Department of Control and Engineering design
Technical University of Denmark

Mobile Robot for Weeding

Tommy Ertbølle Madsen & Hans Lavdal Jakobsen

Master Thesis Project
January 31st 2001

Supervisor: Associate professor Ph.d. Torben Sørensen

This project is made by

Tommy Ertbølle Madsen
Student ID: c948429
tommy@ertboelle.dk

Mechanical and control engineering.
Has participated in the development of the BinderWrap™ round bale wrapper.
www.binderwrap.dk

Hans Lavdal Jakobsen
Student ID: c928543
hans@miditec.com

Electro and control engineering.
Has developed the musicians software EarMaster™ in his company MidiTec.
www.earmaster.com

Abstract

The goal with this master thesis has been to develop an autonomous agricultural vehicle. The decisions regarding the design of mechanics, electrics and software are discussed and the experimental results are presented.

The environmental impact of agricultural production is very much in focus, while the competition demands high efficiency. Some years ago, weeding was done manually without the use of pesticides. With the development of an autonomous agricultural vehicle with sensors for weed detection, it will again be possible to avoid the use of pesticides.

The vehicle uses high precision GPS (RTK), encoders, compass, and tilt sensors, to position itself and follow waypoints.

We have developed a module based software system and identified 12 areas of the system (modules) that have each been enclosed in separate dll's with well defined interfaces. The modules include interfaces to sensors, low-level and high-level controllers and more. By using hardware timers, it has been possible to develop a real-time control system using Windows98.

A vehicle suitable for research under field conditions has been developed. The vehicle has 4-WD and 4-WS, which makes it possible to test different steering strategies. The vehicle is specially designed for in-row driving; this has been achieved with the use of in-wheel motors. The vehicle can drive in row crops with a row distance down to about 250 mm and with 500 mm high crops.

Synopsis

Målet med dette speciale har været at udvikle et autonomt landbrugskøretøj. Beslutningerne omkring udviklingen af mekanik, elektronik og software er beskrevet og resultaterne er fremlagt og diskuteret.

Landbrugsproduktionens miljøbelastning er meget i fokus i dag og samtidig øges kravene betydeligt til erhvervets konkurrenceevne. For år tilbage blev ukrudtet i marken fjernet manuelt uden brug af pesticider. Med udviklingen af et autonomt landbrugskøretøj med sensorer for detektering af ukrudt vil det igen blive muligt at undgå brugen af pesticider.

Køretøjet er udstyret med højpræcisions GPS (RTK), enkodere, kompas og tiltsensorer for at kunne positionere sig selv og følge en specificeret rute.

Vi har udviklet et modulbaseret softwaresystem og identificeret 12 dele af systemet (moduler), som hver er placeret i en dll med en veldefineret grænseflade. Modulerne består af drivere til sensorer, lavniveau og højniveau regulatorer mm.
Ved at bruge hardware timere har det været muligt at udvikle et realtidskontrolsystem under operativsystemet Windows98.

Der er udviklet et køretøj specielt til forskning under mark vilkår. Køretøjet har firehjulstræk og styring på alle fire hjul, hvilket gør det muligt at undersøge forskellige styringsstrategier som f.eks. for- eller baghjulsstyring. Konstruktionen er lavet specielt med henblik på at kunne køre mellem afgrøderækkerne, dette er opnået ved brugen af navmotorer. Køretøjet kan køre i afgrøder med ned til ca. 250 mm mellem rækkerne og i ca. 500 mm høje afgrøder.

Preface

In this project we have focused on:
- Making a vehicle with optimum design for agricultural use: cross-country, large ground clearance and slim wheel and transmission design for in-row driving.
- Making a flexible hardware and software design. The vehicle is designed for further experiments and not for commercial use. The design opens up possibility for testing sensors, new weeding technologies and high precision in-row field steering.
- Getting experimental results with RTK/GPS, because we had the chance to borrow the (so far expensive) equipment. RTK/GPS is a promising new technology for high precision agriculture.

During the development of mechanics, electronics and software we have focused on making the vehicle flexible. The choice of technology for the various functions has big impact on functionality, flexibility and performance.
During the development process we have had safety in the back of our minds and put an effort in eliminating or reducing the risks concerning an autonomous vehicle.

