Control of a mobile robot

*Citation for published version (APA):*
Control of a mobile robot

J.F. van den Eerenbeemt

DCT 2001.57

Supervisors:
Dr.ir. H.A. van Essen
Dr.ir. D.A. Lizarraga
Chapter 1  Introduction

1.1  Introduction

In recent years a lot of research has been done on the field of non-linear mobile robots. There are many possible configurations, like carts with two or four wheels, or even extended with trailers. The model of such a system belongs to a special class of non-linear systems, called non-holonomic systems. Because of this property traditional control methods (like linearization and state feedback) fail when it comes to stabilization. Solutions to the control problem therefore are sought in more advanced mathematical controllers, like time-variant feedback or switching controllers. These non-linear controllers are proven in theory and simulations, but hardly in practice.

1.2  Statement of the problem

The goal of this project is to construct and operate a two wheeled mobile robot, powered by stepper motors, in order to validate non-linear controllers in an experimental environment.

1.3  Approach and overview

When I started the project, the stepper motors together with their microstepping drivers (to control the velocities) were already available. So the first step was to investigate the behavior of these motors and to control the input signal of the motors using a PC.

The next step is to implement the kinematical model of the two-wheel robot in Simulink. At this stage some simulations are done with existing controllers to get insight in the model and the control problem.

The control laws compute the input to the system based on the state variables (in this case the displacement and orientation of the system). We want to obtain the state vector by integrating the system model equations (assuming that the physical system is describe accurately by it's model). In this case we use the number of steps applied to the motors to reconstruct the angular displacement and velocity of the wheels.

To test the performance of the system using this integration scheme, it is compared to the simulation results of some existing controllers. Finally some real time experiments are performed.
Chapter 2 Hardware configuration

For this project a two-wheeled robot cart is constructed. The final design is schematically drawn in figure 2.1.

Figure 2.1: Schematic drawing of the robot

Figure 2.2 gives a systematic overview of the hardware configuration, used to control the robot.

Figure 2.2: Schematic overview of the hardware configuration

To drive the robot, two stepping motors are available. These motors are controlled by microstepping drivers, which are mounted on an interface board. This board consists of a 15 pin terminal, which is connected by a shielded cable to the digital io-port of a dSpace control board. The system is controlled by a PC using Matlab/Simulink. Furthermore, the power for the motors comes from a regulated power supply.

In this chapter each part will be discussed in more detail. In section 2.1 we start with some basics on stepping motors. In the next section (2.2), the motors and drivers that are used to drive the robot are introduced in detail. And finally in section 2.3 the problem of generating the input signal to the motors is addressed.
2.1 Basics on stepper motors

It's not the purpose of this section to discuss in detail how a stepping motor works. Because we are only interested in its input and output, the stepping motor is introduced as a ‘black box’ system (see figure 2.3).

2.1.1 Black box approach

Form this point of view a stepping motor may be considered as an electromotor that responds to a pulse input signal by rotating its output shaft over some elementary angle $\theta$. This angle depends on the number of steps the motor advances each complete revolution. For this traineeship ‘200 steps per revolution’ motors are available (they will be discussed in more detail in section 2.1.2), so this means that every step equals a displacement of 1.8 degrees. Using the microstepping drivers, each step is divided into smaller (micro) steps. If we let ‘q’ be the selected microstep resolution (in microsteps per revolution), the each time a pulse is applied to the input of the drivers the motor shaft will rotate over an angle which is equal to $\frac{2\pi}{q}$.

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

$$\omega_{\text{max}} = \frac{2\pi}{q} \cdot f_{\text{pulse}} \quad (2.1)$$
2.1.2 Accuracy and torque

In reality all stepping motors exhibit some non-linearity, meaning the microsteps bunch together rather than being spread evenly over the span of a full step. This is illustrated in figure 2.4. This has two effects, statically the motor position is not optimum and dynamically low speed resonances occur because of the cyclic acceleration where the microsteps are spread apart and deceleration where they bunch up.

![Motor position (mechanical) with bad linearity](image1)

![Motor position (mechanical) with excellent linearity](image2)

Figure 2.4: Motor with bad linearity (left) and with excellent linearity (right)

The torque provided by a stepper motor is the inverse of motor speed. Figure 2.5 shows the torque and power output as a function of the motor speed.

![Torque and Power](image3)

Figure 2.5: Torque and output power as a function of motor speed
2.1.3 Why using stepping motors

Using stepping motors has two main advantages:

*Equivalence to mathematical model*

The computed output of the controllers (the input to the stepping motors) is in terms of velocities. These velocities directly can be converted into the equivalent 'steps per second'. This means that no additional velocity controllers are required.

