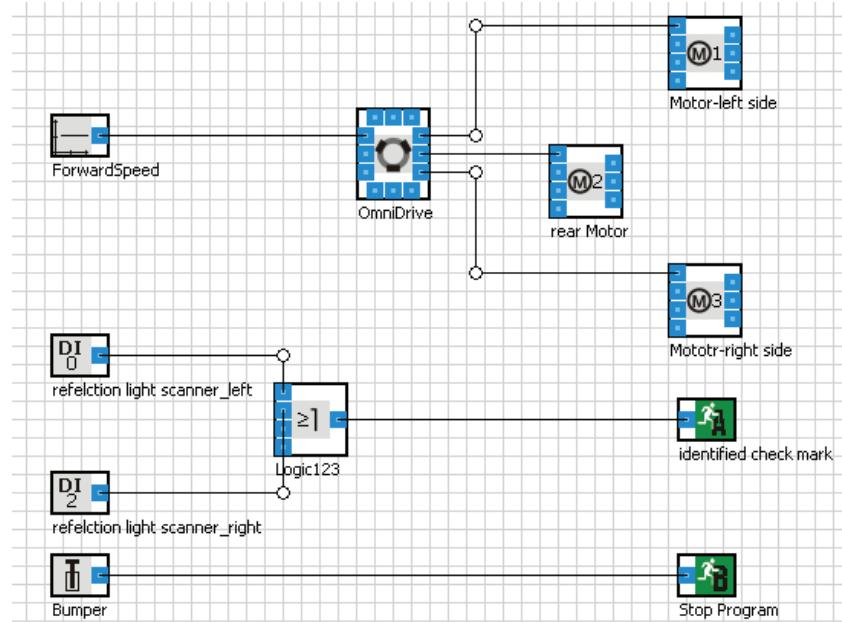


Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors – solution

Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors	
Name:	Date:
Development of the open-loop control program	Sheet 3 of 7

- Amend the sample program Aufg-P4-02.rvw according to your specification.

Solution

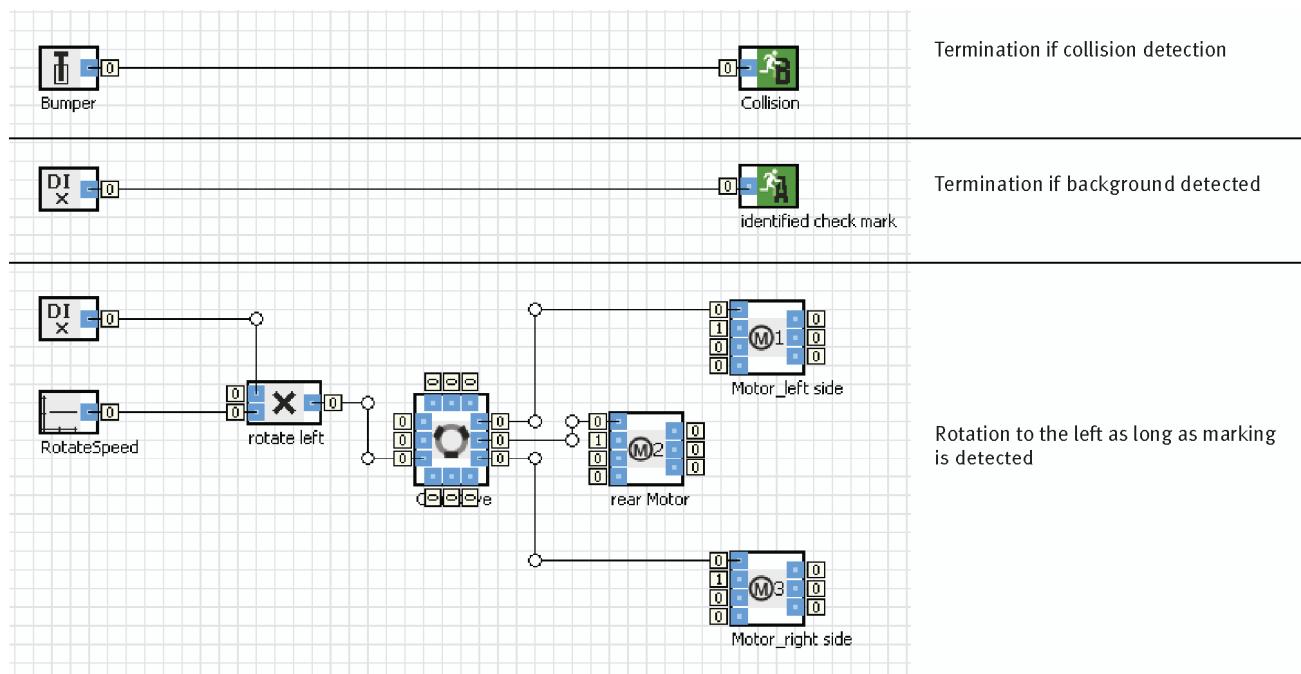


## Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors – solution

Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors	
Name:	Date:
Development of the open-loop control program	Sheet 4 of 7

- Proceed in exactly the same way within the sample program Aufg-P4-03.RVW as with program Aufg-P4-02.RVW. First designate the individual function groups in order to understand the program behaviour.  
Enter this description in the diagram below.

### Solution



- Describe how you identify the direction of rotation.

### Solution

Explanation of direction of rotation:

Since the constant for the rotational speed has a positive value, the Robotino® rotates to the left, i.e. in anti-clockwise direction.

## Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors – solution

Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors	
Name:	Date:
Development of the open-loop control program	Sheet 5 of 7

- Assign the appropriate input to the input function blocks (DI0 to DI3).

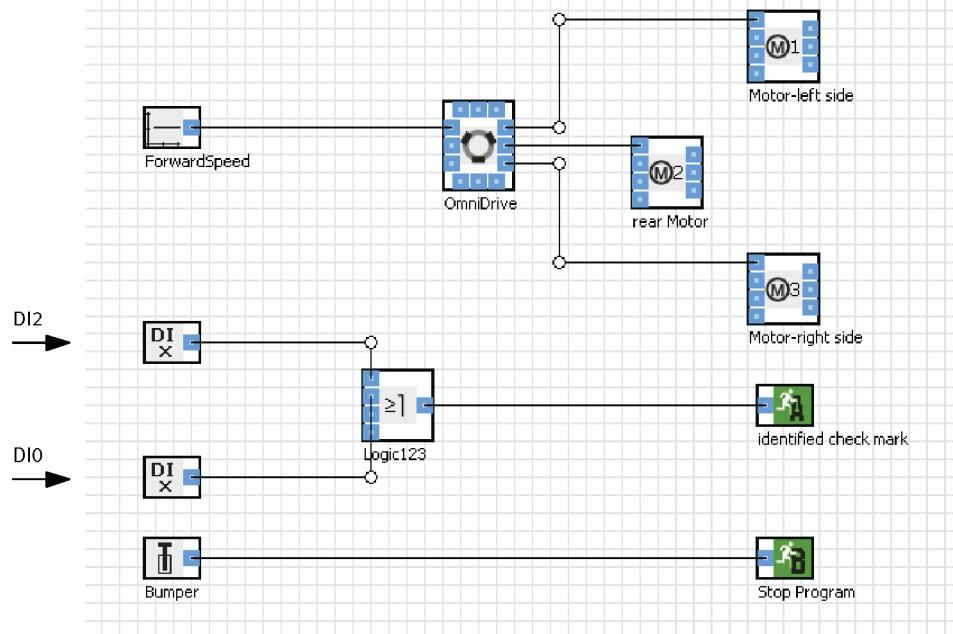
  1. Consider which of the travel situations is applicable if the Robotino® is to execute a rotation in anti-clockwise direction. Which sensor needs to be checked in this case?

Note                    Use your table with the sensor signals of the individual travel situation.

Solution              The Robotino® deviates to the right of the marking. The lefthand sensor is located above the marking. Rotation in anti-clockwise direction is therefore required for an extended period until it is located above the surface again.

2. Enter the relevant inputs in the diagram below, taking into consideration which input behaviour is bright-switching and which dark-switching.

Solution

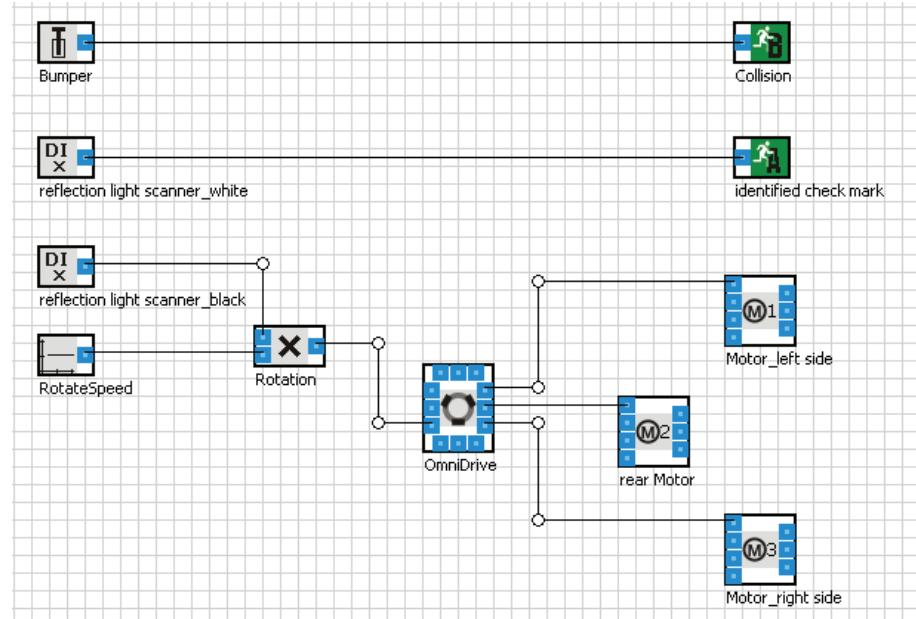


Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors – solution

Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors	
Name:	Date:
Development of the open-loop control program	Sheet 6 of 7

Amend the sample program Aufg-P4-03.rvw according to your specification.

Solution

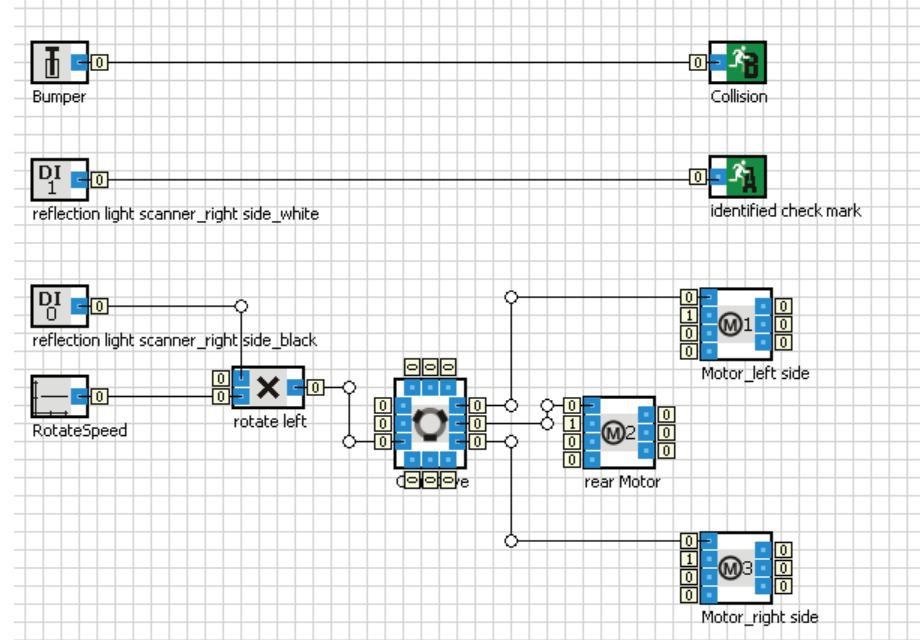


## Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors – solution

Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors	
Name:	Date:
Development of the open-loop control program	Sheet 7 of 7

- Now realise a control program for one rotation of the Robotino® in clockwise direction.  
Amend the sample program Aufg-P4-04.rvw accordingly.  
Note also the rotational speed.

Solution



## Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors – solution

<b>Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors</b>	
Name:	Date:
Testing of the subprograms	Sheet 1 of 1

- Travel the length of the marking by starting the three subprograms according to the situation. Try to travel the path as fast as possible and note your best time.

Solution

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

<b>Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors</b>	
Name:	Date:
Combination of the individual subprograms into a sequence control	Sheet 1 of 1

- To improve the reaction time of the individual subprograms to the respective travel situations, the individual subprograms are to be combined into a sequence control. Establish the relevant information in the theory section and in the Robotino®View help.

Solution

Open all required subprograms in Robotino®View.

Open a workspace to create a sequence control by means of clicking onto the



icon.

Via the library class ‘programs’, drag the required function block diagrams in the form of a function block



onto the workspace with the mouse.

To start a sequence program, drag a start button from the ‘sequence control’ library



onto the workspace.

Connect output S of the start button to input I1 of the function block for one of the three subprograms.

Output B for the collision protection function has been used in all sample programs. Interconnect all function blocks in sequence by connecting output A of a function block to input I1 of the next function block. Connect output A of the last subprogram to input I1 of the first subprogram so that the overall sequence is not terminated once all 3 subprograms are executed.

Save the sequence control in the form of programs.

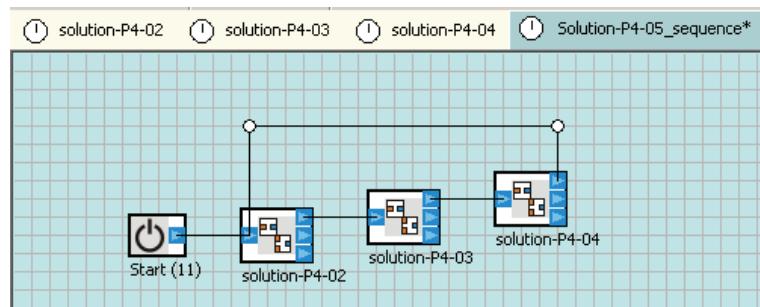
## Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors – solution

Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors	
Name:	Date:
Combination of the individual subprograms into a sequence control	Sheet 1 of 1

### Note

Output A is set the respective subprogram in all programs via the termination condition. The subsequent subprogram can be started via this output signal. The individual subprograms can be combined in any sequence. If the start condition of a subprogram is not fulfilled, the subprogram is aborted. The output is set and the next subprogram is started.  
Also save the sequence control as a program.

### Solution



## Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors – solution

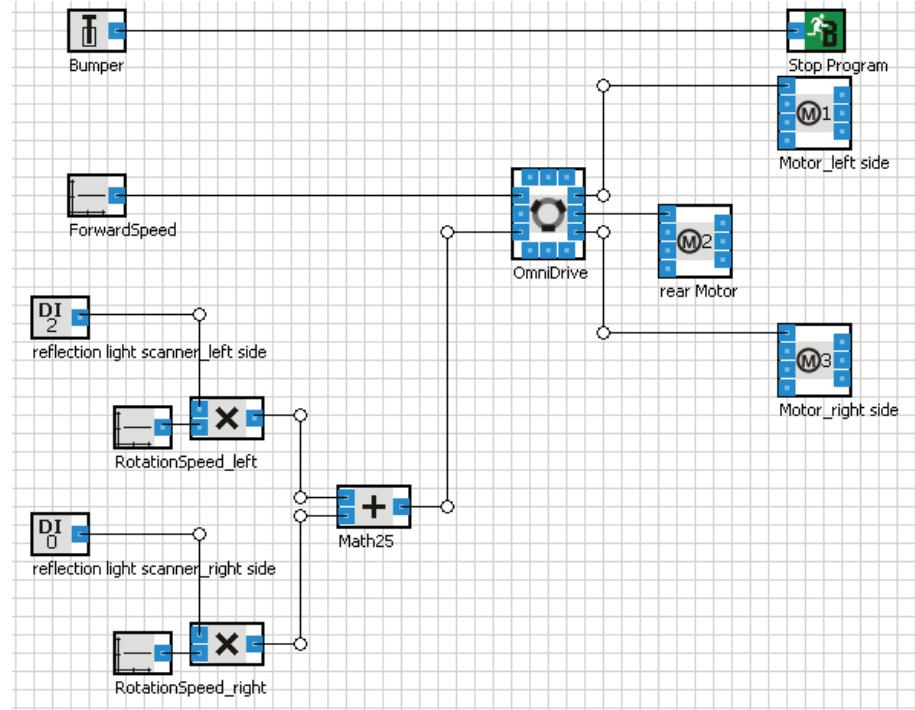
Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors	
Name:	Date:
Realisation of a closed-loop control program	Sheet 1 of 1

- Integrate all functions into a single function block diagram to convert the sequence control into a closed-loop control program.

General conditions

The closed-loop control program must not be aborted if the loading station is reached or any obstacle is touched.

Solution



<b>Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors</b>	
Name:	Date:
Realisation of a closed-loop control program	Sheet 1 of 1

#### Functional description

##### **Control of the rotation**

Multiplication of the rotational speed by the sensor value causes a rotation of the Robotino®. If the value of the input is 0, multiplication by the respective rotational speed equals 0.

$$(0 \times \text{rotational speed} = 0)$$

However, if the marking is detected, the value of the input is 1 and the multiplication results in the rotational speed.

$$(1 \times \text{rotational speed} = \text{rotational speed})$$

Rotation is maintained until the sensor value is 0 again. The controlled variable is reached.

##### **Selection of the direction of rotation**

The direction of rotation is selected via the addition of rotational speeds (right / left). If both diffuse sensors fail to detect the marking, the sensor value is 0 and multiplication by the rotational speed also equals 0. Addition of the two values results in

$$0 + 0 = 0$$

If the value of the inputs is 1, the result of the addition is

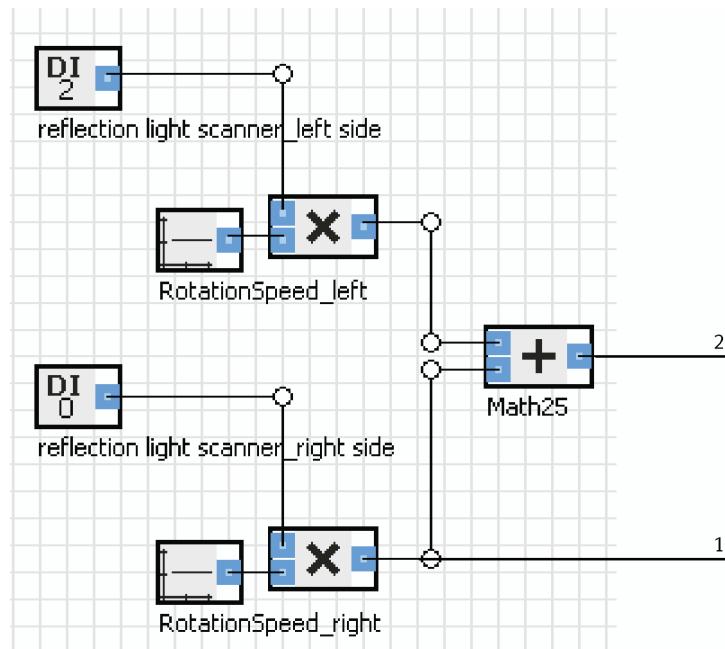
$$\text{Rotational speed} + 0 = \text{rotational speed}$$

If the value of both inputs is 1, addition would result in 0, since both rotational speeds are identical and one of the two results in a minus value.

$$-(\text{rotational speed}) + \text{rotational speed} = 0$$

Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors – solution

Project 4: Path tracking of an automated guided vehicle system using two diffuse sensors	
Name:	Date:
Realisation of a closed-loop control program	Sheet 1 of 1



1 = Rotation, 2 = Direction of rotation

## Project 5

### Accurately positioned approach of a loading station – solution

<b>Project 5: Accurately positioned approach of a loading station</b>	
Name:	Date:
Position detection of distance sensors	Sheet 1 of 4

In order to respond to the individual distance sensors, you need to determine which sensor is mounted at what position of the Robotino®.

- Describe how you proceed to determine the position of the individual distance sensors of the Robotino®.

Procedure:

The position of a sensor is determined by visualising its output value.

