

## A wheeled mobile robot

***Citation for published version (APA):***

Berg, van den, H. C. (2002). *A wheeled mobile robot: creating an experimental environment for non-linear controllers*. (DCT rapporten; Vol. 2002.020). Technische Universiteit Eindhoven.

***Document status and date:***

Published: 01/01/2002

***Document Version:***

Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

***Please check the document version of this publication:***

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

***General rights***

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

[www.tue.nl/taverne](http://www.tue.nl/taverne)

***Take down policy***

If you believe that this document breaches copyright please contact us at:

[openaccess@tue.nl](mailto:openaccess@tue.nl)

providing details and we will investigate your claim.

**A wheeled mobile robot  
Creating an experimental environment  
for non-linear controllers**

H.C. van den Berg

DCT 2002-20

Technische Universiteit Eindhoven

Professor: Henk Nijmeijer  
Supervisor: Harm van Essen

Eindhoven, March 2002

# Table of contents

<b>1</b>	<b>Introduction .....</b>	<b>3</b>
1.1	Introduction .....	3
1.2	Problem statement.....	3
1.3	Approach and overview .....	3
<b>2</b>	<b>Hardware configuration.....</b>	<b>4</b>
2.1	The mobile robot.....	4
2.1.1	<i>The stepper motors</i> .....	4
2.1.2	<i>Microstepping drivers and interface boards</i> .....	5
2.2	Voltage to frequency converter cards .....	6
2.3	The hardware configuration using dSpace .....	7
<b>3</b>	<b>Control strategy and simulations .....</b>	<b>8</b>
3.1	Non-holonomic systems .....	8
3.2	Tracking control.....	9
3.2.1	<i>Linear control design</i> .....	10
3.2.2	<i>Non-linear control design</i> .....	11
3.3	The chained form .....	11
3.4	Point stabilization .....	12
3.4.1	<i>Time-varying control</i> .....	13
3.4.2	<i>Discontinuous time-invariant control</i> .....	14
3.4.3	<i>Hybrid control</i> .....	16
3.4.4	<i>Point stabilization controller evaluation</i> .....	17
3.5	Trailers .....	18
3.5.1	<i>Kinematic model for a tractor pulling n trailers</i> .....	18
3.5.2	<i>Controller design</i> .....	19
3.5.3	<i>Simulation results</i> .....	19
<b>4</b>	<b>Dead-reckoning reconstruction and calibration.....</b>	<b>21</b>
4.1	Dead-reckoning.....	21
4.2	Calibration.....	21
4.2.1	<i>First calibration experiment</i> .....	22
4.2.2	<i>Second calibration experiment</i> .....	23
4.3	Realtime experiments.....	24
<b>5</b>	<b>Measurement systems.....</b>	<b>25</b>
5.1	The pantograph system.....	25
5.2	The CAD/CAM-board .....	26
5.3	The electronic compass .....	27
5.4	Active beacon navigation .....	27
5.5	The whiteboard system.....	28
5.6	Measurement system comparison and choice.....	28
<b>6</b>	<b>The eBeam whiteboard system.....</b>	<b>29</b>
6.1	How does eBeam work?.....	29
6.2	eBeam performance .....	30
6.3	Adapting the system for eBeam.....	32

6.3.1	<i>Adapting the sleeves and robot</i> .....	32
6.3.2	<i>Switching from dSpace to TUEDACs</i> .....	33
6.3.3	<i>Software development</i> .....	34
6.4	Realtime experiments using the eBeam system .....	34
<b>7</b>	<b>Conclusion and recommendations</b> .....	<b>37</b>
	<b>Literature and internet-sites</b> .....	<b>39</b>
	Literature .....	39
	Used Internet-sites .....	40
	<b>Appendices</b> .....	<b>41</b>
	Appendix A1: Stepper motor connections and specifications .....	42
	Appendix A2: Micro stepping driver and 15-pin interface board specifications .....	44
	Appendix A3: Micro stepping driver and 19-pin interface board specifications .....	46
	Appendix A4: Voltage-to-frequency converter card specifications .....	48
	Appendix B1: Electro technical scheme for step motors, microstepping drivers and V/f-converter cards .....	49
	Appendix B2: Electro technical scheme for MOBROB unit and eBeam transmitters .....	50
	Appendix B3: Electro technical scheme for TUEDACs QAD .....	51
	Appendix C1: Simulink scheme for tracking controllers .....	52
	Appendix C2: Simulink scheme for point stabilization controllers .....	53
	Appendix C3: Simulation/realtime Matlab initialisation file example .....	54
	Appendix C4: Matlab-file for a tractor with one trailer point stabilization simulation .....	55
	Appendix C5: Matlab-file for a tractor with two trailers point stabilization simulation .....	56
	Appendix D: Maximum position error calculation for the pantograph system .....	57
	Appendix E1: eBeam software development kit C++-code .....	58
	Appendix E2: C++-code for final.dll .....	60
	Appendix E3: C++-code for eBeam S-function .....	63

# 1 Introduction

## 1.1 Introduction

In the last decade a lot of research has been done on the field of wheeled mobile robots. These robots belong to a special class of non-linear systems, called non-holonomic systems. Many configurations are possible like carts with two or four wheels, possibly extended with trailers. Because of the constraints on this kind of systems traditional control methods like linearization and state feedback fail when it comes to point stabilization of the system. Solutions to this control problem can be found using advanced mathematical, non-linear controllers like time-varying, discontinuous or hybrid feedback controllers. These controllers are proven in theory and simulations, but hardly in practice.

A second problem of this kind of systems is the measurement of the position and orientation of the robot. Since dead-reckoning methods, like the use of incremental encoders for wheel rotation measurement, are not reliable enough, other methods should be used. A lot of research has been done on the field of beacon navigation, GPS-like systems and landmark recognition. Less literature can be found on easier and cheaper ways of measuring position and orientation.

## 1.2 Problem statement

The goal of this project is to create an experimental environment to validate non-linear controllers for non-holonomic systems, especially for a two-wheel mobile robot. This includes selecting and implementing an absolute position and orientation measurement system.

## 1.3 Approach and overview

When I started this project, a wheeled mobile robot, including two stepping motors and microstepping drivers, was already available. A dSpace unit was used to control the robot, which could already perform some realtime experiments.

Two voltage-to-frequency converters were available, but those were not implemented yet. The first step was to test these cards and integrate them in the system. The hardware configuration of the wheeled mobile robot system is described in chapter 2.

Some controllers from literature, both tracking and point-stabilization controllers, were simulated in Matlab/Simulink to get familiar with the way they work and their performance. The system kinematics, the basic principles of these controllers and performed simulations are discussed in chapter 3. Also an extended problem, about a robot pulling one or two trailers, is described and simulated in this chapter.

After calibrating the robot, the simulated controllers could be tested realtime by dead-reckoning position reconstruction, which is described in chapter 4. However, for properly controlling the robot and for the validation of the performance of the robot, an absolute position measurement system is needed. In chapter 5, several potential measurement systems are discussed and one system is selected. The next step is to adapt the existing system for this measurement system, which is described in chapter 6. Software needed to be written in C++ to obtain the coordinates in Simulink and the dSpace unit was replaced by a TUEdACS unit. A new, so-called MOBROB, unit was designed especially for this project to supply power and to make the TUEdACS signals usable for the other devices. At the end of the project an experimental environment for non-linear controllers was ready.

Finally, in chapter 7, the conclusions and some recommendations are given.

## 2 Hardware configuration

This project deals with a small two-wheel mobile robot. The most important components of this robot, i.e. the stepper motors and the micro stepping drivers, will be briefly described in section 2.1. To actuate the stepper motors properly two voltage-to-frequency converter cards were implemented, which are discussed in section 2.2. To control the robot a dSpace unit was available at the start of the project. Halfway the project this unit was replaced by a TUEDACs QAD unit. In this chapter only the hardware configuration using the dSpace unit will be discussed (section 2.3). The modified configuration for the TUEDACs unit will be discussed in section 6.3.2.

### 2.1 The mobile robot

The mobile robot contains two driven wheels, each independently actuated by a stepper motor. A third, ball-like, castor wheel is added to make the robot stable and move smooth. The two stepper motors are controlled by microstepping drivers, which are mounted on an interface board.

A schematic overview of the used mobile robot is given in figure 2.1.a. In figure 2.1.b a 3D-CAD-drawing of the robot is given.

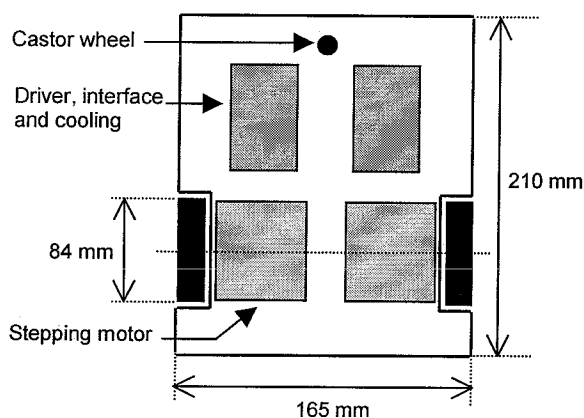


Figure 2.1.a: Schematic view of the mobile robot

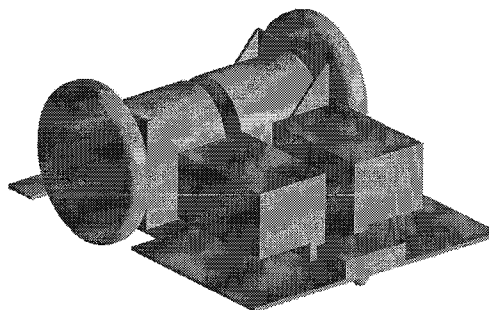


Figure 2.1.b: 3D-CAD-drawing of the robot

In section 2.1.1 the stepper motors will be discussed briefly, section 2.1.2 takes a closer look at the microstepping drivers and interface boards.

#### 2.1.1 The stepper motors

It is not the purpose of this section to give a detailed description of how stepper motors work. We are only interested in the inputs and outputs and some important properties of this kind of motors.

A stepper motor may be considered an electromotor that responds to a pulse signal by rotating its output shaft over some elementary angle  $\theta$ . The use of stepper motors has two main advantages. The first is the equivalence to the mathematical model. The computed output of the controllers is in terms of velocities. These velocities can be converted into the equivalent 'steps per second'. This means no additional velocity controllers are required.

Secondly, no encoders are required. Every pulse to the stepping motors results in a fixed displacement of the motor shaft. The total angular displacement of the motor depends on the total number of pulses applied and its angular velocity on the pulse rate. By knowing the input signal

we can reconstruct the position and velocity of the motor. This gives the possibility of dead-reckoning measurements and position reconstruction.

The stepping motor can be considered an electromotor, responding to a pulse signal. Every input pulse causes a rotation on the output shaft over some elementary angle  $\theta$ . This angle depends on the number of steps in one complete revolution 'ns' of the motor. The stepper motors used for this project have 200 steps per revolution, which makes the output shaft rotate 1.8 degrees for every input pulse. Using the microstepping drivers every step can be divided into smaller steps. When we let 'msr' be the number of microsteps per step, the output shaft will move over an angle  $\theta_{ms}$ ,

$$\text{which equals: } \theta_{ms} = \frac{2 \cdot \pi}{ns \cdot msr} \quad [2.1]$$

The angular velocity is proportional to the step rate, which is the number of steps per second, and therefore depends on the frequency of the applied pulse signal,  $f_{pulse}$ . The maximum angular speed of the motor shaft is given by:

$$\omega_{max} = \frac{2 \cdot \pi}{ns \cdot msr} \cdot f_{pulse} \quad [2.2]$$

Although assumed in theory, all stepper motors exhibit some non-linearity in practice. This means that the microsteps do not spread evenly over the span of a step, but they bunch together as illustrated in figure 2.2.

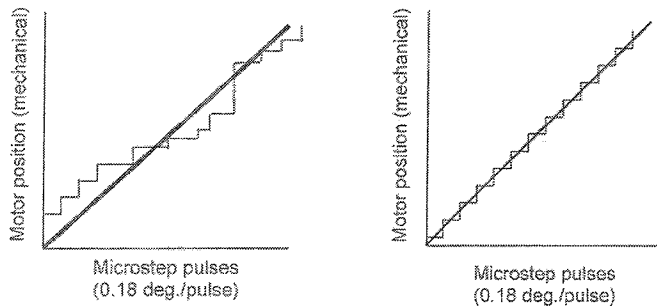


Figure 2.2: Stepper motor with bad linearity (left) and good linearity (right)

This non-linearity has two important effects. Statically the motor position is not optimal and dynamically low speed resonances occur because of the cyclic acceleration where the microsteps are spread apart and deceleration where the microsteps bunch up.

The stepper motors are manufactured to a certain tolerance, typically of +/- 5% non-accumulative error regarding the location of any given step. For the given 200 step per revolution motors this means that every step will be within an error range of 0.18 degrees. Stated otherwise, the motor can accurately resolve 2000 radial locations, which corresponds to a resolution of 10 microsteps per step.

More specifications on the stepping motors and its connections can be found in appendix A1.

### 2.1.2 Microstepping drivers and interface boards

To control the stepper motors, two IMS-IM481H microstepping drivers are used. Each driver is mounted on an interface board (IMS-INT481) together with a heat sink (H-481). Mechanical specifications and the connection of the pins are given in appendices A2 and A3.

The microstep resolution can be varied from 2 microsteps per step up to 256 microsteps per step. For the used 200 step per revolution motors a microstep resolution of 10 microsteps per step results in the maximum accuracy. Any resolution beyond this rate yields no extra accuracy, only empty resolution. Nevertheless there are two reasons justifying higher resolutions. In the case of controlling the robot, very low speeds are required. In this situation higher microstep resolution

guarantees smooth operation. Secondly, microstepping can replace mechanical gearing. As the microstepping resolution affects the maximum velocity of the robot, high resolutions can be used at low speed to guarantee smooth performance and smaller microstep resolutions can be used at higher speeds. To obtain this, the microstep resolution should be switched during operation, dependent on the velocity.

In the case of our robot here lies a problem. The two interface boards are not identical: One of them has a 15-pin terminal, the other one a 19-pin terminal. The four extra pins on the 19-pin terminal give the possibility of controlling the microstep resolution from Matlab/Simulink while the robot is moving. The microstep resolution of the 15-pin interface board can only be adjusted manually by changing the configuration of the four dip switches. As a result it is not possible to change the microstep resolution during operation.

In this project a microstepping rate of 50 microsteps per step is used, which is derived experimentally. Lower rates result in non-smooth behaviour at low speed; at higher rates the maximum speed of the robot becomes too low. The maximum velocity for the robot using the chosen sample rate is about 0.4 [m/s]. As a result the maximum angular velocity of the robot is about 5.3 [rad/s].

## 2.2 Voltage to frequency converter cards

In order to apply high frequency inputs to the stepper motors, two voltage-to-frequency converter cards are implemented. In earlier work [9] an input signal was applied to the stepper motors directly from Matlab/Simulink. In this case only relative low frequencies could be generated, resulting in low velocity movement only.

The new V/f-converters allow us to generate frequencies up to 16 [kHz]. The cards can be used in four modes, converting a 0 to 10 [V] analogue input signal to a frequency range from 0 up to 1, 4, 8 or 16 [kHz]. These modes can be chosen by adjusting a set of dip switches. In this project the cards are always used in the 16 [kHz] mode.

To test the converters they were connected to a power supply and the input voltage was increased with steps of 1 [V] from 0 [V] up to 10 [V]. At every input voltage point the frequency was measured using a scope. The result is shown in figure 2.3. The cards show linear behaviour, even at low and high input voltages. Two other conclusions resulting from the experiments were that the frequency signal almost equals a block signal and the reaction time of the cards is that small that it can be neglected.

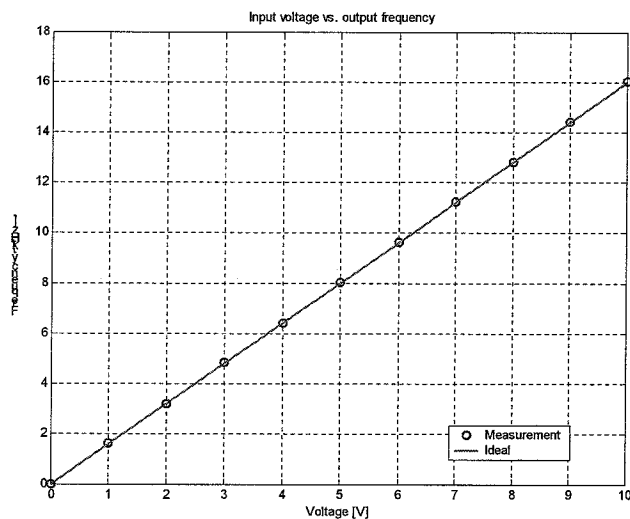


Figure 2.3: Input-output relation for the voltage to frequency converter cards



Besides a 0-10 [V] input signal a 24 [V] power supply needs to be connected to the converters. The output of the V/f-cards is a frequency signal with an amplitude of 24 [V]. The needed input voltage of the micro stepping drivers is lower, about 5 [V], so a set of resistors is added to decrease the voltage. More information about the V/f-converters is given in appendix A4.

### 2.3 The hardware configuration using dSpace

At the start of the project a dSpace DS1102 unit, connected to a PC with Matlab/Simulink, was used to actuate the mobile robot. In this mode the two voltage-to-frequency converter cards receive a 0 to 10 [V] signal from two analogue output ports on the dSpace system. This signal is used to adjust the angular velocity of the wheels.

From the digital in/out port on the dSpace unit each microstepping driver receives a signal containing the direction of the wheel rotation. Two other digital outputs on dSpace are used for the opto supply of the micro stepping drivers. This opto supply is needed to power the opto couplers, which process the frequency and direction signals. Even though this is a simple solution that works properly, dSpace should not be used for this kind of power supply. When adapting the system for TUE DACS another solution for this opto supply is needed.

Two power supplies feed the stepper motors and frequency converter cards with 30 [V] and 24 [V] respectively. Since the stepper motors still work properly when only 24 [V] is supplied, later on in the project only one power supply is used to feed both the motors and the converter cards. A schematic view of the hardware configuration at the start of the project is given in figure 2.4.

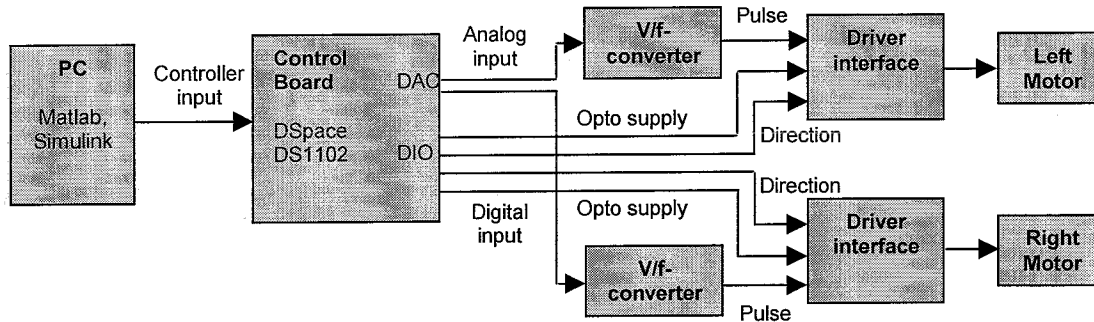


Figure 2.4: Schematic view of the hardware configuration using dSpace

### 3 Control strategy and simulations

Three different kind of control problems can be recognized concerning the wheeled mobile robot, i.e. the stabilization of the motion of the robot along a time-indexed trajectory (tracking control), along a geometric path (path following) and along a fixed point (point stabilization). Since the control strategies for the first two problems are quite similar, path following will not be discussed. In section 3.1 the class of non-holonomic systems is discussed, together with the kinematics of the wheeled mobile robot. Section 3.2 discusses tracking control. An often-used state transformation into the chained form, which is needed for many point stabilization controllers, is described in section 3.3. Section 3.4 actually presents some point stabilization controllers and section 3.5 discusses the point stabilization control of a mobile robot pulling one or two trailers. Since it is not the purpose of this project to actually design controllers, but just to create a test environment for them, only some existing controllers from literature will be discussed. Simulations in Matlab/Simulink are performed to recognize the behaviour of the controllers.

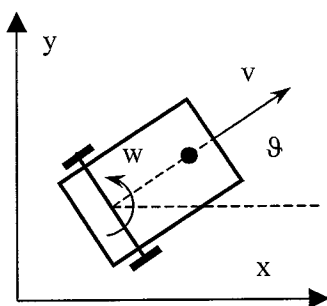
#### 3.1 Non-holonomic systems

Consider the mobile robot, schematically drawn in figure 3.1. The front (castor) wheel is free to rotate without any restrictions. The rear wheels can be rotated independently. This makes our robot a (2,0)-type robot, as described in [6] In this article several types of wheeled mobile robots are recognized by comparing their degree of mobility and the degree of steerability. The degree of mobility is equal to the number of driven wheels and is 2 in our case. The degree of steerability is equal to the number of steering wheels, which equals 0 in our case.

We assume that the non-slipping condition holds in this case. Now it is impossible for the robot to drive in side-way directions. Only forward and backward movement and rotation is possible. This kind of constraint is called a non-holonomic constraint and it is given in equation 3.1. This is by definition a non-holonomic constraint, because it cannot be integrated. It is not the time derivative of a function of the generalized coordinates.

Since no side-way movement is possible, the control problem becomes under-actuated. We need to control three coordinates,  $x$ ,  $y$  and  $\vartheta$ , with only the forward velocity  $u$  and the steering velocity  $w$  to achieve this. The kinematic model for this kind of robot is given in equation 3.2.

These equations form the basic model for a larger class of non-holonomic systems like an equivalent four-wheeled robot or a truck with one or more trailers. These models can be easily extended to a model with dynamic input extension, i.e. torque as input instead of velocity. Since we are using stepper motors, there is no use to convert the model and we will be using the model given in equation 3.2.



$$\text{Non-holonomic constraints:} \\ \dot{x} \cdot \sin \vartheta - \dot{y} \cdot \cos \vartheta = 0 \quad [3.1]$$

$$\text{Kinematic model:} \\ \dot{x} = v \cos(\vartheta) \\ \dot{y} = v \sin(\vartheta) \\ \dot{\vartheta} = w \quad [3.2]$$

Figure 3.1: Schematic view of the mobile robot in Cartesian coordinates.

Interesting about this kind of systems is that no smooth time-invariant stabilizing feedback laws exist [5]. It can be proven that the system is controllable [7], i.e. it can be steered from any state configuration to another state configuration in finite time by using finite inputs, but for non-linear systems this does not imply the existence of stabilizing smooth static state feedbacks. In a similar way it can be proven that tangent linearization and full-state feedback linearization of the system are not possible [7]. Other control structures are needed to stabilize the system, as we will see in the next sections.

### 3.2 Tracking control

The tracking control problem for a wheeled mobile robot can be solved using relatively classical non-linear control techniques [7]. Simply stated, the problem is to follow a virtual reference robot equal to the used robot. The reference robot follows a time-dependent trajectory and the actual robot should be stabilized to this trajectory. As a result not only the position and orientation of the robot are important, but its forward and rotational velocity as well. The tracking control problem is illustrated in figure 3.2, where the subscript 'r' stands for reference and the subscript 'c' for the centre of the actual robot.

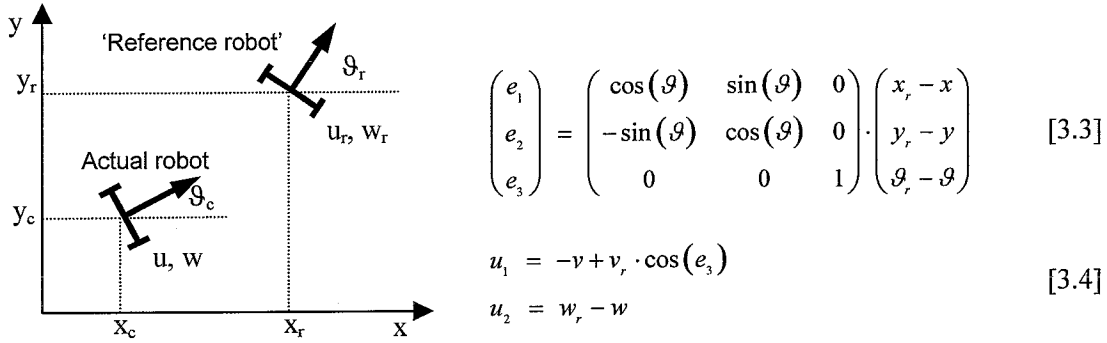


Figure 3.2: Tracking control problem

The above-defined tracking problem involves error equations, which describe the time evolution of the difference between the actual state and the reference state. In order to simulate two different tracking controllers a change of coordinates is used as shown in equation 3.3. Together with this change of coordinates, the change of inputs as given in equation 3.4 is introduced.