*No encoders required*

Every pulse to the stepping motor results in a fixed displacement of the motor shaft. The total angular displacement of the motor shaft depends on the total number of pulses applied and its angular velocity on the pulse rate (#pulses/second). So by knowing the input signal we are able to reconstruct the position and velocity of the motor. This means that no encoders are required to measure them.
2.2 Specifications of the motors and interface

In this section the specifications of the stepper motors, their drivers and the driver interface are presented.

2.2.1 The stepping motor

Specifications
Two 8 wire stepper motors are used (type RS-V9728) to drive the robot. Table 2.1 gives an overview of the technical specifications.

<table>
<thead>
<tr>
<th>Specification</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rated voltage (V)</td>
<td>5</td>
</tr>
<tr>
<td>Rated current (I)</td>
<td>1</td>
</tr>
<tr>
<td>Resistance (Ω)</td>
<td>5</td>
</tr>
<tr>
<td>Inductance (mH)</td>
<td>9</td>
</tr>
<tr>
<td>Detent torque (mNm)</td>
<td>30</td>
</tr>
<tr>
<td>Holding torque (mNm)</td>
<td>500</td>
</tr>
<tr>
<td>Step angle accuracy (%)</td>
<td>5</td>
</tr>
<tr>
<td>Step angle (degrees)</td>
<td>1.8</td>
</tr>
</tbody>
</table>

Table 2.1: Technical specifications of the stepping motor

Motor connections
The motor be operated in two configurations: parallel or serial connected. These wiring configurations are schematically shown in the figures below.

Figure 2.6: 8-wire motor, parallel connected (left) and series connected (right)

Because a parallel connected motor delivers twice as much power as a serial connected one at a given power supply voltage we would prefer the parallel configuration. However because in that case the current is two times the current in serial operation, the motor will heat up earlier. Experiments with both configurations showed that motor heat up is excessive in parallel connection, so for the project the motors are series connected.
2.2.2 Microstepping driver and Interface board

To control the motors, the IMS-IM481H microstepping driver is used. Together with a heat sink (H-481) the drivers are mounted on an interface board (IMS-INT481: see Appendix 1), which is schematically drawn in figure 2.7.

![Figure 2.7: Schematic drawing of the INT-481 interface board](image)

The inputs and outputs of the interface board are connected to the digital io-port of a DS1102 dSpace control board. Not all of them are used, so in the next text only those which are relevant for the control of the robot are addressed briefly.

**Pin 1 & 2: Phase B**
Phase B of the stepping motor is connected between pin 1 and 2.

**Pin 3 & 4: Phase A**
Phase A of the stepping motor is connected between pin 3 and 4.

**Pin 7: Opto supply**
This is the power supply of the optically isolated inputs to the board (see Appendix 1 for details). At this input a constant voltage of +5 V is applied.

**Pin 8: Direction**
This input is used to change the direction of the motor.

**Pin 9: Step clock**
At this input a pulse signal (+5V amplitude) of a variable frequency is applied. A positive going edge advances the motor one increment. The size of the increment depends on the selected microstep resolution.
Pin 12: Supply voltage input
This input is used to connect the motor DC power of 30 Volts to the interface board.

Pin 14 & 15: Output current and current reduction
Applying a voltage to this input (by connecting a pair of resistors) sets the output current send to the motor. The current reduction input allows to automatically reduce the current in the motor windings after the completion of a move. A pair of resistors sets the amount of reduction (see also Appendix 1).

Dip switch: Microstep resolution
The number of microsteps per step is selected by four dip switches. The microstep resolution varies from 2 microsteps/step up to 256 microsteps/step. The motors are manufactured to a certain tolerance, typically of +/- 5% non-accumulative error regarding the location of any given step. For the 200 step per revolution motors we are using this means that any step will be in an error range of 0.18 degrees. Stated otherwise, the motor can accurately resolve 2000 radial locations. This corresponds to a resolution of 10 microsteps per step. Any microstep resolution beyond 10 yields no additional accuracy, only empty resolution. Now, there are two reasons, justifying higher resolutions: in the case of controlling the robot very low speeds are required and in such situation higher microstepping resolution guarantees smooth operation especially at low speeds. Besides, in many instances mechanical gearing can be replaced with microstepping, reducing cost and eliminating potential maintenance. According to relation 2.1 the resolution affects the maximum velocity at which the robot will be able to drive. So determining the optimal resolution will be a trade of between smoothness and high operating speeds. Because we only need high resolution to guarantee smooth operation at very low speeds, it would be an option to make the resolution depend on the robot speed. In this study we will keep it constant during operation.
2.3 Generating the pulse signal