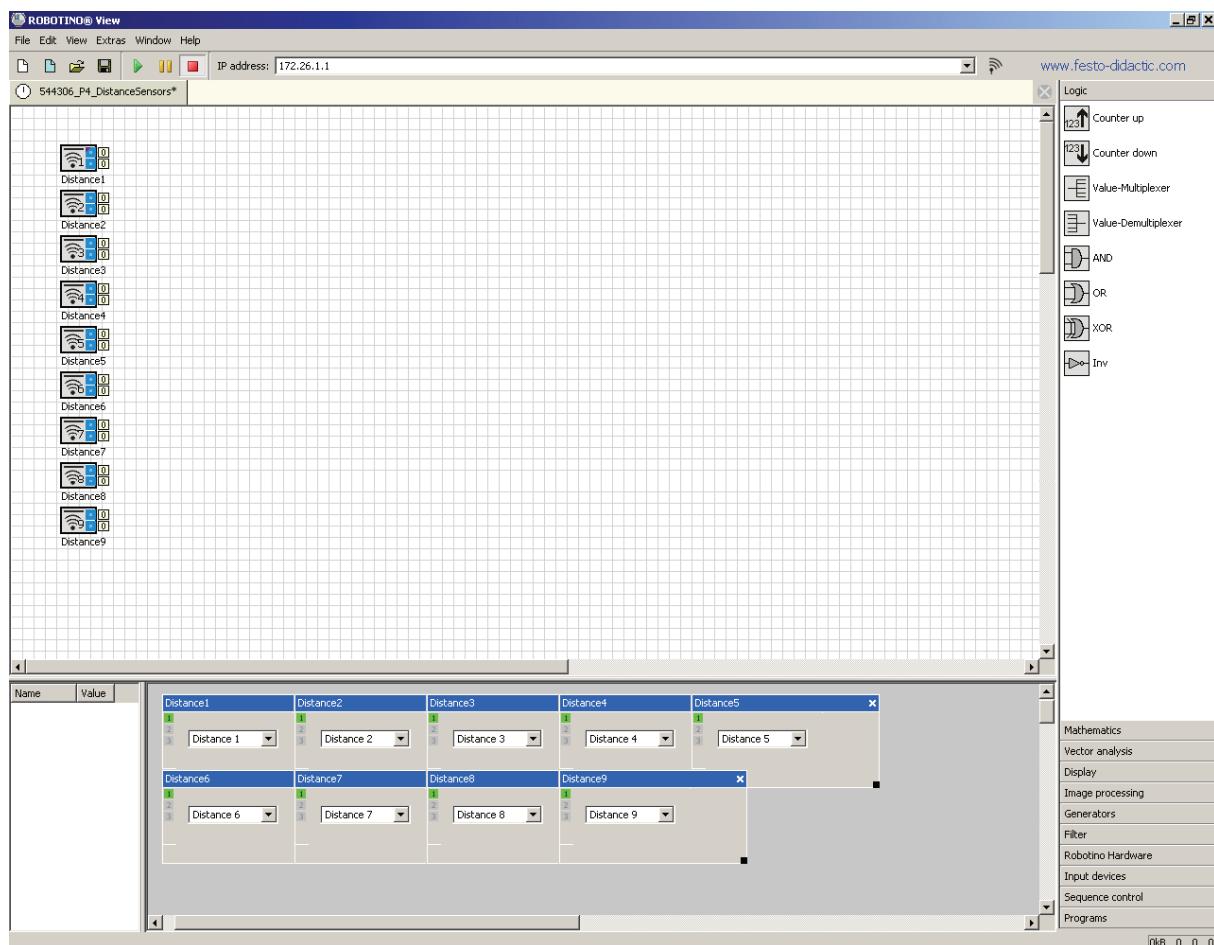
The date display must be switched on in Robotino® View (Ctrl-D).

An obstacle is placed in front of the sensor. If the output value of the sensor changes, the sensor position is determined and its function tested.

## Project 5: Accurately positioned approach of a loading station – solution

Project 5: Accurately positioned approach of a loading station	
Name:	Date:
Determining the position of the distance sensor	Sheet 2 of 4

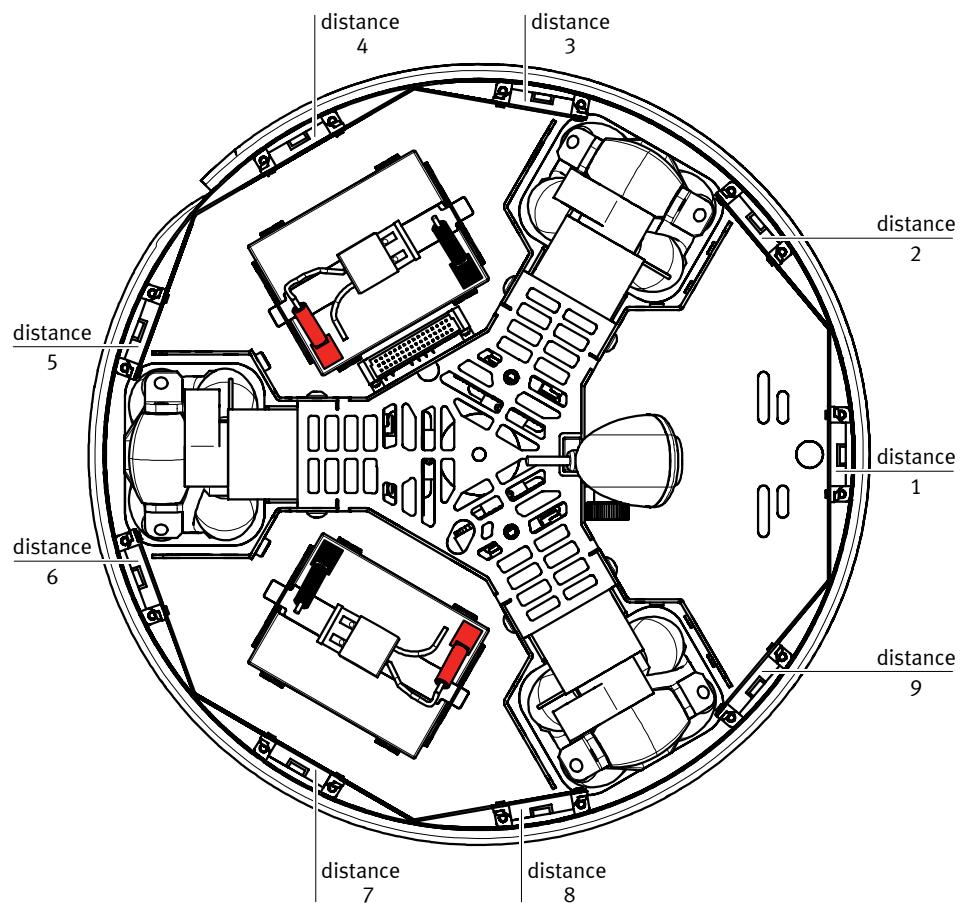
- Create a program in Robotino® View to determine the position of the distance sensors “distance 1” to “distance 9” and save the program.
- Jacked up the Robotino® and switch it on.
- Open a blank function block diagram in Robotino® View.
- Drag a distance sensor onto the function block diagram.
- Assign one of the distance sensors to the function block by opening the function block dialogue via a double click onto the distance sensor symbol.
- Define a distance sensor in the function block dialogue by selecting a sensor “Distance x” from the drop-down menu.
- Expand the program for all distance sensors and save it under the file name 544307\_P5\_00.rvw.



## Project 5: Accurately positioned approach of a loading station – solution

<b>Project 5: Accurately positioned approach of a loading station</b>	
Name:	Date:
Determining the position of the distance sensors	Sheet 3 of 4

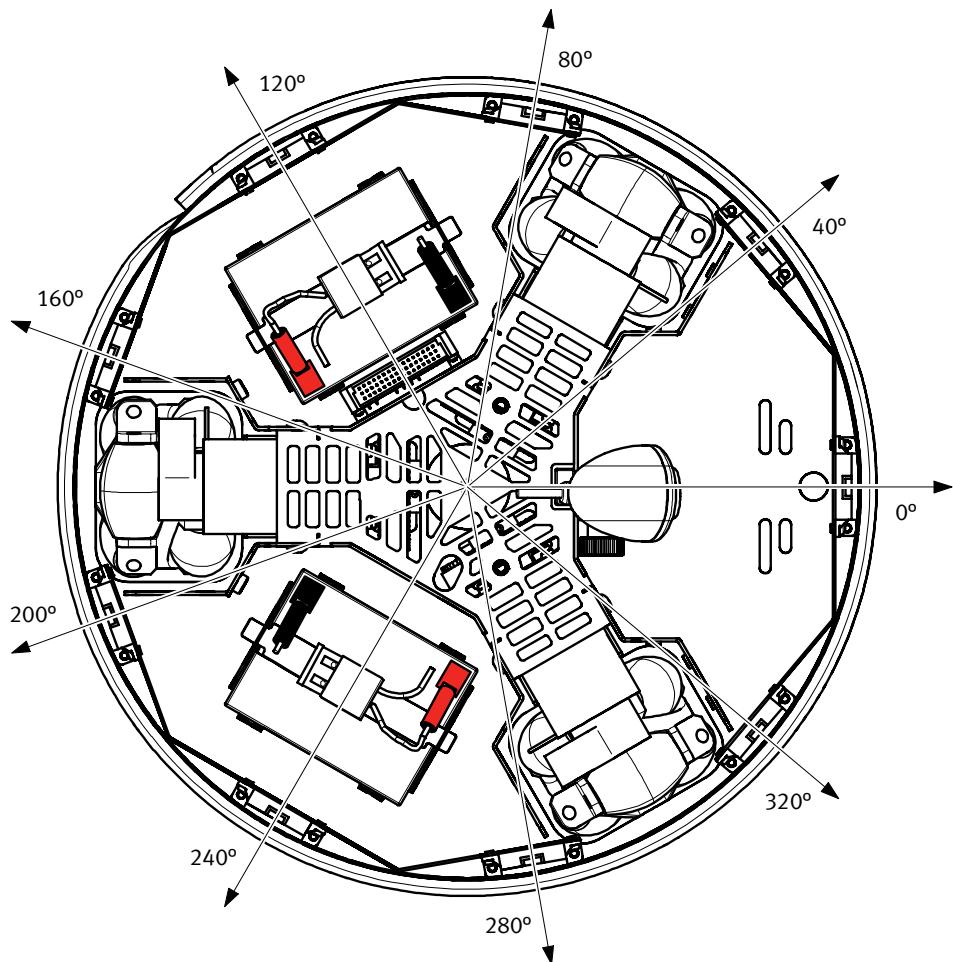
- Start your open-loop control program, determine the position of the sensors and enter the number of the respective distance sensors in the diagram below.
- Complete the check list for the functional test.



Sensor	Function OK	Sensor	Function OK
Distance 1	<input checked="" type="checkbox"/>	Distance 6	<input checked="" type="checkbox"/>
Distance 2	<input checked="" type="checkbox"/>	Distance 7	<input checked="" type="checkbox"/>
Distance 3	<input checked="" type="checkbox"/>	Distance 8	<input checked="" type="checkbox"/>
Distance 4	<input checked="" type="checkbox"/>	Distance 9	<input checked="" type="checkbox"/>
Distance 5	<input checked="" type="checkbox"/>		

<b>Project 5: Accurately positioned approach of a loading station</b>	
Name:	Date:
Determining the position of the distance sensors	Sheet 4 of 4

- The infrared sensors used in the Robotino® emit their light beam vertically. Determine or calculate the beam angle of the individual sensors in relation to the centre of the Robotino®. The beam direction of „distance 1“ sensor is 0°.
- Enter the sensing direction of the individual sensors in the diagram below and enter the respective number of degrees.



Note

In Robotino® View, the respective number of degrees is assigned to each function block for distance sensors. The number of degrees can also be selected here.

## Project 5: Accurately positioned approach of a loading station – solution

<b>Project 5: Accurately positioned approach of a loading station</b>	
Name:	Date:
Initial considerations regarding distance measurement	Sheet 1 of 1

The Robotino® is to stop at a predetermined distance from the loading station.  
Consider how this can be achieved.

1. First determine which distance sensors are required for the distance measurement and make a note of this. Explain your choice.

Solution      “Distance 1” sensor is required if you take into account the fact that the Robotino® is to face the loading station. The other sensors are not required for this exercise.

2. How do you need to proceed to detect the required distance from the loading station in an open-loop control program? Take into account that possibly different distances may be required. Describe how you proceed.

Solution:      For different distances, the respective sensor value needs to be determined. A characteristic curve of the distance sensor must be recorded.

<b>Project 5: Accurately positioned approach of a loading station</b>	
Name:	Date:
Recording of the characteristic curve	Sheet 1 of 3

- Describe your procedure for the recording of the characteristic curve of the distance sensor.

Procedure:

Create and start a program with a distance sensor.

Switch on the display of values in Robotino® View.

Place an obstacle directly in front of the selected distance sensor.

Distance this obstacle from the sensor in steps of 1 cm and record the output values of the sensors.

Transfer the pairs of values determined to a coordinate system and link these with a curve.

Project 5: Accurately positioned approach of a loading station – solution

<b>Project 5: Accurately positioned approach of a loading station</b>	
Name:	Date:
Recording of the characteristic curve	Sheet 2 of 3

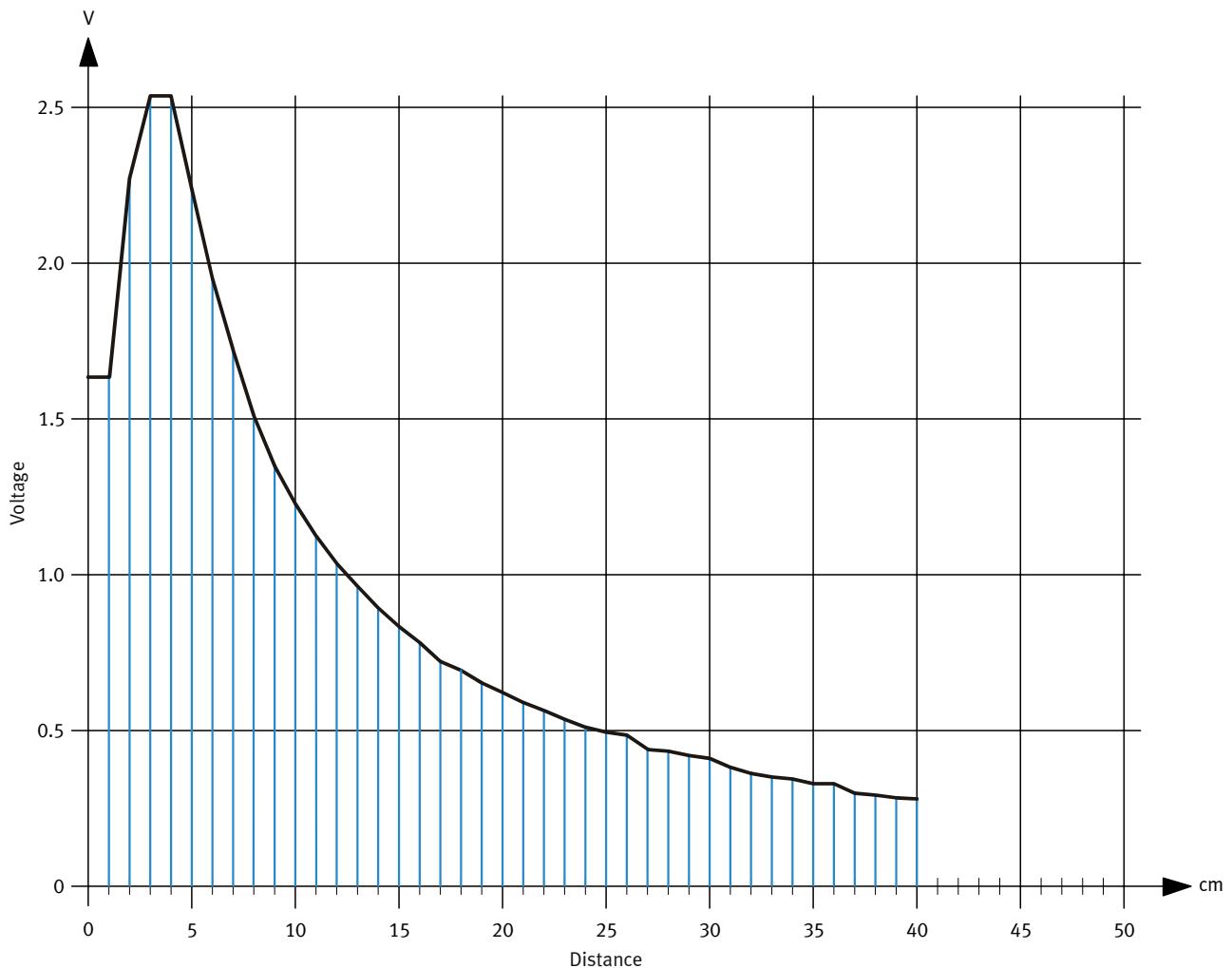
- Record the characteristic curve coordinates and enter these in the table below.

<b>Distance (cm)</b>	<b>Voltage (V)</b>	<b>Distance (cm)</b>	<b>Voltage (V)</b>
1	1.9	21	0.57
2	2.54	22	0.55
3	2.54	23	0.53
4	2.51	24	0.5
5	213	25	0.48
6	1.83	26	0.46
7	1.61	27	0.45
8	1.42	28	0.44
9	1.28	29	0.42
10	1.19	30	0.41
11	1.08	31	0.39
12	1.01	32	0.38
13	0.92	33	0.36
14	0.87	34	0.64
15	0.81	35	0.34
16	0.75	36	0.32
17	0.71	37	0.31
18	0.67	38	0.3
19	0.63	39	0.28
20	0.61	40	0.28

## Project 5: Accurately positioned approach of a loading station – solution

Project 5: Accurately positioned approach of a loading station	
Name:	Date:
Recording of the characteristic curve	Sheet 3 of 3

- Record the characteristic curve and enter it in the coordinate system.

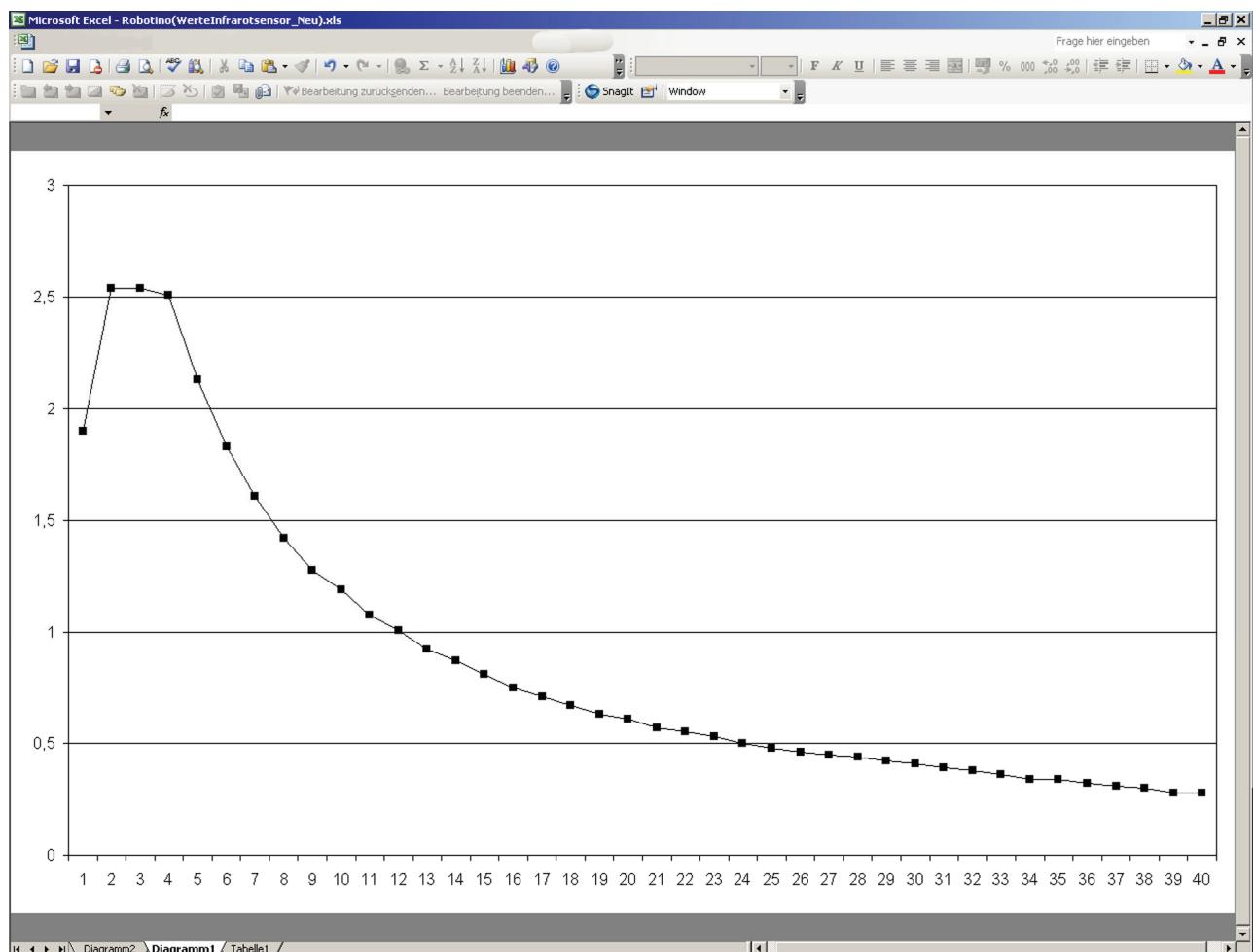


## Project 5: Accurately positioned approach of a loading station – solution

<b>Project 5: Accurately positioned approach of a loading station</b>	
Name:	Date:
Representation of the characteristic curve using MS Excel	Sheet 1 of 1

- Represent the characteristics in the form of a curve using MS-Excel.