Some assumptions should be made to find a stable feedback control law for the tracking problem. First, the reference velocities  $v_r$  and  $w_r$  should be bounded and have bounded derivatives. Secondly, the reference should be persistently exciting. When the reference robot is at rest, the tracking problem becomes a point stabilization problem, which requires more difficult control techniques.

We can now differentiate equation 3.3 to gain the tracking error equations:

$$\begin{pmatrix} \dot{e}_1 \\ \dot{e}_2 \\ \dot{e}_3 \end{pmatrix} = \begin{pmatrix} 0 & w & 0 \\ -w & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} e_1 \\ e_2 \\ e_3 \end{pmatrix} + \begin{pmatrix} 0 \\ \sin(e_3) \\ 0 \end{pmatrix} \cdot v_r + \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \quad [3.5]$$

Using these error equations, simulations are performed with one linear and one non-linear feedback design. More information about the controller design can be found in [7]. Other controllers are proposed [12, 23].

The Simulink-scheme for the tracking controllers can be found in appendix C1. An example of a Matlab initialization file can be found in appendix C3. Four blocks can be identified. The first one, the reference block, applies the trajectory, which the robot should follow. The reference states  $x_r$ ,  $y_r$  and  $\theta_r$  are derived from the input reference velocities  $v_r$  and  $w_r$ . In the second block the error equations (3.5) are modeled. The third block contains the controller; the fourth contains the system as given in equation 3.2. In order to reconstruct the states of the robot an integration scheme is used. This scheme was proposed and tested in [9].

All simulations for tracking controllers are performed with a circular reference. A constant reference forward velocity and angular velocity are applied in the reference block, resulting in a reference circle with radius  $r_r = v_r/w_r$ . The starting point of the reference is  $(x_r, y_r) = (0, y_r)$ , the starting point of the robot is  $(x, y) = (0,0)$ . In the simulations the maximum forward and angular velocities are set to the values calculated in section 2.1.2.

### 3.2.1 Linear control design

The linear control design results in the following control laws:

$$\begin{aligned} u_1 &= -k_1 \cdot e_1 \\ u_2 &= -k_2 \cdot \text{sgn}(v_r) \cdot e_2 - k_3 \cdot e_3 \end{aligned} \quad [3.6]$$

$$\text{with: } k_1 = 2 \cdot \xi \cdot \sqrt{w_r^2 + b \cdot v_r^2}, \quad k_2 = b \cdot |v_r|, \quad k_3 = 2 \cdot \xi \cdot \sqrt{w_r^2 + b \cdot v_r^2}$$

Here the control parameters  $\xi$  and  $b$  are positive real numbers. Although this control law is called linear in [7], a sign term is present. Therefore, we assume the reference forward velocity will always have the same sign. In practice this means the robot will drive only forward or only backward during one experiment.

The above-designed error equations and control laws are implemented in Matlab/Simulink in order to test the controller. A forward velocity of 0.3 [m/s] and a angular velocity of 1 [rad/s] were supplied; the control parameters  $\xi$  and  $b$  were set to 1.5 [-] and 0.5 [m<sup>2</sup>] respectively. The reference trajectory is started after 2 seconds. In figure 3.3 the  $x,y$ -trajectory and the forward and angular velocities resulting from this experiment are given.

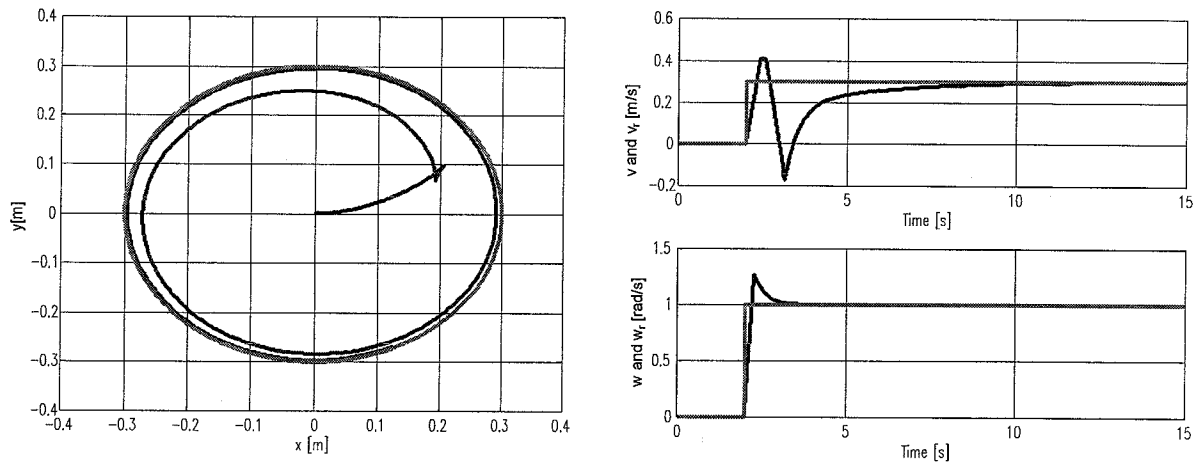


Figure 3.3: Simulation results for a linear tracking controller

The simulation results show that the robot state converges to the reference state within 15 seconds. This holds for both the velocity trajectory and the position trajectory.

By adjusting the control parameters the configuration of the trajectory can be adjusted. Increasing the parameter  $\xi$  makes the trajectory converge slower. Too high values of  $\xi$  make the control

unstable. Decreasing  $\xi$  results in overshoot in the trajectory. In this case it means that the robot would drive outside the reference circle while converging to it. Increasing the parameter  $b$  results in higher inputs, but not necessarily in faster convergence. This is a result of the changing form of the robot trajectory; in some cases it has to make one or more extra ‘turns’. To high values of  $b$  make the control unstable. Taking smaller values for  $b$  then the chosen  $0.5 \text{ [m}^2\text{]}$  does not have much influence. The convergence becomes just a little slower.

### 3.2.2 Non-linear control design

The non-linear control design from [7] results in the following control laws:

$$\begin{aligned} u_1 &= -k_1 \cdot e_1 \\ u_2 &= -k_4 \cdot v_r \cdot \frac{\sin(e_3)}{e_3} \cdot e_2 - k_3 \cdot e_3 \end{aligned} \quad [3.7]$$

Here  $k_1$  and  $k_3$  are defined as in section 3.3.1 and  $k_4 = b$ . For this controller the extra assumption is made that  $e_3$  does not equal 0. Again, this controller is implemented in Simulink. The forward and angular velocity were  $0.3 \text{ [m/s]}$  and  $1 \text{ [rad/s]}$  respectively. The control parameters  $\xi$  and  $b$  were set to  $1.5 \text{ [-]}$  and  $0.5 \text{ [m}^2\text{]}$  respectively. In figure 3.4 the  $x,y$ -trajectory and the forward and angular velocities resulting from this experiment are given.

It shows from figure 3.4 that the results for this controller, using the same trajectory and control parameters, are almost equal to the results obtained using the linear controller as discussed in section 3.2.1. This is a result of the used control structure, which is more or less the same in both cases. Again the state converges to the reference state within 15 seconds. When the control parameters are varied, the same correlations are obtained as in section 3.2.1.

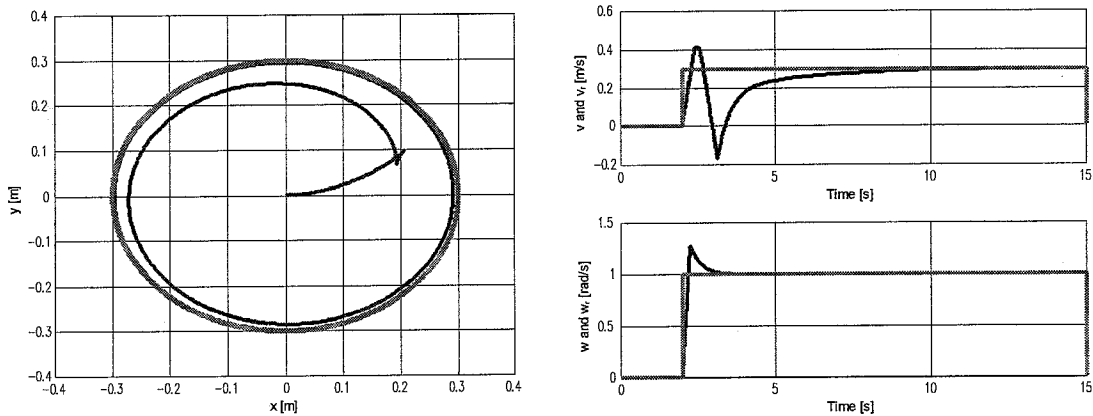


Figure 3.4: Simulation results for a non-linear tracking controller

### 3.3 The chained form

In most point stabilization control design cases for wheeled mobile robots a preliminary change of state coordinates is used which transforms the model equations into a simpler ‘canonical’ form. For instance, the following change of coordinates and inputs:

$$\begin{aligned} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} &= \begin{pmatrix} 0 & 0 & 1 \\ \cos(\vartheta) & \sin(\vartheta) & 0 \\ \sin(\vartheta) & -\cos(\vartheta) & 0 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ \vartheta \end{pmatrix}, & \quad \begin{aligned} u_1 &= w \\ u_2 &= v - w \cdot x_3 \end{aligned} \end{aligned} \quad [3.8]$$

transforms the system (3.2) into:

$$\begin{aligned}\dot{x}_1 &= u_1 \\ \dot{x}_2 &= u_2 \\ \dot{x}_3 &= x_2 \cdot u_1\end{aligned}\tag{3.9}$$

This canonical form can be expanded for systems with more than three states. In that case, two different techniques can be used to obtain the new coordinates and inputs. The first leads to so-called chained form equations; the second to power form equations. When those higher order systems are concerned, for example when trailers are added to the system, it depends on the situation which conversion is used. The chained form is preferred for tracking control and path following, but often no canonical form at all is not used for this kind of controls, as shown in section 3.2. For point stabilization the control strategy designer is free to pick one of both transformations. More information on the chained form and power form can be found in [26, 29] and [10, 16] respectively.

For our robot only three states are of interest. In this case the power form equals the chained form.

### 3.4 Point stabilization

The most difficult problem in mobile robot control is the point stabilization problem. The definition of this problem is: “Given an arbitrary desired state of the robot, find a control law that stabilizes the difference between the desired state and the actual state about zero, independent on the initial state of the robot”. One example of a point stabilization problem is the parking problem as illustrated in figure 3.5.

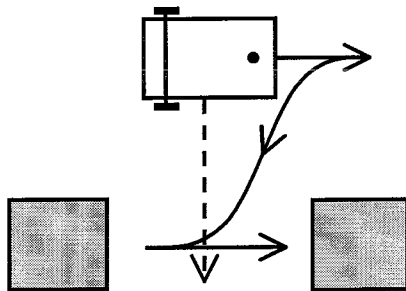


Figure 3.5: The parking problem: non-feasible motion (dashed line) and feasible motion (solid line).

Since there do not exist any smooth control laws to solve this problem (section 3.1), alternative control techniques should be used. Those are more complicated than, for example, tracking controllers, since our reference is not persistently exciting; our reference is a fixed point all the time. Basically, three different control techniques are used for point stabilization: Smooth time-varying non-linear feedbacks, discontinuous time-invariant control laws and hybrid controls. Those control techniques will be discussed and simulated in the next subsections. The simulation scheme for those controllers is given in appendix C2. This scheme contains three important blocks. The first block contains the coordinates transformation into the chained form as mentioned in section 3.3. The second block contains the controller. The third block contains the system. The starting point of the robot is set by adjusting the initial values of the integrators in this block. In the simulations in the next sessions the robot will have to perform a parking movement starting 0.2 [m] from the origin in y-direction, with its orientation equal to 0. The reference point for the controllers is always the origin (0,0), with the orientation of the robot equal to 0.

### 3.4.1 Time-varying control

Quite a lot of research has been done in the field of time-varying controls [1, 7, 13, 22]. Its basic principle is to add one or more sinusoidal terms to the control laws, which keep the robot moving. The amplitude of these terms becomes smaller when the difference between desired and actual state decreases. An example of this kind of controllers from [13] is given in equation 3.10:

$$\begin{aligned} u_1 &= -k_1 \cdot x_1 + k_2 \cdot x_3 \cdot (\sin(t) - \cos(t)) \\ u_2 &= -k_3 \cdot x_2 - k_4 \cdot x_1 \cdot x_3 - k_5 \cdot x_1 \cdot (x_1 + x_3 \cdot \cos(t)) \cdot \cos(t) \end{aligned} \quad [3.10]$$

Simpler, but also much more difficult can be found in literature. In figure 3.6 the  $x,y$ -trajectory and the parameter convergence for the controller in equation 3.10 are given. The parameters  $k_1, k_2, k_3, k_4$  and  $k_5$  were set to 1, 1, 0.1, 10 and 1 respectively.

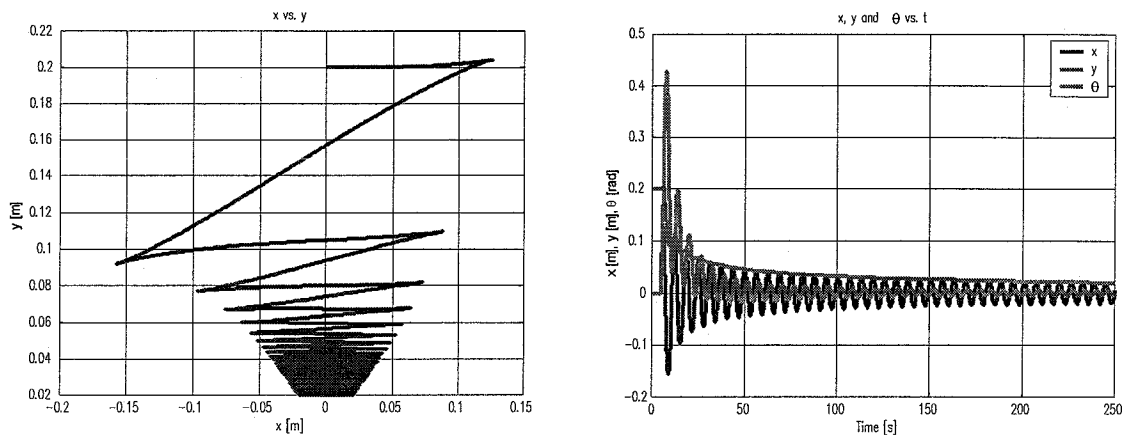


Figure 3.6: Simulation results for a time-varying controller

It shows that the controller is stable and asymptotically stabilizes the robot to the origin, but it is obvious that it is very slow; after 250 seconds the robot is still 0.02 [m] away from its reference point. This is one of the main disadvantages of time-varying controllers. Two main advantages of this type of controllers are that they are easily understood and quite accurate. You can get your robot as close as possible near the reference point, by just waiting long enough.

By adjusting the control parameters  $k_i$  preferences can be given to the state variables to control. Basically this changes the form of the  $x,y$ -trajectory. For example, when converging the  $x$ -coordinate gets high priority and converging the  $\theta$ -coordinate gets low priority, the trajectory will stay within a small  $x$ -domain and it turn more. The convergence speed can be increased by adjusting the control parameters, but just a little. Really high or low values for the parameters can make the controller unstable, but within a normal range the controller is always stable.

A possibility to obtain not only asymptotic stability, but also exponential stability, which guarantees faster convergence, is to design a controller which is smooth in the entire domain except at the origin [13]. In equation 3.11 a controller of this kind is given.

$$\begin{aligned} u_1(x,t) &= -k_1 \cdot x_1 + k_2 \cdot \frac{x_3}{\rho(x)} \cdot \cos(t) \\ u_2(x,t) &= -k_3 \cdot x_2 - \frac{x_3^2}{\rho^3(x)} \cdot \sin(t) \\ u_1(0,t) &= u_2(0,t) = 0, \quad \rho(x) = (x_1^4 + x_2^4 + x_3^2)^{1/4} \end{aligned} \quad [3.11]$$

In figure 3.7 the simulation results for this controller are given. The parameters  $k_1, k_2, k_3$  and  $k_4$  were all set to 1.

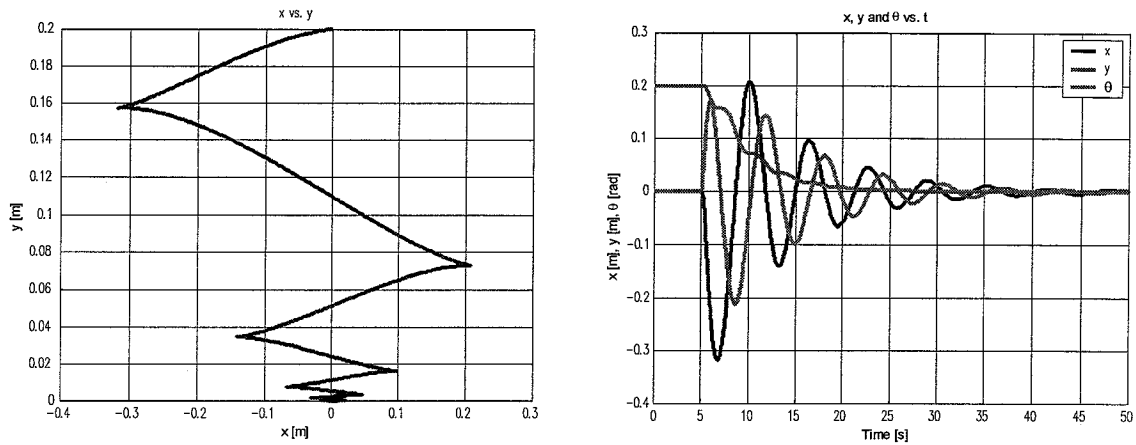


Figure 3.7: Simulation results for a non-smooth time-varying controller.

As stated above, this controller converges the state not only asymptotically to the origin, but also exponentially. In 50 seconds the robot stands within a 5 [mm] radius of its final point. Using this kind of small adaptations and a non-smooth origin, time-varying controllers form a very useful technique for solving point stabilization problems.

Again, changing the control parameters will change the form of the trajectory, but it merely affects the convergence speed. Only extreme values can make the controller unstable or make it converge significantly slower.

### 3.4.2 Discontinuous time-invariant control

Discontinuous time-invariant controllers can be classified into two types: piecewise continuous controls and sliding mode controls. In this project the sliding mode controllers are not discussed. Piecewise continuous controllers can differ a lot from each other. Different techniques can be used to design this type of controllers and they often result in complex control laws [2, 7, 8, 10, 16, 27,]. Generally, piecewise continuous will show faster parameter convergence than time-varying controllers. A disadvantage is that the discontinuity of the controllers reduces the accuracy. The controller will make the states jump over the origin time after time. In fact it is impossible to reach the origin.

An example of a piecewise continuous controller is given in [7,8]. The idea behind this controller is that there always exists a circle with radius  $r(x,y)$  and its centre on the  $y$ -axis, which passes through the origin and the control point of the robot (figure 3.8).

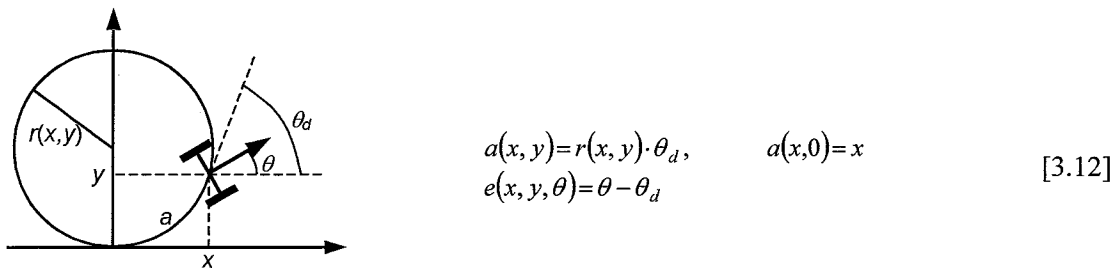


Figure 3.8: Illustration of the variables  $a$ ,  $e$  and  $\theta_d$



We can now define the arc length  $a(x,y)$  between the control point and the origin and an error angle  $e(x,y,\theta)$ , which is the difference between the angle of the tangent of the defined circle and the robot orientation (equation 3.12). At the  $x$ -axis the arc length is equal to the  $x$ -coordinate of the control point of the robot.

To obtain the desired behaviour, a controller is developed, which is given in equation 3.13.

$$\begin{aligned} u_1 &= -\gamma \cdot b_1 \cdot a \\ u_2 &= \gamma \cdot b_1 \cdot b_2 \cdot a - k \cdot e \end{aligned} \quad [3.13]$$

With  $k$  and  $\gamma$  two positive real control parameters and  $b_1$  and  $b_2$  defined as:

$$\begin{aligned} b_1 &= \cos(\vartheta) \cdot \left( \frac{\vartheta_d}{\beta} - 1 \right) + \sin(\vartheta) \cdot \left\{ \frac{\vartheta_d}{2} \cdot \left( 1 - \frac{1}{\beta^2} \right) + \frac{1}{\beta} \right\} \\ b_2 &= \cos(\vartheta) \cdot \frac{2 \cdot \beta}{(1 + \beta^2) \cdot x} - \sin(\vartheta) \cdot \frac{2}{(1 + \beta^2) \cdot x} \end{aligned} \quad [3.14]$$

with:  $\beta = y/x$ .

This controller is based on the idea that first the orientation should be adjusted to equal  $\theta_d$  and the arc length should be decreased. Figure 3.9 gives an idea of kind of trajectories are generated by this kind of controller.

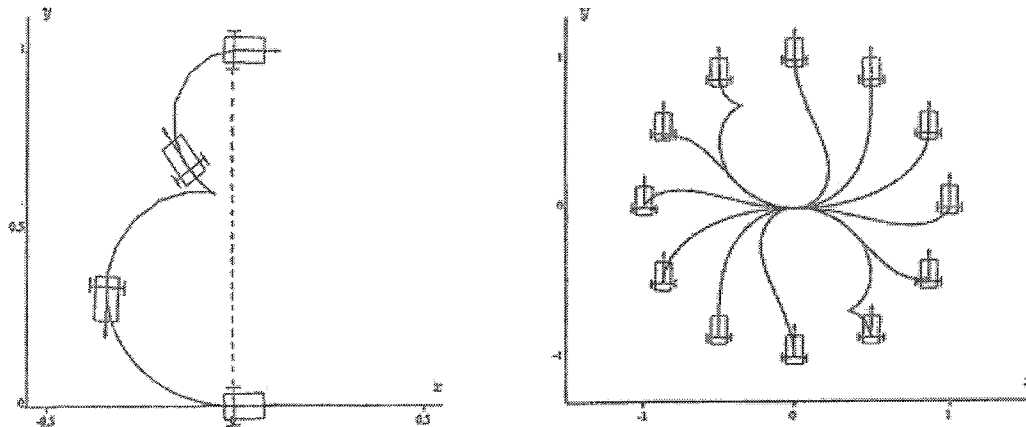


Figure 3.9: Trajectories for a piecewise continuous controller

The control laws have been implemented in Simulink to test the controller. One of the main problems implementing this kind of controllers is how to handle the discontinuities. In this case some solution should be found when crossing the  $y$ -axis, where numerical problems appear. To avoid this problem in the first place, in this case the starting point of the trajectory is not placed on the  $y$ -axis, but a little besides it. The results are given in figure 3.10.

The shape of the trajectory shown in 3.10 is quite similar to the ones shown in 3.9. The main difference is, that the results from the simulations are much slower, than those claimed in the literature. Possibly, this is a result of the fact that the simulations are done in discrete time. More likely, numerical problems are the reason for the slow convergence. Since the controller is quite complex, it is difficult to find the trouble spot. However, if an error would be present in the simulations, it would be pure luck to create a controller, which converges the parameters to the origin as it does as shown in figure 3.10.