The control input of the model (chapter 3) will be in terms of the wheel velocities. These velocities are proportional to the frequency of the pulse signal applied at the 'step clock' input of the driver interface. So, to control our system, we have to generate a pulse signal with a variable frequency to be able to change the velocity of the wheels. One way to do this is by using frequency converters. These commercially available cards convert an input voltage into a pulse signal which frequency varies linearly with the input. The specifications of the cards that will be used are presented in Appendix 2. Due to long delivery times these cards were not available within the time span of this project. For this reason the pulse will be generated using a Simulink model (see figure 2.8), instead of using the cards. Basically it works like shown in the scheme on the next page (figure 2.9).

Figure 2.8: Simulink implementation of the pulse generator
Calculating the desired period time of the pulse (T_scheduled)

At the input of the pulse generator the reference velocity, 'w_ref' of the wheels is specified. This velocity is scaled to range from zero to one by means of the gain '1/w_max' ('gain' in figure 2.7). The exact relation for w_max will be discussed later on. We know that the wheel velocity is proportional to the frequency of the pulse signal at the step input of the motors. So the period time of the pulse is proportional to 1/w_ref. Using this relation, the period time of the pulse signal is calculated in Simulink using the following function (function 2 in figure 2.7)

\[
T_{\text{scheduled}} = \text{floor}(0.5 + \frac{K_s}{\text{eps} + \text{abs}(u)})
\]  

(2.2)

Here 'eps' is used to avoid division by zero and the constant '0.5' makes the floor function behave like a rounding function (because we cannot define values of the period time in between two sample moments).
The input (form 0 to 1) is scaled by a factor $K_s$. To explain the effect of this factor, let's take a look at function 2.2 for $K_s$ equal to one.

$$T \text{ as a function of } u \text{ for } K_s = 1$$

From figure 2.10 we can see that for small values of $u$ the output "T" is more sensitive to changes in its input then for large inputs. By using the scaling factor $K_s$ we limit ourselves to the first (more accurate) part of the curve is used. In the next figure the output frequency ($1/T$) is plotted for $K_s = 1$ and $K_s = 20$.

$$\text{Frequency as a function of } u \text{ for } K_s = 1$$

$$\text{Frequency as a function of } u \text{ for } K_s = 20$$

In the left figure $K_s = 1$, so no input scaling is used. It's clear that from $u = 0.3$ the output frequency is very insensitive to applied values of $u$ and the behavior is very non-linear. An input of 0.75 e.g. will generate a pulse of the same frequency as an input of 1. Using an input scaling of $K_s = 20$ produces a much more linear behavior (figure 2.10 right). Also note from this figure that by using the input scaling $K_s$, the maximum pulse frequency that can be generated reduces by a factor $K_s$. The 'max' function in figure 2.7 is used to assure the pulse period to be equal to or bigger than one sample time.
The current time is counted by a discrete time integrator ('counter' in figure 2.7). Each sampling time the current time is being compared to the scheduled period time of the pulse. Now two cases may occur. If the current time is smaller than the scheduled pulse period (see figure 2.11), the counting proceeds and no pulse will be generated. But if the current time is bigger than or equal to the scheduled pulse period, a logical one is send to the digital output of the control board, which is connected to the step clock input of the driver. This means the motor advances one step. After the pulse is generated, the counter is reset (after one sample time of delay) and the process repeats.

![Figure 2.11: Time base](image)

For $K_s$ bigger than or equal to two, the smallest possible pulse period equals $K_s^*T_s$ (where $T_s$ is the sampling time). So the maximum pulse frequency will be $1 / (K_s^*T_s)$ and the maximum angular wheel velocity equals:

$$\omega_{\text{max}} = \frac{2\pi}{q \cdot T_s \cdot K_s}$$  \hspace{1cm} (2.3)
Chapter 3  The control problem

In this chapter the control of the two dimensional two-wheel robot cart is considered. In section 3.1 the mathematical model of the robot is introduced. Section 3.2 considers the stabilization and tracking problem.

3.1 Mathematical model

Consider the mobile robot, schematically drawn in figure 3.1. The front wheel is free to rotate (castor wheel) without any restrictions and may therefore be considered as a gliding point or a rolling ball. Two independent forces can act on the rear wheels. If we assume that there is no slip between the wheels and the ground, this implies that no sideways movement is possible and the robot can only move in $\theta$ direction.