Enter the coordinate of the determined pairs of values in an Excel table and represent these in the form of a diagram.



<b>Project 5: Accurately positioned approach of a loading station</b>	
Name:	Date:
Linearisation of the characteristic curve	Sheet 1 of 1

- Determine the characteristic curve area required for the exercise given.
- Linearise the characteristic curve within this area and enter the linearised characteristic curve in your drawing or represent it in an Excel diagram. The distance should be expressed in cm.
- Document the calculation in all the individual steps on the worksheet.

Sample calculation for area of 5 to 10 cm

The formula for a straight line is:

$$D = MX + B$$

Measured values for points P1 and P2

$$D1 = 5$$

$$X1 = 2.13$$

$$D2 = 10$$

$$X2 = 1.19$$

Determining the slope of the line

$$M = \Delta D / \Delta X$$

$$\Delta D = D2 - D1 = 10 - 5 = 5$$

$$\Delta X = X2 - X1 = 1.19 - 2.13 = -0.94$$

$$\text{Therefore } M = 5 / -0.94 = -5.3$$

The following equation is therefore obtained for the first measuring point:

$$5 = -5.3 * 2.13 + B$$

$$5 = -11.3 + B$$

$$\text{Hence } B = 11.3 + 5 = 16.3$$

The equation for this curve is therefore:

$$D = -5.3 * X + 16.3$$

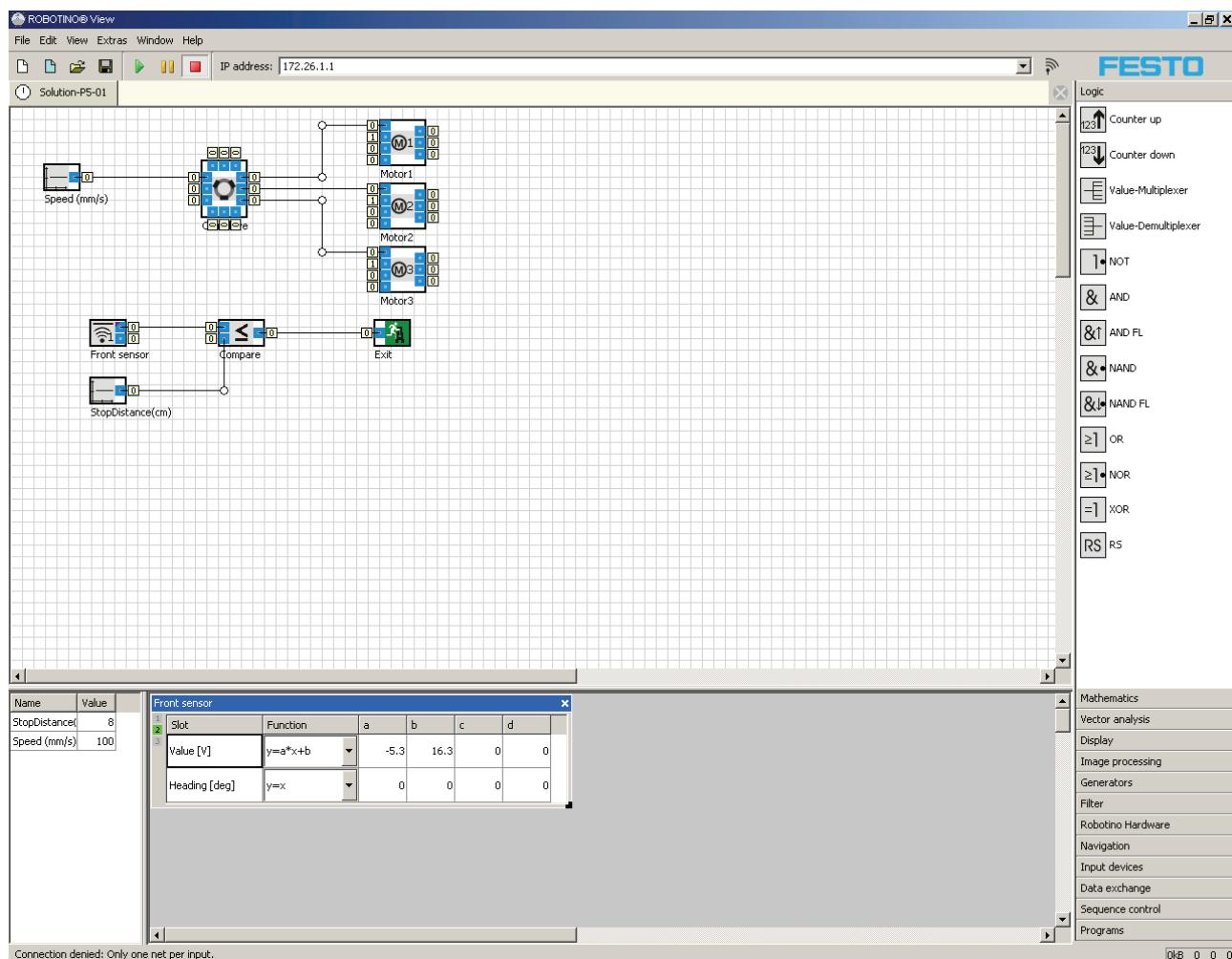
This correlation now applies for all points along this line.

## Project 5: Accurately positioned approach of a loading station – solution

Project 5: Accurately positioned approach of a loading station	
Name:	Date:
Adaptation of the open-loop control program	Sheet 1 of 1

- Adapt the program Aufg-P5-01.rvw so that the Robotino® stops 8 cm in front of an obstacle. Integrate the calculated correlation between the output values and the distance as parameters into the function block dialogue of the distance sensor. Define the desired distance.

### Solution



<b>Project 5: Accurately positioned approach of a loading station</b>	
Name:	Date:
Creating a test plan	Sheet 1 of 1

- Create a test plan for your open-loop control program and describe your test procedure.

Test plan

- The Robotino® is jacked up and switched on.
- The connection via WLAN is established.
- Start the program.
- Move an obstacle towards the distance sensor until the Robotino® drive switches off.
- Check the distance of the obstacle from the sensor.
- Allow the Robotino® to travel towards the obstacle.
- Check the distance.

## Project 5: Accurately positioned approach of a loading station – solution

<b>Project 5: Accurately positioned approach of a loading station</b>	
Name:	Date:
Testing the control program	Sheet 1 of 1

- Test the function of your open-loop control program in accordance with your test plan. Document your findings.

Note                   The step mode in Robotino® View must be set to fast.  
                         Extras → Options → Step Mode → Fast

<b>Test protocol</b>	
Function in jacked up state	OK
Accuracy of distance	8.0 cm
Function in travel mode	OK
Accuracy of distance in travel mode	7.8
Comments	Robotino® travels further than necessary (braking process?)

Date, signature

---

<b>Project 5: Accurately positioning approach of a loading station</b>	
Name:	Date:
Testing the accuracy of the stop process	Sheet 1 of 1

- Travel towards the obstacle using different speeds and measure the accuracy of the distance maintained from the obstacle. Compare the distances with regard to the different speeds.
- Record the cause for the different distances.

Note

You can change the speed of travel of the Robotino® by assigning a higher value to the "speed (mm/s)" constant.

The step mode in Robotino® View must be set to fast.

Extras → Options → Step Mode → Fast

Speed	Measured distance from the obstacle
20	6.8
50	6.1
100	5.4
200	3.2

Explanation

The reasons for the reduction in distance to the obstacle are as follows.

A time delay is created as a result of:

1. The time required for the transfer and processing of signals in the Robotino®.
2. The time required for the transfer of data from the Robotino® via the WLAN.
3. The time required by Robotino® View to process the data.
4. The time required to transfer back the data and transmit it to the drives.

This delay is independent of the speed of travel of the Robotino® and is virtually identical at any speed. The braking process really plays a subordinate role since the Robotino® is not slowed down but stopped immediately although, depending on the speed, the Robotino® covers a different distance within this time delay.

## Project 5: Accurately positioned approach of a loading station – solution

<b>Project 5: Accurately positioned approach of a loading station</b>	
Name:	Date:
Project documentation	Sheet 1 of 1

- Compile the project documentation.

<b>Exercises</b>	<b>Required documents</b>
Determining the position of the distance sensors	Worksheet 3, program file, hardcopy of program
Recording the characteristic curve	Worksheet 2
Representation of the characteristic curve using MS Excel	Excel file, printout of diagram
Linearisation of the characteristic curve	Worksheet
Adaptation of the control program	Program, hardcopy of program
Creating a test plan	Work plan
Test	Test protocol

Project 5: Accurately positioned approach of a loading station – solution

## Project 6

### Approaching an obstacle and maintaining a defined distance – solution

Project 6: Approaching an obstacle and maintaining a defined distance	
Name:	Date:
Approaching and readjusting to a distance of 60 mm	Sheet 1 of 4

- Create and test a program in Robotino® View whereby the Robotino® stops in front of an obstacle at a distance of 60 mm.
- The Robotino® is to automatically adjust the distance to a setpoint value of 60 mm if the obstacle is moved.
- Move the obstacle whilst running the program and observe what happens. Explain the possibilities of optimising the program.
- Explain the difference between the Distance Sensor program from project 5.
  
- Create a test a program in Robotino® View whereby the Robotino® stops 60 mm in front of an obstacle.
  
- Save the program from project 5 (distance sensor) under another name.
- Determine the direct sensor value from the distance sensor diagram (project 5), and enter this as a minimum distance.

#### Direct sensor value

Minimum distance of 60 mm corresponds to the sensor value of 2 V.

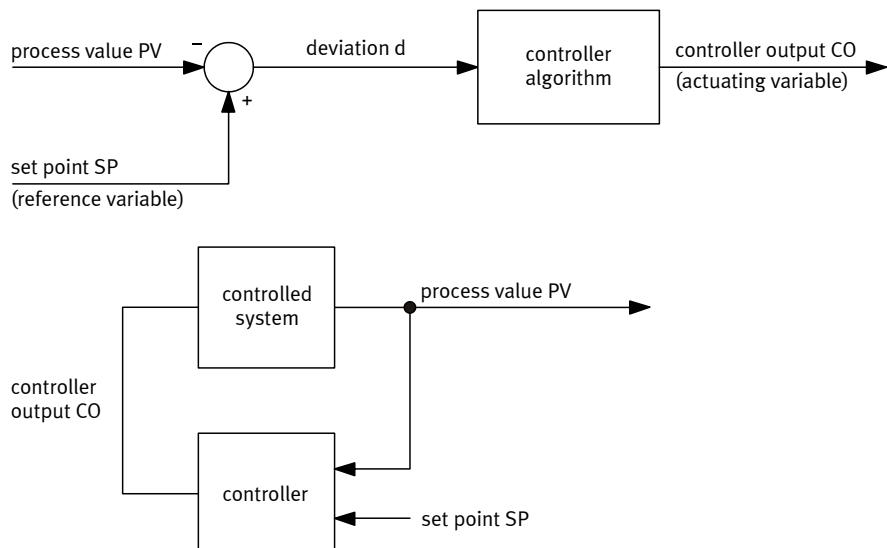
#### Note

Make sure that the correct value is entered in the function block diagram of the distance sensor.

## Project 6: Approaching an obstacle and maintaining a defined distance – solution

<b>Project 6: Approaching an obstacle and maintaining a defined distance</b>	
Name:	Date:
Approaching and readjusting a distance of 60 mm	Sheet 2 of 4

- If the obstacle is moved, the Robotino® is to automatically readjust the distance to a setpoint value of 60 mm.
- Answer the following questions using the block diagram:



Questions	Answer
What is the controlled variable?	Sensor value of front distance sensor
How is the actual value of the controlled variable measured?	Sensor transmits a signal to robot controller via analogue input.
What is the setpoint value of the controlled variable?	Setpoint value = 2.0 V
What is the controlled system?	Robotino®
What is the disturbance variable?	Displacement of the obstacle
Determine the system deviation. Use the functionality of a P-controller for the solution.	<p>System deviation = 2.0 V – current sensor value</p> <p>System deviation is multiplied by a velocity factor. This factor is to be established by means of experiments and largely determines the control behaviour.</p> <p>The control algorithm is supplied by a P controller</p>
Determine the manipulated variable .	The manipulated variable is the speed whereby the Robotino® is moved forward or backward.

## Project 6: Approaching an obstacle and maintaining a defined distance – solution

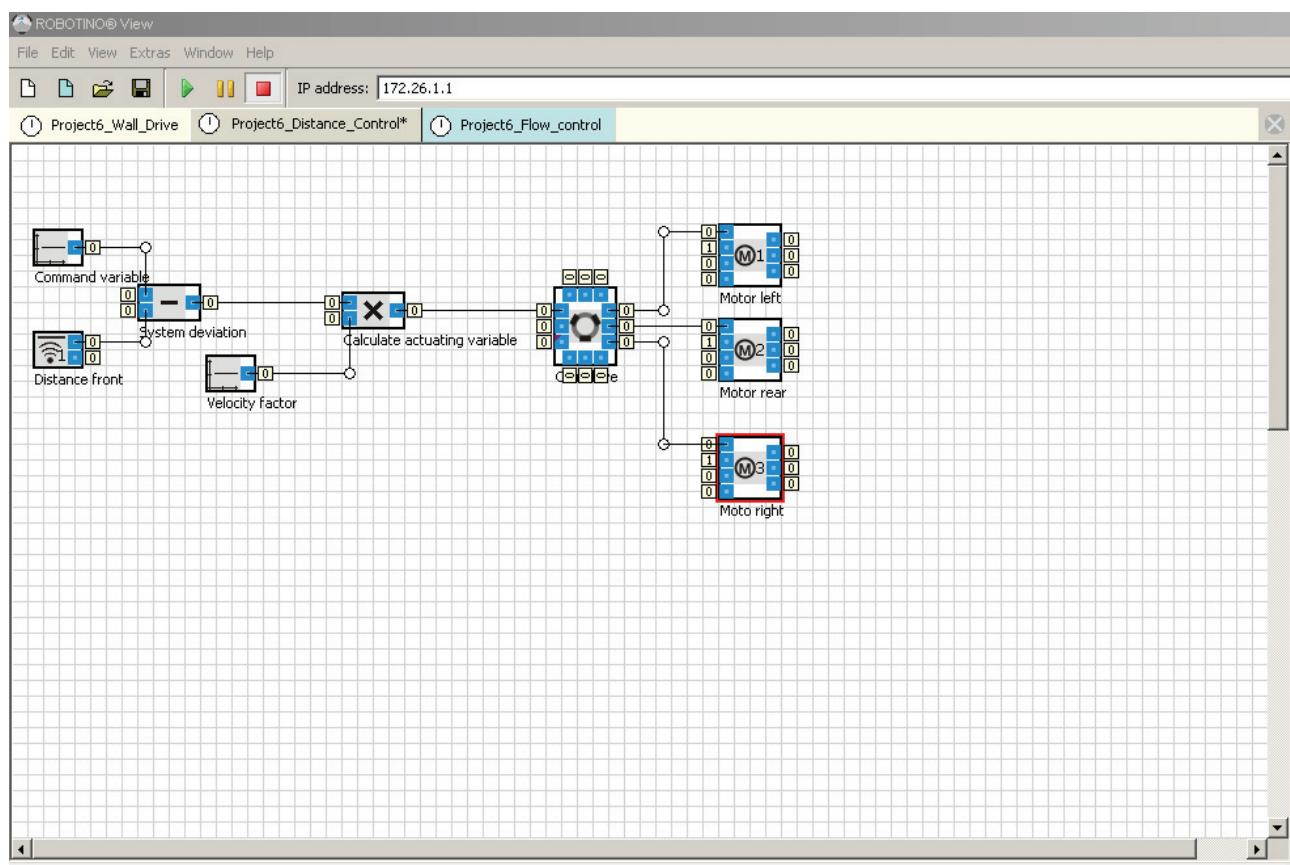
Project 6: Approaching an obstacle and maintaining defined distance	
Name:	Date:
Approaching and readjusting to a distance of 60 mm	Sheet 3 of 4

- Create and test the program

Note

Reference variable = 2 V / Velocity factor = 75

The manipulated variable is calculated from the system deviation via multiplication by a velocity factor. For example, if the Robotino® is positioned 10 cm in front of an obstacle, the Robotino® approaches at a speed of 90 [mm/s]. However, if the Robotino® is only 7 cm from the obstacle, it only travels forward at a speed of 30 [mm/s].



<b>Project 6: Approaching an obstacle and maintaining a defined distance</b>	
Name:	Date:
Approaching and readjusting a distance of 60 mm	Sheet 4 of 4

- Move the obstacle whilst running the program and observe what happens. How can you improve the closed-loop control?

#### **Change if the obstacle is moved, if closed-loop control is improved**

If the obstacle is moved, the desired position of 60 mm is approached again. Both the setpoint value and actual value are adjusted. If the actual value is changed, in this case the position of the obstacle, correction is effected.

The sensor value is not 100% steady, hence there is continuous correction. For the purpose of optimisation, very minor system deviations may be ignored. This can be achieved by means of adjusting the velocity factor.

- Explain the difference between the **distance sensor** program from project 5.

#### **Difference compared to the distance sensor program in project 5**

The distance sensor program is an open-loop control program. The sequence is stopped as soon as the distance limit is reached. The process is not closed-loop controlled.

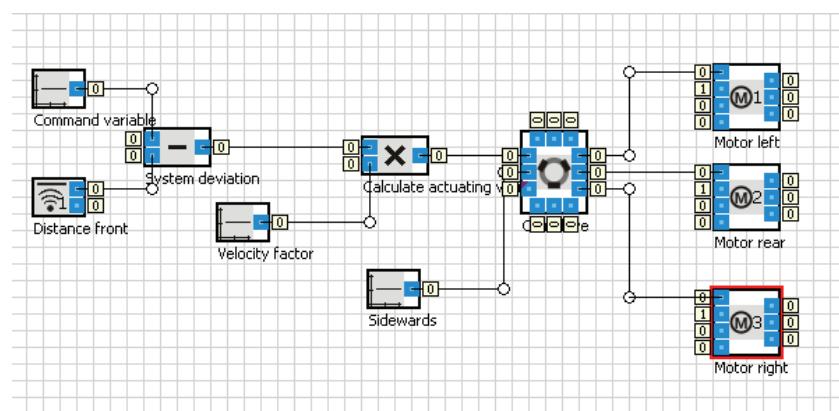
## Project 6: Approaching an obstacle and maintaining a defined distance – solution

<b>Project 6: Approaching an obstacle and maintaining a defined distance</b>	
Name:	Date:
Travelling along a wall	Sheet 1 of 3

- Create a program whereby the Robotino® travels sensor-guided along a wall at a distance of 60 mm with its line of vision facing towards the wall.
- Test the program using the following cases:
  - The start position of the Robotino® is 60 mm away from the wall
  - The start position of the Robotino® is more than 100 mm away from the wall
- Create a sequence program so that the Robotino® approaches the wall facing it up to a distance of 60 mm and then travels along the wall at a constant distance of 60 mm.
- Test and explain the sequence program.
- Add lateral travel to the existing program and describe what happens.
- Test the program using the following cases:
  - The Robotino® start position is 60 mm away from the wall
  - The Robotino® start position is more than 100 mm away from the wall

### Note

Make sure that the line of vision of the Robotino® is towards the wall when starting.  
Sample value for lateral travel = 100 [mm/s].



<b>Project 6:Approaching an obstacle and maintaining a defined distance</b>	
Name:	Date:
Travelling along a wall	Sheet 2 of 3

#### **What happens? Case 1**

The Robotino® travels along the wall.

#### **Was happens? Case 2**

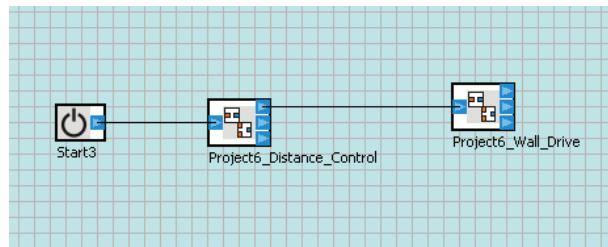
The Robotino® approaches the obstacle in the form of a curve since forward travel and lateral travel are executed simultaneously. The sequence of steps “straight-ahead travel and lateral travel” does not take place. The motors are actuated simultaneously which results in the superimposing of the straight-ahead and lateral travel processes. The superimposing of processes causes a mixture of straight- ahead and lateral, i.e. travel in the form of a curve.