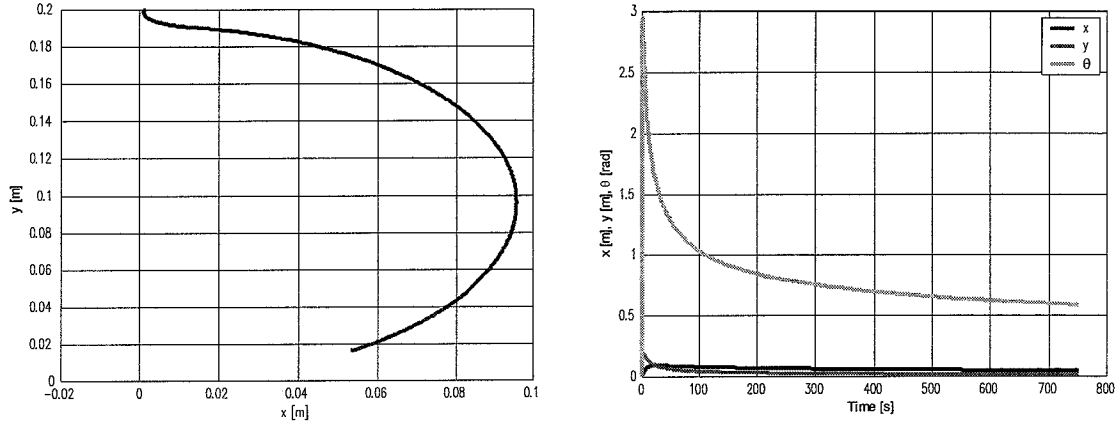


Figure 3.10: Simulation results for a piecewise continuous controller

Adjusting the control parameters does change the shape of the trajectory, but merely the convergence speed. The parameters should be chosen carefully, because instability appears in many configurations.

### 3.4.3 Hybrid control

In hybrid control two or more different control techniques are combined. In the case of point stabilization two different types of hybrid control can be recognized. The first uses a control law in which several control techniques are combined along the entire working area. In fact, the non-smooth time-varying controller discussed in section 3.4.2 could be called a hybrid controller, because it combines the time-varying control technique with the discontinuous technique. Because the time-varying behaviour is dominant, we still call it a time-varying controller. Other combinations of control techniques can be found [1, 21].

The second type of hybrid controller uses two different controllers on two different parts of the working area. This way the advantages of the controllers can be combined. A controller can be designed for example, which makes the robot drive as fast as possible to a small area around the origin. A second controller, for example a time-varying controller, will then be used in this area to reach the origin and control the orientation. An example of this kind of controller is given in [21]. The proposed controller shows parameter convergence and proper performance, but the used time-varying controller is quite slow. To increase the performance this controller is replaced by the non-smooth time-varying controller, discussed in section 3.4.1. The controller that makes the robot move to a circle around the origin with radius  $R$  is given in equation 3.15.

$$\begin{aligned} u_1 &= \alpha(x, y, \vartheta) \cdot (x \cdot \sin(\vartheta) - y \cdot \cos(\vartheta) + R) \\ u_2 &= -\frac{1}{R} \cdot \alpha(x, y, \vartheta) \cdot (x \cdot \cos(\vartheta) + y \cdot \sin(\vartheta)) \end{aligned} \quad [3.15]$$

$$\text{with: } \alpha(x, y, \vartheta) = \frac{k_1}{k_2 + \sqrt{X^2 + Y^2}}$$

$$\text{and: } X = x + R \cdot \sin(\vartheta), \quad Y = x - R \cdot \cos(\vartheta)$$

$k_1$  and  $k_2$  are two positive real control parameters.

Simulations with the controller from equation 3.15 combined with the controller from equation 3.11 were performed. The radius of the area around the origin was set to 20 [mm] and the control

parameters  $k_1$  and  $k_2$  both to 1. The time-varying controller is tuned the same way as done in section 3.4.1. The results are given in figure 3.11.

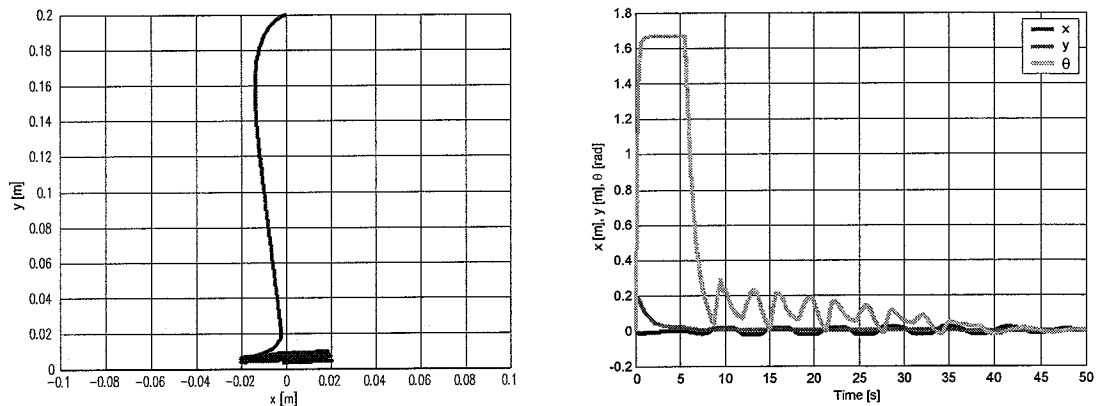


Figure 3.11: Simulation results for a hybrid controller

The two controllers can be identified quite easy from figure 3.11. The hybrid controller shows fast parameter convergence as expected, although it is not much faster than the non-smooth time-varying controller alone. This is a result of the fact that the first controller works only in a small area. When the starting point is further away from the origin and the area of controller 2, this hybrid controller will gain more profit of the first controller.

By adjusting the control parameter  $R$ , the area around the origin can be made larger and smaller. Making this area too small will make it almost impossible for the second controller to reach the origin. Making the area too large will take away the advantage of the first controller.

Increasing  $k_1$  or decreasing  $k_2$  will make the first controller go faster to the area around the origin. As a result the needed inputs will increase.

### 3.4.4 Point stabilization controller evaluation

It is proven that all three kinds of controllers can solve the parking problem. Not all controllers are useful in all cases however. Time-varying controllers have the advantage of their simplicity and accuracy. By just waiting long enough, the controller will stabilise the robot as close to the reference point as possible. The final accuracy depends on the system and measurement accuracy. The main drawback of this kind of controllers is their slow convergence rate.

Discontinuous time-invariant controllers are far more complex than the time-varying controllers are, which makes their tuning more difficult. Their advantage is in most cases the fast convergence. Their discontinuities however bring a lot of disadvantages. The accuracy is reduced because the robot keeps jumping over its final position. Secondly, numerical problems appear when simulations are performed. In literature however, very promising controllers are proposed, which could give excellent performance when the numerical problems are solved.

Hybrid controllers form the most promising group of controllers. By adding the good properties of two or more different kind of controllers and reducing their disadvantages, a fast control law for every application can be found. A drawback might be that the designed controllers are not useful for all applications so the robot task should be well-defined.

### 3.5 Trailers

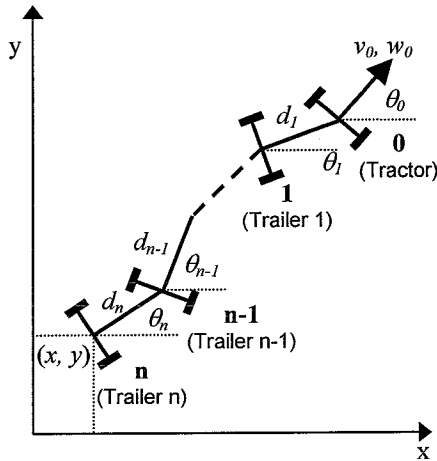
An interesting expansion of the point stabilization problem is the addition of trailers to the system. This makes the system and the used controllers more complex. The relative easy point stabilization ‘parking’-problem now changes into a problem of parking a tractor pulling  $n$  trailers. Everyone who has ever parked a car with trailer can imagine how difficult it is to park a car with two or more trailers. The pulling robot will be called tractor from now on.

Quite a lot of research has been done on this topic [10, 15, 18, 26, 29] resulting in some interesting controllers. The first thing to do is to expand the kinematic model for the trailers and transform it into the chained form. This will be discussed in section 3.5.1. In the next session a controller design will be discussed and in section 3.5.3 simulations with a tractor pulling one and two trailers are presented.

Again it is not my goal to actually design a controller for this kind of systems, but only to study and simulate their behaviour. Adding trailers to the robot might be an interesting topic in future research.

#### 3.5.1 Kinematic model for a tractor pulling $n$ trailers

The kinematic model of the robot given in equation 3.2 should now be expanded for the addition of  $n$  trailers [26]. A schematic view of the new system is given in figure 3.12. In this figure the parameters  $d_i$  are the distances between the axle between the two wheels of one trailer (or the tractor) and the next. The parameters  $\theta_i$  are the orientations of the trailers and tractor.  $v_0$  and  $w_0$  are the applied forward and angular velocity respectively. The kinematic model for this system, seen from the tractor is given in equation 3.16. Here the point  $(x,y)$  is the absolute position of the axle between the wheels of the rear trailer.



$$\begin{aligned}
 \dot{x} &= \cos(\theta_n) \cdot v_n \\
 \dot{y} &= \sin(\theta_n) \cdot v_n \\
 \dot{\theta}_n &= \frac{1}{d_n} \cdot \sin(\theta_{n-1} - \theta_n) \cdot v_{n-1} \\
 &\vdots \\
 \dot{\theta}_i &= \frac{1}{d_i} \cdot \sin(\theta_{i-1} - \theta_i) \cdot v_{i-1}, \quad i=1, \dots, n \\
 &\vdots \\
 \dot{\theta}_1 &= \frac{1}{d_1} \cdot \sin(\theta_0 - \theta_1) \cdot v_0 \\
 \dot{\theta}_0 &= w
 \end{aligned} \tag{3.16}$$

Figure 3.12: Schematic view of a tractor pulling  $n$  trailers

The non-holonomic constraints now take the following form:

$$\sin(\theta_i) \cdot \dot{x} - \cos(\theta_i) \cdot \dot{y} - \sum_{j=i+1}^n d_j \cdot (\theta_i - \theta_j) \cdot \dot{\theta}_j = 0 \tag{3.17}$$

In most cases of controlling a tractor pulling trailers the chained form is used. In order to obtain this chained form, first a transformation of coordinates from the tractor to the rear trailer should be made. Then, conversion in the chained form is possible. It goes too far to discuss this conversion in detail here, but it is worked out in [26].

### 3.5.2 Controller design

In literature many controllers can be found, which are capable of converging more than the standard number of three parameters to 0. In this section a controller described in [10] will be discussed.

The basic idea behind this controller is that one state is varied over time, while a second algorithm tries to converge the other states to 0. The amplitude of the varied state decreases over time. To obtain this to different control laws are designed. These control laws are too complex to discuss all their parameters here, but the essence will be briefly described. The complete control laws can be found in [10]. The first control law is given in equation 3.25:

$$u_1 = k(x(t_i)) \cdot f(t) \quad [3.21]$$

Here,  $f(t)$  is a sinoid function and  $k(x(t_i))$  is a saturated function that depends on the sign of  $x_i$ , the function  $f(t)$  and some constant control parameters.

The second control law is given by equation 3.22:

$$u_2 = \begin{cases} \Gamma(k(x(t_i)), t)^T \cdot z, & x(t_i) \neq 0 \\ 0 & x(t_i) = 0 \end{cases} \quad [3.22]$$

Here,  $z$  is the state vector and  $\Gamma(k(x(t_i)), t)$  is a set of functions depending on the function  $f(t)$  and its time derivatives and some constant control parameters. For every extra state, an extra parameter  $\Gamma$  is introduced. Adding more states results in more complex parameters  $\Gamma_i$ , which then depend on more time derivatives of the function  $f(t)$ .

Other controllers can be found in literature and most of them work with the same principles as the one described above. In fact, these controllers can be seen as extended time-varying controllers. Examples of extended discontinuous time-invariant controllers can also be found, because of their extreme complexity, no simulations have been performed with these.

### 3.5.3 Simulation results

Using the kinematic model discussed in section 3.5.1 and the controller discussed in section 3.5.2, simulations have been performed with one and two trailers. The distances  $d_i$  between the axes are all set to 0.2 [m]. The starting point for the tractor is always  $(x, y, \theta) = (0, 0.2, 0)$ , just like the starting point used with the ordinary point stabilization simulations. The trailers are lined up in a straight line behind the tractor.

The results for the one-trailer-case are shown in figure 3.13. In this case four states need to be controlled to 0. A Matlab-file for this simulation is given in appendix C4.

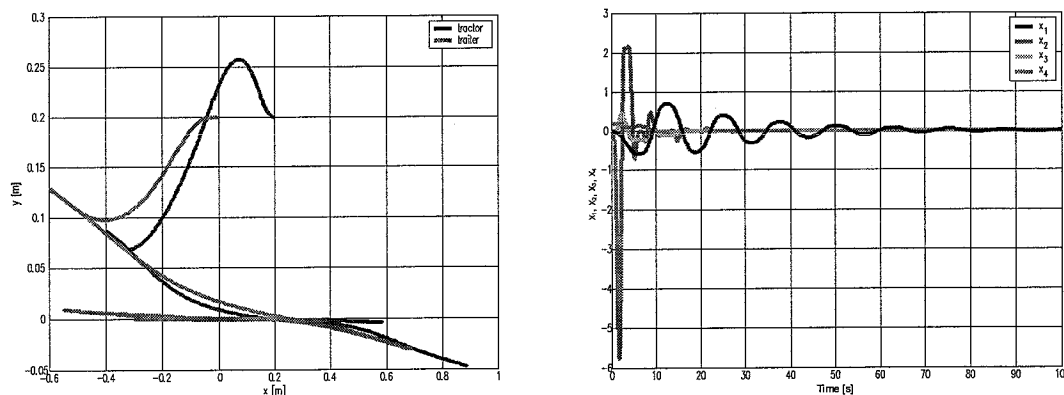


Figure 3.13: Simulation results for a tractor pulling one trailer

It shows from figure 3.13 that the controller converges the four states to 0 quite fast. The  $x,y$ -trajectory shows the typical behaviour of a trailer: When driving backward you need to steer to the right to make the trailer go to the left.

In figure 3.14 the simulation results for a tractor pulling two trailers are given. Now five states need to be converged to 0. A Matlab-file for this simulation is given in appendix C5.

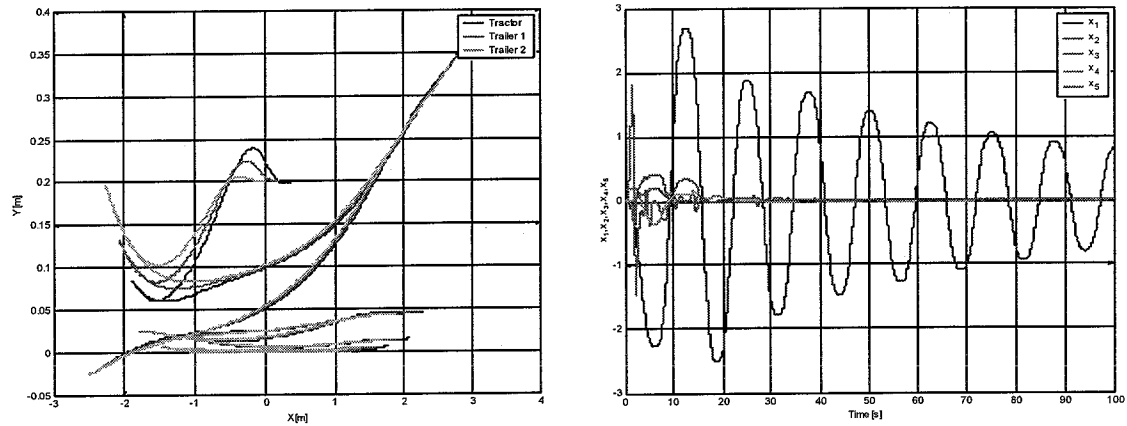


Figure 3.14: Simulation results for a tractor pulling two trailers

Again, these figures show parameter convergence. However some other remarks should be made. First, in the  $x,y$ -trajectory-plot, the distance between the trailers seems to increase and decrease along the trajectory. In fact, this is just a result of the scaling of the figure. The trailers stay all fixed at 0.2 [m] from each other. In the second plot the difference between the two controllers  $u_1$  and  $u_2$  is very obvious. The  $x_1$ -state is varied over time, while the other states are converged to 0. It is possible to make  $x_1$  converge to 0 faster, but only limited. To fast convergence of  $x_1$  causes instability. It is also obvious that every extra trailer makes the convergence process slower.

## 4 Dead-reckoning reconstruction and calibration

In order to implement the controllers discussed in chapter 3, realtime state reconstruction is needed. In this phase of the project the only way to obtain this is by measuring the wheel rotations. This kind of so-called dead-reckoning methods is discussed in section 4.1. In order to make the robot do exactly what we want, it needs to be calibrated. This calibration is discussed in section 4.2. Some realtime experiments, the adapted Simulink-scheme, and results are given in section 4.3.

### 4.1 Dead-reckoning

Dead-reckoning (from deduced reckoning) methods are used in a large amount of applications, starting from sailboat navigation using a compass and velocity measurements up to almost every mobile robot [4]. Dead-reckoning methods have two main advantages: First of all, dead-reckoning sensors like incremental encoders and gyroscopes are cheap and easy to implement. In our case, the stepper motor input can be used for this purpose, without adding any hardware to the system.

Secondly, dead-reckoning methods can easily be fused with other measurement systems. In some cases this fusion increases the accuracy of the system, in other cases this makes less frequent absolute position updating possible. Dead-reckoning keeps the robot sufficiently on its track, while its computer generates new coordinates or while its sensors search for new beacons. As a result, larger computing times are supported and less beacons or landmarks are needed to travel a given distance.

The main disadvantage of dead reckoning is the fact that only relative positions and orientations can be measured. As a result small errors will grow larger in time, without the possibility to correct them. Therefore calibration of the robot and its sensors is very important, since most systematic errors can be excluded or minimized in advance. Systematic errors include unequal wheel diameters, wrong wheel-base estimation, the misalignment of the wheels, finite encoders resolution and finite encoder sampling rate. More difficult to exclude are non-systematic errors, like travelling over uneven floors, travelling over unexpected objects on the floor and wheel slippage.

Because our robot drives on a horizontal whiteboard, wheel slippage is the main issue. Slippage is caused by a slippery floor, over acceleration, internal (castor wheels) and external forces, fast turning, and non-point wheel contact with the floor. This slippage can never be corrected by calibration or dead-reckoning reconstruction. One possibility to avoid this is to set a maximum forward and angular acceleration, as will be discussed in section 4.3.

### 4.2 Calibration

In order to correct the most of the systematic errors mentioned in the previous section calibration is needed. The most important errors are in this case unequal wheel diameters and wrong wheel base estimation. Before we can perform measurements to correct these errors, we need to be sure no other errors occur between the PC and the actual wheel rotation. This means we need to check all components in figure 2.4 on their accuracy.

We assume that dSpace exactly outputs the voltage we want to the voltage to frequency converter cards. These cards were tested and show perfect linear behaviour (section 2.2). The combination of microstepping driver and stepper motor results in an accuracy on the wheel rotation of  $0.18^\circ$ . Since these devices do not provide systematic errors, we will assume the occurring errors are a

result of unequal wheel radii and wheelbases. The first way to obtain these parameters is just to measure them manually. This gives the following results:

$$\begin{aligned} r_1 &= 41[\text{mm}], & r_2 &= 42[\text{mm}], \\ B_1 &= 78[\text{mm}], & B_2 &= 80[\text{mm}] \end{aligned}$$

These results can be compared to the results of two calibration experiments, which will be discussed in the next subsections.

#### 4.2.1 First calibration experiment

The first experiment consists of two parts. In the first part, the robot should drive a straight line. This means the same wheel rotations are supplied to both wheels. When the two wheels have equal diameters and the wheel-bases are equal, then the robot should drive a straight line. If not, the trajectory will typically look like this:

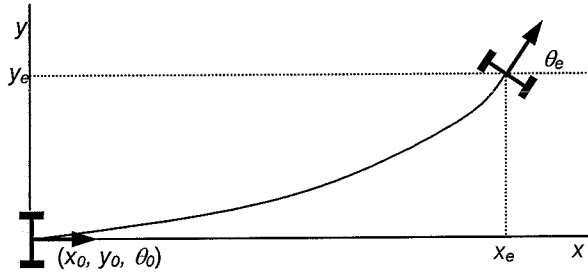
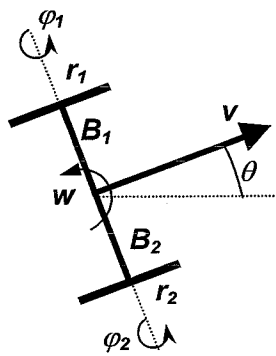


Figure 4.1: Typical trajectory for bad calibrated mobile robot

In the second part of the experiment the robot will rotate around the centre of the axis between the wheel centres. This can be obtained by making the wheel rotations exactly the opposite of each other.

The first thing to do is to find out how the wheel diameters and wheelbases are related to the input variables. We define the wheel radii and bases as given in figure 4.2. Equation 4.1 gives the relation between the wheel rotations  $\varphi_1$  and  $\varphi_2$  and the forward and angular velocity.



$$\begin{pmatrix} v \\ w \end{pmatrix} = \frac{1}{B_1 + B_2} \cdot \begin{pmatrix} B_2 \cdot r_1 & B_1 \cdot r_2 \\ -r_1 & r_2 \end{pmatrix} \cdot \begin{pmatrix} \varphi_1 \\ \varphi_2 \end{pmatrix} = M \cdot \begin{pmatrix} \varphi_1 \\ \varphi_2 \end{pmatrix} \quad [4.1]$$

$$\begin{aligned} r_1 &= \frac{c_1 \cdot c_4 - c_2 \cdot c_3}{\varphi_{s,1} \cdot c_4 - \varphi_{r,1} \cdot c_2}, & r_2 &= \frac{c_1 \cdot c_4 - c_2 \cdot c_3}{\varphi_{s,2} \cdot c_4 - \varphi_{r,2} \cdot c_2}, \\ B_1 &= \frac{c_1 \cdot \varphi_{r,1} - c_3 \cdot \varphi_{s,1}}{\varphi_{r,1} \cdot c_2 - \varphi_{s,1} \cdot c_4}, & B_2 &= \frac{c_1 \cdot \varphi_{r,2} - c_3 \cdot \varphi_{s,2}}{\varphi_{r,2} \cdot c_4 - \varphi_{s,2} \cdot c_2} \end{aligned} \quad [4.2]$$

$$\text{with: } \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = M \cdot \begin{pmatrix} \varphi_{s,1} \\ \varphi_{s,2} \end{pmatrix}, \quad \begin{pmatrix} c_3 \\ c_4 \end{pmatrix} = M \cdot \begin{pmatrix} \varphi_{r,1} \\ \varphi_{r,2} \end{pmatrix}$$

Figure 4.2: Definitions of wheel radii and wheelbases

Now the radii and bases can be obtained by doing the straight line experiment with wheel rotations  $\varphi_{s,i}$  and the rotation experiment with wheel rotations  $\varphi_{r,i}$ . This results in the equations given in 4.2.



The forward and angular velocity can be computed as follows:

$$\int_0^T v dt = \frac{x_e \cdot \theta_e}{\sin(\vartheta_e)} = \frac{y_e \cdot \theta_e}{1 - \cos(\theta_e)}, \quad \int_0^T w dt = \theta_e \quad [4.3]$$

If both the forward and angular velocity ( $v, w$ ) and the wheel rotations for both experiments  $\varphi_{j,i}$  are known, the matrix  $M$  can be solved and the wheel radii and bases will be known.

Since we have only dead-reckoning methods available for measuring the position and orientation of the robot, we should be really careful with the results. If we place the robot on its starting point, we do not know its exact position. We will now take a look at the straight line experiment. Lets assume we can place the robot on its starting point with an accuracy of 1 [mm] in the  $x$ - and  $y$ -direction, which is quite reasonable. This results in a starting orientation error of:

$$\theta_{er_0} = \arctan\left(\frac{2 \cdot d}{L}\right) = \arctan\left(\frac{2}{210}\right) = 0.55^\circ \quad [4.4]$$

with  $L$  the length of the robot and  $d$  our placing error. We also assume we can measure the end position of the robot with an accuracy of 1 [mm] in the  $x$ - and  $y$ -direction and with an accuracy of  $1^\circ$  in the angular direction. This results in the following measurement errors on the end position of the robot:

$$\begin{aligned} dx &= x_e - \cos(\theta_e + \theta_{er_0}) \cdot L + x_{er_0} + x_{er_{end}} = 3[mm] \\ dy &= y_e - \sin(\theta_e + \theta_{er_0}) \cdot L - y_{er_0} + y_{er_{end}} = 12[mm] \\ d\vartheta &= \theta_{er_0} + \theta_{er_{end}} = 1.6^\circ \end{aligned} \quad [4.5]$$

For the rotation experiments these errors can be estimated the same way.

The straight line and rotation experiments were performed and the following results were obtained:

$$\begin{aligned} r_1 &= 41 \pm 4[mm], \quad r_2 = 43 \pm 4[mm], \\ B_1 &= 77 \pm 8[mm], \quad B_2 = 80 \pm 8[mm] \end{aligned}$$

The uncertainty is about 10% after these experiments, which is too much. The values lay quite close to those measured by hand. A second experiment will be performed to fine-tune the parameters.