This results in the following nonholonomic constraint on the velocities:

$$\dot{x} \cdot \sin(\theta) - \dot{y} \cdot \cos(\theta) = 0 \quad (3.1)$$

The system can move to any point in the state-space, but because of the nonholonomic constraint only by restricted trajectories.

Under previous conditions the kinematical model (in terms of velocities) can be formulated by the following set of differential equations:

$$\begin{cases} 
\dot{x} = v \cdot \cos(\theta) \\
\dot{y} = v \cdot \sin(\theta) \\
\dot{\theta} = \omega
\end{cases} \quad (3.2)$$

where $v$ is the forward velocity and $\omega$ the steering velocity. $(X,Y)$ is the position of the robots' center of gravity and $\theta$ denotes its heading angle from the horizontal axis. Normally the velocities $v$ and $\omega$ are subject to the following constraints:

$$|\omega| \leq \omega_{\text{max}}, \quad |v| \leq v_{\text{max}}$$
If we take $u = [u_1 \ u_2] = [v \ w]$ and $x = [x_1 \ x_2 \ x_3] = [x \ y \ \theta]$ equations 3.2 can be rewritten as:

\[
\begin{align*}
\dot{x}_1 &= u_1 \cdot \cos(x_3) \\
\dot{x}_2 &= u_1 \cdot \sin(x_3) \\
\dot{x}_3 &= u_2
\end{align*}
\]  

(3.3)

The forward and the rotational velocity, $u_1$ and $u_2$, are not the actual inputs of the system. By sending a pulse signal to the motors, we manipulate the angular velocity of the left and the right wheel, $w_1$ and $w_2$ respectively (see also figure 3.2). In literature [1] the relation between the 'w' and 'u' is defined as $u = M \cdot w$ according to:

\[
\begin{pmatrix}
u_1 \\
u_2
\end{pmatrix} =
\begin{pmatrix}
r & r \\
r & r \\
2R & 2R
\end{pmatrix}
\begin{pmatrix}
w_1 \\
w_2
\end{pmatrix}
\]  

(3.4)

Figure 3.2: Velocities
3.2 Stabilization

In this section the stabilization problem is considered. An example of a stabilizing controller (available from literature) is presented and simulated.

In figure 3.3 the basic closed loop control configuration of a nonlinear system is shown.

![Figure 3.3: Stabilization](image)

The classical way to stabilize the nonlinear system is by choosing a state feedback control law, i.e. of the form

\[ u(t) = k(x(t)) \]  \hspace{1cm} (3.6)

in such a way as to make the origin of (3.3) an asymptotically stable equilibrium point of the system

\[ \dot{x} = f(x, k(x)) \]  \hspace{1cm} (3.7)

Earlier research (Brocket, 1983) showed that when a system is nonholonomic constraint, like our model (3.3), there exists no such a smooth state feedback law like (3.6) which makes a specified equilibrium of the closed loop locally asymptotically stable. The solution to this problem is found in using time-varying controllers, i.e. of the form

\[ u = u(x, t) \]  \hspace{1cm} (3.8)

In the following text we will introduce such a controller which is used to stabilize our control system. Also some simulation results are presented, which can be used as a reference for testing the controllers on the physical system.
Before presenting the feedback laws, the kinematical equations (3.3) are converted into a special normal form, called ‘chained form’, by means of a coordinate change. Define the new coordinates as

\[
\begin{align*}
    z_1 &= x_3 \\
    z_2 &= \sin(x_3)x_1 - \cos(x_3)x_2 \\
    z_3 &= \cos(x_3)x_1 + \sin(x_3)x_2
\end{align*}
\] (3.9)

Since the coordinate change in (3.9) is a global diffeomorphism, the feedbacks are defined over a large domain of the state space. One can confirm that the vector fields in these coordinates have the form:

\[
\begin{align*}
    \dot{z}_1 &= v_1 \\
    \dot{z}_2 &= v_1z_3 \\
    \dot{z}_3 &= v_2
\end{align*}
\] (3.10)

Where:

\[
\begin{align*}
    v_1 &= u_2 \\
    v_2 &= u_1 - u_2(\sin(x_3)x_1 - \cos(x_3)x_2)
\end{align*}
\] (3.11)

The following time varying controller [2] is used to stabilize the system:

\[
\begin{align*}
    v_1 &= -z_1 + |z_2|^{0.5}\sin(t) \\
    v_2 &= -z_3 - \text{sign}(z_2)|z_2|^{0.5}\cos(t)
\end{align*}
\] (3.12)
The behavior of this controller is simulated in Simulink, using the model shown in figure 3.4. (The block 'state reconstruction' contains the model (3.3) of the robot).