If the Robotino® has reached the wall at a specified distance, it maintains the distance for a specified time.

Since lateral movement is not precisely in parallel with the wall, the orientation to the wall changes and cannot be corrected via the front distance sensor.

<b>Project 6: Approaching an obstacle and maintaining a defined distance</b>	
Name:	Date:
Travelling along a wall	Sheet 3 of 3

- Create a sequence program so that the Robotino® approaches a wall facing it up to a distance of 60 mm and then along the wall at a constant distance of 60 mm.
- Test and explain the sequence program.
  
- Open the programs **Distance.rvm** and **WallTravel.rvm** and open a new sequence program.
- First start the Distance program and then the WallTravel program.



#### Explanation of the program

Once started, the Robotino® travels in the direction of the wall until it reaches a minimum distance of 60 mm. Then the second program is started and the Robotino® travels along the wall.

When the end of the wall is reached and if the program is not stopped beforehand, the robot travels ahead in a direction of approx. 45 °.

Project 6: Approaching an obstacle and maintaining a defined distance – solution

## Project 7

### Circling a station and approaching various transfer positions – solution

Project 7: Circling a station and approaching various transfer positions	
Name:	Date:
Initial considerations	Sheet 1 of 4

- Consider:
  - How many degrees of freedom are required to approach and circle an obstacle. Use the positional sketch for this
  - How, using an omnidrive function block, you can generate a circular path and create the appropriate program.
  - How and, by using which distance sensors, can you monitor the distance to the obstacle whilst circling it
- What should the value be of the two distance sensors 2 and 9 in order for the camera to be aligned with the obstacle? How can you use the sensor values for closed-loop control of the orientation towards the obstacle?
- Consider how many degrees of freedom are required to approach and circle the obstacle (circling of the station).

#### Note

Use the positional sketch from the problem definition and the program from project 5.

#### How many degrees of freedom are required to circle a station?

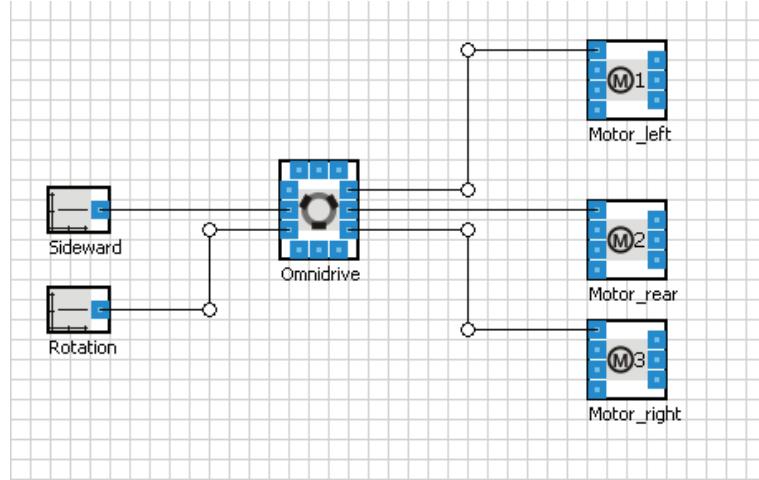
Three degrees of freedom are required to circle a station. The Robotino® travels in the x- and y-direction to the input distance (distance sensor 1) from the station. In order to generate a circular path; it requires a rotation around the z-axis (Omega setpoint). However, to ensure that the Robotino® does not rotate around its own axis (z-axis), it has to simultaneously continue travelling laterally in the y-direction.

## Project 7: Circling a station and approaching various transfer positions – solution

Project 7: Circling a station and approaching various transfer positions	
Name:	Date:
Initial considerations	Sheet 2 of 4

- Consider how, using the omnidrive function block, you can generate a circular path and create the appropriate program.

Circular movements



### What happens and why?

The processes lateral travel and rotating are superimposed. This creates a circular movement of the robot system, which you can create by means of the two velocity constants Lateral and Rotation, e.g.

Lateral = 100 [mm/s]

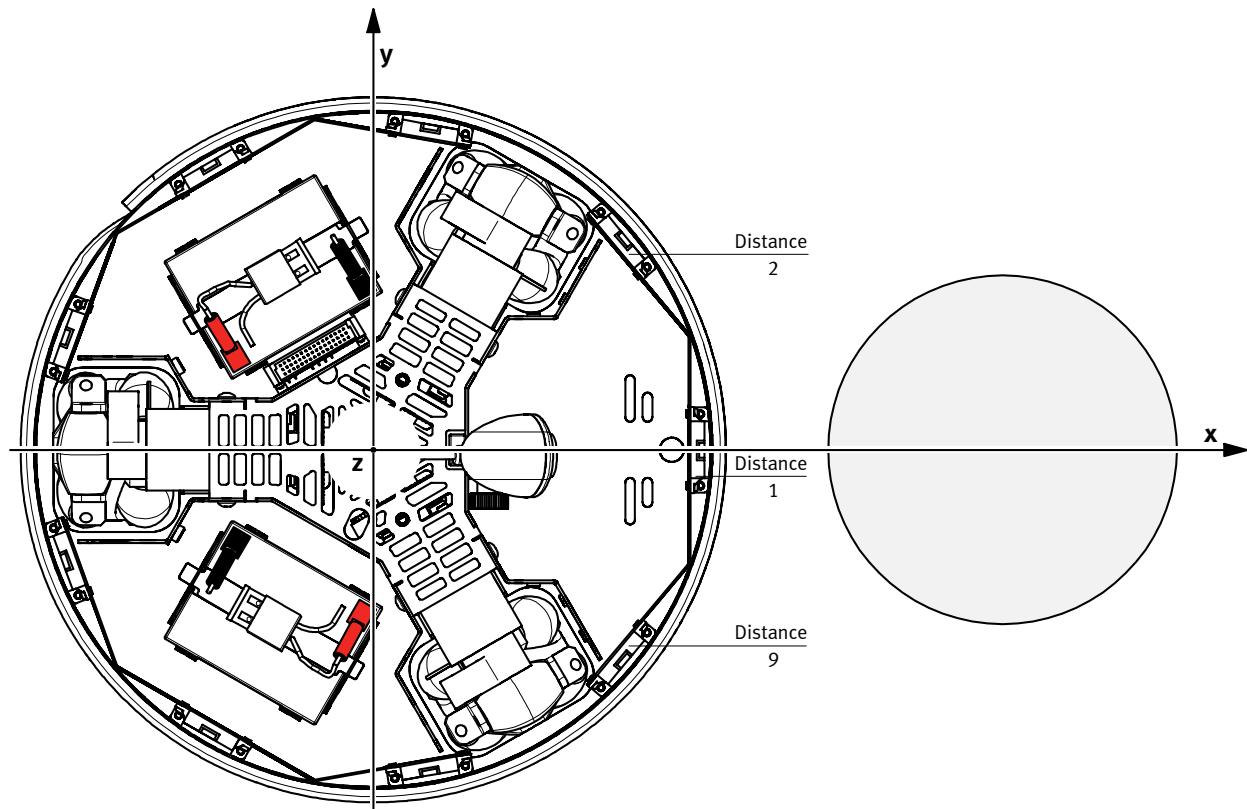
Rotating = -15 [degrees/s]

Project 7:Circling a station and approaching various transfer positions	
Name:	Date:
Initial considerations	Sheet 3 of 4

- Consider how and with which distance sensors you can monitor the distance from the obstacle when circling the station. Use the positional sketch for this.

#### Monitoring of distance – which sensors?

Distance sensor 1 is used to monitor the vertical distance to the obstacle. Distance sensors 2 and 9 are used to check the deviation from the orientation to the obstacle (see illustration below).



<b>Project 7: Circling a station and approaching various transfer positions</b>	
Name:	Date:
Initial considerations	Sheet 4 of 4

- What should the value be of the two distance sensors 2 and 9 in order for the camera to be aligned with the obstacle? How can you use the sensor values for closed-loop control of the orientation towards the obstacle?

**Values of distance sensors 2 and 9 – use of sensor values for closed-loop control of the orientation towards the obstacle**

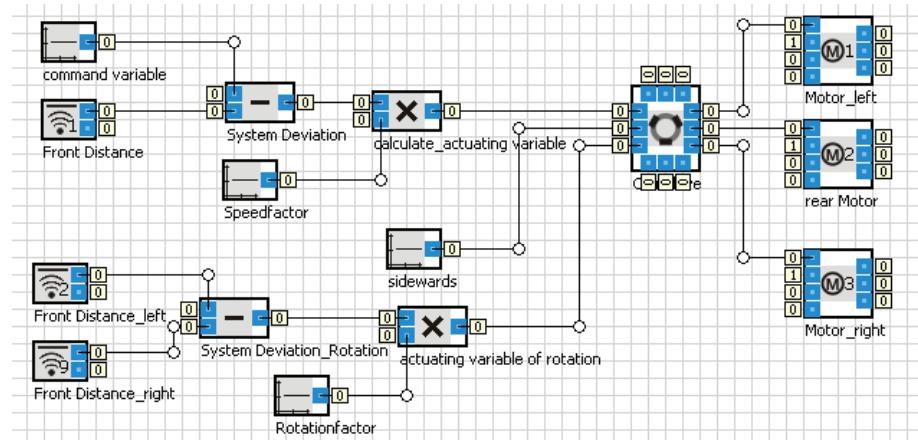
The values of the two distance sensors must be identical.

The controlled variable is the orientation towards the obstacle and is measured by means of the difference between the two distance sensors 2 and 9.

## Project 7: Circling a station and approaching various transfer positions – solution

Project 7: Circling a station and approaching various transfer positions	
Name:	Date:
Program	Sheet 1 of 2

- Position the Robotino® at a distance of 6 cm from the obstacle so that the camera faces in the direction of the obstacle. Add a rotational movement to the program WallTravel from project 6, which maintains the orientation of the camera vertical to the obstacle throughout.
- Test and optimise your program.
- Create a sequence program so that the Robotino® approaches the station up to a distance of 6 cm and then circles it at a distance of 6 cm. Test your program.



### Note

The solution consists of a path control using two P-controllers, which on the one hand control the distance to the obstacle and on the other hand the alignment with the obstacle.

Using the three constants

velocity factor = 75

lateral = 100

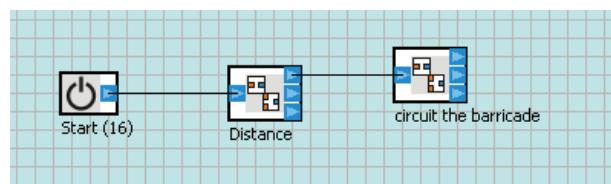
rotational factor = 30

you can significantly influence the motion behaviour.

## Project 7: Circling a station and approaching various transfer positions – solution

Project 7: Circling a station and approaching various transfer positions	
Name:	Date:
Program	Sheet 2 of 2

- Create a sequence program so that the Robotino® approaches the station up to a distance of 6 cm and then circles it at a distance of 6 cm. Test your program.
- Select the program **Distance** from project 5 and combine it with the above program.



## Project 8

### Path tracking of an automated guided vehicle system using an analogue inductive sensor – solution

<b>Project 8: Path tracking of an automated guided vehicle system using an analogue inductive sensor</b>	
Name:	Date:
Preparatory work	Sheet 1 of 3

To make it easier for you to carry out your task, you should first of all prepare a work plan for the assembly and commissioning of the sensors. Draw up a list of tools and materials prior to starting assembly.

Subsequently use your work plan as a check list for project documentation.

Work plan	Work step	Time required	Completed
	Mounting of sensor on the Robotino®		
	Connection of the sensor cable to the I/O strip and sensor		
	Establishing access to sensor data via Robotino® View.		
	Functional test of sensor.		

Overall time required: \_\_\_\_\_

Material & tools required	Quantity	Tool / Material
	1	Wire cutter
	1	Crimping tool
	3	Cable end sleeves 0.25 mm <sup>2</sup>
		Cable binder

<b>Project 8: Path tracking of an automated guided vehicle system using an analogue inductive sensor</b>	
Name:	Date:
Preparatory work	Sheet 2 of 3

- Determine the necessary sensor connections in accordance with the Robotino®-manual, operating instructions and data sheets of the sensor. Explain your choice.

Solution            0 V, 24 V and the signal line of the sensor.

Determine the colours of the required wires and their designations in the operating instructions. Explain your choice.

Solution            24 V BN = Brown  
                    0 V BU = Black  
                    Signal line: BK = Black

Explanation:

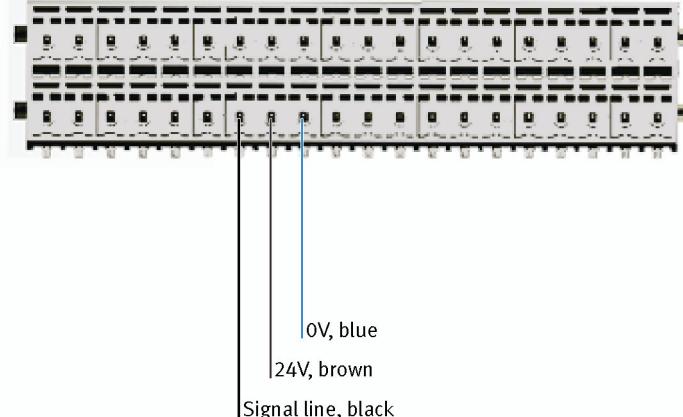
The black signal line must be selected since the I/O interface of the Robotino® exclusively processes voltage values. The white signal line supplies a current intensity signal in amperes.

<b>Project 8:</b> <b>Path tracking of an automated guided vehicle system using an analogue inductive sensor</b>	
Name:	Date:
Preparatory work	Sheet 3 of 3

The sensor is to be connected to the analogue input AI0. Create a connection diagram for the I/O interface.

- Enter the wires on the drawing shown below and indicate the characteristics and colour.

Solution



<b>Project 8: Path tracking of an automated guided vehicle system using an analogue inductive sensor</b>	
Name:	Date:
Mounting of sensor and connection to the I/O interface	Sheet 1 of 2

- Attach the sensor in accordance with the Robotino® manual. What should be noted here?

Solution Since the sensing range of the sensor is 0 - 6 mm, the sensor must be mounted correspondingly low.

Connect the wires shown in your drawing to the contact strip accordingly.

<b>Project 8: Path tracking of an automated guided vehicle system using an analogue inductive sensor</b>	
Name:	Date:
Mounting of sensor and connection to the I/O interface	Sheet 2 of 2

- Carry out a functional test of the sensor. To do so, establish access to the sensor data in Robotino®View and represent the input values using the oscilloscope. Document how you proceed.

#### Solution

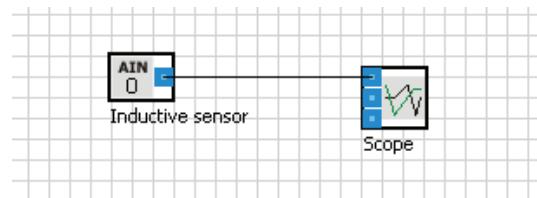
Open Robotino®View.  
Open new function block diagram.  
Drag an analogue input onto the workspace.  
Assign input AIN0 to the analogue input with the help of the function block dialogue.  
Connect the output of the input block to an input of the oscilloscope.

Carry out a functional test of the sensor and describe how you proceed.

A metal object is to be moved towards the sensor. The sensor function is ensured if the output voltage changes within the sensing range.

Save the function block diagram.

#### Solution-P6-01.rvw



Project 8: Path tracking of an automated guided vehicle system using an analogue inductive sensor – solution

<b>Project 8:</b> <b>Path tracking of an automated guided vehicle system using an analogue inductive sensor</b>	
Name:	Date:
Experiment: Sensing behaviour of the sensor	Sheet 1 of 3

- Check the reaction behaviour of the inductive sensor in respect of different materials (alloys) and determine the correlation between distance and surface of the object and the sensor values.

Use Euro coins ranging from 1 Cent of 2 Euros as test objects.

Place one of the coins on the surface within the sensing range of the sensor and enter the output voltage measured (V) in the table below. In order to determine the values for different distances, you should place a non-metallic base (paper, cardboard or plastic material) underneath the coin used.

Value	Diameter (mm)	Thickness (mm)	Material	Magnetic	Output voltage (V)	Output voltage (V) with base
1 Cent	16.25	1.67	Steel with copper coating (Fe, Cu)	Yes	<b>8.88</b>	<b>5.08 / 2.72</b>
2 Cent	18.75	1.67	Steel with copper coating	Yes	<b>7.92</b>	<b>5.16 / 2.80</b>
5 Cent	21.25	1.67	Steel with copper coating	Yes	<b>6.40</b>	<b>5.36 / 2.92</b>
10 Cent	19.75	1.93	Nordic gold (Cu89 Al5 Zn5 Sn1)	No	<b>10.20</b>	<b>7.12 / 3.4</b>
20 Cent	22.25	2.14	Nordic gold (Cu89 Al5 Zn5 Sn1)	No	<b>9.84</b>	<b>5.80 / 3.00</b>
50 Cent	24.25	2.38	Nordic gold (Cu89 Al5 Zn5 Sn1)	No	<b>7.92</b>	<b>4.76 / 2.72</b>
1 Euro	23.25	2.33	External: Brass-Ni (Cu75 Zn20 Ni5) Internal: Cu-Ni, Ni, Cu-Ni coated	Weak	<b>9.76</b>	<b>3.36 / 1.88</b>
2 Euro	25.75	2.20	External: Cu-Ni (Cu75 Ni25), Internal: Brass-Ni, Ni, brass-Ni coated	Weak	<b>4.40</b>	<b>3.24 / 2.96</b>

Note

The sensor values vary according the attachment height of the sensor. This value table merely serves as an example.

<b>Project 8: Path tracking of an automated guided vehicle system using an analogue inductive sensor</b>	
Name:	Date:
Experiment: Sensing behaviour of the sensor	Sheet 2 of 3

- Analyse the values determined in respect of alloy, surface and distance to the sensor and make a note of these.

Solution

Of identical alloy and same height (1, 2, 5 Cent):

The diameter changes the output voltage: The larger the area, the smaller is the output voltage.

Different alloys influence the sensor signal differently:

A 1 Euro coin generates a higher signal than a 2 Euro coin although the 1 Euro coin is thicker and the area of the 2 Euro coin is greater. The same applies in the case of a 2 Cent and 10 Cent coin.

The smaller the output voltage using an identical coin, the smaller is the distance between the coin and sensor.

Project 8: Path tracking of an automated guided vehicle system using an analogue inductive sensor – solution

<b>Project 8:</b> <b>Path tracking of an automated guided vehicle system using an analogue inductive sensor</b>	
Name:	Date:
Experiment: Sensing behaviour of the sensor	Sheet 3 of 3

What conclusions can be drawn from the two series of measurements in the table below with regard to the sensor position if the objects of both series of measurements were placed on the same base?

Value	Diameter (mm)	Thickness (mm)	Material	Magnetic	Output voltage Measurement 1	Output voltage Measurement 2
1 Cent	16.25	1.67	Steel with copper coating (Fe, Cu)	Yes	8.88	5.08
2 Cent	18.75	1.67	Steel with copper coating	Yes	7.92	5.16
5 Cent	21.25	1.67	Steel with copper coating	Yes	6.40	5.36

Solution

The sensor is mounted at a different height from the base for the two series of measurements. The sensor is mounted at a greater distance from the base in the case of measurement series 1.

<b>Project 8:</b> <b>Path tracking of an automated guided vehicle system using an analogue inductive sensor</b>	
Name:	Date:
Recording the characteristic curve	Sheet 1 of 2

Determine the parameters for tracking the guide line in order to develop a strategy. Record the characteristic curve of the sensor when travelling along the aluminium strip.

For the object to be measured, use a 5cm wide aluminium strip, which is glued to a level surface. This aluminium strip should correspond in width and material to the one used subsequently as guideline for the Robotino®.

#### Solution

Travel across the aluminium strip with the Robotino® step-by-step and record the sensor data.

Use the function block diagram 'Aufg-P8-02.rvw' provided for this. Adapt the program to your requirements accordingly.

Document your changes and save the program together with your changes.