#### 4.2.2 Second calibration experiment

The second calibration experiment is proposed in [4]. In this experiment the two above mentioned experiments are combined. A square-formed trajectory is applied to the robot (figure 4.3).

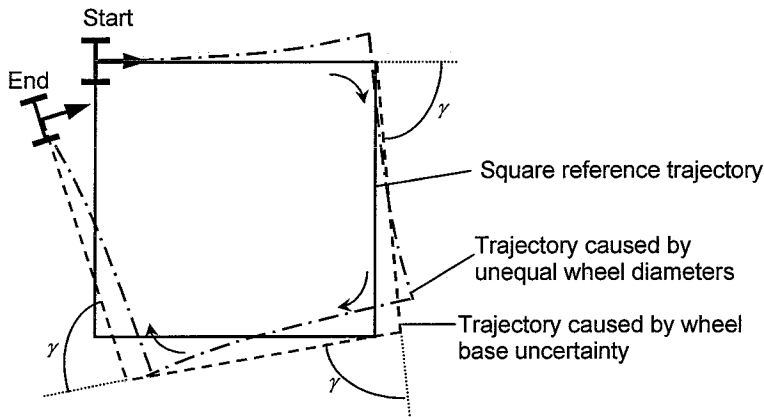


Figure 4.3: The square path experiment

Typically two different types of errors will occur now, which can be recognized easily. The first error is a result of unequal wheel diameters and appears on the straight parts of the trajectory. The robot will not follow the straight line, but it will bend off as described in the straight line experiment in the previous section. The second error is a result of uncertainty about the wheel-base and appears in the corners of the trajectory. The robot will not make a 90°-turn, but it will make a larger or smaller turn. The experiment will be performed in both directions.

The advantage of this experiment is that the calibration can be done instinctively by trial-and-error. By adjusting the parameters and applying the square trajectory to the robot, the effect of the adjustments becomes clear. Because both the straight line and the rotation are applied four times, differences in the end position become quite obvious.

This trial-and-error method results in the following values for the wheel radii and wheel bases:

$$\begin{aligned} r_1 &= 40[mm], & r_2 &= 42[mm], \\ B_1 &= 77[mm], & B_2 &= 78[mm] \end{aligned}$$

Although these results are different from the results obtained in the previous section, they lay within the estimated uncertainty domain. The given values will not be the exact values of the wheel radii and wheel bases, but using these the best performance is obtained. The parameter values found in this subsection will be used for all realtime experiments performed with only dead-reckoning reconstruction. As soon as a proper absolute measurement system is available, this calibration should be performed again to obtain even more accurate parameter values.

### 4.3 Realtime experiments

Using the dead-reckoning reconstruction method, some realtime experiments can be performed. The purpose of these experiments will not be the tuning of the applied controllers, but more gaining the insight in the way they work. Since our calibration is not perfect and non-systematic errors like wheel-slippage cannot be corrected, it is hard to give a judgement about the performance of the controller and as a result tuning is not useful. More interesting is the question if the controllers will still work in the realtime environment with constraints on the velocity and the acceleration. Because we want to prevent the robot from slipping, we need to apply this maximum acceleration. This is done by adding a special scheme, which works the same way a saturation function works. A proper value for this maximum is obtained by trial-and-error experiments and set to 1 [m/s<sup>2</sup>]. The maximum angular acceleration is set to 5 [rad/s<sup>2</sup>]. The Simulink scheme is in fact the same as for the simulations (appendix C1 and C2) with two exceptions: An extra subsystem was added in the System-block to implement the acceleration constraints. Secondly, dSpace analogue and digital output blocks were added to send signals to the robot. The reconstruction is still performed by an integration scheme, which uses the applied inputs for the robot.

All tracking, time-varying and hybrid controllers discussed in chapter 3 have been implemented and tested successfully. The piecewise continuous controller gave more problems, mostly because of the discontinuity on the y-axis. All controllers show more or less the same behaviour realtime as they did in the simulations. The acceleration constraint however makes the most controllers a little or even a lot slower. As a result of the imperfect calibration, the robot often does not exactly end on the reference point. In other cases the final orientation does not equal the wanted orientation. These effects however are more the result of the lack of an absolute position measurement system than the result of bad functioning controllers. In fact the tested controllers show promising behaviour for realtime experiments using an absolute position measurement system.

## 5 Measurement systems

At this point of the project, we can think of several reasons to implement an absolute position measurement system. First of all, our dead-reckoning reconstruction is not reliable enough. In order to test this dead-reckoning behaviour, we need absolute position information. Using an absolute measurement system makes it possible to increase calibration accuracy. Besides, we might be able to model some kind of errors, like wheel slippage, and add these to the kinematics of the robot. Finally we are able to reconstruct the state using absolute position information. In this chapter a survey through several types of measurement systems is performed. Although a lot of research has been done on the field of beacon navigation and GPS-like systems, the solutions for small-scale applications like our mobile robot are limited. Five systems will be discussed, ending up with the comparison and choice in section 5.6.

### 5.1 The pantograph system

To find both position and orientation of the robot, a simple construction can be built, containing two arms and three rotational encoders. We will call this system the pantograph system. A sketch of what this could look like is given in figure 5.1.

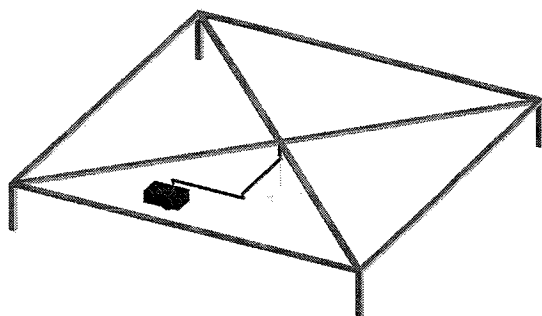


Figure 5.1.a: The pantograph measurement system

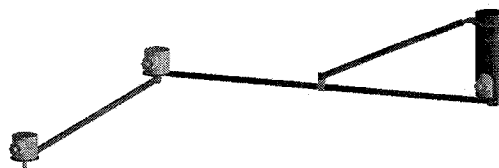


Figure 5.1.b: The encoder arm

When constructing the arm the following parameters should be considered:

- The lengths of the arms.
  - The reachable area of the robot depends on the lengths of the arms. Longer arms enable a larger working area. If the entire workspace is to be covered, both arms should have the same length. Otherwise the robot cannot reach a circular area (with a radius equal to the difference between the arm lengths) in the middle of the workspace.
  - The accuracy of the system depends on the arm lengths. The maximum position error of the robot can be calculated by using the encoder resolutions and the arm lengths as shown in appendix D. As seen in figure 5.2a longer arms result in larger position errors. In this figure, the two arm lengths are kept the same and so are the two encoder resolutions.
- The encoder type.

In order to obtain an absolute position measurement, ordinary rotational encoders are not sufficient, as they only supply relative rotations. As a result three absolute encoders should be used. A disadvantage of these absolute encoders is that they are larger, heavier

and more expensive than the standard rotational encoders are. Several producers and distributors of encoders can be found on the Internet [A, B, C, D].

- The encoder resolution.

Together with the arm lengths, the encoder resolutions set the accuracy of the measurement system. The encoder attached to the frame has the largest influence on the accuracy, the encoder attached to the robot the least. The relation between the number of pulses per revolution of the first two encoders and the maximum position error of the system is given in figure 5.2.b.

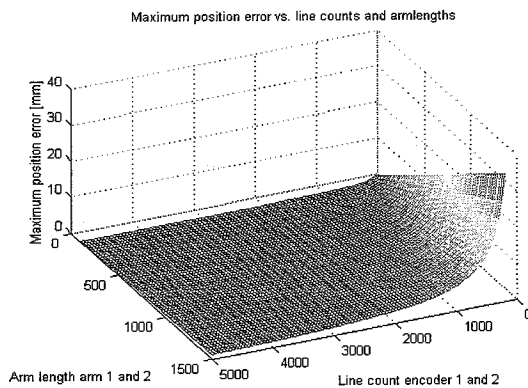


Figure 5.2.a: Arm lengths vs. position error

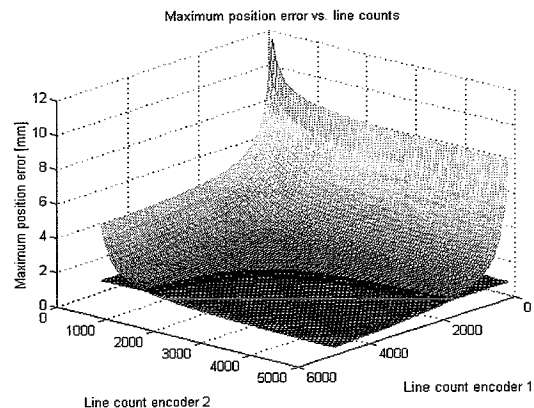


Figure 5.2.b: Encoder resolutions vs. position error

Some problems will be encountered while constructing the pantograph system. The first problem is how to lead the power supply cable from the robot through or along the arm. When wrongly constructed, the cable will get stuck if the robot makes too much rotations around one of the turning points (encoders) of the arm. This is a very tricky problem, since all cables should be led through the arms, which should be constructed hollow, but also through the encoders. Otherwise the cables will wind up around the encoders. Even when this could be achieved, the cable itself would wind up when too many rotations in one direction are made.

The second problem is how to protect the robot from driving out of its workspace. Mechanical or software solutions should be considered to prevent the robot from damaging itself and the measurement system.

A third problem is that a solution should be found for the situation where the robot drives exactly over the centre of the workspace. On this spot the robot might damage itself and the measurement system if no precautions are made. In this case a mechanical solution should be found.

Finally, the construction should be as light as possible to avoid that the measurement system influences the robot behaviour. On the other hand the arms should be stiff and they should not bend down. Deformation of the measurement system reduces the accuracy.

Although the above-mentioned construction problems make the pantograph measurement system more difficult to realize, it has some serious advantages. The needed components are relatively cheap and most of them are available at the TU/e. Secondly, the state of the robot is obtained easily from the rotations of the encoders by using goniometry. Finally, choosing different arm lengths and encoder resolutions can easily change the accuracy and working area of the system.

## 5.2 The CAD/CAM-board

CAD/CAM-boards are used by engineers for creating 2D or 3D product drawings. Several types of transducers are available, which enable the engineer to draw on the board and import this drawing into the computer [J, K]. By positioning the board horizontally and attaching the

transducer to our mobile robot, its position coordinates can be imported into the computer. The board provides only the position of the robot, since only one transducer at the same time can be used. To obtain the orientation of the robot as well, another measurement system, like for example an electronic compass, is needed.

One advantage of the CAD/CAM system is its accuracy, which is approximately 0.05 [mm]. Secondly, the sample frequency, 200 coordinate pairs per second, is very acceptable. However, there are several serious drawbacks in using the board. In first place, it is a very expensive system, which will cost at least 3000 Euro. This amount will even increase when considering the second system to be bought for obtaining the orientation of the robot.

Secondly, the working area of the robot is limited to the dimensions of the board. In the best case this is about 1.2 x 1.5 [m]. Besides, there are no possibilities to expand the working area.

### **5.3 The electronic compass**

An electronic compass works just like an ordinary compass, but its output is digital instead of analogue. Most electronic compasses output both heading and tilting information. Conventional and cheap electronic compasses only work properly in an outdoor environment. Ferrous metals often magnetise over time, misdirecting magnetic compass readings. Soft iron objects can misdirect or magnify existing magnetic fields, making indoor calibration of the compass extremely difficult. More advanced systems use correction algorithms to counter the effects of both soft and hard iron objects and these maintain a high degree of accuracy, even in demanding environments [L, M].

The accuracy of the electronic compass is dependent on the tilting. When held level, heading accuracies up to 1° can be reached, when tilted up to 2°. Tilting accuracy reaches 0.4°. Cheap electronic compasses cost about 200 Euro, more advanced compasses cost about 800 Euro. Since the electronic compass does not provide the position of the mobile robot, it is not possible to use it without any other measurement system. However, it might be a useful addition to other systems, like the CAD/CAM-board system.

### **5.4 Active beacon navigation**

Active beacon navigation systems are the most common navigation aids on ships and airplanes. Perhaps the best-known active beacon navigation systems are GPS-like systems. Active beacons can be detected reliably and provide accurate information with minimal processing. The drawback of this kind of systems is often that they are very expensive. A lot of information on these type of systems can be found in [4].

Two different types of active beacon systems can be recognized: trilateration systems and triangulation systems. In the first case the distance to three beacons is measured. In the second case the angles between the signals of three beacons are measured.

In the sense of this project beacon navigation systems using infrared or radio signals or lasers are far too expensive. Ultrasonic systems however offer a medium- to high accuracy, low-cost solution for the position measurement problem [E, F]. A drawback is the relative small working area as a result of the short range of ultrasound. Two general implementations exist. The first includes a single transducer on the robot and multiple fixed-location receivers; the second includes a single receiver listening on the robot, with multiple fixed-location transmitters serving as beacons. The first case is preferable when only one robot is used; the second is preferred when more robots are used at the same time. Complete systems exist, but these are still quite expensive. The transmitter-sensor pairs itself can be found starting from 20 Euro per pair. The main drawback in this case is that the entire measurement system should still be designed, including

hardware connection, software development and system calibration, which is a time-consuming activity. However ultrasonic beacon systems are promising for our wheeled mobile robot.

## **5.5 The whiteboard system**

In daily practise the whiteboard system is used during meetings to import whiteboard notes immediately into the computer, to send them to a printer or to export them over the Internet. The whiteboard system consists of a set of sleeves for whiteboard markers, which contain ultrasonic/infrared transducers, and two receivers, which are attached to the whiteboard. A pressure switch in the back of the sleeve activates the transducer, when pushing a marker inside a sleeve on the whiteboard. The sensors measure the elapsed time between the incoming ultrasonic and infrared signal and by using goniometry the coordinates of the sleeve are obtained. The system can be compared with the ultrasonic beacon systems discussed in the previous section, however there are some differences. The whiteboard system has only two receivers and it measures only two distances instead of three. Normally two possible positions result from two known distances, but since the receivers are placed on the edge of the board, the second point lies 'behind' the receivers and only one point is feasible. To correctly obtain this point, the assumption that the transmitter is always on the same height from the board should be made. Several producers of interactive whiteboards can be found on the Internet [G, H, I]. A complete system costs about 650 Euro. The accuracy of the system is said to be 1 [mm]. The maximum working area of the system is about 2.4 x 1.5 [m]. The sample frequency is 70 [Hz]. Advantages of this measurement system are its portability and simplicity. Besides, the system can provide the orientation of the robot, but it requires extra work. The system can read the position of only one sleeve at the time, so two sleeve transducers should be activated after each other to obtain the orientation.

## **5.6 Measurement system comparison and choice**

Comparing the measurement systems in the previous subsections results in a choice for the whiteboard system. The main drawbacks for the pantograph system are the construction difficulties and the fact that the robot is attached to the measurement system. This reduces the working area and might influence the robot behaviour.

The CAD/CAM drawing board is too expensive and its working area is limited. Moreover, a second measurement system, like an electronic compass needs to be used to obtain the orientation.

Beacon navigation systems are quite promising, but most of them are far too expensive to use for this project. They are often used for larger robots, working in larger areas. Beacon navigation using ultrasonic sensors is a promising option, but it requires a lot of work before results can be obtained, since transmitters and receivers should be tuned and software and hardware needs to be developed.

The whiteboard system is in fact a simple ultrasonic/infrared beacon navigation system, which already provides coordinate information. A second advantage of the whiteboard is that it is relatively cheap. Besides, it is very portable, which makes it ideal for presentations and demonstrations in combination with a notebook. Finally, both position and orientation can be measured by using alternating transmitters. Of course, still a lot of adaptations should be made before the system works properly for our application.

## 6 The eBeam whiteboard system

Among other products the eBeam whiteboard system was the best choice. It has a good price, it is available from distributors in the Netherlands resulting in a short delivery time and the employees of the company were very willing to give a helping hand.

A standard eBeam-set consists of a set of four sleeves for markers, one eraser, two receivers, the connection cables and a CD-ROM containing software, a software development kit and instructions. This set is shown in figure 6.1.



Figure 6.1: The eBeam whiteboard system components

In section 6.1 the working of the eBeam system is explained. Section 6.2 discussed the system performance. Section 6.3 describes the adaptations to the system, needed to implement the measurement system. Finally, section 6.4 gives some comments on realtime experiments using the whiteboard system.

### 6.1 How does eBeam work?

The eBeam sleeves each contain four infrared transmitters and four ultrasonic transmitters. These are connected to a small electric circuitry, which is mounted in the cylindrical part of the sleeve. The upper end of the sleeve can be opened to insert two 3V-batteries, which supply power for the circuitry and the transmitters. Also in the back of the sleeve a pressure switch is present, which activates the transmitters when a marker is pushed on it.

The eraser contains the same elements as the markers and works the same way. The difference is that it has a large flat brush, with which notes can be removed from the whiteboard.

The receivers (or pods) can be fixed on the whiteboard using suction cups. The left pod contains a infrared and a ultrasonic receiver. It is connected to the right pod. This pod contains a infrared and a ultrasonic receiver as well, but it also has a ultrasonic transmitter. By sending a signal with this transmitter and receiving it with the other pod the width of the board can be measured, which is needed for proper absolute position measurements. The right pod is connected to the computer with a 9-pin serial port. Power is supplied by an adaptor, which is connected to the net power supply.

Together with the eBeam system a USB to serial 9-pin converter was bought. This device, the PortGear PGSD9 from Xircom, is useful to connect the eBeam system to the computer. The advantage of the USB-connection is that it is faster. Together with the converter software and documentation was supplied.

The supplied CD-ROM contains software to install the eBeam system. A program, which makes the whiteboard notes visible on the screen and exportable to other devices or the Internet, is present. Documentation and instructions are present, as well as a software development kit (SDK). This SDK, which is written in C, contains information on messages the receivers send to the PC, the way to handle this messages and how to pick the coordinates and the color of the sleeve from them. This SDK is essential for writing an application, which imports the coordinates from the eBeam system and exports them to Matlab/Simulink.

## 6.2 eBeam performance

Before we are going to use the eBeam system we want to know how good it works. Of interest are the working area, the accuracy, the repeatability, the sample rate and the effect of certain disturbances on the system.

The working area of the eBeam system is given in the documentation supplied by the CD-ROM. The maximum active area is 2.4x1.5 [m], the minimum is 0.6x0.6 [m]. The dimensions of our whiteboard are 1.5x1.0 [m], so no problem should be expected.

A first indication on the accuracy of the system can be obtained by measuring the width of the board manually and then measure the width of the board using eBeam. Before we are capable of doing so, we need to write an software application, which will be described in section 5.4.3. The width of the board is 1.5 [m] as mentioned before, which is divided in 3000 lines by the eBeam system, resulting in a line every 0.50 [mm]. Similarly, the height of the whiteboard is 1.0 [m], which is divided in 1800 lines, resulting in a line every 0.56 [mm]. Concluding, the accuracy is 0.25 [mm] in the  $x$ -direction (width) and 0.28 [mm] in the  $y$ -direction (height).

An experiment was performed to test the accuracy of the system. A sleeve was placed on an arbitrary spot on the board and kept on that position for a while. The coordinates were imported into Matlab/Simulink. Now it showed that over time not one constant value was exported by eBeam, but the output jumps between two or three successive values. Two different values can be easily explained by the fact that the marker could be placed between two lines. Three different values however indicate some uncertainties in the measurement. The experiment was repeated on several spots on the board covering the working area, resulting in the same differences as in the first experiment. This uncertainty is the same in the  $x$ - and  $y$ -direction. This results in a 0.75 [mm] accuracy of 0.75 [mm] in the  $x$ -direction and 0.84 [mm] in the  $y$ -direction, which is quite acceptable.

In order to test the repeatability of the system a simple experiment was performed. A sleeve was placed on an arbitrary spot on the board. After this it was taken off, the software application was started again and the sleeve was placed on the same spot again. The measured coordinates were the second time exactly the same as the first time, resulting in a repeatability of the system equal to the accuracy.

The sample rate of the system was given by an eBeam engineer and said to be 70 coordinate pairs per second. Simple experiments using a stopwatch and a counter, which value increased every time a message was received from eBeam, confirmed this value.

Also of interest is the switching behaviour. The sleeves will transmit for 6 samples after the switch is disconnected, which means the sleeve will stay active for 0.1 second. When the case of switching between two sleeves, in order to obtain the orientation of the robot, is concerned, the sample rate is reduced to about 10 coordinate pairs per second. A probable solution for this is to



activate the second sleeve only once in a while and using a dead-reckoning method for updating the orientation of the robot between these moments.

An important effect, which could affect the measurements, is the misalignment of the pods. Experiments were performed in order to test this effect. A grid was drawn on the whiteboard, with each line 200 [mm] from each other. Near the pods an extra line was placed in  $x$ -direction on 100 [mm] from the last one, in order to measure effects on the edge of the board. A sleeve was activated on each grid point and these points were plotted in Matlab/Simulink. Five experiments were done, in which both pods were placed under the same angles of  $-40^\circ$ ,  $-20^\circ$ ,  $0^\circ$ ,  $+20^\circ$  and  $+40^\circ$ . Here  $-40^\circ$  means the pods were rotated outwards over this angle. The results are shown in figure 6.2.

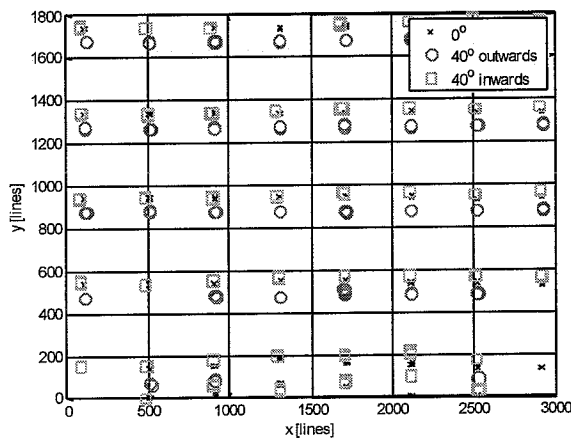


Figure 6.2: Results for a misalignment experiment for the pods

Two effects appear from figure 6.2. First, when the pods are placed under certain angle, some parts of the board cannot be measured any more. In the case of turning the pods outwards, this area lies between the pods, in the case of turning them inside, this area lies at the sides of the board. A second effect is the movement of the points along the  $y$ -axis. This can be explained by the fact that the infrared and the ultrasound receivers are no longer on the same line when the pods are rotated. If for example the distance between the point and the infrared sensor increases and the distance between the point ultrasonic sensor decreases as a result of the turning of the pods, the measured time difference between those two signals will be smaller, resulting in a smaller distance. Since in this experiments the effect is equal for both pods, only movement along the  $y$ -axis results. Experiments in which only one of the pods was rotated also show  $x$ -movement, which confirms the above given motivation.

If the pods are placed properly, or just a few degrees misaligned, the above mentioned effects lie within the accuracy of the system and can be neglected. More important however, the pods should not be moved or turned during or in between experiments, since then the measurement results cannot be compared.

A second issue is to find the effect of activating the transmitters on a certain height above the board. When the transmitters are placed on the robot, they will not be on the same height as the receivers are on. The first thing to do is to find out if signals will be received from higher placed transmitters. It shows that the signal is received well everywhere in the height range needed for our robot. Correcting the coordinates however does not only involve goniometry but also some side-effects, which seem difficult to explain. Combining this with the higher risk of signal blocking by the robot itself when the pods are placed on the board while the transmitters are

placed higher, the best solution is to simply place the pods on the same altitude above the board as the transmitters are on.

A large advantage of the eBeam system is the fact that it provides information on the color of the sleeve that is writing on the board. A special code is sent with the coordinates, which makes distinguishing between the sleeves possible. By using this property, the orientation of the robot can be easily obtained by placing two transmitters on the robot and making them send signals one after the other as proposed above.

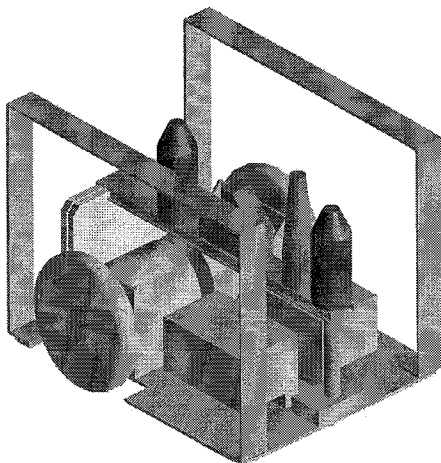
### 6.3 Adapting the system for eBeam

Before the eBeam system can be used as measurement system a lot of adaptations should be made to the system. The sleeves should be adapted and mounted on the robot, dSpace should be replaced by TUEDACs and software should be written. In the next subsections these adaptations are discussed.