![Diagram](image)

**Figure 3.4:** Simulink model for stabilization

The simulation results of stabilizing the system from the initial condition \( x_0 = [-0.5 \ 0.5 \ \pi/10] \) are plotted in figure 3.5. It shows the convergence of the state variables \( x, y \) and \( \theta \) in time. It must be remarked that there are no constraints on the controller inputs.

![Graph](image)

**Figure 3.5:** Stabilization from initial condition \([-0.5 \ 0.5 \ \pi/10]\)
3.3 Tracking

In this section we address the tracking problem. As in the previous section an example of an excising tracking controller is presented and simulated. Figure 3.6 gives an schematic overview of the tracking problem.

Figure 3.6: Tracking

Assume that feasible reference dynamics \( (x_r, y_r, \theta_r, v_r, w_r)^T \) is given, i.e., dynamics that satisfies

\[
\begin{align*}
\dot{x}_r &= v_r \cos(\theta_r) \\
\dot{y}_r &= v_r \sin(\theta_r) \\
\dot{\theta}_r &= \omega_r
\end{align*}
\] (3.13)

Where \( w_r \) and \( v_r \) are bounded reference velocities. Now the tracking problem boils down to finding time-varying state-feedback controllers of the form

\[
\begin{align*}
w &= w(t, \theta, x, y) \\
v &= v(t, \theta, x, y)
\end{align*}
\]

such that \( x(t) - x_r(t), y(t) - y_r(t) \) and \( \theta(t) - \theta_r(t) \) tend to zero as \( t \to +\infty \)

For solving this control problem the following global change of coordinates is used (Kanayama et al., 1990):

\[
\begin{bmatrix}
x_e \\
y_e \\
\theta_e
\end{bmatrix} =
\begin{bmatrix}
\cos(\theta) & \sin(\theta) & 0 \\
-\sin(\theta) & \cos(\theta) & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x_r - x \\
y_r - y \\
\theta_r - \theta
\end{bmatrix}
\] (3.14)
This global change of coordinates from $[x_i - x, y_i - y, \theta_i - \theta]^T$ to $[x_e, y_e, \theta_e]^T$ makes the error variables become independent from the choice of the inertial coordinate frame; the errors are considered in a frame attached to the mobile robot. In these new coordinates the tracking error dynamics becomes:

$$
\begin{align*}
\dot{x}_e &= \omega y_e - v + v_y(t) \cos(\theta_e) \\
\dot{y}_e &= -\omega x_e + v_y(t) \sin(\theta_e) \\
\dot{\theta}_e &= \omega_r(t) - \omega
\end{align*}
$$

(3.15)

The objective is to find appropriate control laws for $v$ and $w$ such that the tracking error $(x_e, y_e, \theta_e)^T$ converges to zero.

Jiang and Nijmeijer (1997) remarked that by means of Lyapunov theory the control law

$$
\begin{align*}
v &= v_r(t) \cos(\theta_e) + k_1 x_e \\
w &= w_r(t) + v_y(t) \frac{\sin(\theta_e)}{\theta_e} + k_2 \theta_e
\end{align*}
$$

(3.16)

yields global asymptotic stability (GAS) of the tracking error system (3.15), provided that $v_r(t)$ or $w_r(t)$ does not converge to zero.
The tracking controller (3.16) is simulated, using the model presented in figure 3.7. Again, the reconstruction block contains the kinematical model of the robot (3.1)

Figure 3.7: Simulink model for tracking

Figure 2.3 shows the simulation results of tracking a circle, starting from initial condition $x_0 = [2 -2 \pi/10]$.

Figure 3.5: Tracking error of tracking a circle from initial condition $[2 -2 \pi/10]$
Chapter 4  ‘Reconstructing’ the state

In the previous chapter the stabilizing and tracking problem was introduced. The outputs of the control laws, in terms of velocities, are computed based on the position and orientation of the system. This means to evaluate the controllers we need to know the complete state vector at any time. In previous simulations, they were calculated, using the differential equations of the robot (equations 3.2 and 3.3). This chapter considers the problem of obtaining the state variables, when dealing with the physical system.

4.1 Measuring and observers

One way to obtain the state would be to measure all of its components. In our case this would mean that the position and the orientation have to be measured.

Another way is to use a non-linear observer (i.e. Kalman filter), in combination with the model equations (3.2 or 3.3). Under certain conditions, with such an observer it is possible to reconstruct the complete n-dimensional state vector, only measuring n-p outputs.