#### Solution

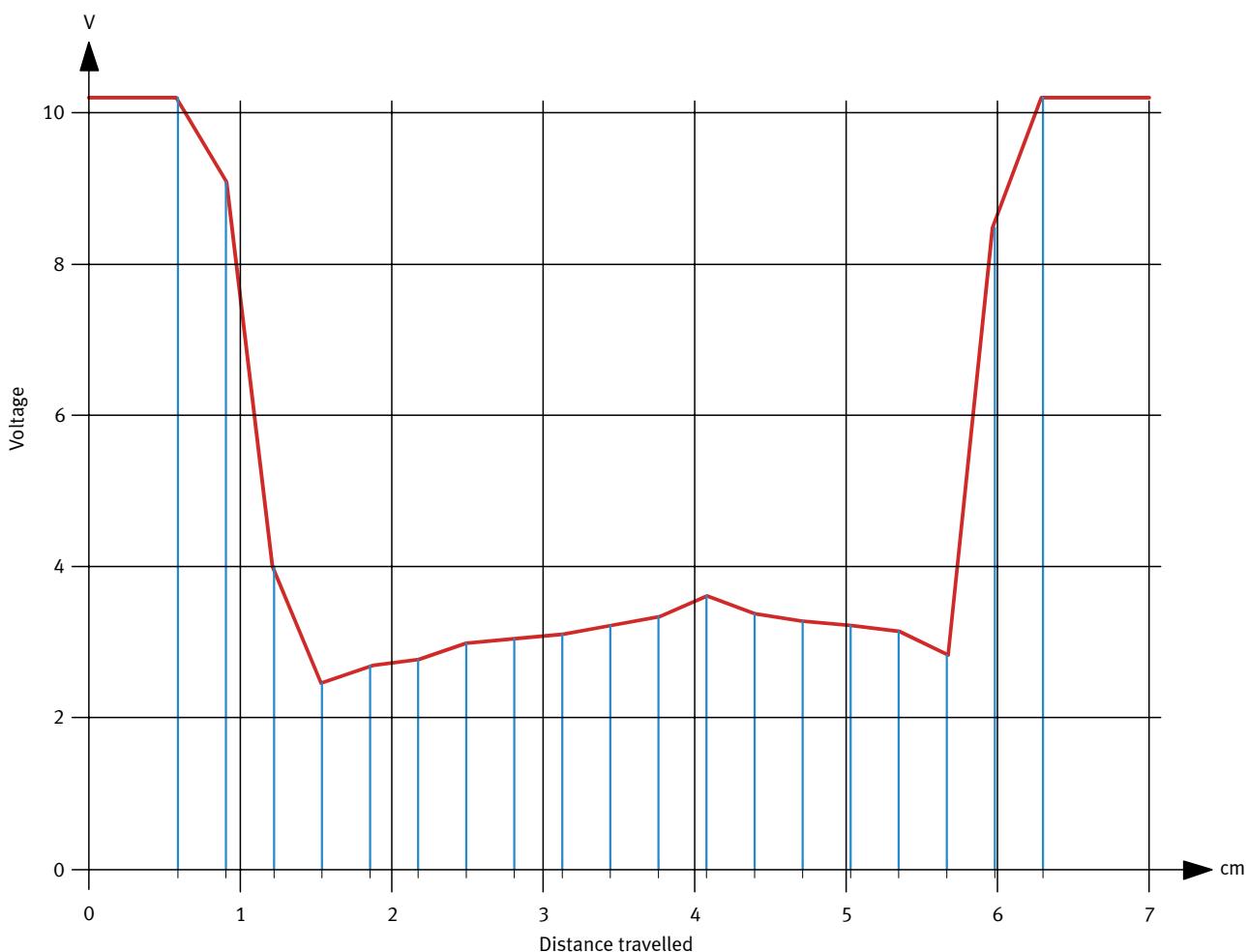
Function block	Value
Constant 'path'	2
Constant 'duration'	5
Analogue input	AIN0

## Project 8: Path tracking of an automated guided vehicle system using an analogue inductive sensor – solution

<b>Project 8: Path tracking of an automated guided vehicle system using an analogue inductive sensor</b>	
Name:	Date:
Recording the characteristic curve	Sheet 2 of 2

Record the characteristic curve and enter the values in the diagram below.

### Solution



### Notes

The voltage values determined may vary depending on the installation height above the base. The curve pattern is, however, always similar.

<b>Project 8: Path tracking of an automated guided vehicle system using an analogue inductive sensor</b>	
Name:	Date:
Developing a control strategy	Sheet 1 of 1

- Examine the value pattern determined and develop possible strategies to track the line. Make a note of your strategies. Select a strategy and explain your choice.

#### Solution

The value pattern is virtually identical across large sections of the aluminium strip. Within this range, it is not possible to identify from the sensor value whether the Robotino® is in the middle of the strip or to the left or right of it. The sensor generates the same values whether the Robotino® travels beyond the edge of the strip to the right or left side.

Since we are only working with one sensor here, it is not possible to identify whether the robot travels beyond the righthand or lefthand edge. You therefore need to travel along with one of the two wheels.

The following strategies are possible:

##### 1. Travelling along righthand edge:

With an output voltage of  $> 9.00$ , rotate in clockwise direction & continue travel

With an output voltage of  $< 5.00$ , rotate in anti-clockwise direction & continue travel

##### 2. Travel along lefthand edge:

With an output voltage of  $< 9.00$ , rotate in clockwise direction & continue travel

With an output voltage of  $> 5.00$ , rotate in anti-clockwise direction & continue travel,

#### Selection and evaluation

##### Re: 1 and 2:

The value pattern of the sensor only changes significantly at the edge of the aluminium strip. Reaction to changes is therefore only possible there.

<b>Project 8: Path tracking of an automated guided vehicle system using an analogue inductive sensor</b>	
Name:	Date:
Analysis of the enclosed control program	Sheet 1 of 1

- Analyse the enclosed sequence program 'Aufg-P8-04.rvw'.  
Place the Robotino® on the aluminium strip and execute the sequence program.  
Describe the behaviour of the Robotino®.

**Note** The limit values of the sensor signal used in the same program may vary depending on the installation height of the sensor and the type of aluminium strip in your configuration. If necessary, adapt the subprograms to these conditions.

**Solution** The Robotino® travels forward and stops.  
One of the following subprograms is then executed.  
Either the Robotino® turns to the right by a defined distance if it is far from the righthand edge and then stops.  
Or the Robotino® turns to the left by a defined distance if it is close to the righthand edge and then stops.

This process is repeated until the Robotino® has reached the target (an obstacle).

Project 8: Path tracking of an automated guided vehicle system using an analogue inductive sensor	
Name:	Date:
Optimisation of the control program	Sheet 1 of 1

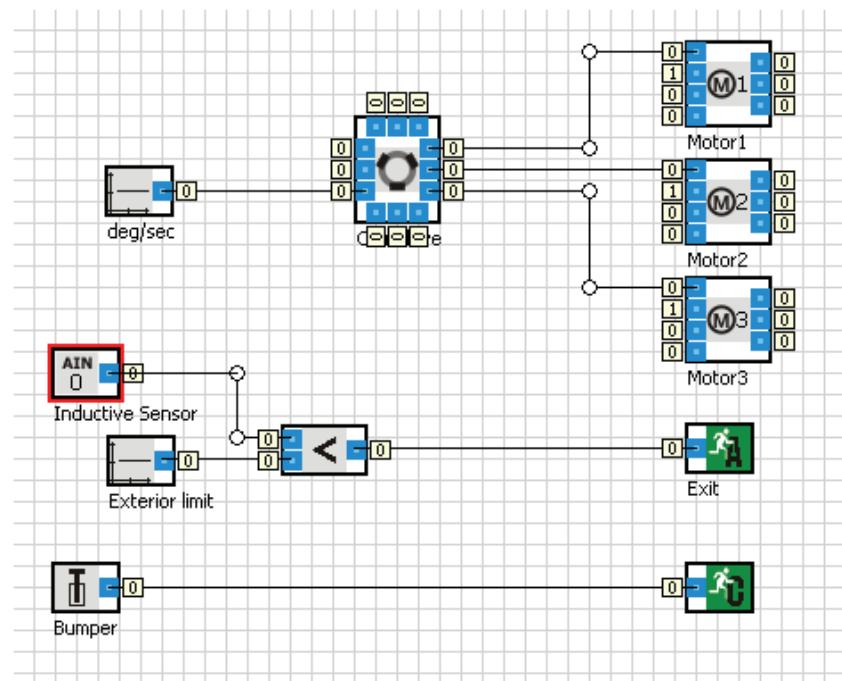
- You are to optimise the sequence program so that a smoother overall sequence is achieved and the defined rotation used in the sample program is replaced by a rotation dependent on the output voltage.
- Determine the subprograms to be modified and record your solution; then amend the respective subprogram.

#### Solution

The subprograms 'Aufg-P8-04b.rvw' and 'Aufg-P8-04c.rvw' must be modified. The rotational movement must be changed so that it is terminated when the internal or external limit value is reached.

#### P8-04b.rvw solution

#### Rotation to the right: Subprogram

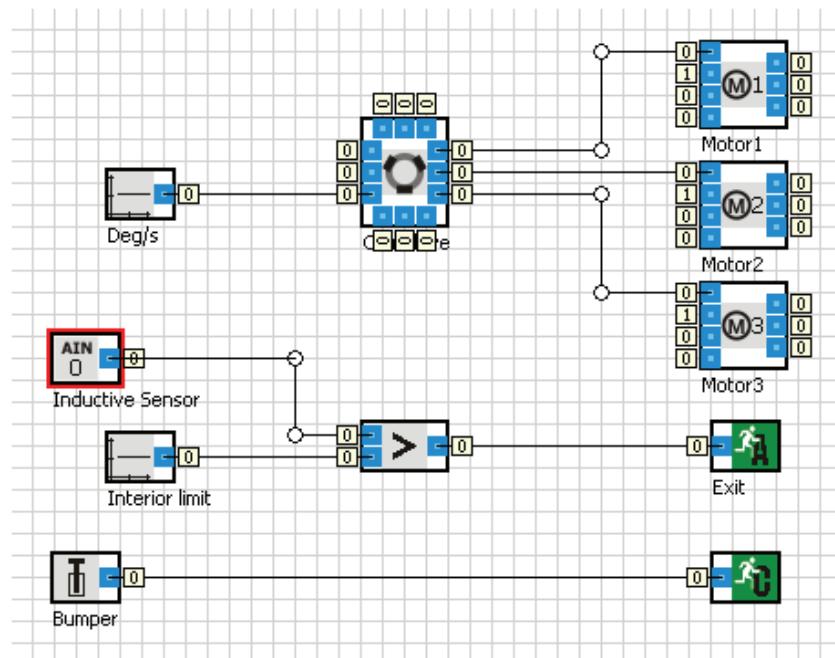


Project 8: Path tracking of an automated guided vehicle system using an analogue inductive sensor – solution

<b>Project 8:</b> <b>Path tracking of an automated guided vehicle system using and analogue inductive sensor</b>	
Name:	Date:
Optimisation of the control program	Sheet 1 of 1

P8-04c.rvw solution

Rotation to the left: Subprogram



Project 8: Path tracking of an automated guided vehicle system using an analogue inductive sensor	
Name:	Date:
Programming of a closed-loop control program	Sheet 1 of 1

The sequence program optimised by you is to be converted into a closed-loop control program.

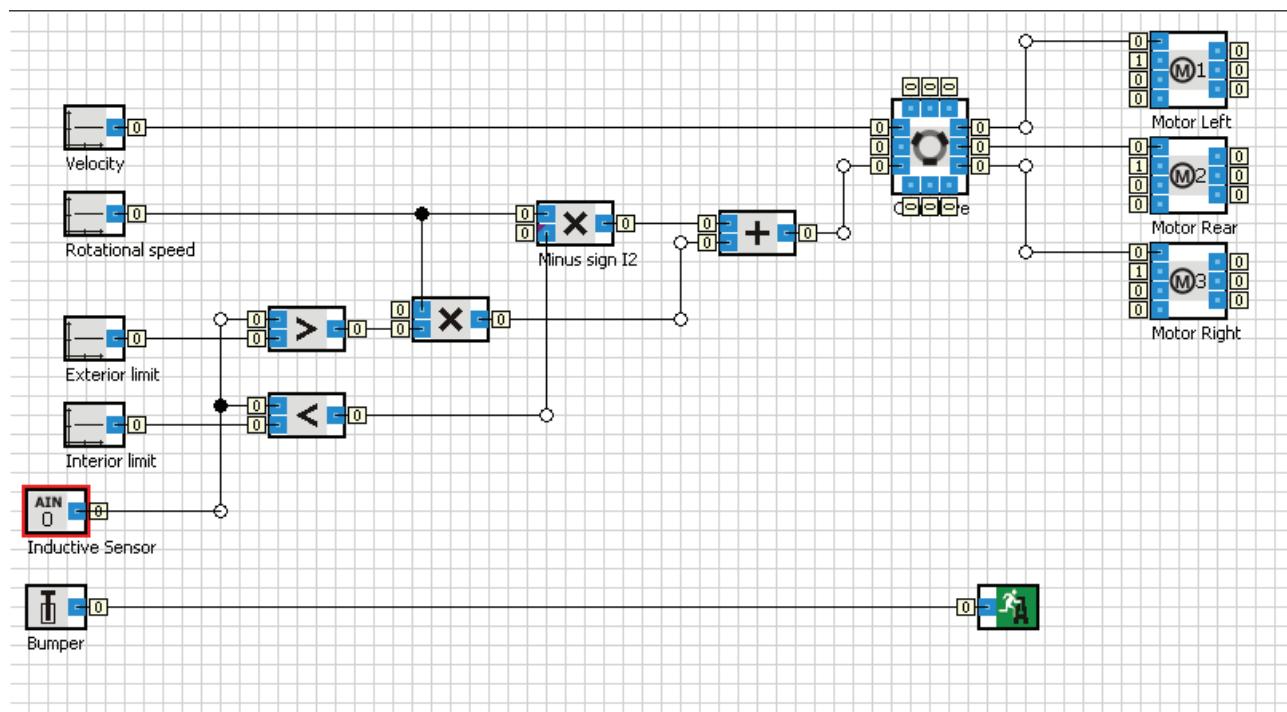
- Make a note of the advantages of a closed-loop control program and realise the required closed-loop control program.

#### Solution

A change in value can be immediately reacted to without stopping. Closed-loop control can proceed during forward travel, thus resulting in shorter travel time.

#### Program

'P8-05.rvw solution'



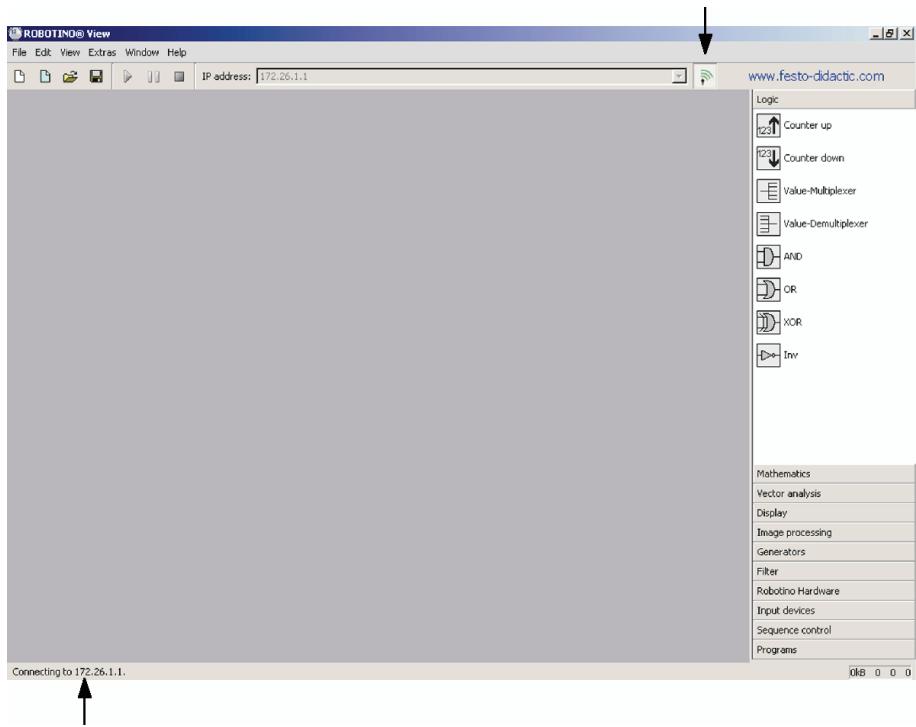
Project 8: Path tracking of an automated guided vehicle system using an analogue inductive sensor – solution

## Project 9

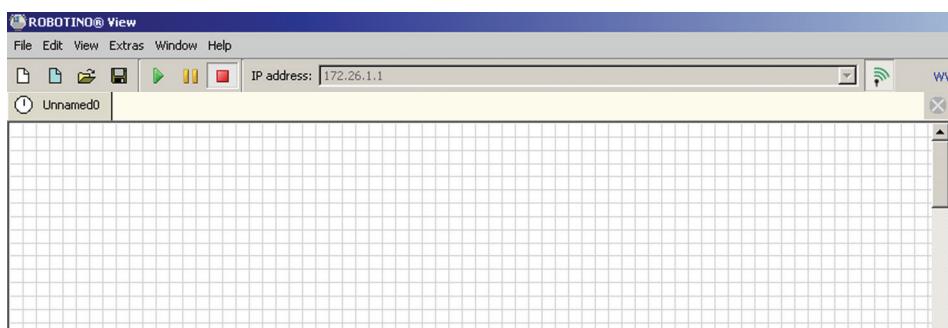
### Determining the optimal motion behaviour – solution

Project 9: Determining the optimal motion behaviour	
Name:	Date:
Motor actuation program	Sheet 1 of 4

- Create a program for the motor actuation in Robotino® View.
- Visualise and observe the setpoint/actual characteristics.
- Jack up the Robotino®.
- Start Robotino® View and establish a connection between the control of Robotino® and Robotino® View.



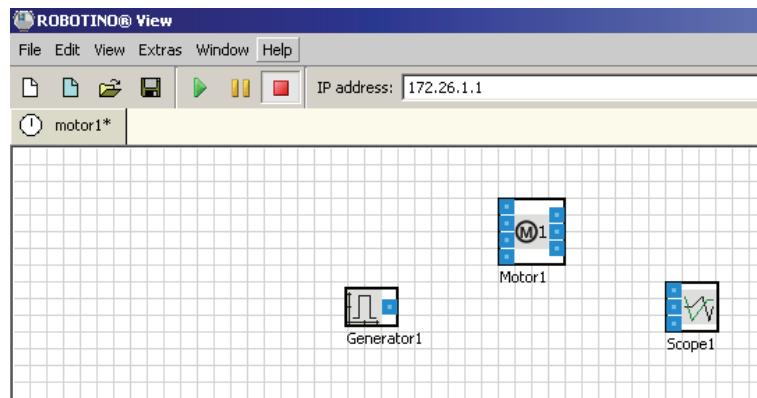
- Open a blank function block diagram in Robotino® View.



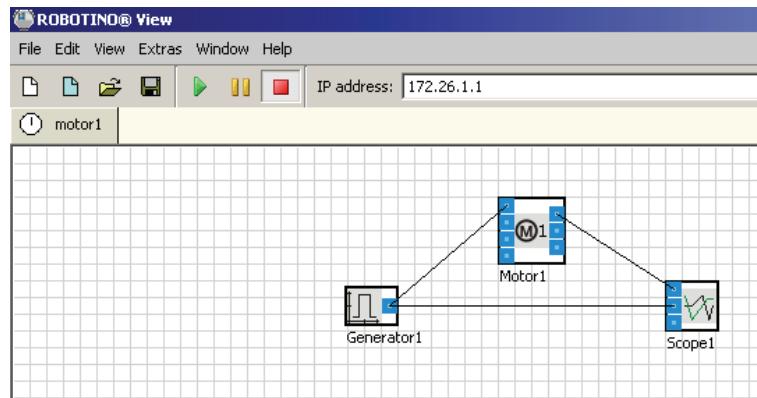
## Project 9: Determining the optimal motion behaviour – solution

Project 9:Determining the optimal motion behaviour	
Name:	Date:
Motor actuation program	Sheet 2 of 4

- Create a function block diagram from the function blocks “motor”, “square-wave generator” and “oscilloscope”.