We have been under time pressure all the way; just to get the vehicle driving. This has forced us to cut corners in some areas, especially the control engineering area.

Thanks to:

Associate Professor Torben Sørensen for your help with robotics and for not setting limits for our goal.
Professor Christian Boe for your help with design methods and mechanical reviews.
Associate Professor Nils Andersen for letting us draw on your experiences with autonomous vehicles.
Deputy head of department Svend Christensen, head of research unit Martin Heide-Jørgensen and senior scientist Henning Tangen Søgaard, Department of Agricultural Engineering for ideas and encouragement.
Ph.d. student Anna B.O. Jensen, National Survey and Cadastre for borrowing us the GPS equipment and giving valuable support.
MSc Henrik Holm, Department of Product Development for help with electronics.
Thomas Thriges foundation for economical support.

Our wives Karen and Birgitte for support and understanding (!)

1. Introduction

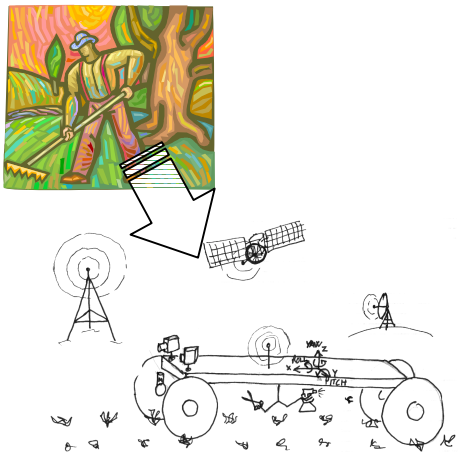


Figure 1.1 Precision farming

1.1. Project description

This project is based on ideas from Svend Christensen, DJF, proposed at the conference "Technology in the Primary Agriculture" 1999 arranged by the Danish Agricultural and Veterinary Research Council [HIGHTECH].

Motivation

Today the environmental impact of agricultural production is very much in focus and the demands to the industry is increasing. The production of agricultural products is growing and the competition is getting bigger, therefore the farmer has to be very efficient to be able to compete. At the same time the demands, for less use of pesticides and fertilizers, from the consumer and the legislation are increasing. Therefore the farmers have to use high technology in the fields for weeding, spraying, etc., earlier weeding were done manually but today it is neither profitable or possible to get a sufficient number of labourer for this job.
In this project, an autonomous agricultural vehicle, for test and development of sensors, tools and information technology in the field, is going to be developed.

Problem statement

Development and construction of an autonomous robot for weed control in row crops e.g. sugar beets or maize. The robot can be divided into four main modules:

1. A vehicle as a platform for carrying e.g. weeding tools for in-row weeding. The vehicle could be equipped with the control modules described below.
2. A control unit, with input from Vision, GPS, and other necessary sensors, are providing the vehicle and the tools with the necessary control signals.
3. A GPS module that provides the vehicle with its global position in real time.
4. A vision system detecting the position of the crop relative to the vehicle position.

The main goal of this project is the development of the vehicle and the control unit, with possibility of using different sensor technologies. We want to test the vehicle and control unit and show that the robot is capable of following a path under field conditions.

The vehicle

The autonomous vehicle should be developed to test different sensor systems and tools in the field. The vehicle should be able to work under field conditions, where the vehicle can be disturbed by irregularities, when following a specified path.

The control unit

The control software is intended to be executed on a PC physically attached to the vehicle. Interface cards, the serial and parallel ports will be used for communication with the peripherals.
The control software is going to be developed with a high level programming language and is going to work on a Windows operating system. This is chosen because of our expertise in this area, the flexibility, and the large number of available interface cards with drivers.
The control of motors and the higher-level path control are intended to be solved with classical methods as e.g. PID control.