In both cases we would have to invest in measuring devices. In the case of using stepping motors it is possible to determine the position and orientation of the system by knowing the steps applied to the motors. This is done by integrating the differential equations of the system (figure 4.1). This procedure, which will be explained in section 4.2, will (erroneously) be referred to as a ‘reconstruction’ of the state.

Figure 4.1: Reconstructing the state
4.2 Integrating the system equations

By integrating the system equations (3.3) we obtain the expressions for the state components as a function of time:

\[
\begin{align*}
    x_1(t) &= x_{10} + \int_0^t \cos(x_3)u_1 d\tau \\
    x_2(t) &= x_{20} + \int_0^t \sin(x_3)u_1 d\tau \\
    x_3(t) &= x_{30} + \int_0^t u_2 d\tau
\end{align*}
\]  

(4.1)

When we assume the value of \( x_3 \) to be constant during each integration step, we only need to calculate the integrals \( \int_0^t u_1 d\tau \) and \( \int_0^t u_2 d\tau \) to obtain \( x(t) \). Where the integral of \( u_1 \) corresponds to the forward displacement and the integral of \( u_2 \) is equal to the rotation of the center of gravity. For we know the number of steps applied to each motor during a sample interval, the angular displacement of the left and the right wheel is known and equal to \( \frac{2\pi}{q} N \), where \( N \) is the number of steps in each sample time interval for the corresponding wheel. Using the transformation matrix \( M \) (see equation 3.4) we can transform these displacements to the forward displacement and the rotation of the center of gravity. So we can calculate the integrals of \( u_1 \) and \( u_2 \) by measuring the number of steps applied to each motor within a sample time interval. In this way it’s possible to obtain the state by integrating the system equations, without directly measuring its components. The Simulink model to implement this integration scheme (from this point on we will call it ‘reconstruction’ scheme) is presented in Appendix 3.

Using the pulse generator described in section 2.3 the maximum number of pulses within a sample time interval will be limited to one. In this case the number of steps will be counted directly from the output of the pulse generator. When the frequency converter cards are available, it will be possible to have more step counts within a sampling time interval. In that case the counting will be done using the encoder inputs of the dSpace board. So using the cards will not only provide a higher output frequency, but it also increases accuracy of the state reconstruction.
Chapter 5  Validation of the reconstruction scheme

To validate the performance of the reconstruction it was tested in Simulink. In section 5.1 we compare the output of the reconstruction scheme to the output of the model, applying the same input to both. In section 5.2 and 5.3 we repeat the simulations of the stabilizing and tracking controllers respectively, but in both cases we replace the model of the robot by the reconstruction scheme in order to examine the difference.

5.1 Same input to model and reconstruction

In this case a constant and a sinusoidal input were applied to both the model and the reconstruction, to get a first insight in the performance of the reconstruction. In figure 5.1 this principle is shown.

Reconstructing a constant input

\[ u_1 \quad \begin{array}{l} \text{Reconstruction} \\ \text{(integrating 3.2)} \end{array} + \quad \begin{array}{l} \text{System model} \\ \text{(equations 3.2)} \end{array} \]

**Figure 5.1:** Reconstruction error

**Figure 5.2:** Reconstruction of ‘x’ using constant inputs \( u_1 \) and \( u_2 \)
Figure 5.3: Reconstruction of 'y' using constant inputs $u_1$ and $u_2$

Figure 5.4: Reconstruction of 'y' using constant inputs $u_1$ and $u_2$
Reconstructing a sinusoidal input

\( u_1 \)
Sine wave
Amplitude: 0.5
Frequency: 1 Hz

\( u_2 \)
Sine wave
Amplitude: 0.05
Frequency: 0.2 Hz

Figure 5.5: Reconstruction of 'x' (left) and 'y' (right) using sinusoidal inputs

Figure 5.6: Reconstruction of 'θ' using sinusoidal inputs
5.2 Stabilization

In this case the stabilization from an initial condition is simulated, using the reconstruction scheme, instead of the model of the robot, to obtain the state vector. To make a comparison with the results of section 3.2 the same controllers were used and again the initial condition is [-0.5 0.5 pi/10]. Figure 5.8 shows the results of stabilization from this initial condition (left) and the difference in the state components between the model and the reconstruction (right).

Figure 5.8: Stabilization using the reconstruction and the difference between the model and the reconstruction.
5.3 Tracking

In this section tracking of specified trajectories is considered. The tracking controller introduced in section 3.3 is used. The difference is the fact that in this case the model of the robot is being replaced by the reconstruction scheme.