#### 6.3.1 Adapting the sleeves and robot

The first step in adapting the sleeves is taking off all parts we cannot use. In fact we end up with only the head of the sleeve, containing the transmitters, the piece of electric circuitry and the power supply/pressure switch connection. These parts are all quite small and this makes them very useful for mounting on the robot. In order to protect the circuitry a hollow cylinder is glued on the head. The backside of this cylinder is closed by gluing a metal piece with a thread end to it, which makes the construction on the robot easy. The power will be supplied externally and herefore the wires are soldered to the power supply connections. The pressure switch will be replaced by a relais, which can be switched from Matlab/Simulink. This results in two extra wires leaving the new transmitter module. The connection scheme of the transmitters is shown in appendix E2.

A 3D-drawing of the adapted mobile robot is given in figure 6.3.



*Figure 6.3: 3D-drawing of the adapted wheeled mobile robot*

In order to mount the transmitters on the robot, some adaptations should be made to it. Because no large obstacles should come between the transmitters and receivers, the transmitter modules should be mounted above the existing parts. To achieve this a Weidmuller mounting rail is

mounted on the robot by two spacer sleeves. This rail is placed over the length axis of the robot on a 70 [mm] height. The advantage of using this rail is that the voltage-to-frequency converter cards can be mounted on the robot as well. The first spacer sleeve will be placed on the axis between the wheel centres and the first transmitter module will be attached to it. This guarantees that this transmitter is on, or at least very close to, the control point of the robot. Tests are needed to exactly find the location of this module. The second transmitter module is mounted as far in the front as possible. The voltage-to-frequency converter cards are mounted at the rear end of the rail. Finally the power supply and signal connectors are mounted on the rail.

At the sides of the robot two brackets are mounted which are useful for several reasons. In the first place these brackets protect the transmitters and the rest of the robot hardware. Secondly, new additional devices can be mounted on these brackets. The height of the brackets is chosen to make sure no other components will block the transmitter signal. When mounting new components under the brackets this signal blocking should be considered seriously. The used brackets are that small that they do not influence the signal sent to the receivers.

A last adaptation should be made to this part of the system. Because the transmitters are now placed about 140 [mm] from the ground, the eBeam pods should be placed on this height as well. This is done temporarily by placing the pods on a pile of steel blocks. In the future a final solution for the height correction should be found.

### 6.3.2 Switching from dSpace to TUEdACS

Because the eBeam system provides Windows messages to the PC, the TUEdACS system is more appropriate than the dSpace system. The processing of these messages needs to be done under Windows. Now keeping the control of the robot under Windows too, is easier and more efficient than processing that part with dSpace.

Together with the addition of the transmitters and the wish to reduce the number of used power supplies, this results in a new hardware configuration. A schematic overview of this configuration is given in figure 6.4. The complete connection scheme is given in the appendices B1, B2 and B3.

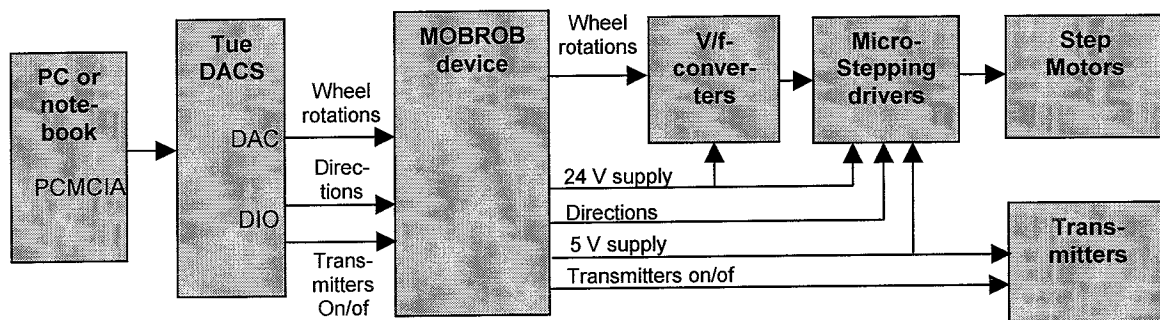


Figure 6.4: Schematic overview of the hardware configuration using TUEdACS

The most important change compared to the system used with dSpace (figure 2.4) is the addition of the MOBROB device, specially designed for this application. The following functions are performed by the MOBROB-block:

- The TUEdACS unit supplies the wheel rotations from an analogue output port, but the output voltage ranges from  $-2.5$  to  $+2.5$  [V]. The voltage-to-frequency converter cards need a 0 to  $+10$  [V] input voltage. The MOBROB device converts the TUEdACS output to the voltage-to-frequency cards input using op-amp's.

- The MOBROB unit supplies both the 24 [V] and 5 [V] power needed for the V/f-converters and step motors and for the microstepping drivers and transmitters respectively.
- The MOBROB unit contains two relays for switching the transmitters on and off. It converts the incoming digital 0 or 1 into an actual current.
- All MOBROB output wires are put together into one cable, so no additional cables or wires have to be connected to the robot.

The TUEdACS now supplies the wheel rotations from two DAC-ports. From the DIO-port the direction signals and the transmitter on/of signals are supplied. After passing the MOBROB unit, the wires will all be connected as done before.

### 6.3.3 Software development

In order to get the coordinates from the Windows-messages sent by the eBeam system, a software application should be written. It has no use to discuss the developed software in detail here, but the main idea behind it will be discussed.

The software has been written in Microsoft Visual C++, the same language the supplied software development kit is written in. This SDK consists of a list of functions, messages and structures and some documentation about them. Also implementation examples are provided. The functions can be used to ask a special action from the eBeam system. This includes searching for and connecting to communication ports, setting or measuring the board width and height and setting the pod locations. The messages contain information about the success of the called functions. The output may be an error code or the asked information about coordinates, connection ports or board width and height. The coordinates are received in a structure format together with the color of the sleeve. Also all error codes and color codes are each placed in structure formats.

The first step was to connect the eBeam system and then try to make contact with it. Once a communication port is connected properly, messages will be received from the eBeam system. The next step is to extract the needed information from those messages and send them to a simple Windows window. All these programs were application programs with .exe extension. The next thing to do now is to write a dynamic link library (.dll) file, which contains the same functions as the previous designed applications. To test this dll-file a small application file (.exe) can be written. The last step is to write a S-function, which can be used in Matlab/Simulink and has the same functions as the small test-application. This is done using the template file 'sfuntmpl.c'. The S-function file calls the earlier designed dll-file and receives the coordinates and the sleeve color from it. This S-function can be used in Matlab/Simulink. The C++-code for the software development kit, the dll-file and the S-function are given in appendix E1, E2 and E3 respectively.

## 6.4 Realtime experiments using the eBeam system

The final step in this project is to test the robot using the absolute position measurement system. The system calibration needs to be improved and algorithms need to be designed to combine the dead-reckoning with the absolute position measurement and to obtain the orientation. This will be quite a lot of work and will be the first thing to do for future researchers. Without these algorithms however, also some interesting experiments can be performed.

We can, for example, perform the same realtime experiments as we did with only the dead-reckoning reconstruction. We can now measure the position of the robot quantitatively and qualitatively. The first option is the easiest one. By just starting up the original eBeam software

and making the robot move, the driven trajectory can be followed on the screen. This way, information on the shape of the trajectory is provided.

By using the especially designed new software, we can obtain the robot coordinates in Matlab/Simulink and process them. Now qualitative performance validation is possible too. Using the measurement system this way, we can improve the calibration of the robot and gain more information about wheel slippage and other non-systematic errors. These are topics for future researchers.

In this project only some experiments to test the performance of several controllers are performed. Those were done using a notebook attached to the TUE DACS QAD. During the first experiments, the eBeam receivers were placed on the board without adjusting their height. The tracking controller described in section 3.2 and the non-smooth time-varying point stabilization controller described in section 3.4.1 were implemented and the results were plotted by the standard eBeam software. The results of these experiments are given in figure 6.5.

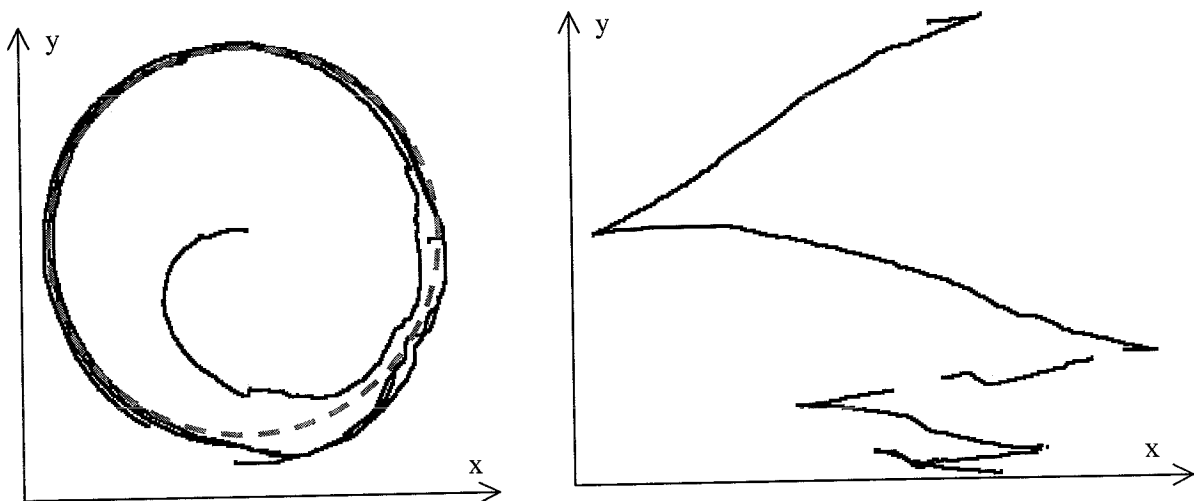


Figure 6.5: Results of first realtime experiments using the TUE DACS QAD. Left: Tracking controller. Right: Point stabilization controller.

In both cases some two effects appear: There are disturbances present on the trajectories and there are gaps in the data. Both seem to be a result of some kind of signal blocking, for example caused by the connectors on the robot, the brackets or the cable itself. In both cases however, other effects cause these errors. The disturbances, which can be seen best in the tracking control experiment, are a result of the height difference between the transmitters and receivers. When placing the pods on about the same height as the transmitters, this problem is reduced a lot, as can be seen from simulation results later on in this paragraph.

The gaps in the data are also not the result of signal blocking, but a result of the limited data processing capability of the used notebook. For further research, a PC with a faster processor and using Windows 2000 was prepared for experiments. On this PC the latest PCMCIA-drivers were installed and the same experiments were performed as on the notebook. The eBeam hardware was connected to this PC too. It appeared that the gaps in the data were absent now, as can be seen from figure 6.6. To verify if the gaps are really results of slow data processing and data-loss, a final experiment was performed. Now the robot was controlled by the PC and the eBeam data was received by the notebook. This way, the notebook has less realtime functions and as a result more data processing time. In fact, there were less gaps in the data during this experiment, but

there were still some present. Concluding, the computer processing speed should be high enough to both control the robot and gain the eBeam coordinates to obtain proper measurement results.

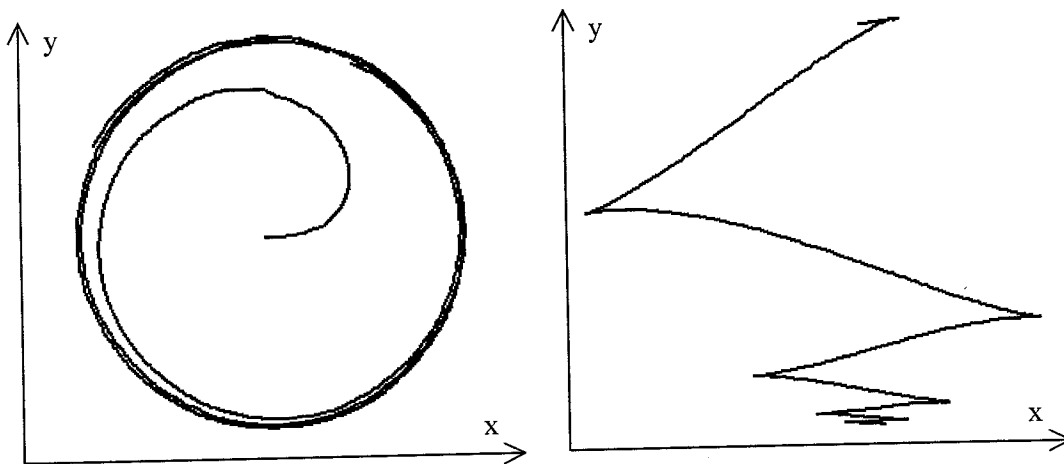


Figure 6.6: Experimental results using a PC and high placed pods. Left: Tracking controller. Right: Point stabilization controller.

From figure 6.6 the following can be concluded: The measurement system provides good position information. The connectors on the robot do not or merely have influence on the measured signal, and the same counts for the brackets and the cable on the robot.

Finally, it can be seen from the tracking control experiment that the robot converges to the reference circle and stays on it. This gives an indication that the calibration experiments have been successful and the used parameters for the wheel diameters and wheelbases are properly chosen.

The robot is now ready for all kind of future realtime experiments...

## 7 Conclusion and recommendations

The original wheeled mobile robot system has been adapted for better performance and is now ready for the realtime implementation of non-linear controllers and other applications. Two voltage-to-frequency converter cards were added to the system successfully, making faster and smoother movement of the robot possible. A measurement system survey has been performed resulting in the choice for the eBeam whiteboard system. This proves to be a proper solution for absolute position measurements. The system uses two infrared/ultrasonic transmitters and receivers and an accuracy of less than 1 [mm] is obtained. The orientation of the mobile robot can be obtained when two transmitter units are used alternating. Each transmitter can be recognized independently. The first available dSpace unit is replaced by a TUE DACS unit and an extra unit, the so-called MOBROB unit is added for power supplies and signal transformations. Software was designed to import the coordinates and information about which transmitter is used into Matlab/Simulink. The robot itself has been adapted for the montage of the transmitters and voltage-to-frequency cards.

Several non-linear controllers have been tested in simulations and realtime, using dead-reckoning methods. Simulation results of both tracking controllers and point-stabilization controllers are presented. The dead-reckoning reconstructions give a first impression on realtime behaviour of the controllers and the first realtime experiments using the absolute measurement system have been performed. Those were only tests to validate the dead-reckoning performance however and improved calibration and realtime absolute position reconstruction still need to be done. In order to be able to use the dead-reckoning reconstruction method the mobile robot was calibrated first. Finally some simulations have been performed using the mobile robot as a tractor pulling one or two trailers.

A lot of recommendations for the future can be made, because this project is just at the start of several other projects.

- The mobile robot needs to be calibrated using the absolute measurement system so systematic errors like unequal wheel diameters and wrongly estimated wheel bases can be corrected properly.
- An algorithm should be designed to combine position, orientation and dead-reckoning measurements. This includes a strategy on how often the orientation is measured and how it is reconstructed in between the sample moments.
- The simulated controllers should be tested in the realtime environment in order to obtain not only their performance but also the robot performance.
- The design of two simple trailers makes the implementation of more complicated controllers possible, which is certainly an interesting field of research.





## Literature and internet-sites

### Literature

- [1] Emad Al-Regib, Ilya Kolmanovsky and N. Harris McClamroch: "Stabilization of wheeled vehicles by hybrid non-linear time-varying feedback laws", *Proceedings on the 1996 IEEE International Conference on Control Applications*, 1996.
- [2] A. Astolfi: "Discontinuous control of non-holonomic systems", *System and control letters* 27, 1996.
- [3] A.M. van Beek: "3D measurement systems for robot manipulators", *Project report for the Eindhoven University of Technology*, 1998.
- [4] J. Borenstein, H.R. Everett and L. Feng: "'Where am I?', Sensors and methods for mobile robot positioning", *Technical report from the University of Michigan*, 1996.
- [5] R.W. Brockett: "Asymptotic stabilization and feedback stabilization", *Differential geometric control theory*, 1983.
- [6] Guy Campion, Georges Bastin and Brigitte D'Andréa-Novel, "Structural properties and classification of kinematic and dynamic models of wheeled mobile robots", *IEEE Transactions on robotics and automation*, vol 12, nr.1, 1996.
- [7] C. Canudas de Wit, H. Khennouf, C. Samson and O.J. Sørдалen: "Non-linear control design for mobile robots", 1992.
- [8] C. Canudas de Wit and O.J. Sørдалen: "Exponential stabilization of mobile robots with non-holonomic constraints", *IEEE Transactions on automatic control*, 1992.
- [9] J.F. van den Eerenbeemt: "Control of a mobile robot", *Project report for the Eindhoven University of Technology*, 2001.
- [10] John-Morten Godhavn and Olav Egeland: "A Lyapunov approach to exponential stabilization of non-holonomic systems in power form", *IEEE Transactions on automatic control*, 1997.
- [11] Zhong-Ping Jiang and Henk Nijmeijer: "Tracking control of mobile robots: A case study in backstepping", *Automatica*, 1997.
- [12] Zhong-Ping Jiang and Henk Nijmeijer: "A recursive technique for tracking control of non-holonomic systems in chained form", *IEEE Transactions on automatic control*, 1999.
- [13] Ilya Kolmanovsky and N. Harris McClamroch: "Developments in non-holonomic control problems", *IEEE Control Systems*, 1995.
- [14] Ilya Kolmanovsky and N. Harris McClamroch: "Hybrid feedback laws for a class of cascade non-linear control systems", *IEEE Transactions on automatic control*, 1996.
- [15] David A. Lizárraga, Pascal Morin and Claude Samson: "Chained form approximation of a driftless system. Application to the exponential stabilization of the general N-trailer system", *International Journal of control*, 2001.
- [16] Jihua Luo and Panagiotis Tsiotras: "Exponentially convergent control laws for non-holonomic systems in power form", *Systems and control letters*, 1998.
- [17] Robert T. M'Closkey and Richard M. Murray: "Exponential stabilization of driftless non-linear control systems using homogeneous feedback", *IEEE Transactions on automatic control*, 1997.
- [18] H. Michalska and F.U. Rehman: "Set point stabilizing control for a mobile robot with trailer", *ICAR '97*, 1997.
- [19] Richard M. Murray and S. Shankar Sastry: "Non-holonomic Motion Planning: Steering using sinusoids", *IEEE Transactions on automatic control*, 1993.

- [20] KyuCheol Park, Hakyoung Chung and Jang Gyu Lee: "Point stabilization of mobile robots via state-space exact feedback linearization", *Robotics and computer integrated manufacturing*, 2000.
- [21] J.B. Pomet, B. Thuilot, G. Bastin, G. Campion: "A hybrid strategy for the feedback stabilization of non-holonomic mobile robots", *IEEE International conference on robotics and automation*, 1992.
- [22] Claude Samson: "Control of chained systems, application to path following and time-varying point-stabilization of mobile robots", *IEEE Transactions on automatic control*, 1995.
- [23] C. Samson and K. Ait-Abderrahim: "Feedback control of a non-holonomic wheeled cart in cartesian space", *IEEE International conference on robotics and automation*, California, 1991.
- [24] Shankar Sastry: *Non-linear Systems: analysis, stability and control*, Berlin: Springer, 1999.
- [25] Jean-Jacques E. Slotine and Weiping Li, *Applied non-linear control*, London: Prentice Hall International, 1991.
- [26] O.J. Sørдалen: "Conversion of the kinematics of a car with  $n$  trailers into a chained form", *IEEE*, 1993.
- [27] O.J. Sørдалen and O. Egeland: "Exponential stabilization of non-holonomic chained systems", *IEEE Transactions on automatic control*, 1995.
- [28] Zhendong Sun, S.S. Ge, Wei Huo, T.H. Lee: "Stabilization of non-holonomic chained systems via non-regular feedback linearization", *Systems and control letters*, 2001.
- [29] D. Tilbury, O. Sørдалen, L. Bushnell and S. Sastry: "A multi-steering trailer system: conversion into chained form using dynamic feedback", *IEEE Transactions on robotics and automation*, 1995.

### Used Internet-sites

- [A] [www.baumerelectric.com](http://www.baumerelectric.com): Homepage of the Baumer Electric company, which produces rotary encoders and ultrasonic transmitters and receivers.
- [B] [www.automationdirect.com](http://www.automationdirect.com): Homepage where (price-)information on Koyo rotary encoders can be found.
- [C] [www.hengstler.de](http://www.hengstler.de): Homepage of the Hengstler company, which produces rotary encoders.
- [D] [www.heidenhain.de](http://www.heidenhain.de): Homepage of the Heidenhain company, which produces rotary encoders.
- [E] [www.ultrasonicsensors.com](http://www.ultrasonicsensors.com): Homepage of the Senix company, which produces ultrasonic sensors.
- [F] [www.mindspring.com/~sholmes/robotics/ultrasnd.htm](http://www.mindspring.com/~sholmes/robotics/ultrasnd.htm): Information and examples for ultrasonic measurement systems.
- [G] [www.e-Beam.com](http://www.e-Beam.com): Homepage of the e-Beam whiteboard system from Electronics for Imaging (EFI).
- [H] [www.mimio.com](http://www.mimio.com): Homepage of the Mimio whiteboard system.
- [I] [www.pegatech.com](http://www.pegatech.com): Homepage of the Pegasus whiteboard system from Pegasus Technologies.
- [J] [www.gtccalcomp.com](http://www.gtccalcomp.com): Homepage for CAD/CAM drawing boards.
- [K] [www.ICN-Solutions.nl](http://www.ICN-Solutions.nl): Homepage for CAD/CAM drawing boards.
- [L] [www.pnicorp.com](http://www.pnicorp.com): Homepage for electronic compasses.
- [M] [www.electronic-compass.com](http://www.electronic-compass.com): Homepage for electronic compasses.