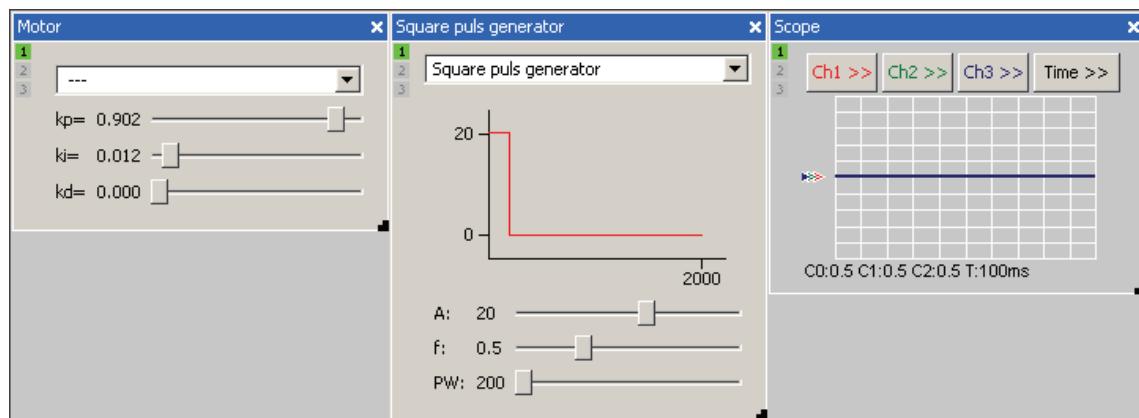
- Connect the elements so that the setpoint and actual signal are displayed.



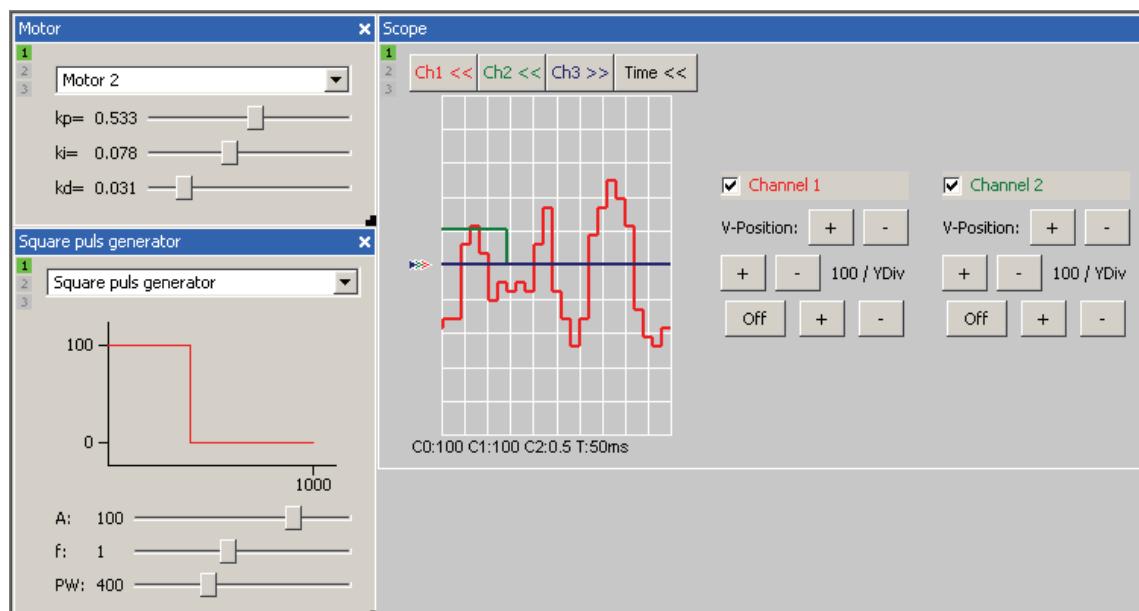
## Project 9: Determining the optimal motion behaviour – solution

Project 9: Determining the optimal motion behaviour	
Name:	Date:
Motor actuation program	Sheet 3 of 4

Parameterisation of function blocks.



- Set the oscilloscope.
- Set the amplitude of the square-wave generator to 100.



- Start the program by clicking onto the Start symbol.



<b>Project 9:Determining the optimal motion behaviour</b>	
Name:	Date:
Motor actuation program	Sheet 4 of 4

- Observe the behaviour of the multidirectional caster and describe and explain it (square-wave generator).

**Description: Behaviour of the multidirectional caster**

The multi directional casters do not travel in constant motion, but interrupt travel.

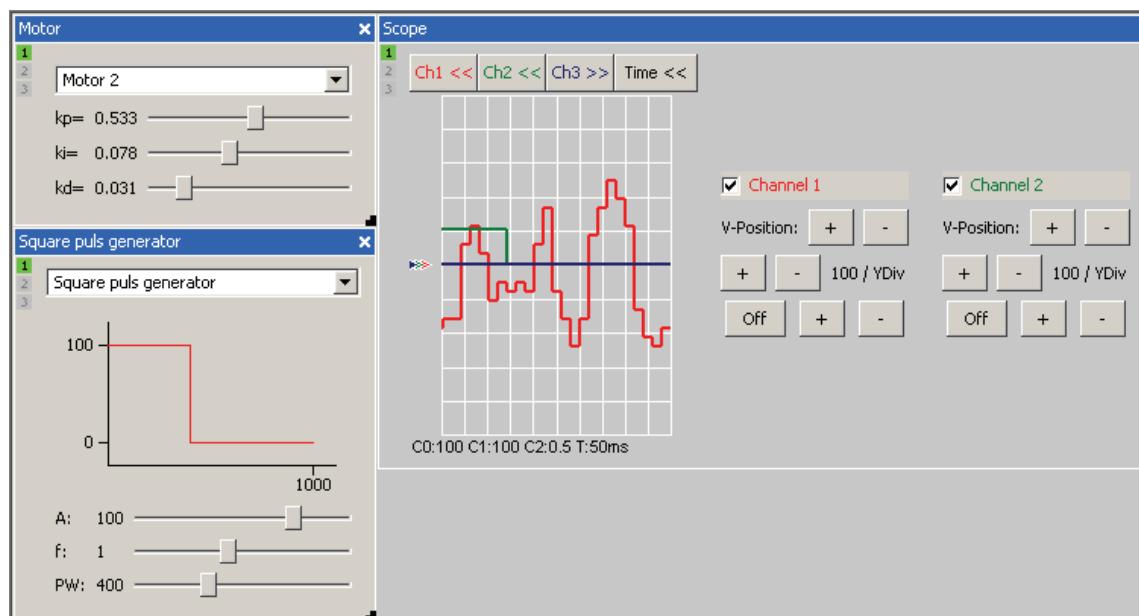
**Explanation: Behaviour of the multidirectional caster**

The square-wave generator sends a strong pulse to the multidirectional casters. These quickly achieve their travel speed (the speed of the motor is achieved immediately). A short distance is travelled at constant speed, after which motion is quickly decelerated again.

## Project 9: Determining the optimal motion behaviour – solution

Project 9: Determining the optimal motion behaviour	
Name:	Date:
Setting the PID controller	Sheet 1 of 2

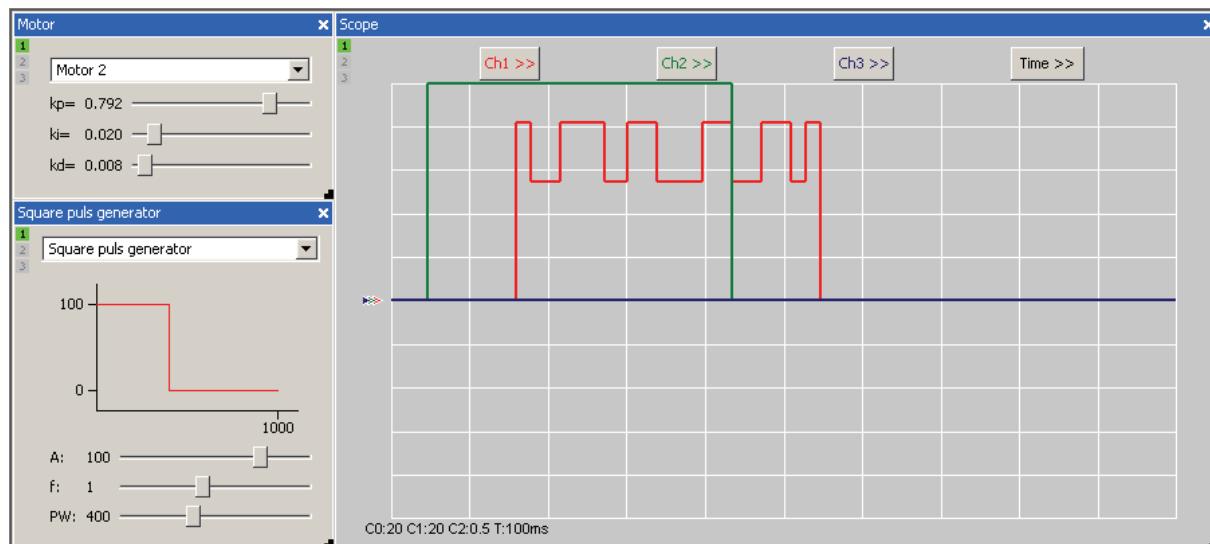
- Set the PID controller parameters so that the setpoint and actual signal behaviour match optimally and describe the effects on the motion behaviour. Explain and document your findings.
- Set the PID controller parameters on the motor. Change kp, ki, kd in succession.
- Observe the setpoint/actual behaviour displayed by the virtual oscilloscope.



## Project 9: Determining the optimal motion behaviour – solution

Project 9: Determining the optimal motion behaviour	
Name:	Date:
Setting the PID controller	Sheet 2 of 2

- Optimise the transient response so that the setpoint and actual curves virtually coincide.



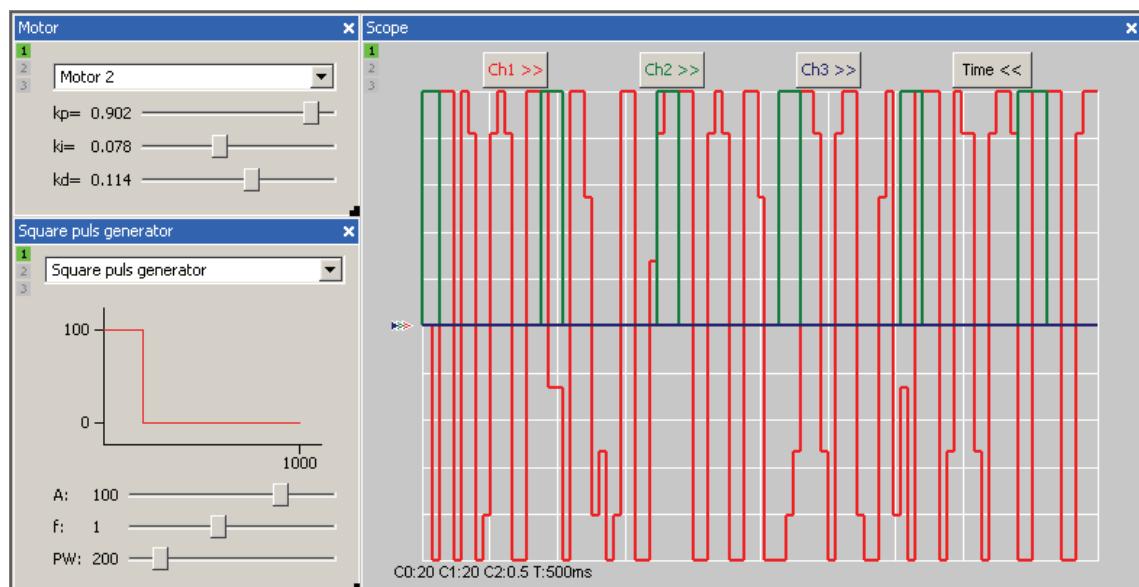
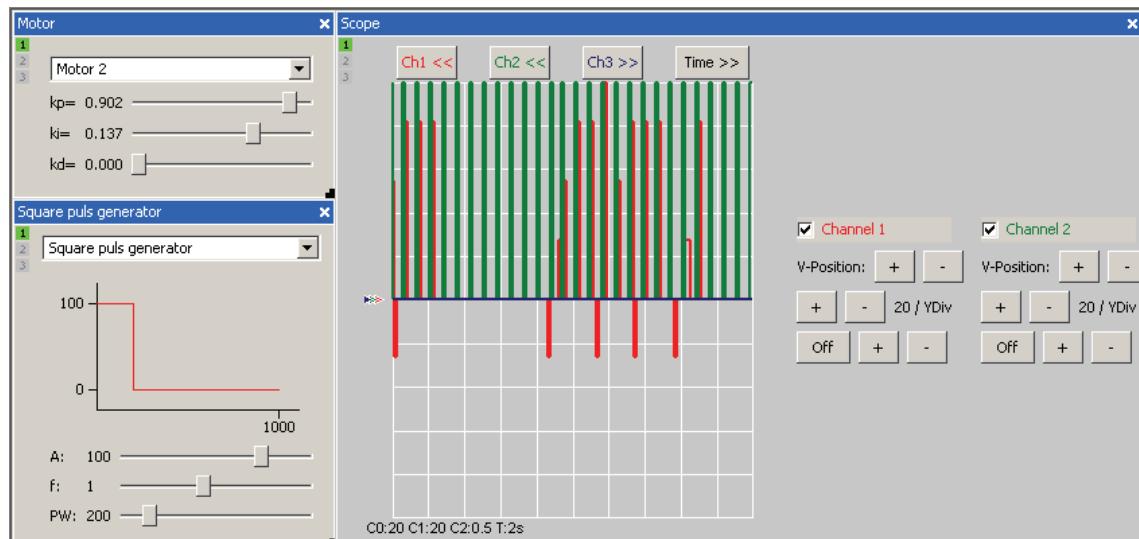
- Determine a range of values for the control parameters within which the motion behaviour is acceptable.
- Document the parameter values determined.

Range of values of PID controller parameters	
kp	0.690 – 0.900, e.g. kp = 0.9
ki	0.010 – 0.030, e.g. ki = 0.020
kd	0.001 – 0.010 , e.g. kd = 0.004

## Project 9: Determining the optimal motion behaviour – solution

Project 9: Determining the optimal motion behaviour	
Name:	Date:
Oscillations	Sheet 1 of 2

- Describe the effect on the movement of the multidirectional casters if high oscillations occur as a result of deviations in the setpoint/actual behaviour.
- Explain the effect of major oscillations on motion behaviour.
- Change the parameters  $k_i$  and  $k_d$  so that high overshoots and undershoots occur within short periods.



<b>Project 9: Determining the optimal motion behaviour</b>	
Name:	Date:
Oscillations	Sheet 2 of 2

- Observe the setpoint/actual behaviour and the movement of the multidirectional casters.

**Description: How does the multidirectional caster move?**

It moves jerkily to and fro. The desired motion generated via the square-wave signal is not achieved.

**Why do high oscillations influence the motion behaviour?**

The Robotino<sup>®</sup> travels an uncontrolled path, i.e. a curved path. Travel straight ahead is not possible with high oscillations since the setpoint value is not achieved.

Oscillations occur because the i-action and d-action of the controller attempt to correct disturbances in short cycles.

<b>Project 9: Determining the optimal motion behaviour</b>	
Name:	Date:
Functions of the PID controller parameters	Sheet 1 of 1

- Answer the questions regarding the functions of the PID controller parameters.
- Refer to the theory section to find out what the parameters p, i, and d mean and cause.

#### **Meaning of p-action and its effect**

In the case of a proportional controller, the control signal is calculated proportional to the system deviation. If the system deviation is large, the value of manipulated variable is also high. If the system deviation is small, the value of the manipulated variable is low. In the ideal state, the timing of the P-controller is exactly the same as that of the input variable. The advantage is that it intervenes very quickly and without delay. The P-controller converts an erratic input signal directly into an erratic output signal which results in rapid response characteristics.

#### **Meaning of i-action and its effect**

The integral controller is the more effective, the longer a system deviation exists. Even a very minor system deviation causes a large output signal if applied for a sufficiently long period. It converts erratic input signals into ramp-shaped output signals by means of continuous summation. This means that manipulated variable changes occur continuously and much slower than in the case of the proportional controller. If a constant signal is applied at the input of an integral controller, the output changes continuously until the system deviation is corrected. Integral controllers are however frequently used in order to eliminate the disadvantage of the proportional controller which cannot fully correct the system deviation. This is why they can generally be found in combination with proportional controllers in industrial practice.

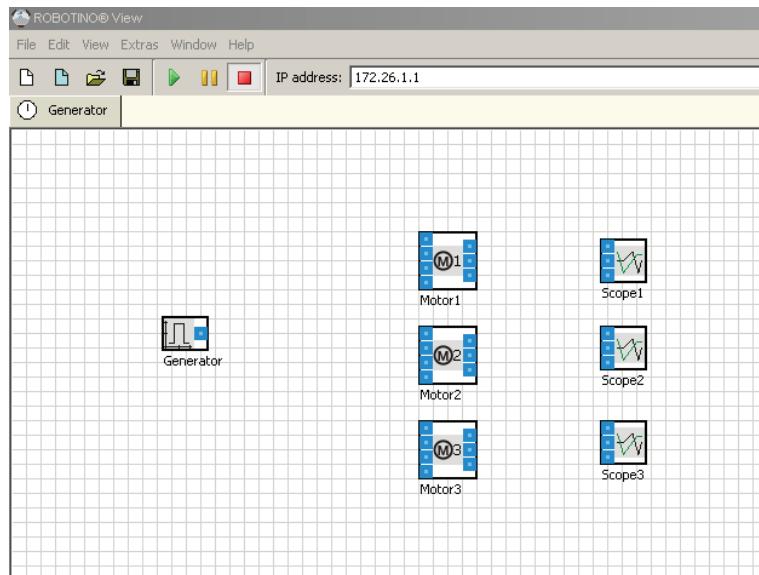
#### **Meaning of d-action and its effect**

In some controlled systems, major disturbance variables can quickly become operative in that the controller variable greatly deviates from the reference variable within a short time. Deviation such as these can be compensated with a D-controller. The output variable of a D-controller is proportional to the temporal change in the system deviation. An erratic change in the system deviation therefore generates an infinitely large manipulated variable at the output of the controller. The differential controller is the more effective, the faster the change in the system deviation occurs.

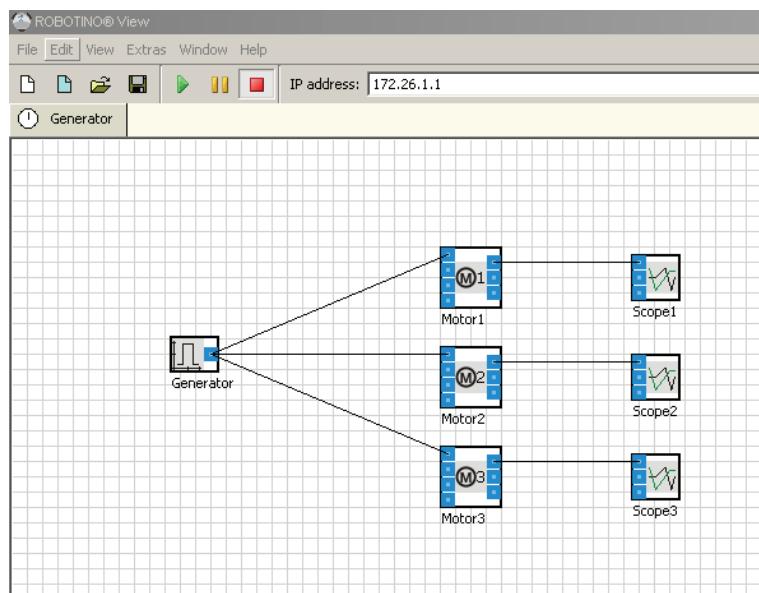
## Project 9: Determining the optimal motion behaviour – solution

Project 9: Determining the optimal motion behaviour	
Name:	Date:
Program for the actuation of three motors	Sheet 1 of 3

- Create a program for the actuation of three motors in Robotino® View.