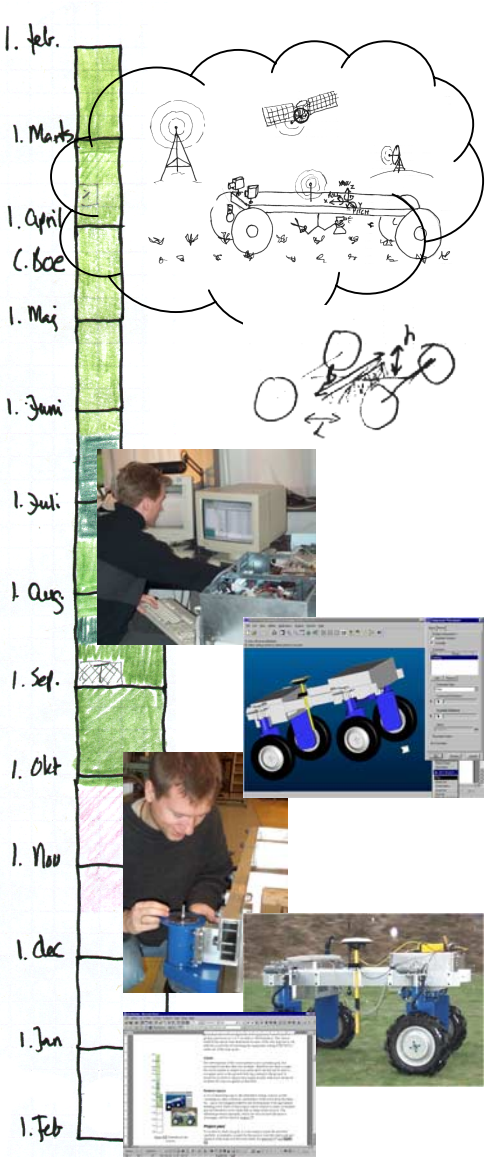


Figure 1.2 Timetable for the project.

The GPS module

From the National Surveys and Cadastre, Denmark we were offered the possibility of borrowing RTK/GPS equipment, which is able of giving a precision on 1 to 5 cm with a 5 Hz frequency. This sensor method had earlier been abandoned because of the very high price, but with the possibility of borrowing the equipment, testing RTK/GPS is made one of the main goals.

Vision

The development of the vision system is not a primary goal. Therefore, we want to make the vision system simple and it should just be able to recognize spots on the ground with good contrast to the ground. It should be possible to replace this simple module with more advanced modules for crop recognition in the field.

Related topics

Many interesting topics, like alternative energy sources, power consumption, data collection, surveillance of the robot from the farm, etc., can be investigated related to the development of an agricultural weeding robot. Each of these topics can be subject to years of research and will therefore not be dealt with in detail in this project. The different problems and tasks, which we will not have the time to investigate, will be listed in “Recommendations”, page 151.

Project plan

To be able to reach our goal, it is necessary to plan the activities carefully. The time for the project is 3/4 of a year distributed over one year. The first 6 months are on half time and the last 6 months on full time. A timetable is made for the project, were the start points and durations of the main activities are listed, see appendix 1.1.1 and Figure 1.2.
The analysis stage started with brainstorming, literature study and meetings with people, who have know-how about and interest in the topic for this project. Then the project statement is detailed and a specification for the robot were made and the synthesis is started. The search for principle concepts is made parallel with the vision software. In August the first CAD drawings are made and the control architecture is determined. The workshop starts making the parts in September and the vehicle is ready for assembly in late October.
The first tests should have begun primo November. Unfortunately, the assembly and tests of the robot was delayed because we had

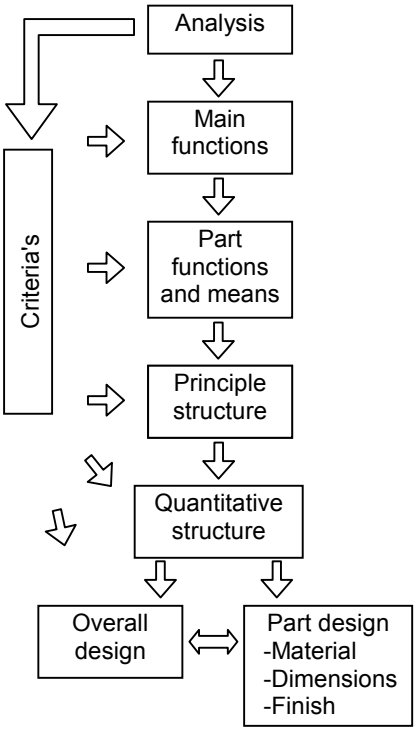


Figure 1.3 Product synthesis model [TJALVE]

underestimated the time necessary in the workshop. Therefore, more of the documentation has been made parallel with the development of the robot and the first outdoor tests did not begin until primo January. Because the test started so late, we have not had time for optimising the controllers, which we had chosen.