## Appendices

- Appendix A1: Stepper motor connections and specifications
- Appendix A2: Micro stepping driver and 15-pin interface board specifications
- Appendix A3: Micro stepping driver and 19-pin interface board specifications
- Appendix A4: Voltage-to-frequency converter card specifications
  
- Appendix B1: Electro technical scheme for step motors, microstepping drivers and V/f-converter cards
- Appendix B2: Electro technical scheme for MOBROB unit and eBeam transmitters
- Appendix B3: Electro technical scheme for TUE DACS QAD
  
- Appendix C1: Simulink scheme for tracking controllers
- Appendix C2: Simulink scheme for point stabilization controllers
- Appendix C3: Simulation/realtime Matlab initialization file example
- Appendix C4: Matlab-file for a tractor with one trailer point stabilization simulation
- Appendix C5: Matlab-file for a tractor with two trailers point stabilization simulation
  
- Appendix D: Maximum position error calculation for the pantograph system
  
- Appendix E1: eBeam software development kit C++-code
- Appendix E2: C++-code for eBeam.dll
- Appendix E3: C++-code for eBeam S-function



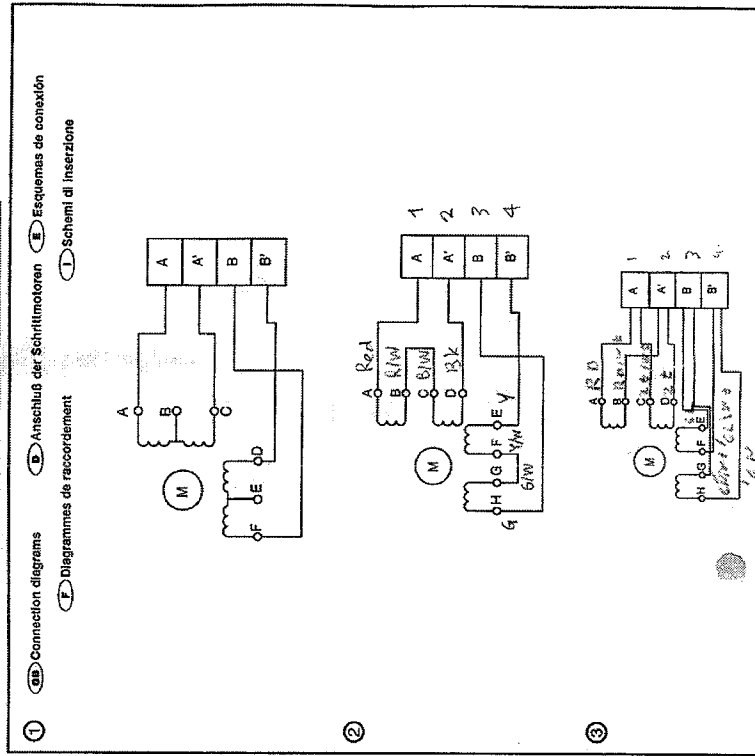
# Appendix A1: Stepper motor connections and specifications (continued)



Instruction Leaflet  
 Bedienungsanleitung  
 Hojas de instrucciones  
 Feuille d'instructions  
 Foglio d'istruzioni

- Hybrid Stepping Motors **(GB)**
- Hybrid-Schrittmotoren **(D)**
- Motores híbridos paso a paso **(E)**
- Moteurs pas à pas hybrides **(F)**
- Motori ibridi passo-passo **(I)**

Figures / Abbildung / Figura



## Appendix A2: Micro stepping driver and 15-pin interface board specifications



**INTELLIGENT MOTION SYSTEMS**

*Excellence in Motion™*

# INT-481

## IM481H INTERFACE BOARD OPERATING INSTRUCTIONS

370 N. MAIN ST., PO BOX 457, MARLBOROUGH, CT 06447  
PH. (860) 295-6102, FAX. (860) 295-6107  
Internet: <http://imshome.com>, E-Mail: [info@imshome.com](mailto:info@imshome.com)

PIN ASSIGNMENT AND DESCRIPTIONS		
PIN #	PIN NAME	DESCRIPTION
1,2	Phase B	Phase B output
3,4	Phase A	Phase A output
5	Enable	Active high motor phase enable input
6	Reset	Active low reset input
7	Opto Supply	+5Vdc external optical isolator power supply
8	Direction	Motor direction input
9	Step Clock	Motor step clock input
10	Fault	Open drain fault output
11	Full Step	Open drain full step output
12	+V	Supply voltage input
13	Ground	Supply voltage ground (return)
14	Current Adj.	Phase current adjustment input
15	Current Red	Phase current reduction input

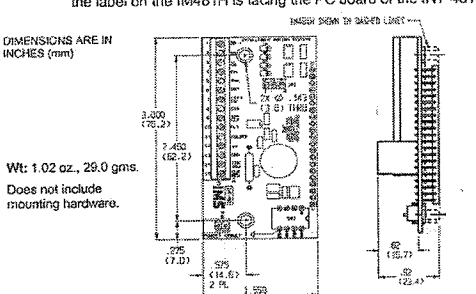
ELECTRICAL SPECIFICATIONS					
PARAMETER		MIN	TYP	MAX	UNITS
Opto Supply	Isolated Inputs	5		40	V
Input Forward Current	Isolated Inputs	8	10	12	mA
Opto Input Forward Voltage	Isolated Inputs	1.5	1.7		V
Reverse Breakdown Voltage	Isolated Inputs	5			V
Signal Output Current	Full Step, Fault			25	mA
Drain-Source Voltage	Full Step, Fault			100	V
Drain-Source Resistance	Full Step, Fault $I_{DS} = 25\text{mA}$		6.5		$\Omega$

Test Parameters:  $T_A = 25^\circ\text{C}$ ,  $+V = 45\text{Vdc}$

### DIMENSIONAL AND MOUNTING INFORMATION

**NOTE:** The IM481H is mounted to the underside of the INT-481 such that the label on the IM481H is facing the PC board of the INT-481.

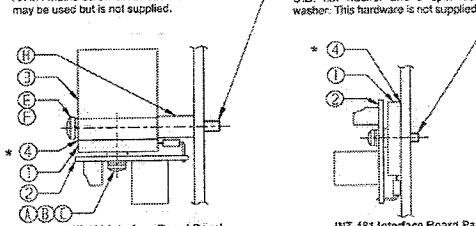
DIMENSIONS ARE IN INCHES (mm)



Wt: 1.02 oz., 29.0 gms.  
Does not include mounting hardware.

An M4 Main Screw substitution for item "E" may be used but is not supplied.

\*\* #6-32 (M3.5) screw with a .250" (6.3) O.D. flat washer and a split lock washer. This hardware is not supplied.



**INT-481 Interface Board Panel Mounted with H-481 Heat Sink**

**INT-481 Interface Board Panel Mounted without H-481 Heat Sink**

\* The isolating thermal pad (TI-461) item "4" is supplied with the Interface Board (INT-481). If the INT-481 is not used it must be ordered separately.

\*\* The hardware items "A" thru "H" are supplied with the Heatsink kit (H-481). If the H-481 is not used no mounting hardware is supplied.

ITEM	DESCRIPTION	QTY.
1	IM481H Microstepping Driver	1
2	INT-481 Interface Board	1
3	H-481 Heat Sink	1
4	TI-481 Isolating Thermal Pad	1

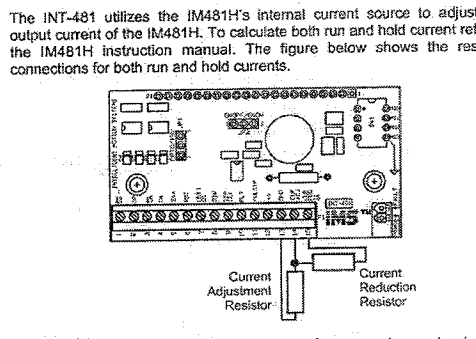
ITEM	DESCRIPTION	QTY.
A	#6-32x.50" Pan Head Screw	2
B	#6 Split Lock Washer	2
C	#6 Flat Washer, 250 O.D., 145 I.D., .030 Thick	2
D	#6-32x1.38" Pan Head Screw	2
E	#6-32x2" Pan Head Screw	2
F	#8 Split Lock Washer	2
G	#8-32 Internally Threaded Broaching Nut	2
H	Spacer, 8710 O.D., .171 I.D., .500" long	2

**NOTE:** Torque specification for # 6-32 INT-481 and IM481H mounting screws:  
**5.0 - 7.0 in-lbs**

**WARNING!** The heat sink mounting surface must be a smooth, flat surface with no burrs, protrusions, cuttings or other foreign objects.

### CURRENT ADJUSTMENT

The INT-481 utilizes the IM481H's internal current source to adjust the output current of the IM481H. To calculate both run and hold current refer to the IM481H instruction manual. The figure below shows the resistor connections for both run and hold currents.



**NOTE:** When connecting both the current reference and current reduction resistors, connections should be made as short as possible to minimize the noise coupled into the driver.

## Appendix A2: Micro stepping driver and 15-pin interface board specifications (continued)

<h3 style="text-align: center;">ISOLATED INPUTS</h3> <p>The following schematic shows the optically isolated inputs to the INT-481 along with the associated circuitry:</p>	<h3 style="text-align: center;">MICROSTEP RESOLUTION SELECTION</h3> <p>The number of microsteps per step is selected by the dip switch (SW1). The following table shows the standard resolution values along with the associated switch settings.</p> <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 10px;"> <thead> <tr> <th>RESOLUTION Microsteps/step</th> <th>STEPS/REV 1.8" motor</th> <th>SWITCH 1 Pin 19</th> <th>SWITCH 2 Pin 18</th> <th>SWITCH 3 Pin 17</th> <th>SWITCH 4 Pin 16</th> </tr> </thead> <tbody> <tr> <td colspan="6"><b>DECIMAL</b></td> </tr> <tr> <td>5</td> <td>1,000</td> <td>ON</td> <td>ON</td> <td>ON</td> <td>OFF</td> </tr> <tr> <td>10</td> <td>2,000</td> <td>OFF</td> <td>ON</td> <td>ON</td> <td>OFF</td> </tr> <tr> <td>25</td> <td>5,000</td> <td>ON</td> <td>OFF</td> <td>ON</td> <td>OFF</td> </tr> <tr> <td>50</td> <td>10,000</td> <td>OFF</td> <td>OFF</td> <td>ON</td> <td>OFF</td> </tr> <tr> <td>125</td> <td>25,000</td> <td>ON</td> <td>ON</td> <td>OFF</td> <td>OFF</td> </tr> <tr> <td>250</td> <td>50,000</td> <td>OFF</td> <td>ON</td> <td>OFF</td> <td>OFF</td> </tr> <tr> <td colspan="6"><b>BINARY</b></td> </tr> <tr> <td>2</td> <td>400</td> <td>ON</td> <td>ON</td> <td>ON</td> <td>ON</td> </tr> <tr> <td>4</td> <td>800</td> <td>OFF</td> <td>ON</td> <td>ON</td> <td>ON</td> </tr> <tr> <td>8</td> <td>1,600</td> <td>ON</td> <td>OFF</td> <td>ON</td> <td>ON</td> </tr> <tr> <td>16</td> <td>3,200</td> <td>OFF</td> <td>OFF</td> <td>ON</td> <td>ON</td> </tr> <tr> <td>32</td> <td>6,400</td> <td>ON</td> <td>ON</td> <td>OFF</td> <td>ON</td> </tr> <tr> <td>64</td> <td>12,800</td> <td>OFF</td> <td>ON</td> <td>OFF</td> <td>ON</td> </tr> <tr> <td>128</td> <td>25,600</td> <td>ON</td> <td>OFF</td> <td>OFF</td> <td>ON</td> </tr> <tr> <td>256</td> <td>51,200</td> <td>OFF</td> <td>OFF</td> <td>OFF</td> <td>ON</td> </tr> </tbody> </table>	RESOLUTION Microsteps/step	STEPS/REV 1.8" motor	SWITCH 1 Pin 19	SWITCH 2 Pin 18	SWITCH 3 Pin 17	SWITCH 4 Pin 16	<b>DECIMAL</b>						5	1,000	ON	ON	ON	OFF	10	2,000	OFF	ON	ON	OFF	25	5,000	ON	OFF	ON	OFF	50	10,000	OFF	OFF	ON	OFF	125	25,000	ON	ON	OFF	OFF	250	50,000	OFF	ON	OFF	OFF	<b>BINARY</b>						2	400	ON	ON	ON	ON	4	800	OFF	ON	ON	ON	8	1,600	ON	OFF	ON	ON	16	3,200	OFF	OFF	ON	ON	32	6,400	ON	ON	OFF	ON	64	12,800	OFF	ON	OFF	ON	128	25,600	ON	OFF	OFF	ON	256	51,200	OFF	OFF	OFF	ON
RESOLUTION Microsteps/step	STEPS/REV 1.8" motor	SWITCH 1 Pin 19	SWITCH 2 Pin 18	SWITCH 3 Pin 17	SWITCH 4 Pin 16																																																																																																		
<b>DECIMAL</b>																																																																																																							
5	1,000	ON	ON	ON	OFF																																																																																																		
10	2,000	OFF	ON	ON	OFF																																																																																																		
25	5,000	ON	OFF	ON	OFF																																																																																																		
50	10,000	OFF	OFF	ON	OFF																																																																																																		
125	25,000	ON	ON	OFF	OFF																																																																																																		
250	50,000	OFF	ON	OFF	OFF																																																																																																		
<b>BINARY</b>																																																																																																							
2	400	ON	ON	ON	ON																																																																																																		
4	800	OFF	ON	ON	ON																																																																																																		
8	1,600	ON	OFF	ON	ON																																																																																																		
16	3,200	OFF	OFF	ON	ON																																																																																																		
32	6,400	ON	ON	OFF	ON																																																																																																		
64	12,800	OFF	ON	OFF	ON																																																																																																		
128	25,600	ON	OFF	OFF	ON																																																																																																		
256	51,200	OFF	OFF	OFF	ON																																																																																																		
<h3 style="text-align: center;">JUMPERS</h3> <p><b>JP1:</b> If the shunt is placed on the "OPTO" side of the jumper the power for the opto isolators must be provided by the user at the P1 connector. If the shunt is placed on the "+5V" side of the jumper then the opto isolators will be powered by the on board supply and electrical isolation between the inputs and the drive power will be eliminated.</p> <p><b>JP2:</b> If the shunt is placed on the "ENON" side of the jumper then the drives outputs will be automatically disabled approximately .5 seconds after the last step clock input. <b>NOTE:</b> In this mode the current reduction resistor <b>MUST NOT</b> be used or it will cause erratic operation of the driver. If the shunt is placed on the "ENOFF" side of the jumper then a current reduction resistor can be used to set the level of current in the motor after the last step clock input.</p>	<h3 style="text-align: center;">ILLEGAL SETTINGS</h3> <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 10px;"> <thead> <tr> <th></th> <th>ON</th> <th>OFF</th> <th>OFF</th> <th>OFF</th> </tr> </thead> <tbody> <tr> <td></td> <td>OFF</td> <td>OFF</td> <td>OFF</td> <td>OFF</td> </tr> </tbody> </table> <p><b>NOTE:</b> In the above table ON is ground and OFF is floating when using the terminal strip inputs to set the microstep resolution.</p>		ON	OFF	OFF	OFF		OFF	OFF	OFF	OFF																																																																																												
	ON	OFF	OFF	OFF																																																																																																			
	OFF	OFF	OFF	OFF																																																																																																			
<h3 style="text-align: center;">LED'S</h3> <p>The green LED is controlled by the on board +5vdc power supply.</p> <p>The red LED is controlled by the fault output of the IM481H. If the red LED is illuminated turn off power and check for a system fault. A fault may be caused by a short or miss wiring of the motor or power supply.</p> <p>A fault condition can only be reset by cycling power or toggling of the reset input on P1 pin 6. In the case of an over temperature fault allow the drive to cool before re-applying power.</p> <p>For additional trouble shooting information refer to the drivers operating instructions.</p>	<h3 style="text-align: center;">RECOMMENDED WIRING</h3> <p>Logic level cables must not run parallel to power cables. Power cables will introduce noise into the logic level cables and make your system unreliable.</p> <p>Logic level cables must be shielded to reduce the chance of EMI induced noise. The shield needs to be grounded at the signal source to AC ground. The other end of the shield must not be tied to anything, but allowed to float. This allows the shield to act as a drain.</p> <p>Motor cabling in excess of 1 foot requires twisted pair shielded cable to reduce the transmission of EMI. The shield must be connected to AC ground at the driver. The other end of the shield must not be tied to anything, but allowed to float. This allows the shield to act as a drain.</p> <p>Power supply leads to the driver need to be twisted. If more than one driver is to be connected to the same power supply, run separate power and ground leads from the supply to each driver.</p> <p>Refer to the IM481H operating instructions for recommended motor and power supply cables.</p>																																																																																																						
<h3 style="text-align: center;">FAULT PROTECTION</h3> <p>The INT-481 adds phase to ground fault protection to the IM481H. If a phase to ground fault is detected the IM481H will latch the signal, set the fault output and illuminate the red fault LED. To clear the fault condition, the IM481H will have to be reset or power will need to be cycled.</p> <p>The INT-481 buffers the IM481H's fault output signal through an open drain N-channel FET. The signal at the terminal strip is inverted and is active low.</p> <p>In the case of an over temperature fault, neither the red LED of the fault output become activated. The IM481H's motor outputs will disable. They will not re-enable until the drive cools to a safe operating level.</p>																																																																																																							
<h3 style="text-align: center;">FULL STEP OUTPUT</h3> <p>The INT-481 buffers the IM481H's full step output through an open drain N-channel FET. The signal available at the terminal strip is inverted and is active low.</p>																																																																																																							

# Appendix A3: Micro stepping driver and 19-pin interface board specifications



**INTELLIGENT MOTION SYSTEMS**

*Excellence in Motion™*

## INT-481

### IM481H INTERFACE BOARD

### OPERATING INSTRUCTIONS

370 N. MAIN ST., PO BOX 457, MARLBOROUGH, CT 06447  
 PH. (860) 295-6102, FAX. (860) 295-6107  
 Internet: <http://imshome.com>, E-Mail: [info@imshome.com](mailto:info@imshome.com)

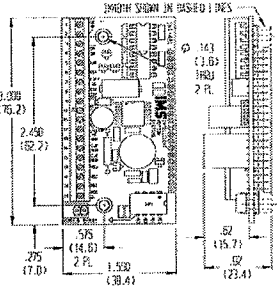
PIN ASSIGNMENT AND DESCRIPTIONS		
PIN #	PIN NAME	DESCRIPTION
1,2	Phase B	Phase B output
3,4	Phase A	Phase A output
5	Enable	Active high motor phase enable input
6	Reset	Active low reset input
7	Opto Supply	+5Vdc external optical isolator power supply
8	Direction	Motor direction input
9	Step Clock	Motor step clock input
10	Fault	Open drain fault output
11	Full Step	Open drain full step output
12	+V	Supply voltage input
13	Ground	Supply voltage ground (return)
14	Current Adj.	Phase current adjustment input
15	Current Red.	Phase current reduction input
16-19	Res. Sel. 0-3	Microstep resolution selection inputs 0-3

ELECTRICAL SPECIFICATIONS					
PARAMETER		MIN	TYP	MAX	UNITS
Input Forward Current	Isolated Inputs		7	15	mA
Input Forward Voltage	Isolated Inputs		1.5	1.7	V
Reverse Breakdown Voltage	Isolated Inputs	5			V
Signal Output Current	Full Step, Fault			25	mA
Drain-Source Voltage	Full Step, Fault			100	V
Drain-Source Resistance	Full Step, Fault $I_{DS} = 25mA$		6.5		$\Omega$
Quiescent Current (Including IM481H)	Inputs/Outputs Floating			45	mA

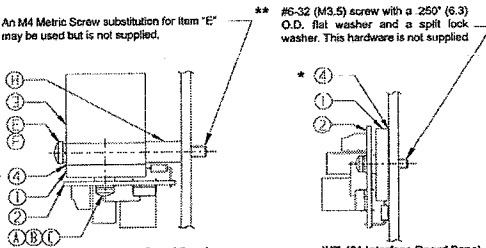
### DIMENSIONAL AND MOUNTING INFORMATION

DIMENSIONS ARE IN INCHES (mm)

**NOTE:** The IM481H is mounted to the underside of the INT-481 such that the label on the IM481H is facing the PC board of the INT-481.



Wt: 1.04 oz., 29.6 gms.  
Does not include mounting hardware.



An M4 Metric Screw substitution for item "E" may be used but is not supplied.

\*\* #6-32 (M3.5) screw with a 250" (6.3) O.D. flat washer and a split lock washer. This hardware is not supplied.

\* The isolating thermal pad ( T1-481 ) item "4" is supplied with the Interface Board ( INT-481 ). If the INT-481 is not used it must be ordered separately.

\*\* The hardware items "A" thru "H" are supplied with the Heatsink kit ( H-481 ). If the H-481 is not used no mounting hardware is supplied.

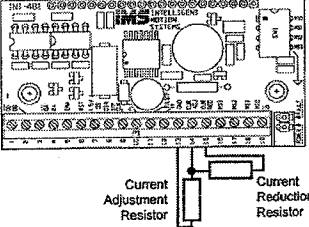
ITEM	DESCRIPTION	QTY.	ITEM	DESCRIPTION	QTY.
1	IM481H Microstepping Driver	1	A	#5-32x5/8" Pan Head Screw	2
2	INT-481 Interface Board	1	B	#6 Split Lock Washer	2
3	H-481 Heat Sink	1	C	#6 Flat Washer, 250 OD, .145 ID, .039 Thick	2
4	T1-481 Isolating Thermal Pad	1	D	#6-32x1 1/2" Pan Head Screw	2
			E	#6-32x1 1/2" Pan Head Screw	2
			F	#5 Split Lock Washer	2
			G	#5-32 Internally Threaded Boreing Nut	2
			H	Spacer, 5/16" OD, .171 ID, .500" long	2

**NOTE:** Torque specification for # 6-32 INT-481 and IM481H mounting screws:  
5.0 - 7.0 in-lbs

**WARNING!** The heat sink mounting surface must be a smooth, flat surface with no burrs, protrusions, cuttings or other foreign objects.

### CURRENT ADJUSTMENT

The INT-481 utilizes the IM481H's internal current source to adjust the output current of the IM481H. To calculate both run and hold current refer to the IM481H instruction manual. The figure below shows the resistor connections for both run and hold currents.



**NOTE:** When connecting both the current reference and current reduction resistors, connections should be made as short as possible to minimize the noise coupled into the driver.



## Appendix A3: Micro stepping driver and 19-pin interface board specifications (continued)

### ISOLATED INPUTS

The following schematic shows the optically isolated inputs to the INT-481 along with the associated circuitry:

The isolated inputs may be powered by a DC voltage other than +5vdc. In doing so, care should be taken **NOT TO EXCEED** the maximum input forward current. To do so, an external resistor should be placed in series with the input pins (6,6,8,9).

The value of the resistor is calculated as follows:

$$\text{SERIES RESISTOR} = ((\text{Opto Supply Voltage} - 1.5) \div .007) - 470$$

EXAMPLE: Opto Supply Voltage = 24vdc:

$$((24 - 1.5) \div .007) - 470 = 2,700 \text{ ohms}$$

### MICROSTEP RESOLUTION SELECTION

The number of microsteps per step is selected by the dip switch (SW1) or by pins 16 through 19 on the terminal strip with the dip switch in the off position.

When using the terminal strip, the resolution selection inputs are pulled up through 10K resistors to +5 VDC. These are non-isolated inputs and care should be taken when connecting the controller and driver grounds. Refer to the IM481H operating instructions for more information.

The following table shows the standard resolution values along with the associated switch and input settings:

RESOLUTION Microsteps/step	STEPS/REV 1.8" motor	SWITCH 1 Pin 19	SWITCH 2 Pin 18	SWITCH 3 Pin 17	SWITCH 4 Pin 16
5	1,000	ON	ON	ON	OFF
10	2,000	OFF	ON	ON	OFF
25	5,000	ON	OFF	ON	OFF
50	10,000	OFF	OFF	ON	OFF
125	25,000	ON	ON	OFF	OFF
250	50,000	OFF	ON	OFF	OFF

Handwritten note:  $1.8" \rightarrow \frac{360}{1.8} = 200 \text{ steps}$

### BINARY

RESOLUTION	STEPS/REV	SWITCH 1	SWITCH 2	SWITCH 3	SWITCH 4
2	400	ON	ON	ON	ON
4	800	OFF	ON	ON	ON
8	1,600	ON	OFF	ON	ON
16	3,200	OFF	OFF	ON	ON
32	6,400	ON	ON	OFF	ON
64	12,800	OFF	ON	OFF	ON
128	25,600	ON	OFF	OFF	ON
256	51,200	OFF	OFF	OFF	ON

ILLEGAL SETTINGS

ON	OFF	OFF	OFF
OFF	OFF	OFF	OFF

**NOTE:** In the above table ON is ground and OFF is floating when using the terminal strip inputs to set the microstep resolution.

---

### FAULT PROTECTION

The INT-481 adds phase to ground fault protection to the IM481H. If a phase to ground fault is detected the IM481H will latch the signal, set the fault output and illuminate the red fault LED. To clear the fault condition, the IM481H will have to be reset or power will need to be cycled.

The INT-481 buffers the IM481H's fault output signal through an open drain N-channel FET. The signal at the terminal strip is inverted and is active low.

### FULL STEP OUTPUT

The INT-481 buffers the IM481H's full step output through an open drain N-channel FET. The signal available at the terminal strip is inverted and is active low.

---

### BLOCK DIAGRAM

### RECOMMENDED WIRING

Logic level cables must not run parallel to power cables. Power cables will introduce noise into the logic level cables and make your system unreliable.

Logic level cables must be shielded to reduce the chance of EMI induced noise. The shield needs to be grounded at the signal source to AC ground. The other end of the shield must not be tied to anything, but allowed to float. This allows the shield to act as a drain.

Motor cabling in excess of 1 foot requires twisted pair shielded cable to reduce the transmission of EMI. The shield must be connected to AC ground at the driver. The other end of the shield must not be tied to anything, but allowed to float. This allows the shield to act as a drain.

Power supply leads to the driver need to be twisted. If more than one driver is to be connected to the same power supply, run separate power and ground leads from the supply to each driver.

Refer to the IM481H operating instructions for recommended motor and power supply cables.

# Appendix A4: Voltage-to-frequency converter card specifications

## Frequency signal conditioners

- frequency outputs
- LED-indicators
- switchable frequency output

### MCZ VFC

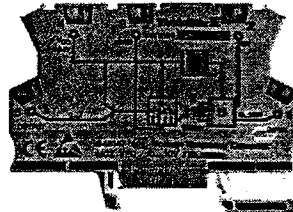
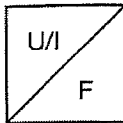
0...10 V

### MCZ CFC

0...20 mA

### MCZ CFC

4...20 mA CLP

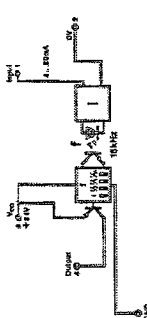
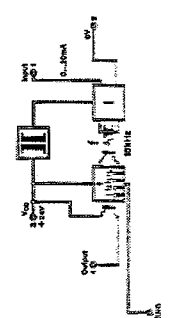
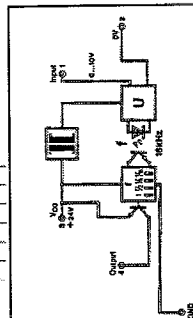


### Important for installation:

By converting analog signals into frequencies, it is possible to read in analog values into the controller counter-inputs. Here it is also recommended that twisted and shielded pairs be used.