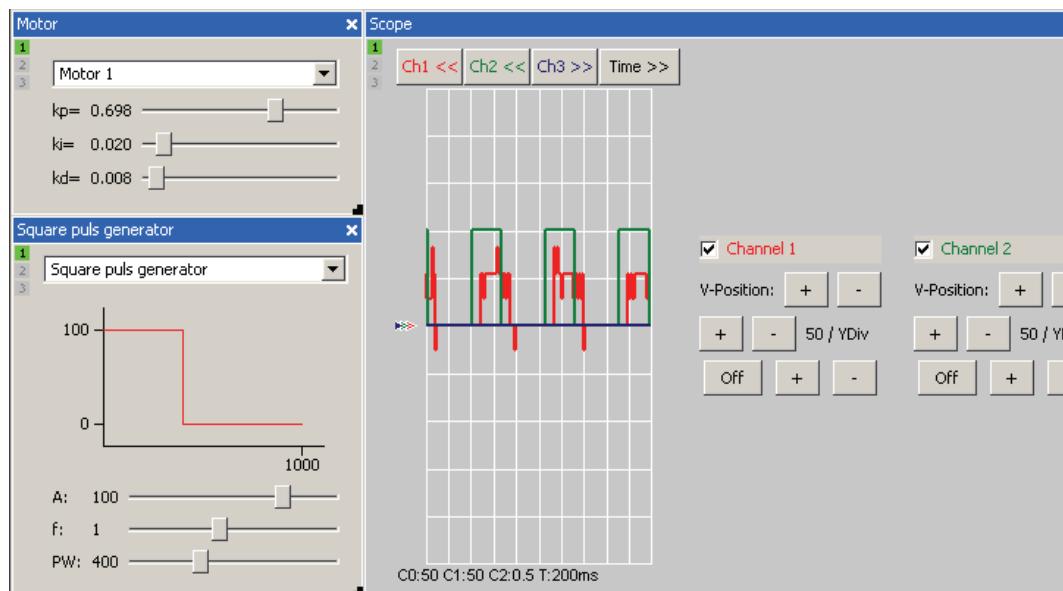
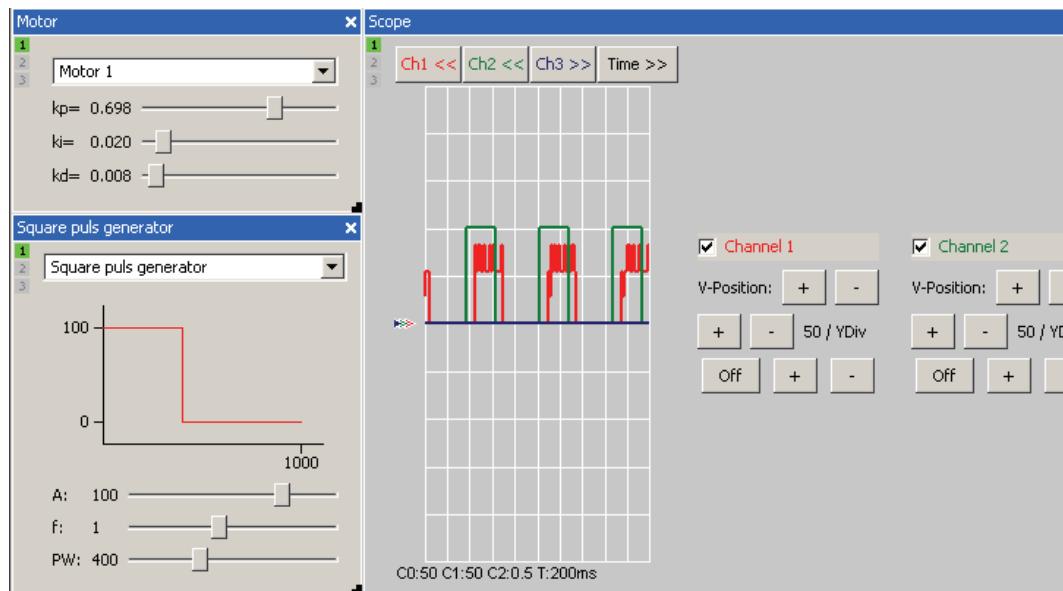
- Connect one generator and one oscilloscope in each case.



## Project 9: Determining the optimal motion behaviour – solution

Project 9: Determining the optimal motion behaviour	
Name:	Date:
Program for the actuation of three motors	Sheet 2 of 3

- Visualise and observe the setpoint/actual behaviour in the jacked-up state and during motion using the same controller setting.



<b>Project 9: Determining the optimal motion behaviour</b>	
Name:	Date:
Program for the actuation of three motors	Sheet 3 of 3

- Describe the behaviour of the curves both in the jacked-up state and during motion.

<b>Description: Behaviour of curve</b>	
Jacked-up behaviour of curve	The actual value of the speed rapidly increases and exceeds the setpoint value. It adjusts to the setpoint value after a short settling time.
Moving behaviour of curve	In the moving state, the actual value also increases rapidly, but does not exceed the setpoint value of the speed. The actual value remains below the setpoint value even after the settling time.

- Explain why the curve during motion deviates from the curve in the jacked-up state.

<b>Explanation: Different behaviour of curves</b>	
In the jacked-up state, the actual value oscillates beyond the setpoint value since it is to reach the required speed as quickly as possible. Once the speed is reached, the actual value oscillates around the setpoint value. Due to the static friction when the Robotino® is in motion, the actual value no longer fully adjusts to the setpoint value, but settles below the setpoint line. In order to obtain optimal motion behaviour with static friction it is therefore necessary to re-adjust the parameters.	

## Project 10

Path tracking of an automated guided vehicle system with the help of a webcam – solution

Project 10: Path tracking of an automated guided vehicle system with the help of a webcam	
Name:	Date:
Commissioning of the Robotino® webcam	Sheet 1 of 1

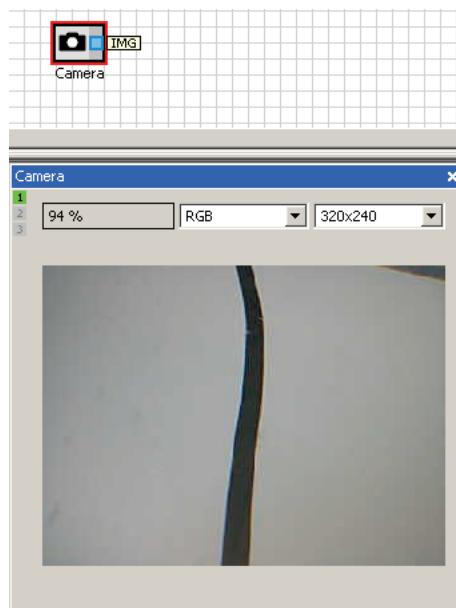
- To realise this project you first of all will need to commission and test the Robotino® webcam. Describe how you proceed and create a function block diagram whereby you can test the functioning of the camera.

Note By turning the lens, the camera can be accurately focussed for the desired image acquisition range.

Solution Loosen the locking screw of the webcam and set up the camera.  
Fix the alignment of the webcam using the locking screw.  
Connect the camera to the USB interface of the command bridge.  
Test the function of the webcam.

Create a test program in Robotino® View.  
Establish a WLAN connection to the Robotino®.  
Drag a camera onto the workspace from the “Robotino® Hardware” functions library.  
Double click onto the camera symbol.  
Start the program.

The functioning of the webcam is ensured if the camera image is displayed in the parameter dialogue.



<b>Project 10: Path tracking of an automated guided vehicle system with the help of a webcam</b>	
Name:	Date:
Alignment of the camera	Sheet 1 of 1

To evaluate the camera image, the camera must be set accordingly.

How does the camera need to be aligned to be able to detect a line in front of the Robotino®?

- Carry out your setting.

Solution

The camera must be aligned in such a way that neither chassis nor cable of the Robotino® is within the image acquisition range. It should be inclined to the extent that the line to be detected is detected as closely as possible in front of the Robotino®. The camera must be aligned straight ahead in the direction of travel. The camera must be focussed for this range.

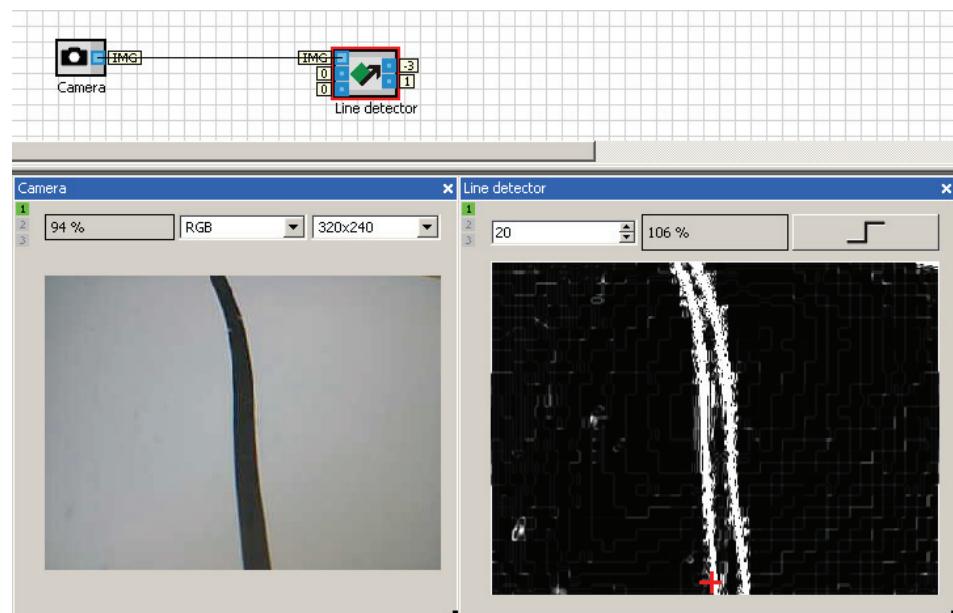
## Project 10: Path tracking of an automated guided vehicle system with the help of a webcam – solution

<b>Project 10: Path tracking of an automated guided vehicle system with the help of a webcam</b>	
Name:	Date:
Programming the line detection function	Sheet 1 of 1

- Expand the program to test the line detection function of the camera.
- Describe in a few words what you need to do.

### Solution

Drag a “line detection” function block onto the workspace from the function block library.  
Connect the “image” input to the output of the camera.



<b>Project 10: Path tracking of an automated guided vehicle system with the help of a webcam</b>	
Name:	Date:
Testing the functions of the line detection function block	Sheet 1 of 1

- Describe the functions of the line detection function block in Robotino® View.
- Test the different parameters (inputs/outputs) of the module.

Solution	Parameter	Function
	Input: Image	Image supplied by the camera. This input is connected to the output of the camera
	Input threshold value	The image is scanned from the bottom up, i.e. scanned for the detection of borders. The limit value is a parameter as to how many image lines are used to detect a line as such. Up to 50 lines of the image can be used.
	Input border type	The image is scanned from left to right. 0 means that transitions from dark to bright are recognised as a line 1 means that transitions from bright to dark are recognised as a line
	Output X	x-Position (pixel) of the line at the bottom border of the image
	Output found	The output has the value 0, if no border is detected, or the value 1, if a border is detected

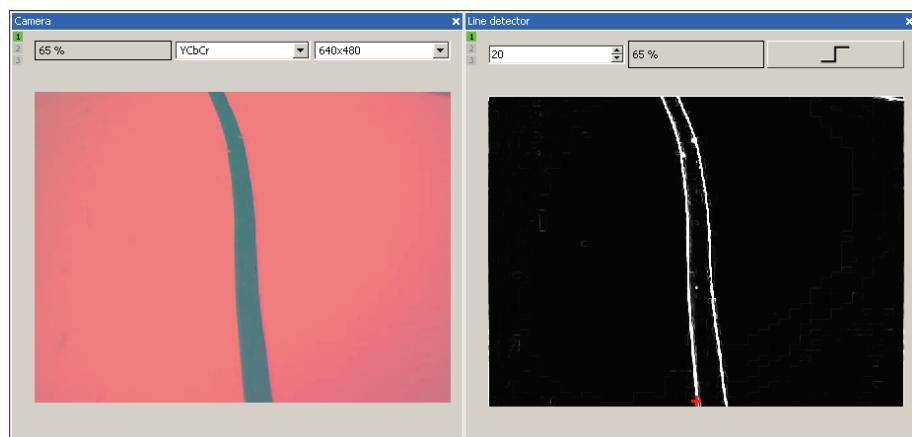
The inputs Threshold Value and Border Type can also be set manually if the corresponding input in the program is not occupied.

<b>Project 10: Path tracking of an automated guided vehicle system with the help of a webcam</b>	
Name:	Date:
Determining the optimal colour spaces for the line detection	Sheet 1 of 2

- Examine the effect of the different colour spaces of the camera in respect of line detection.
- Enter your findings in the table below.

Solution	Colour space	Findings
	RGB	Line is clearly detected
	YCbCr	Line is clearly detected
	HSV	Line is not clearly detected, since additional lines are generated by this colour model.
	HLS	Line is not detected at all, since numerous additional lines are generated by this colour model.

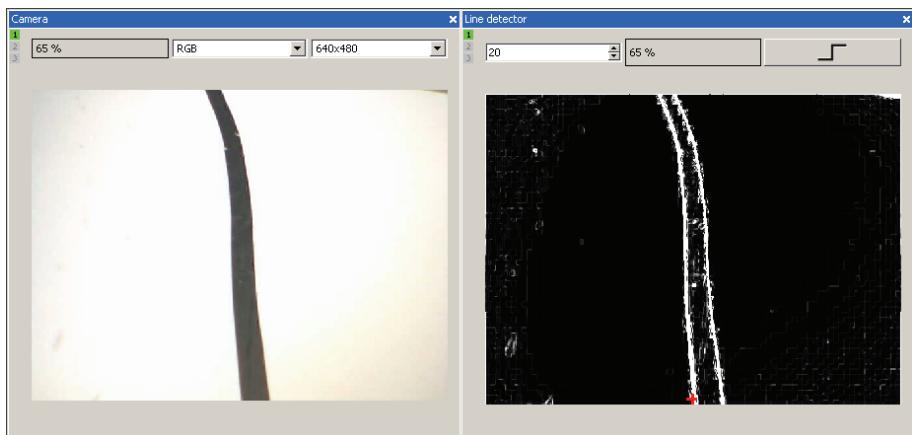
YCbCr



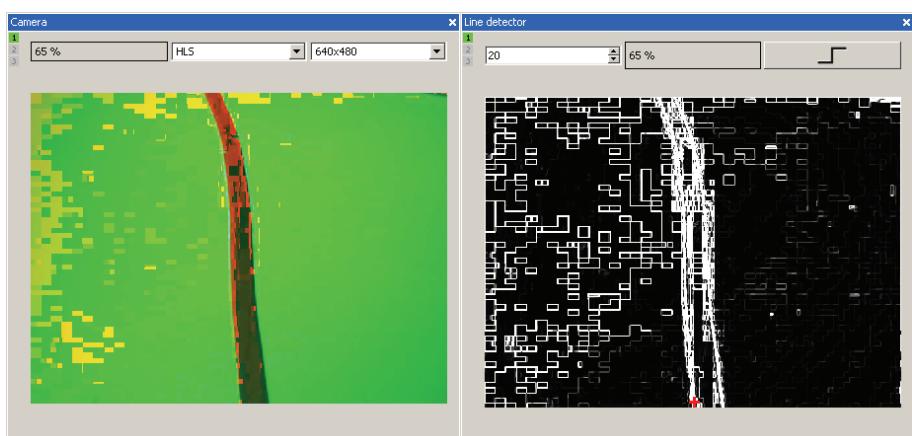
## Project 10: Path tracking of an automated guided vehicle system with the help of a webcam – solution

<b>Project 10: Path tracking of an automated guided vehicle system with the help of a webcam</b>	
Name:	Date:
Determining the optimal colour space for the line detection	Sheet 2 of 2

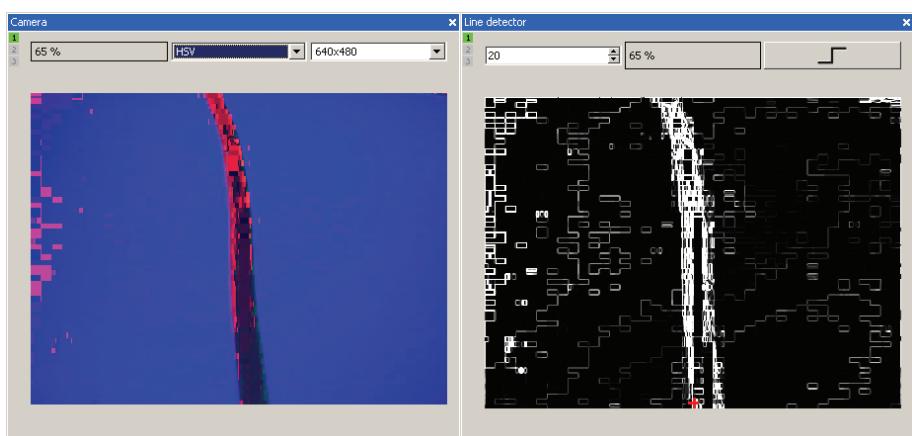
RGB



HLS



HSV



<b>Project 10: Path tracking of an automated guided vehicle system with the help of a webcam</b>	
Name:	Date:
Expansion of the test program with the addition of line detection	Sheet 1 of 1

- Test the individual inputs and outputs of the line detection module and describe the possible settings. Place a DIN A3 sheet with a black line (width > 5mm) in front of the Robotino®. Align the Robotino® and the camera so that the line is within the detection range of the camera.
- By means of experiments, determine the colour model of the camera best suited for line detection and test the input setting options of the line detection module.

How is the detected line displayed?

Solution                   A detected line is signalled by means of a cross-shaped red mark.

The colour model RGB or YCbCR is used for line detection since the two other models do not facilitate clear line detection under normal lighting conditions.

Input functions of the line detection module:

Image: Line detection can be optimised by selecting the appropriate colour model.

Limit value: By changing the input value via the parameter dialogue, the line detection can be set such for the colour model used is reliably detected. (The marking does not "bounce").

Border type: No improvement was noted as a result of this.

<b>Project 10: Path tracking of an automated guided vehicle system with the help of a webcam</b>	
Name:	Date:
Determining the line detection value pattern	Sheet 1 of 1

To realise path tracking you need to determine the output values supplied by the line detection. This will enable you to decide what strategy to consider for the realisation of path tracking.

- What information is supplied by output X of the line detection module?
- Enter the values determined in the table below.

#### Note

Determine the value patterns as follows:

Make sure that the line is detected. You can establish this by the fact that the input "line found" has the value 1.

First, position the Robotino® so that the line detected is located in the bottom righthand corner of the camera image.

Then rotate the Robotino® until the line is directly in front of the Robotino®.

Now position the Robotino® so that the line detected is in the bottom lefthand corner of the camera image.

#### Solution

	<b>Position</b>	<b>Resolution 640 x 480</b>	<b>Resolution 320 x 240</b>
	Line is located outside to the right	320	160
	Line is located directly in front of the Robotino	0	0
	Line is located outside to the left	-320	-160

Output X outputs the x-position of the detected line in pixels. The image centre is indicated by 0, the righthand image border by half of the x-value of the image resolution (640 or 320), and the lefthand image border by the corresponding negative value. If the output value is 0, the line detected is located centrally in front of the Robotino®.

## Project 10: Path tracking of an automated guided vehicle system with the help of a webcam – solution

Project 10: Path tracking of an automated guided vehicle system with the help of a webcam	
Name:	Date:
Determining the strategy for path tracking	Sheet 1 of 1

- Describe the program functions required for path tracking. Use the higher resolution of the camera (640 x 480).

Function – line detected: Determine the sensor value and travel forward:  
 If the sensor value is less than 0: Rotate the Robotino® to the right.  
 If the sensor value is greater than 0: Rotate the Robotino® to the left.

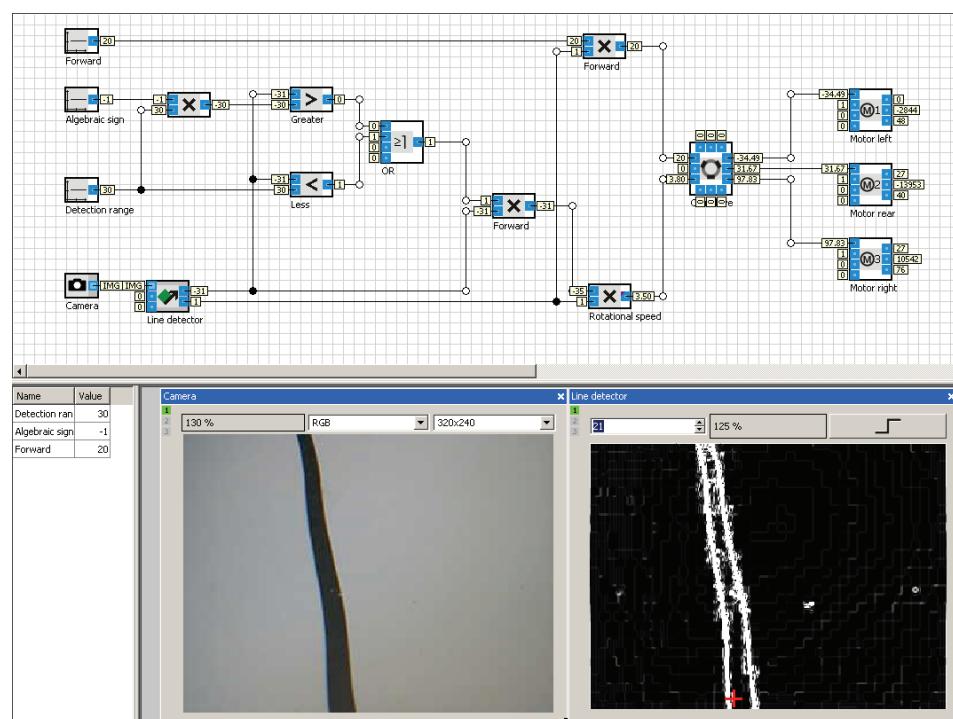
- Realise the necessary functions in a closed-loop control program.