Means (Tools and resources)

Tools

In the project we have mainly used methods for systematically design of industrial products (Figure 1.3) and classical control methods lectured at IKS.
Most of the design has been done with help from different computer tools. The software is developed in Delphi and the mechanical design is detailed with Pro-Engineer. For calculations Matlab, Excel and Win-Beam have been used.

Resources

From the start, we had identified different departments and people, who could help us with know how. The Department of Control and Engineering Design (IKS), the IKS-workshop, the Department of Agricultural Engineering (DJF), Department of Automation (IAU), the Department for Product Development (IPU), National Cadastre and Survey (KMS), see appendix 1.1.2.
The budget for master projects is very limited, but because this project can be used for further research, it has been possible to get the necessary economical resources. The robot is the property of IKS, who has paid most of the robot, in addition Thomas B. Thriges Foundation has given 22.000,- kr to the project (Appendix 1.2.2-3).

Figure 1.4 Advanced Tool Control from Eco-Dan (between the tractor and the cultivator)

1.2. State-of-the-art

This paragraph will describe the current technological level for precision farming in row crops. The content is based on a literature study (appendix 1.2.1.d and literature list), internet study and information from DJF.

State-of-the-art commercial products

So far autonomous in-row vehicles for weed control have not been commercialised, but control systems for controlling the implements towed by the tractor are available. Some products can also control the tractor in the row, but it still needs to be monitored carefully by the driver. The technology used in these commercial products is computer vision and ultrasonic sensors.

Vision and ultrasonic sensors

The first company to launch vision control is Eco-Dan A/S, who has developed a system called Advanced Tool Control (ATC), see Figure 1.4. The system consists of two dedicated vision cameras and a dedicated microprocessor for manipulating the pictures and generating the control signal for positioning the implement sideways. The system makes it possible for one man to manoeuvre a cultivator with a precision of +/- 3 cm with speeds up to 8 – 10 km/h in the field.

John Deere and Danfoss/Reichardt have developed a system using ultrasonic sensors. We do not know the precision of this system, but it is mainly for relieving the driver when following swaths and not for high precision control in row crops, see Figure 1.5.

Figure 1.5 John Deere Pilot System

DGPS in precision farming

The last few years DGPS with a precision of 1 – 2 m has become quite popular for yield monitoring. The concept is that the combine equipped with DGPS makes a yield map and this information together with soil analyses from the field helps the farmer to differentiate the amount of fertilizer used in the field. In this way the farmer can optimise the use of fertilizer. He also gets a better overview of his production, which can help him to make the right decisions about the use of his fields.

RTK/GPS

So far, high precision GPS like RTK has not been used commercially in the agricultural industry for controlling the position of vehicles or implements.
The development of a RTK/GPS reference system in Denmark is going very fast at the moment. Already now it is possible to subscribe to a service delivering the reference signals via the GSM net, then it is not necessary to set up a local base station (www.gpsnet.dk). This is a large step towards commercial agricultural use of high precision GPS.

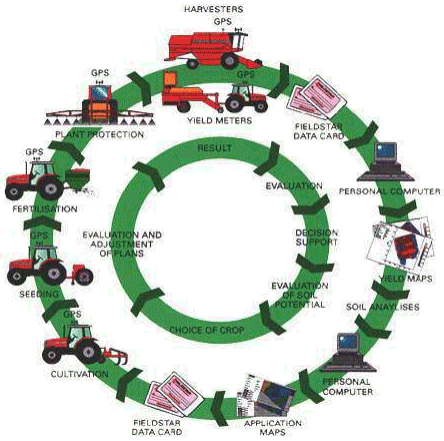


Figure 1.6 The Fieldstar™ precision farming concept.