### Schematic circuit diagrams

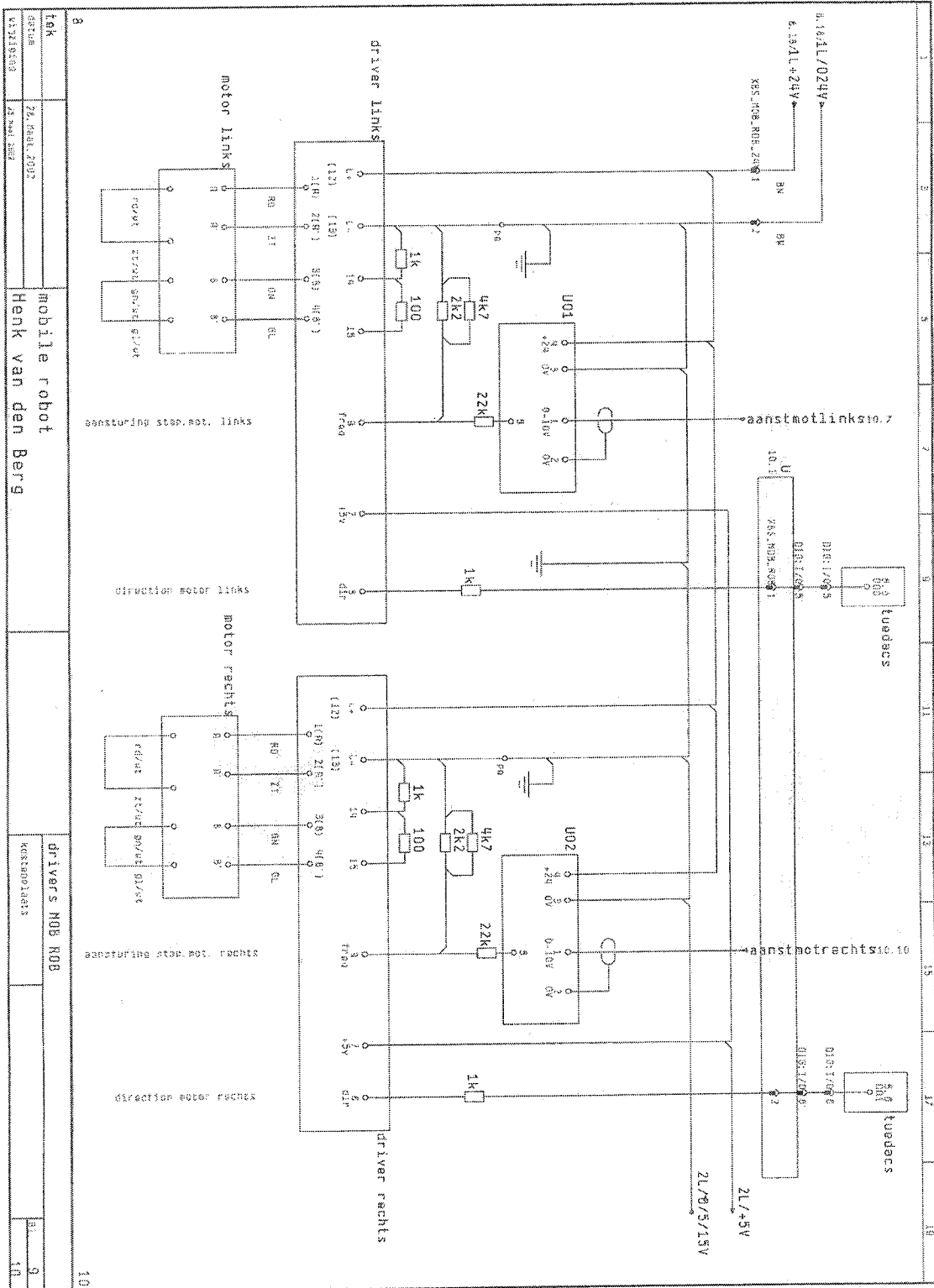
1	2	3	4	DIP switch
1	0	0	0	0...16 kHz
0	1	0	0	0...8 kHz
0	0	1	0	0...4 kHz
0	0	0	1	0...1 kHz



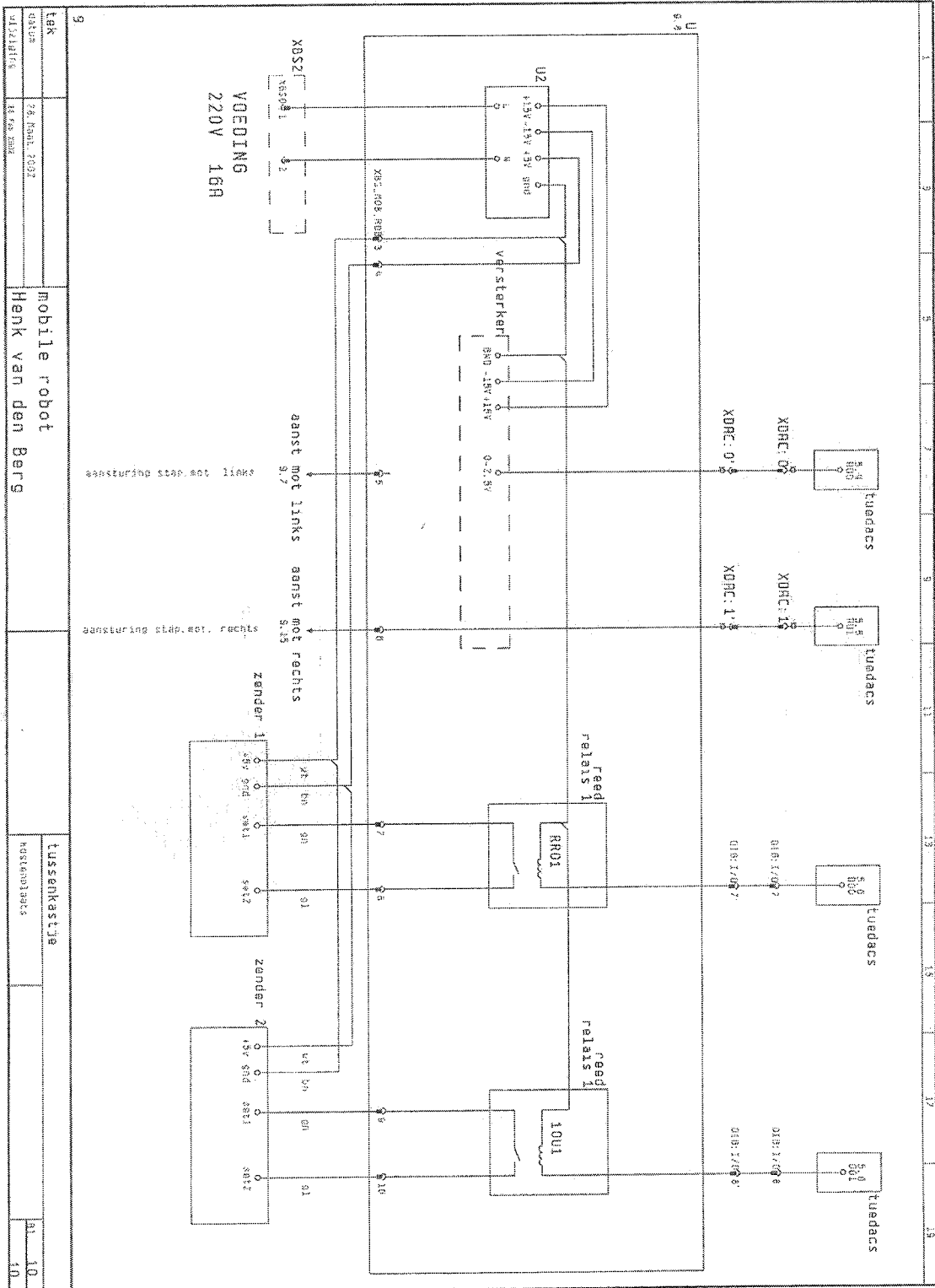
Ordering data	
for TS 35	
Technical data	
Input ranges	
Overload limits, input	
Input impedance	
Voltage drop, input	
Output	
Output frequency, final value	
Frequency setting	
Readjustment range	
Output level	
Output current	
Display	
Supply voltage	
Current consumption	
Making current limit	
Polarization protection	
Precision	
Temperature coefficient	
Isolation coordinates acc. to EN 50178	
Voltage proof input/output	
Rated voltage	
Rated surge	
Overvoltage category	
Voltage proof, input/output mounting rail	
Operating temperature	
Storage temperature	
Total width	
Conductor cross-section	

Type	Cat. No.	Type	Cat. No.	Type	Cat. No.
MCZ VFC	846147	MCZ CFC	846148	MCZ CFC	846149
0...10 V		0...20 mA		4...20 mA LP	
30 V		50 mA		50 mA	
100 kΩ		50 Ω			
		1 V at 20 mA		5.6...6.4 at 20 mA	
1 kHz, 4 kHz, 8 kHz, 16 kHz		1 kHz, 4 kHz, 8 kHz, 16 kHz		1 kHz, 4 kHz, 8 kHz, 16 kHz	
DIP switch		DIP switch		DIP switch	
±10 %, internally		±10 %, internally		±10 %, internally	
PNP, Ub- 0.7 V		PNP, Ub- 0.7 V		PNP, Ub- 0.7 V	
max. 20 mA		max. 20 mA		max. 20 mA	
LED, alternating		LED, alternating		LED, alternating	
24 Vdc ±10 %		24 Vdc ±10 %		24 Vdc ±20 %	
14 mA, no load		14 mA, no load		14 mA, no load	
200 mA		200 mA			
yes		yes		yes	
0.2 % v. FSR		0.2 % v. FSR		0.15 % v. FSR	
< 250 ppm/°C		< 250 ppm/°C		< 250 ppm/°C	
1 kVdc		1 kVdc			
100 V		100 V		150 V	
1.5 kV		1.5 kV		2.5 kV	
III		III		III	
4 kV <sub>eff</sub> / 1 min		4 kV <sub>eff</sub> / 1 min		4 kV <sub>eff</sub> / 1 min	
0 °C...+50 °C		0 °C...+50 °C		0 °C...+50 °C	
-25 °C...+85 °C		-25 °C...+85 °C		-25 °C...+85 °C	
6 mm		6 mm		6 mm	
1.5 mm <sup>2</sup>		1.5 mm <sup>2</sup>		1.5 mm <sup>2</sup>	

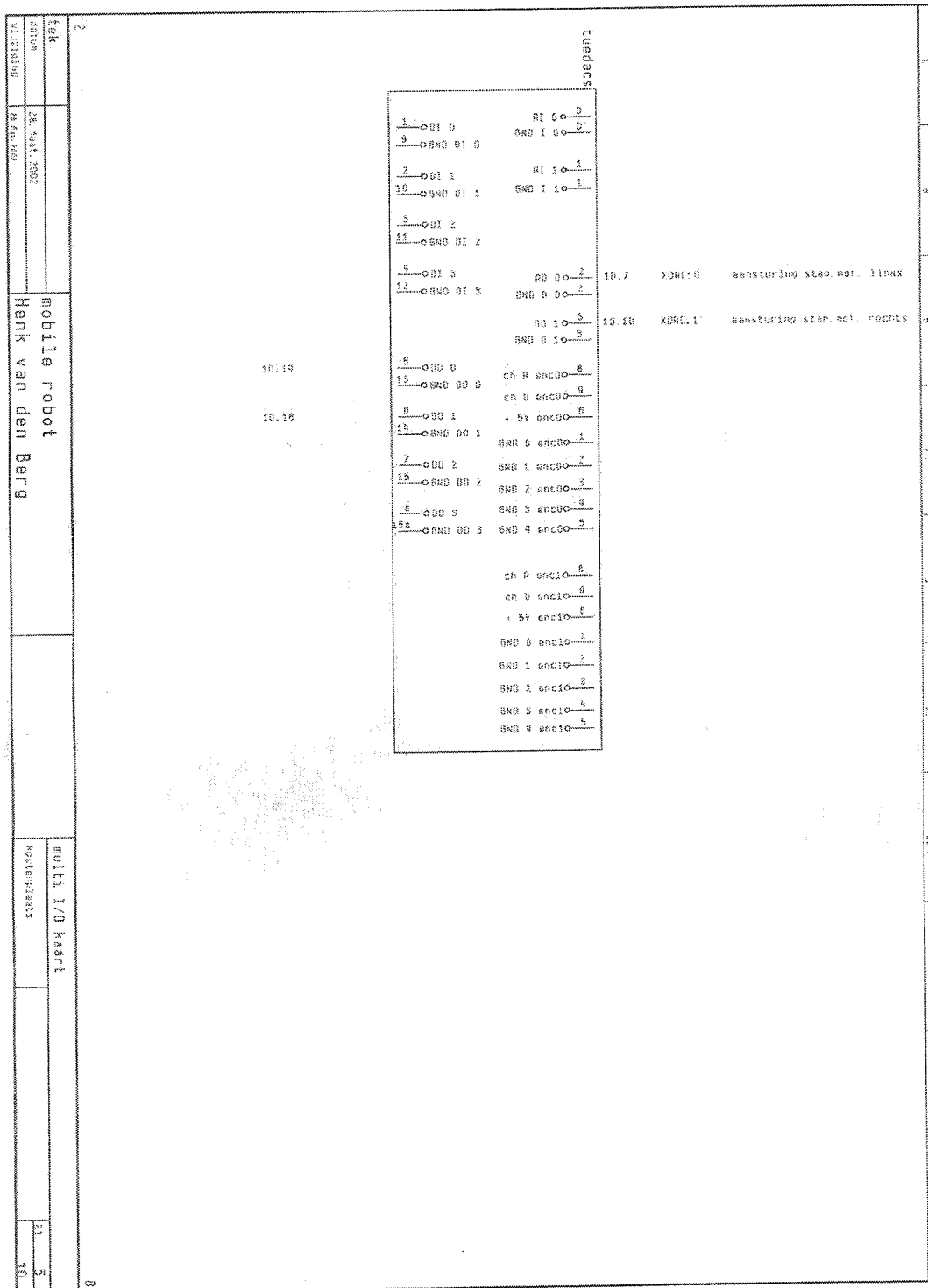
# Appendix B1: Electro technical scheme for step motors, microstepping drivers and V/f-converter cards



## Appendix B2: Electro technical scheme for MOBROB unit and eBeam transmitters

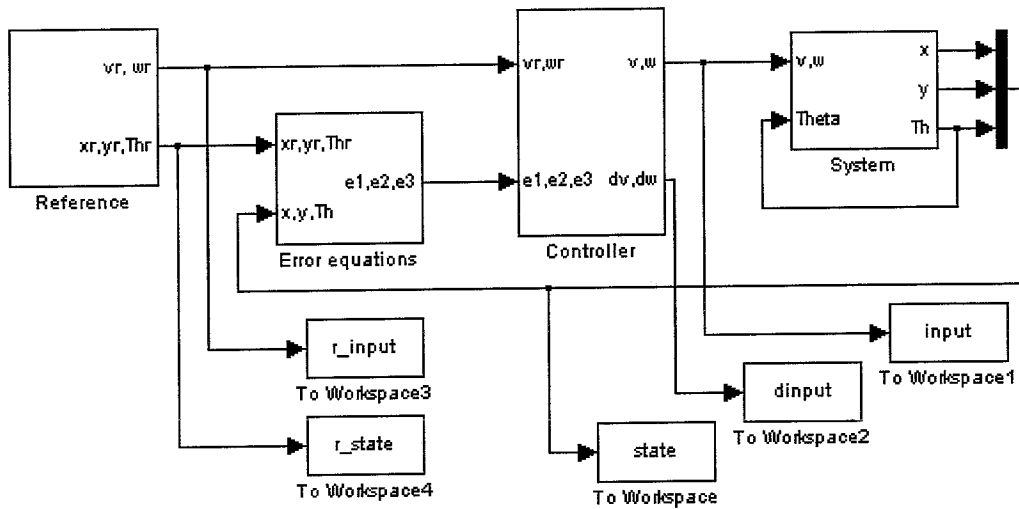


# Appendix B3: Electro technical scheme for TUE DACS QAD

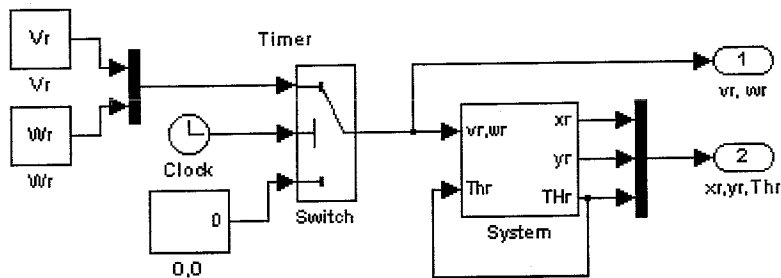


## Appendix C1: Simulink scheme for tracking controllers

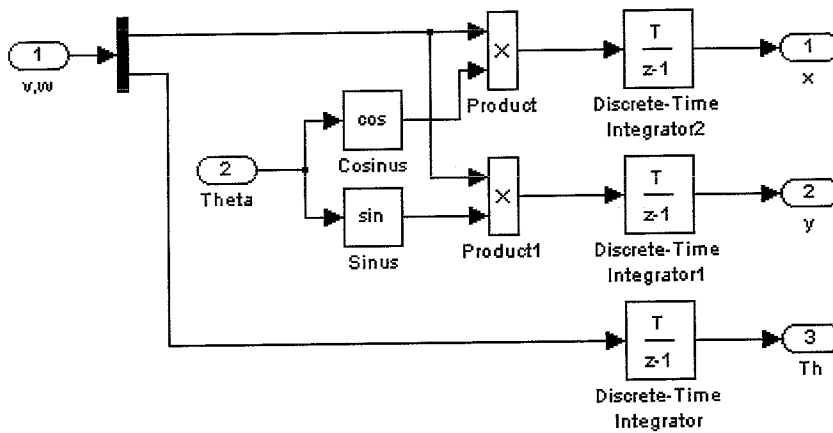
Overview:



Reference:

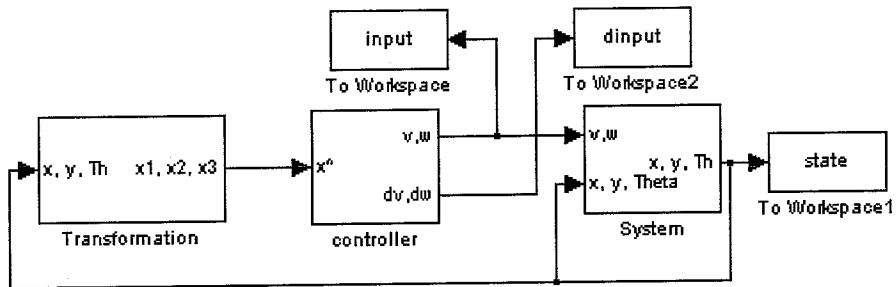


System:

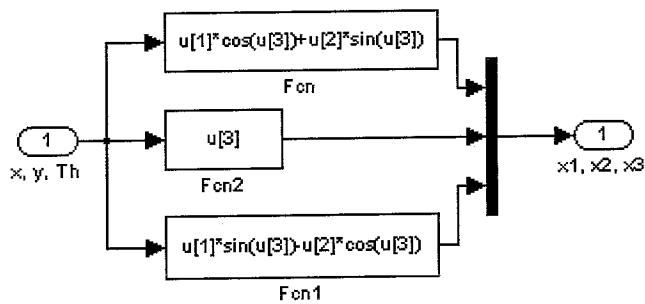


## Appendix C2: Simulink scheme for point stabilization controllers

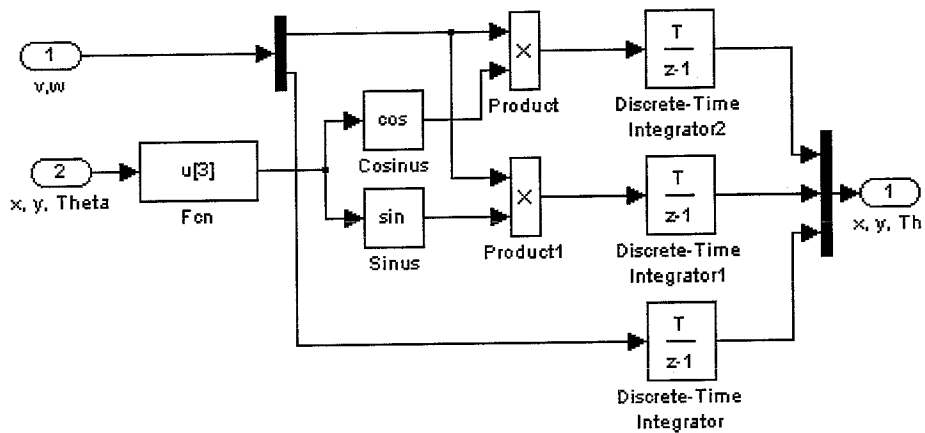
Overview:



Transformation:



System:



## Appendix C3: Simulation/realtime Matlab initialization file example

```
% init_tracking.m: Initialization for tracking controllers models:
% sim_tracklin, rt_tracklin, sim_tracknonlin and rt_tracknonlin

clear all; close all;

Reg = 1;      % Controller choice: Reg = 1 -> linear controller, Reg = 2
              % -> non-linear controller

mode = 1;    % Mode choice: mode = 1: simulation, mode = 2: realtime

% Time variables
dt = 0.001; Tsim = 15; T_init = 2;

% Control parameters
ksi = 1.5; b = 0.000; e3_min = 1e-9;

% Starting point:
x0 = 0; y0 = 0; Th0 = 0; v0 = 0; w0 = 0;

% Reference trajectory and starting point:
Vr = 0.3; Wr = 1; xr0 = 0; yr0 = -0.3; Thr0 = 0;

% Calculation maximal wheel velocity:
msr = 50;          % micro-stepping rate
ns = 200;          % nr. of steps per revolution
f_max = 16000;    % maximal frequency
phi_max = 2*pi*f_max/(ns*msr); % maximal wheel velocity

% Maximal forward and angular acceleration:
dv_max = 1; dw_max = 5;

% Matrix M parameters:
r1 = 0.0402; r2 = 0.0417; R1 = 0.077; R2 = 0.078;

if mode == 1 % Simulation
v_max = (r1*R2+ r2*R1)/(R1+R2)*phi_max;
w_max = (r1+r2)/(R1+R2)*phi_max;

% To Simulink
if Reg == 1
    sim sim_tracklin
elseif Reg == 2
    sim sim_tracknonlin
end

% From Simulink:
x = state(:,1); y = state(:,2); Th = state(:,3);
xr = r_state(:,1); yr = r_state(:,2); Thr = r_state(:,3);
v = input(:,1); w = input(:,2);
vr = r_input(:,1); wr = r_input(:,2);
dv = dinput(:,1); dw = dinput(:,2);

if mode == 2 % Realtime
a = r1*R2/(R1+R2); b = r2*R1/(R1+R2); c = -r1/(R1+R2); d = r2/(R1+R2);
M=[a b;c d];

end
```