Tracking a circle

The same initial condition is used as in section 3.3, to make the comparison between reconstruction and model. Figure 5.9 shows the simulation results of tracking a circle from $x_0 = [2 \ -2 \pi/10]$. When we compare these results to figure 3.5, it can be seen, that the results between the model and the reconstruction are equal.

![Figure 5.9: Trajectory 1: movement in xy-plane (left) and tracking error (right)](image)

Trajectory 2: Tracking straight line

![Figure 5.10: Trajectory 2: movement in xy-plane (left) and tracking error (right)](image)
Control of a mobile robot

Trajectory 3: Moving along straight line with sinusoidal component in angle

Figure 5.11: Trajectory 2: movement in xy-plane (left) and tracking error (right)

5.4 Real time experiments

It must be pointed out that the kind of integration used to reconstruct the state only works fine if the behavior of the physical system is accurately described by the integrated model. Therefore the tracking and stabilizing controllers presented before have been tested in real time experiments using the reconstruction scheme.

Because of the limited time of this traineeship it was not possible to accurately measure the performance, but on first sight it seemed to be working pretty good. There are deviations from the desired trajectories, but probably these are mainly due to the difference between the radii of the wheels. So better performance is obtained after calibration of the robot.
Chapter 6 Conclusion and recommendations

A two-wheeled mobile robot cart, driven by stepping motors, is constructed. It can be controlled by Matlab / Simulink, using a dSpace control board.

The input to the motors is generated using a Simulink model. This causes the maximum speed of the motors to be limited by the maximum sampling time of the PC. Because very high frequency pulses are required, only low operating speeds are allowed. The scaling factor, used to obtain linear input-output behavior of the pulse generator limits the maximum speed even more.

The inputs to the system, in terms of velocities, computed by the control law, are based on the orientation of the system. So to evaluate the controllers the x, y and θ coordinate of the robot have to be available. In this traineeship the system equations are integrated to obtain these state components. To do so, the forward and angular displacement of the wheels have to be known. They were reconstructed by measuring the number of steps applied to each of the motors.

Simulations and real time experiments with different kind of controllers proved that this 'reconstruction' scheme works fine. In real time experiments there were some small differences between the desired and measured behavior, but they are mainly due to differences in the wheel radii. So a big part of this problem can be solved by proper calibration.

The first step in future projects would be to use the frequency converter cards to generate the input to the stepping motors. Compared to the Simulink model it will have some major advantages:

- Higher operating speeds of the robot, because the sampling time will not be a bottleneck anymore.
- Improvement of the accuracy of the integration scheme that is used to reconstruct the state. This is because it is possible to have more than one step count within each sampling interval when the cards are used. In this case the steps could be counted by the encoder inputs of the dSpace board.
References

*Commande de vehicules non-holonomes pour le suivi de trajectoire et la stabilization a une posture desiree*
September, 1993

*Contributions s la stabilization de systemes non-lineaires a la commande de vehicules sur roues*
April, 2000

*Paper: ‘Experiments in exponential stabilization of a mobile robot towing a trailer’.*
June, 1994

*Paper: ‘Exponential stabilization of driftless nonlinear control systems via time-varying, homogeneous feedback’.*
December, 1994

*Paper: ‘Exponential stabilization of driftless nonlinear control systems via time-varying, homogeneous feedback’.*
May, 1997

*Paper: ‘Stabilization and tracking of a nonholonomic mobile robot with saturating actuators’.*

*Paper: ‘Tracking control of a wheeled mobile robot by Lyapunov’s direct method’.*
December, 1996

*Nonlinear systems*
Berkeley, 1999

[9] E. Lefeber
*Proefschrift ‘Tracking control of nonlinear mechanical systems’*
Enschede, 2000
Appendix 1: Microstepping driver and Interface board: specifications
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

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

**(3.5) screw with a .250 (6.3) O.D. flat washer and a split lock washer. This hardware is not supplied.

PIN ASSIGNMENT AND DESCRIPTIONS

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

ELECTRICAL SPECIFICATIONS

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>MIN</th>
<th>TYP</th>
<th>MAX</th>
<th>UNITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opto Supply</td>
<td>Isolated Inputs</td>
<td>5</td>
<td>40</td>
<td>V</td>
</tr>
<tr>
<td>Input Forward Current</td>
<td>Isolated Inputs</td>
<td>8</td>
<td>10</td>
<td>12</td>
</tr>
<tr>
<td>Opto Input Forward Voltage</td>
<td>Isolated Inputs</td>
<td>1.5</td>
<td>1.7</td>
<td>V</td>
</tr>
<tr>
<td>Reverse Breakdown Voltage</td>
<td>Isolated Inputs</td>
<td>5</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>Signal Output Current</td>
<td>Full Step, Fault</td>
<td>25</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>Drain-Source Voltage</td>
<td>Full Step, Fault</td>
<td>100</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>Drain-Source Resistance</td>
<td>Full Step, Fault</td>
<td>Ios = 25mA</td>
<td>6.5</td>
<td>Ω</td>
</tr>
</tbody>
</table>

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.
ISOLATED INPUTS

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

JUMPERS

JP1: 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.