### Note

Use sequence and closed-loop control programs you have already developed and modify these accordingly.

Use the sensor values to adapt the rotational speed of the deviation from the zero position. Parameterise the sensor value using an appropriate function of the parameter dialogue.

Select a low threshold value, such as 20, in order to accelerate the line detection. Remember to integrate collision protection into the program. Adapt the output values of the line detection module so as to achieve constant motion and reasonably jerk-free rotational movement. Move forward slowly.



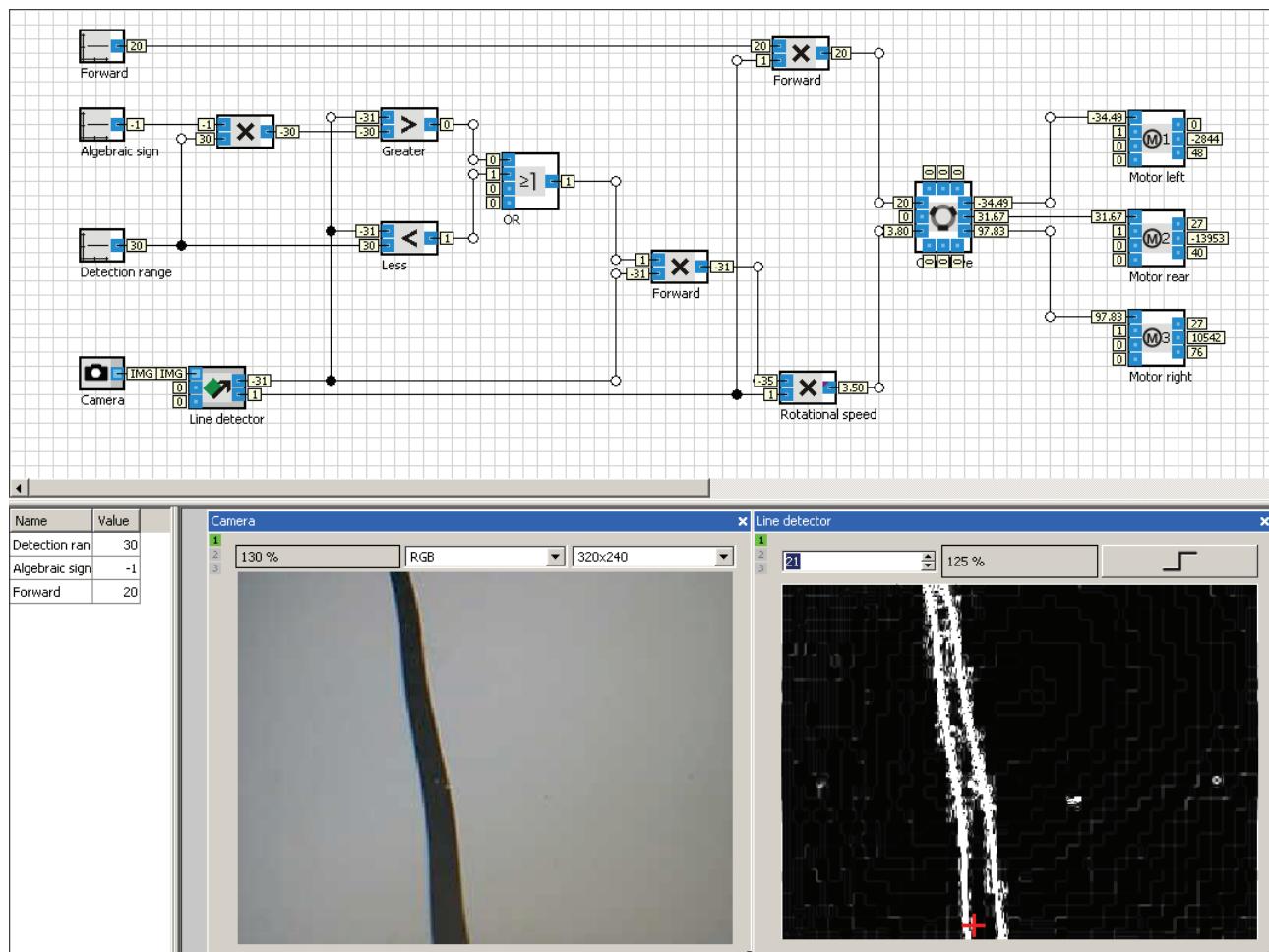
## Project 10: Path tracking of an automated guided vehicle system with the help of a webcam – solution

Project 10: Path tracking of an automated guided vehicle system with the help of a webcam	
Name:	Date:
Expansion of the closed-loop control program: Detection of the end of the guide line	Sheet 1 of 2

The desired destination is reached when the guide line ends. The Robotino® must then be stopped.

- Expand your closed-loop control program with the addition of a function which stops the Robotino®, if the line is not detected.

### Solution



## Project 10: Path tracking of an automated guided vehicle system with the help of a webcam – solution

Project 10: Path tracking of an automated guided vehicle system with the help of a webcam	
Name:	Date:
Expansion of the closed-loop control program: Optimisation of the line detection	Sheet 2 of 2

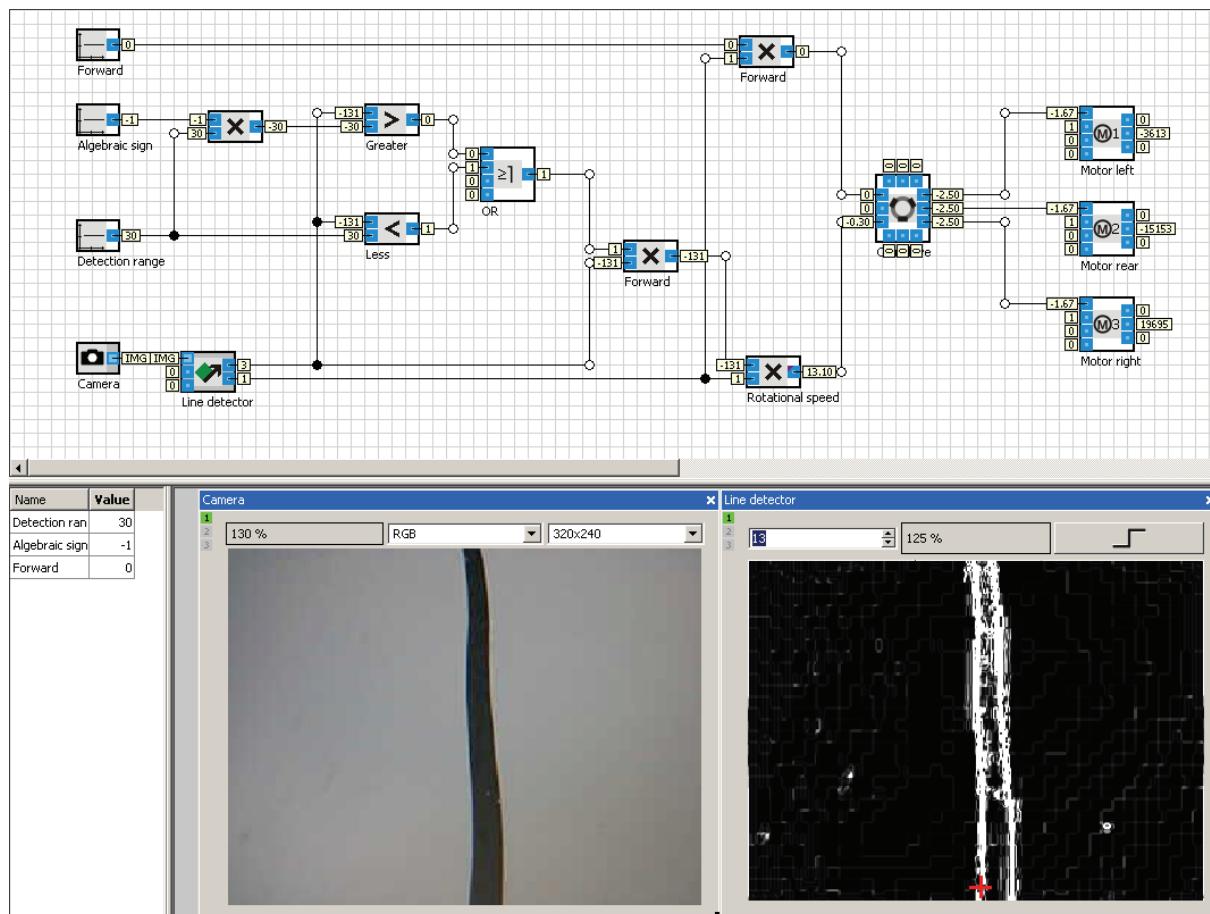
It is quite possible that in addition to the line to be followed, other lines not visible to the naked eye will be detected on the camera image. These may interfere with the detection of the desired line and result in the guide line being abandoned, particularly if these lines are close to the image border.

- Consider how you can minimise these influences on line detection.

### Solution

The range of values must be limited to within a narrow range around the zero position of the range of values, since otherwise the high line detection output values will lead to extensive clockwise or anti-clockwise rotation. This may result in the guide line being lost or the Robotino® following an additionally detected line or stopping if it no longer finds a line.

The magnitude of the range of values is dependent on the curvature of the guide line and the speed travelled.





## Project 11

Searching and approaching a coloured object with the help of a webcam

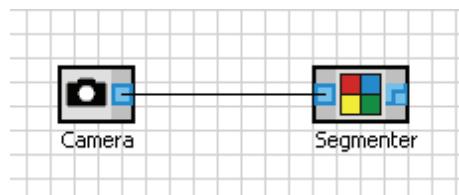
Project 11: Searching and approaching a coloured object with the help of a webcam	
Name:	Date:
Evaluation of the camera image	Sheet 1 of 3

- Determine the function blocks required for colour recognition and create a function block diagram for this.
- Note down the function blocks required and create the necessary function block diagram.

Function blocks required:

Solution

Camera  
Segmenter



Sample program

Solution\_P11\_01.rvw

## Project 11: Searching and approaching a coloured object with the help of a webcam

<b>Project 11: Searching and approaching a coloured object with the help of a webcam</b>	
Name:	Date:
Evaluation of the camera image	Sheet 2 of 3

The camera image must be evaluated for the motion behaviour of the Robotino®.

- Which function block do you need to use to do so and which function block outputs can you thus evaluate?
- Describe their function and how these can be used for the exercise given.

### Solution

#### Segment extractor

Output	Functional description
X	The output indicates the x-position of the segment main focus in relation to the image centre. Values greater than 0 indicate a deviation of the segment midpoint to the right and values less than 0 indicate a deviation of the segment midpoint to the left
Y	The output indicates the y-position of the segment main focus in relation to the image centre. Values greater than 0 indicate a deviation of the segment midpoint upwards and values less than 0 indicate a deviation of the segment midpoint downwards
Area	Size of a continuous segment within the image in pixel
Found	Indicates whether an area has been found which is greater than the specific minimum area.

Output	Use within the function block diagram
X	This output is used for navigation. The Robotino navigates towards the segment main focus.
Y	This output is not evaluated since the height does not need to be determined.
Area	This output can perhaps be used to roughly determine the distance.

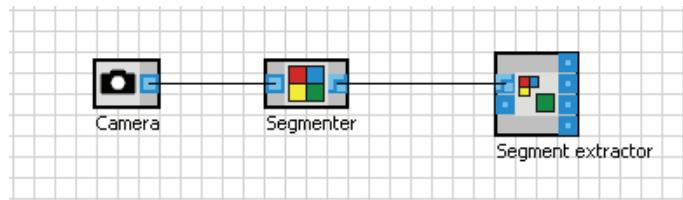
### Note

A higher tolerance can be achieved for colour recognition by adjusting the pixel intensity for individual channels. By increasing the pixel intensity (+ key), it is to some extent possible to compensate colour deviations which can occur, for example, when approaching the object or if the lighting conditions are changed. This means that the colour object will still be detected in this case. Determine the optimal setting by means of experimenting.

**Check these settings whenever you reload the program.**

## Project 11: Searching and approaching a coloured object with the help of a webcam

<b>Project 11: Searching and approaching a coloured object with the help of a webcam</b>	
Name:	Date:
Evaluation of the camera image	Sheet 3 of 3



Sample program

Solution\_P11\_02.rvw

How can you ensure that the object is shown as a full-frame image and in optimal quality?

Describe all the possibilities you are considering and select one of these for your program.

Solution

1. Via evaluation of the image information (size of the area detected), of the segmenter.
2. By determining the distance at which the object is shown as a full-frame image. Approaching at this distance with the help of the distance sensors.

The advantage of variant 2 is that the size of the object found cannot be influenced by lighting conditions. In addition, the lens can be optimally set beforehand for this distance, whereby optimal quality of representation is obtained.

Project 11: Searching and approaching a coloured object with the help of a webcam

<b>Project 11: Searching and approaching a coloured object with the help of a webcam</b>	
Name:	Date:
Definition of the strategy and its subfunctions	Sheet 1 of 2

- Define your strategy for the solution of the exercise given. Note down the necessary subfunctions.
1. Rotate the Robotino® until the segment main focus is in the centre of the image.
  2. Travel forward in the direction of the segment main focus up to the determined optimal distance (for example 8 cm)

Project 11: Searching and approaching a coloured object with the help of a webcam

<b>Project 11: Searching and approaching a coloured object with the help of a webcam</b>	
Name:	Date:
Definition of the strategy and its subfunctions	Sheet 2 of 2

- Describe the step enabling conditions for the program functions required and make a note of the necessary function blocks and outputs required.

Function	Step enabling condition
Finding object	Rotate the Robotino® until output X of the extractor = 0
Approaching object	Travel forward until distance sensor 1 outputs a distance which is less than the desired distance. The segment midpoint is approached by means of closed-loop control.

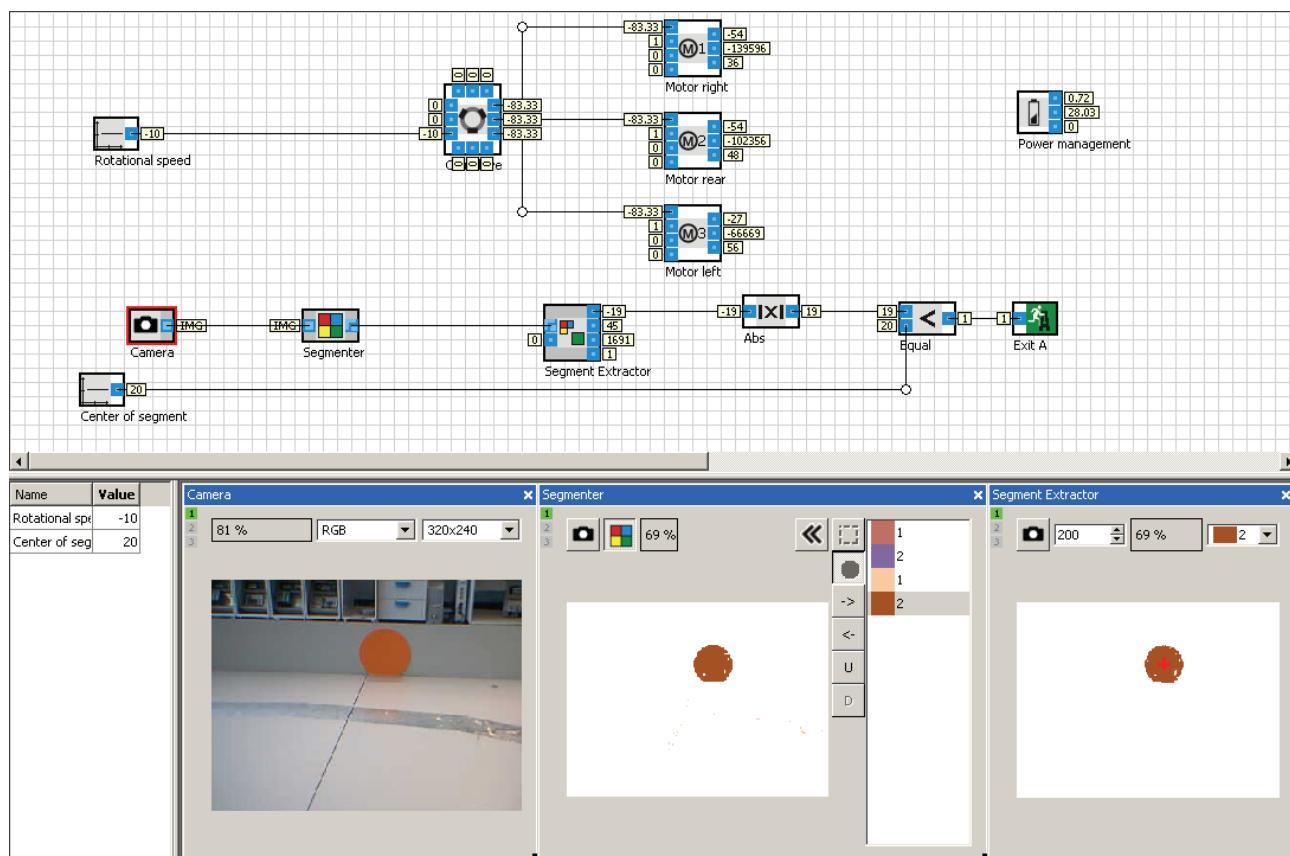
## Project 11: Searching and approaching a coloured object with the help of a webcam

Project 11: Searching and approaching a coloured object with the help of a webcam	
Name:	Date:
Realisation of the sequence program	Sheet 1 of 2

- Realise the individual program functions and combine these into a sequence program.

Solution

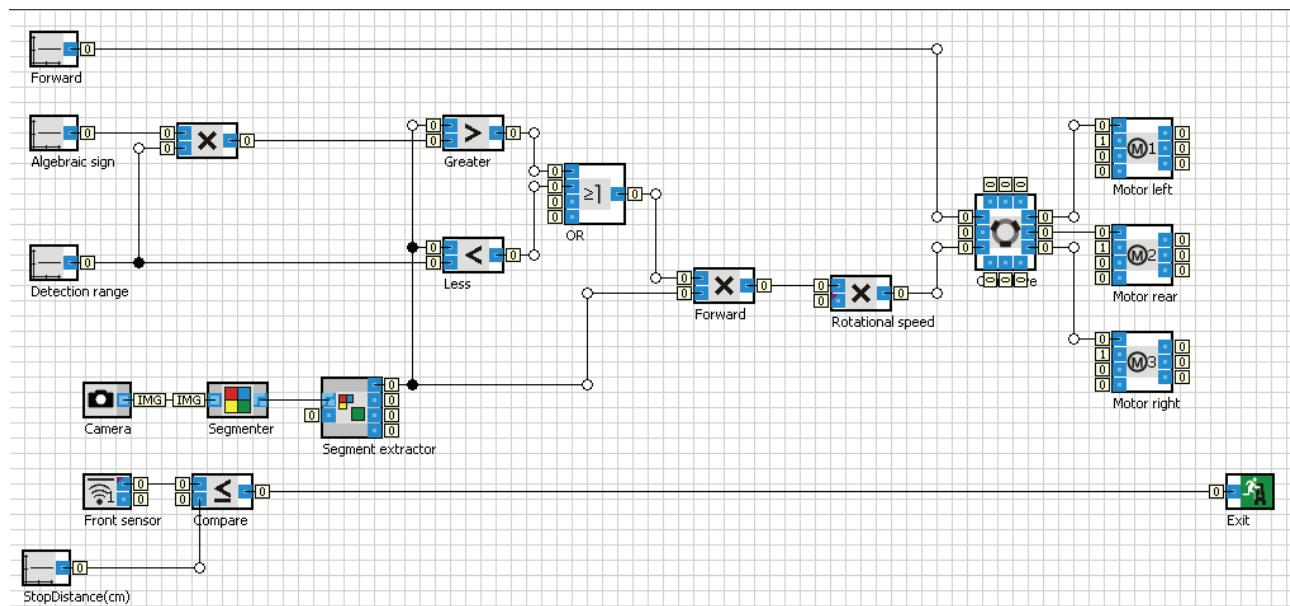
Finding the object



Project 11: Searching and approaching a coloured object with the help of a webcam

Project 11: Searching and approaching a coloured object with the help of a webcam	
Name:	Date:
Realisation of the sequence program	Sheet 2 of 2

Approaching the object

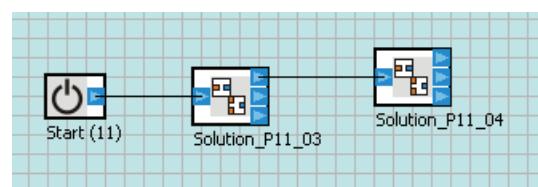


Sample programs

Solution\_P11\_03.rvw

Solution\_P11\_04.rvw

Sequence program



Sample program

Solution\_P11\_05.rvw

Project 11: Searching and approaching a coloured object with the help of a webcam