## Appendix C4: Matlab-file for a tractor with one trailer point stabilization simulation

```

% Trailer1.m: Point stabilization control for tractor with one trailer

clear all; close all;

% Time parameters
dt = 0.002; T_end = 100; t = 0:dt:T_end;

% Matrix initialization
Zero = zeros(size(t)); V0 = Zero; V = Zero; W = Zero;
X = Zero; Y = Zero; TH1 = Zero; X0 = Zero; Y0 = Zero; TH0 = Zero;
GZ = Zero; kk = Zero; GAM2 = Zero; GAM3 = Zero; GAM4 = Zero;
Z1 = Zero; Z2 = Zero; Z3 = Zero; Z4 = Zero;
Ttel = 2*pi; Teller = 0;

kappa = 1; K = 2; L2 = 1; L3 = 1; L4 = 1; % Control parameters
d1 = 0.2; % length trailers

% Starting position REAR trailer (!) and starting orientations
x = 0; y = 1; Th0 = 0.07; Th1 = 0.1;

for it = 2:length(t)+1
    T = t(it-1);

    % Transformation to chained form
    z1 = x; z2 = tan(Th0-Th1)/(d1*cos(Th1)^3); z3 = tan(Th1); z4 = y;

    % Control parameter calculation
    z = sqrt(z2^2+z3^2+z4^2); Gz = kappa*z^0.25;
    f = (1-cos(T))/2; df = 0.5*sin(T); ddf = 0.5*cos(T);
    beta = 1/pi;
    if z1 == 0
        z1 = 1e-9;
    end

    Ttel = Ttel+dt;
    if Ttel >= 2*pi;
        k = -(z1+sign(z1)*Gz)*beta;
        if abs(k) >= K
            k = sign(k)*K;
        end
        Ttel = 0;
    end

    % Controller 1 for state z1:
    v0 = k*f; p1 = cos(Th1); v = cos(Th1)*cos(Th0-Th1)*v0;
    % Controller 2 for state z2,z3 and z4
    Gam2 = -L2 - L3*f^3 - L4*f^5;
    Gam3 = -f*(L2*L3*f + L2*L4*f^4 + 2*L3*df + 8*L4*df*f^2 + L3*L4*f^6)/k;
    Gam4 = -f*(L2*L3*L4*f^5 + 4*L2*L4*df*f + 6*L3*L4*df*f^4 + 8*L4*df^2 + 4*L4*ddf*f)/k^2;
    w = Gam2*z2 + Gam3*z3 + Gam4*z4;

    % Calculate new state:
    Th1_d = tan(Th0-Th1)*v/(d1*p1); Th1 = Th1 + dt*Th1_d;
    x_d = v; x = x + dt*x_d; y_d = tan(Th1)*v; y = y + dt*y_d;

    Th0_d = w; Th0 = Th0 + dt*Th0_d;
    x0 = x + d1*cos(Th1); y0 = y + d1*sin(Th1);

    % Save variables in matrices
    X(it-1) = x; Y(it-1) = y; TH1(it-1) = Th1;
    X0(it-1) = x0; Y0(it-1) = y0; TH0(it-1) = Th0;
    Z1(it-1) = z1; Z2(it-1) = z2; Z3(it-1) = z3; Z4(it-1) = z4;
    V0(it-1) = v0; V(it-1) = v; W(it-1) = w;
    GAM2(it-1) = Gam2; GAM3(it-1) = Gam3; GAM4(it-1) = Gam4;
    kk(it-1) = k; GZ(it-1) = Gz;
end
end

```

## Appendix C5: Matlab-file for a tractor with two trailers point stabilization simulation

```

% Trailer2.m: Point stabilization simulation for tractor pulling two trailers
% Initialize matrices:
Nul = zeros(size(t));
X=Nul; Y=Nul; TH2=Nul; X1=Nul; Y1=Nul; TH1=Nul; X0=Nul; Y0=Nul; TH0=Nul;
V0=Nul; V=Nul; W=Nul; Z1=Nul; Z2=Nul; Z3=Nul; Z4=Nul; Z5=Nul;
Ttel = 2*pi; omega = 1;

dt = 0.002; T_end = 100; t = 0:dt:T_end; % Time variables
kappa=3; K=2; L2=1; L3=1; L4=1; L5=1; eps=1e-9; % Control parameters
d1 = 0.2; d2 = 0.2; % length trailers

% Starting position REAR (!) trailer and starting orientations
x = 0; y = 0.2; Th0 = 0.2*pi; Th1 = 0; Th2 = 0;

for it = 2:length(t)+1
    T = t(it-1);
    % Transformation to the chained form
    p1 = cos(Th2)*cos(Th1-Th2); p2 = cos(Th2);
    f1 = tan(Th0-Th1)/(d1*p1); f2 = tan(Th1-Th2)/(d2*p2); c2 = p1^3*p2*d1*d2;
    tau2=((-1-tan(Th1-Th2)^2)/(d2*cos(Th2)^3)+3*tan(Th1-Th2)*sin(Th2)/(d2*cos(Th2)^4))*f2;
    z1=x; z2=tan(Th0-Th1)/c2+tau2; z3=tan(Th1-Th2)/(d2*cos(Th2)^3); z4 = tan(Th2); z5=y;

    % Calculation control parameters
    z = sqrt(z2^2+z3^2+z4^2+z5^2); Gz = kappa*z^(1/6); beta = omega/pi;
    f=(1-cos(omega*T))/2;df=0.5*omega*sin(T);ddf=0.5*omega^2*cos(T);d3f=-0.5*omega^3*sin(T);
    if z1 == 0
        z1 = 1e-9;
    end
    Ttel = Ttel+dt;
    if Ttel >= 2*pi
        k = -(z1+sign(z1)*Gz)*beta;
        if abs(k) >= K
            k = sign(k)*K;
        end
        Ttel = 0;
    end

    % Controller 1 for state z1
    v0 = k*f; v = cos(Th2)*cos(Th1-Th2)*cos(Th0-Th1)*v0;
    % Controller 2 for state z2, z3, z4 and z5
    g23 = -L3 -L4*f^2 - L5*f^4;
    g24 = -L3*L4*f^4 - L3*L5*f^6 - L4*L5*f^8 - 4*L4*df - 12*L5*df*f^2;
    g25 = -L3*L4*L5*f^10 - 6*L3*L5*df*f^4 - 10*L4*L5*df*f^6 - 24*L5*df^2 - 6*L5*ddf*f;
    dg23 = -2*L4*df*f - 4*L5*df*f^3;
    dg24 = -4*L3*L4*df*f^3 - 6*L3*L5*df*f^5 - 8*L4*L5*df*f^7 - 4*L4*ddf - 12*L5*ddf*f^2 -
    24*L5*df^2*f;
    dg25 = -10*L3*L4*L5*df*f^9 - 6*L3*L5*ddf*f^4 - 24*L3*L4*df^2*f^3 - 10*L4*L5*ddf*f^6 ... -
    60*L4*L5*df^2*f^5 - 48*L5*ddf*df - 6*L5*d3f*f - 6*L5*ddf*df;

    Gam2 = -L2 + f^3*g23;
    Gam3 = f*(L2*f*g23 + 2*df*g23 + f*dg23 + f^2*g24)/k;
    Gam4 = f*(L2*f*g24 + 2*df*g24 + f*dg24 + f^2*g24)/k^2;
    Gam5 = f*(L2*f*g25 + 2*df*g25 + f*dg25)/k^3;
    w = Gam2*z2 + Gam3*z3 + Gam4*z4 + Gam5*z5;

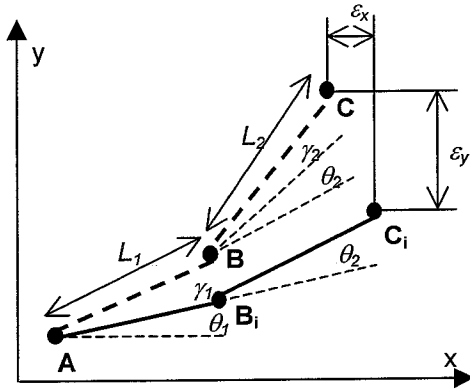
    % Nieuwe toestand bepalen
    x_d = v; x = x + dt*x_d; Th1_d = f1*v; Th1 = Th1 + dt*Th1_d;
    Th2_d = f2*v; Th2 = Th2 + dt*Th2_d; y_d = tan(Th1)*v; y = y + dt*y_d;
    Th0_d = w; Th0 = Th0 + dt*Th0_d; x1 = x + d2*cos(Th2);
    y1 = y + d2*sin(Th2); x0 = x1 + d1*cos(Th1); y0 = y1 + d1*sin(Th1);

    % Save variables in matrices
    X(it-1)=x; Y(it-1)=y; TH2(it-1)=Th2; X1(it-1)=x1; Y1(it-1)=y1; TH1(it-1)=Th1;
    X0(it-1)=x0; Y0(it-1)=y0; TH0(it-1)=Th0; V0(it-1)=v0; V(it-1)=v; W(it-1)=w;
    Z1(it-1)=z1; Z2(it-1)=z2; Z3(it-1)=z3; Z4(it-1)=z4; Z5(it-1)=z5;
end

```

## Appendix D: Maximum position error calculation for the pantograph system

In figure D1 the encoder arm position error is illustrated.



$$\begin{aligned} x_{C_i} &= \cos(\theta_1) \cdot L_1 + \cos(\theta_1 + \theta_2) \cdot L_2 \\ y_{C_i} &= \sin(\theta_1) \cdot L_1 + \sin(\theta_1 + \theta_2) \cdot L_2 \end{aligned} \quad [D.1]$$

$$\begin{aligned} x_{C_e} &= \cos(\theta_1 + \gamma_1) \cdot L_1 + \cos(\theta_1 + \theta_2 + \gamma_1 + \gamma_2) \cdot L_2 \\ y_{C_e} &= \sin(\theta_1 + \gamma_1) \cdot L_1 + \sin(\theta_1 + \theta_2 + \gamma_1 + \gamma_2) \cdot L_2 \end{aligned} \quad [D.2]$$

Figure D1: Sketch of the position errors of the encoder arm

In this figure the solid line shows an arbitrary configuration for the encoder arm. The first encoder A is attached to the framework. The first arm, between encoder A and B, has an orientation angle  $\theta_1$  and a length  $L_1$ . The second arm has an orientation angle  $\theta_2$  compared to the first arm and a length of  $L_2$ . The third encoder C is attached to the robot.

The dashed line represents the encoder arm when both the encoder A and B have finite accuracy. The maximum orientation errors of encoder A and B are  $\gamma_1$  and  $\gamma_2$  respectively. The resulting position errors are  $\epsilon_x$  and  $\epsilon_y$ .

The position of the point  $C_i$ , which is the robot position in the ideal situation, is given in equation D.1; the position of the point  $C_e$ , which is the robot position in the non-ideal situation, is given in equation D.2.

Now, the position errors can be calculated:

$$\begin{aligned} \epsilon_x &= x_{C_i} - x_{C_e} \\ \epsilon_y &= y_{C_i} - y_{C_e} \\ \epsilon_{tot} &= \sqrt{\epsilon_x^2 + \epsilon_y^2} \end{aligned} \quad [D.3]$$

The total orientation error is given by:

$$\gamma = \gamma_1 + \gamma_2 + \gamma_3 = \frac{2 \cdot \pi}{4} \cdot \left( \frac{1}{z_1} + \frac{1}{z_2} + \frac{1}{z_3} \right) \quad [D.4]$$

Here,  $z_1$ ,  $z_2$  and  $z_3$  are the line counts of the encoders. The factor 4 is the results of the fact that most encoders have two disks above each other. This way not only the direction of the rotation can be obtained, but also more accurate measurement, since four different states can be recognized as shown in figure D.2:

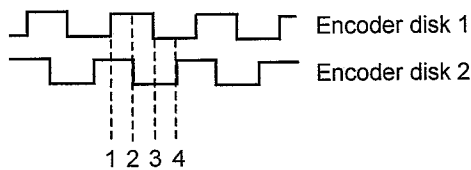


Figure D.2: Gained accuracy when using two encoder disks

## Appendix E1: eBeam software development kit C++-code

```
// The following ifdef block is the standard way of creating macros which make exporting
// from a DLL simpler. All files within this DLL are compiled with the WBAPI_EXPORTS
// symbol defined on the command line. this symbol should not be defined on any project
// that uses this DLL. This way any other project whose source files include this file see
// WBAPI_API functions as being imported from a DLL, whereas this DLL sees symbols
// defined with this macro as being exported.

#ifndef __WBAPI_H__
#define __WBAPI_H__

#ifdef WBAPI_EXPORTS
#define WBAPI_API __declspec(dllexport)
#else
#define WBAPI_API __declspec(dllimport)
#endif

//-----
// Variable types
//-----

enum WB_COLOR {WBBLACK = 0, WBRED = 1, WBBLUE = 2, WBGREEN = 3, WBERASE = 4};

struct WBStruct
{
    WB_COLOR color;
    short x;
    short y;
};

enum POD_LOCATION {TOP, BOTTOM, LEFT, RIGHT};
enum WB_PORT {INVALID_PORT=0, COM1_PORT=1, COM2_PORT=2, COM3_PORT=3, COM4_PORT=4,
              COM5_PORT=5, COM6_PORT=6, COM7_PORT=7, COM8_PORT=8};
enum WB_ERROR_CODE { WB_NO_ERROR=0, // No problem
                    WB_ERROR_COM_UNAVAILABLE_SYNC, // Cannot gain access to com port
synchronously     WB_ERROR_COM_UNAVAILABLE_ASYNC, // Cannot gain access to com port
asynchronously     WB_ERROR_NO_EBEAM_FOUND, // Cannot detect eBeam system
                    WB_ERROR_UNSUPPORTED_VERSION, // H/W version unsupported
                    WB_ERROR_CALIBRATION_FAILED, // eBeam found but could not cal
                    WB_ERROR_PORT_NOT_VALID, // Not a supported COM port
                    WB_ERROR_PORT_NOT_OPEN}; // Operation tried on un-open port

//-----
// Function definitions
//-----
// Attempt to detect eBeam on specified port.
// If eBeam is found, messages will be sent to windowHandle
WBAPI_API WB_ERROR_CODE WBOpenPort(WB_PORT portToOpen, HWND windowHandle);

// Attempt to autodetect eBeam. Returns the port on which eBeam was detected
// or INVALID_PORT if no eBeam could be detected. If no eBeam is detected,
// the find port error structure contains the error codes for each port that
// was attempted. As above, messages will be sent to windowHandle.
typedef struct
{
    WB_ERROR_CODE com1Error;
    WB_ERROR_CODE com2Error;
} SFindPortError;
WBAPI_API WB_PORT WBFindPort(HWND windowHandle, SFindPortError *sFindPortError);

// Close a previously opened port.
WBAPI_API WB_ERROR_CODE WBClose(WB_PORT portToClose);

// Change the window to which messages are sent.
// Returns the old HWND or NULL if an error occurred.
WBAPI_API HWND WBSethWND(WB_PORT port, HWND hwndNew);
```

## Appendix E1: eBeam software development kit C++-code (continued)

```
// The pods may be located on any edge - TOP, LEFT, RIGHT, or BOTTOM.
WBAPI_API WB_ERROR_CODE WBSetPodLocation(WB_PORT port, POD_LOCATION newPodLocation);

// If the BOTTOM or RIGHT is chosen, the DLL needs also to know the board
// "height" - defined to be the dimension away from the pods
WBAPI_API WB_ERROR_CODE WBSetBoardHeight(WB_PORT port, UINT newBoardHeight);

// Retrieve the board width - defined as distance between the pods.
// This is in the same units as the (x,y) coordinates sent in the messages.
WBAPI_API WB_ERROR_CODE WBGetBoardWidth(WB_PORT port, UINT *boardWidth);

// Retrieve the upper bound on the board height for the system.
// Height is the distance perpendicular to the width - see WBSetPodLocation
// and WBSetBoardHeight.
WBAPI_API WB_ERROR_CODE WBGetMaxBoardHeight(WB_PORT port, UINT *maxBoardHeight);

// Retrieve a string containing the version information for the eBeam pods.
WBAPI_API WB_ERROR_CODE WBGetBoardVersion(WB_PORT port, char* versionString);
// The above version string is guarantee to be shorter than this.
#define MAX_VERSION_STRING_LENGTH 20

//-----
// Messages sent to the window specified by windowHandle
//-----
// Pen down: Start of stroke
#define WM_BOARDPENDOWN (WM_APP + 1)
// Pen move: Stroke continuing
#define WM_BOARDPENMOVE (WM_APP + 2)
// Pen up: End of stroke
#define WM_BOARDPENUP (WM_APP + 3)
// For the above WM_BOARDPEN* messages, the WPARAM parameter is a pointer to
// a WBStruct and the LPARAM value is the WB_PORT where writing is taking place.
// ---Warning: THE APPLICATION SHOULD NOT FREE THE MEMORY FOR THE WBStruct*---
#define WM_BOARD_ERR (WM_APP + 4)
// WM_BOARD_ERR possible WPARAM values contain the following error codes.
// The LPARAM is the WB_PORT that originated the error.
enum WM_BOARD_ERR_CODES {WM_TWOPENSDETECTED, WM_SYSTEMRESET, WM_LOWBATTERY};
// This message indicates a major breakdown in communications and should not be
// received.
#define WM_COMM_ERR (WM_APP + 6)
// This message indicates that the eBeam system is no longer responding.
// It usually means that the cables have been unplugged.
#define WM_BOARDCOMMLOST (WM_APP + 7)

#endif
```

## Appendix E2: C++-code for final.dll

```
/* ===== */
/* Final.cpp, file for het creating final.dll. */
/* Needed to gain the coordinates and sleeve color from the eBeam system */
/* ===== */

/* Defines, includes and declarations
----- */

#define STRICT
#include <windows.h>
#include <stdio.h>
#include <string.h>
#include <iostream.h>
#include "wbapi.h"

static short XX, YY; static short Kleur = -1; static UINT width;
static WB_PORT portNr; HANDLE HINST;

// declare WndProc
LONG WINAPI WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam);

/* DllMain
----- */

BOOL APIENTRY DllMain( HANDLE hModule,
                      DWORD ul_reason_for_call,
                      LPVOID lpReserved )
{
    HINST=hModule;
    switch (ul_reason_for_call)
    {
        case DLL_PROCESS_ATTACH:
        case DLL_THREAD_ATTACH:
        case DLL_THREAD_DETACH:
        case DLL_PROCESS_DETACH:
            break; }
    return TRUE;
}

/* WBThread
----- */

static DWORD WBThreadID;
DWORD WBThread(LPVOID);

// Define function WBStart
extern "C" _declspec(dllexport) int WBStart(void)
{
    CreateThread(NULL,0, (LPTHREAD_START_ROUTINE) WBThread,
                (LPVOID) NULL,0,&WBThreadID);
    return 0; }

// Define WBThread
DWORD WBThread(LPVOID param)
{
    char szAppName[] = "voorbeeld";
    WNDCLASS wndclass; HWND hWnd; MSG msg;

    // define windowclass
    wndclass.style = CS_HREDRAW | CS_VREDRAW;
    wndclass.lpfnWndProc = WndProc;
    wndclass.cbClsExtra = 0;
    wndclass.cbWndExtra = 0;
    wndclass.hInstance = (HINSTANCE) HINST;
    wndclass.hIcon = LoadIcon(NULL, IDI_APPLICATION);
    wndclass.hCursor = LoadCursor(NULL, IDC_ARROW);
    wndclass.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);
    wndclass.lpszMenuName = NULL;
    wndclass.lpszClassName = szAppName;
    if (!RegisterClass(&wndclass)) return FALSE;
}
```

## Appendix E2: C++-code for final.dll (continued)

```
// Create window
hWnd = CreateWindow(
    szAppName, "E-Beam output", WS_OVERLAPPEDWINDOW, 0, 0, CW_USEDEFAULT,
    CW_USEDEFAULT, NULL, NULL, (HINSTANCE) HINST, NULL);

// Open eBeam communication port
WB_ERROR_CODE open_error = WB_NO_ERROR;
WB_PORT portNr = COM2_PORT;
open_error = WBOpenPort(portNr, hWnd);

// Gain board width
WB_ERROR_CODE width_error = WB_NO_ERROR;
width = 4000;
width_error = WBGetBoardWidth(portNr, &width);

ShowWindow(hWnd, SW_HIDE); /* SW_SHOW if window needs to be visable */
                          /* SW_HIDE if window needs to be invisible */

UpdateWindow(hWnd);

// Messageloop
while (GetMessage(&msg, NULL, 0, 0))
{
    TranslateMessage(&msg);
    DispatchMessage(&msg);
}

return 0;
}

/* ----- */
/* Define WndProc */
/* ----- */

LONG WINAPI WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    struct WBStruct *pcoor;

    switch (message)
    {
        case WM_DESTROY:
            PostQuitMessage(0); break;
        case WM_BOARDPENDOWN:
            break;
        case WM_BOARDPENMOVE:
            pcoor = (WBStruct*) wParam;
            XX = pcoor[0].x;
            YY = pcoor[0].y;
            Kleur = pcoor[0].color;
            break;
        case WM_BOARDPENUP:
            Kleur = 5; XX = 0; YY = 0;
            break;
        case WM_BOARD_ERR:
            Kleur = -1; XX = 0; YY = 0;
            break;
        case WM_COMM_ERR:
            Kleur = -1; XX = 0; YY = 0;
            break;
        case WM_BOARDCOMMLOST:
            Kleur = -1; XX = 0; YY = 0;
            break;
        default:
            return DefWindowProc(hWnd, message, wParam, lParam);
    }
    return 0L;
}
```

## Appendix E2: C++-code for final.dll (continued)

```
/* ----- */
/* Define functions WBInfo (coordinates and color), WBStatus (system status), */
/* WBPort (connected port) and WBWidth (board width)
*/
/* ----- */

extern "C" _declspec(dllexport) int WBInfo(short* pXX, short* pYY, short* pKleur)
{
    pXX[0]=XX;
    pYY[0]=YY;
    pKleur[0]=Kleur;

    return 0;
}

extern "C" _declspec(dllexport) int WBStatus(void)
{
    if (Kleur < 0)
    {
        return 0;
    }
    else
    {
        return 1;
    }

    return 0;
}

extern "C" _declspec(dllexport) int WBPort(int* pPort)
{
    pPort[0] = portNr;

    return 0;
}

extern "C" _declspec(dllexport) int WBWidth(int* pWidth)
{
    pWidth[0] = width;

    return 0;
}
```



## Appendix E3: C++-code for eBeam S-function

```
/* ebeam.c: Import eBeam-coordinates into Matlab/Simulink
 * More information in: "sfuntmpl.doc"
 * Copyright 2002, Henk van den Berg, Version 1.0 */

/* ===== *
 * Defines en includes *
 * ===== */

#define S_FUNCTION_LEVEL 2
#define S_FUNCTION_NAME eBeam
#include "simstruc.h"
#include <string.h>
#include <windows.h>
#include <stdio.h>

// Functions from final.dll
extern int WBStatus(void);
extern int WBStart(void);
extern int WBInfo(short*, short*, short*);
extern int WBWidth(int*);
extern int WBPort(int*);

static short x,y,color;
static int width;
static int portNr;

/*=====*
 * mdlInitializeSizes *
 *=====*/

static void mdlInitializeSizes(SimStruct *S)
{
    // Start WBStart
    printf("Starting WBThread... \n");
    WBStart();

    // Wait until initialization is complete
    while (!WBStatus())
    { Sleep(100);}
    printf("Initialisation ok... \n");

    // Check port
    WBPort(&portNr);
    printf("Communicatie via poort: %d\n", portNr);

    // Check board width
    WBWidth(&width);
    printf("Breedte bord: %d\n", width);

    ssSetNumSFcnParams(S, 0); /* Number of expected parameters */
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        /* Return if number of expected != number of actual parameters */
        return;
    }

    ssSetNumContStates(S, 0);
    ssSetNumDiscStates(S, 0);

    // No input-ports
    if (!ssSetNumInputPorts(S, 0)) return;

    // 4 output-ports (x, y, color, width)
    if (!ssSetNumOutputPorts(S, 4)) return;
    ssSetOutputPortWidth(S, 0, 1);
    ssSetOutputPortWidth(S, 1, 1);
    ssSetOutputPortWidth(S, 2, 1);
    ssSetOutputPortWidth(S, 3, 1);
}
```

## Appendix E3: C++-code for eBeam S-function (continued)

```
    ssSetNumSampleTimes(S, 1);
    ssSetNumRWork(S, 0);
    ssSetNumIWork(S, 0);
    ssSetNumPWork(S, 0);
    ssSetNumModes(S, 0);
    ssSetNumNonsampledZCs(S, 0);

    ssSetOptions(S, SS_OPTION_EXCEPTION_FREE_CODE);
}

/* ===== *
 * mdlInitializeSampleTimes *
 * ===== */

static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, CONTINUOUS_SAMPLE_TIME);
    ssSetOffsetTime(S, 0, 0.0);
}

/* ===== *
 * mdlOutputs *
 * ===== */

static void mdlOutputs(SimStruct *S, int_T tid)
{
    real_T *xlong = ssGetOutputPortRealSignal(S,0); // poort 1: x-coordinaat
    real_T *ylong = ssGetOutputPortRealSignal(S,1); // poort 2: y-coordinaat
    real_T *colorlong = ssGetOutputPortRealSignal(S,2); // poort 3: kleur
    real_T *widthlong = ssGetOutputPortRealSignal(S,3); // poort 4: breedte bord

    // send signal to specified output ports
    WBInfo(&x,&y,&color);
    {
        printf("x=%d y=%d color=%d\n",x,y,color);
        xlong[0] = (int_T) x;
        ylong[0] = (int_T) y;
        colorlong[0] = (int_T) color;
    }

    WBWidth(&width);
    {
        widthlong[0] = (int_T) width;
    }
}

/* ===== *
 * mdlTerminate *
 * ===== */

static void mdlTerminate(SimStruct *S)
{
}

/*=====*
 * Required S-function trailer *
 *=====*/

#ifdef MATLAB_MEX_FILE /* Is this file being compiled as a MEX-file? */
#include "simulink.c" /* MEX-file interface mechanism */
#else
#include "cg_sfun.h" /* Code generation registration function */
#endif
#endif
```