JP2: 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. NOTE: In this mode the current reduction resistor MUST NOT be used or it will cause erratic operation of the driver. If the shunt is placed on the "ENOFP" 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.

LED'S

The green LED is controlled by the on board +5vdc power supply.
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.

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.

For additional trouble shooting information refer to the drivers operating instructions.

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 IM481Hs fault output signal through an open drain N-channel FET. The signal at the terminal strip is inverted and is active low.

In the case of an over temperature fault, neither the red LED of the fault output becomes activated. The IM481Hs motor outputs will disable. They will not re-enable until the drive cools to a safe operating level.

FULL STEP OUTPUT

The INT-481 buffers the IM481Hs full step output through an open drain N-channel FET. The signal available at the terminal strip is inverted and is active low.

MICROSTEP RESOLUTION SELECTION

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.

<table>
<thead>
<tr>
<th>RESOLUTION</th>
<th>STEPS/REV</th>
<th>SWITCH 1</th>
<th>SWITCH 2</th>
<th>SWITCH 3</th>
<th>SWITCH 4</th>
</tr>
</thead>
<tbody>
<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>OFF</td>
<td>OFF</td>
<td>ON</td>
</tr>
<tr>
<td>250</td>
<td>50,000</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DECIMAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>16</td>
</tr>
<tr>
<td>32</td>
</tr>
<tr>
<td>64</td>
</tr>
<tr>
<td>128</td>
</tr>
<tr>
<td>256</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BINARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON</td>
</tr>
</tbody>
</table>

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 2: Frequency converter card(s): specifications
Frequency signal conditioners
- frequency outputs
- LED-indicators
- switchable frequency output

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

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>DIP switch</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0...18 kHz</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0...8 kHz</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0...4 kHz</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0...1 kHz</td>
</tr>
</tbody>
</table>

Ordering data

Technical data
- Input ranges
- Overload limits, input
- Input importance
- Voltage across, input
- Output
- Output frequency, final value
- Frequency setting
- Resettlement range
- Output level
- Output current
- Decay
- Substrate voltage
- Current consumption
- Mating current limit
- Polarization protection
- Precision
- Temperature coefficient
- Isolation coordinates acc. to EN 60178
- Voltage proof input/output
- Rated voltage
- Rated current
- Overvoltage category
- Voltage proof, input/output, mounting rail
- Operating temperature
- Storage temperature
- Total width
- Conductor cross-section

<table>
<thead>
<tr>
<th>Type</th>
<th>Cat. No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCZ VFC</td>
<td>846147</td>
</tr>
<tr>
<td>MCZ CFC</td>
<td>846148</td>
</tr>
</tbody>
</table>

MCZ VFC
- Nominal output: 0...10 V
- Overload limits: 20 V
- Voltage proof: 200 V
- Storage temperature: -10°C..+65°C
- Total width: 6 mm
- Conductor cross-section: 1.5 mm²

MCZ CFC
- Nominal output: 0...20 mA
- Overload limits: 50 mA
- Voltage proof: 150 V
- Storage temperature: -25°C..+65°C
- Total width: 6 mm
- Conductor cross-section: 1.5 mm²

* without DC/DC-Conditioner
input loop powered
Appendix 3: Simulink implementation of the integration scheme
The following scheme shows the implementation of equations 4.1 in Simulink:

![Simulink diagram](image)

**Figure A1:** Simulink implementation of the integration scheme reconstruction

Elucidation of the components in figure A1:

\[
\begin{align*}
VCO\_out &= \begin{bmatrix} step\_left & dir\_left & step\_right & dir\_right \end{bmatrix}^T \\
fcn.1 &= \frac{-2\pi}{qTime\cdot q} \cdot (u[1] \cdot 2(u[2] - 0.5)) \\
fcn.2 &= \frac{2\pi}{qTime\cdot q} \cdot (u[3] \cdot 2(u[4] - 0.5)) \\
fcn.3 &= u[1] \cdot \cos(u[3]) \\
fcn.4 &= u[1] \cdot \sin(u[3]) \\
fcn.5 &= u[2]
\end{align*}
\]