Crop sensors

Hydro has developed a sensor to detect the amount of nitrogen in the crop. The sensor is an optical sensor that can measure the amount of chlorophyll in the crop, which gives a good estimate of the nitrogen level. The sensor is mounted on the roof of the tractor and connected to a computer, which controls the amount of fertilizer to be spread.

Research results

This paragraph only describes the most relevant research result we have found in relation to our project. An exhaustive list of the literature studied can be found in the literature list.

Vision

Test results from California research on spot spraying and identification of weeds and crops are not very convincing. 24.2% of the tomatoes were sprayed and 52.4% of the weeds were not sprayed [TOMATO]. It is very difficult to identify the different species and it will take years of research to make a satisfactory identification.

DGPS

At Stanford University an automatic tractor guidance system using carrier-phase differential GPS has been developed [GPSTRAC]. They use four GPS antennas on the roof of a tractor to measure both the position and attitude of the tractor. Very good results have been achieved in the field following non-linear trajectories with a controller using a priori information about the trajectory.

At the Institute of Agricultural and Environmental Engineering, IMAG-DLO, the Netherlands researchers have experimented with controlling the lateral movement of the implement using RTK/DGPS. The tests showed an average error of 1 cm at an average speed of 3.6 km/h [GPSIMP].

Off-road and agricultural vehicles

Analogies and comparable products can often provide useful information and give ideas for generating concepts. Therefore, other off-road and agricultural vehicles have been examined to see if this could give some help. In appendix 1.2.1.a-c some of the examined products can be seen.

1.3. Analyses and Product Specification

In this paragraph the different life stages of the product are identified and analysed and the demands, criteria's and comments are summarized in a product specification, appendix 1.3.5.a-b.
In the product specification solutions are delimited from not solutions (demands) and a yardstick for the solution is specified (criteria's). It is important to go through all life stages and remember all the stakeholders to specify the product right. The product specification is a management tool for controlling, that the right product is designed. On the other hand, the product specification should only set the relevant and necessary constraints for the product. The specification is not a static paper, but can be altered during the design process in dialogue with the relevant persons.
As the solution is designed in detail, it can be necessary to add additional demands and criteria's to the specification or make a sub specification. If e.g. a modular design is chosen, it can be necessary to make individual specifications for each module, because the other modules sets additional demands and criteria's to the design.

Figure 1.7 Identify the life stages of a product.

Identification of life stages

This product is only for research purposes and is therefore not going through all the life stages of a "normal" product. The life stages of this product are: design, industrial design, production, service, use, transport, destruction. Use is divided into: input, output, users, environment and safety.

Figure 1.8 Cornfield sown with double distance (250 mm) between the rows for mechanical weeding.

Background material for the Product Specification

The product specification is based on input from literature, DJF and our own knowledge about agriculture and particularly field conditions. Some of the demands and criteria's have been determined from the beginning (appendix 1.3.1-1.3.4) others have been identified but have not been answered until later in the project.
The paragraphs below are comments referring to the product specification in appendix 1.3.5.a-b.

Design

The 3/4-year available for this project is the main constrain on the design stage.

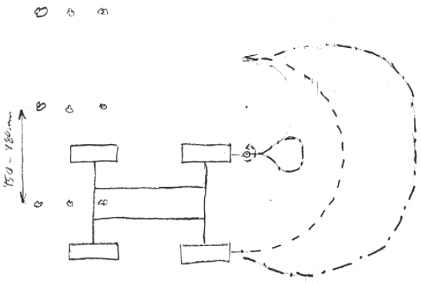


Figure 1.9 Very low turning radius.

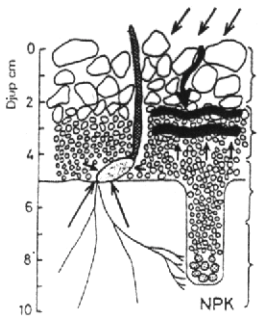


Figure 1.10 The ideal soil structure.

Industrial design

This is not a commercial product, which should appeal to a customer. However, future students could also be seen as customers and it is important that the robot will make students enthusiastic and wanting to work in this area. Therefore, the robot should appear as a functional, high tech and innovative product.

Production

This robot is a prototype for research and the budget is limited, therefore many processes and materials are not economical to use. The robot should be built at the IKS workshop, which is specialized in making prototypes and small batch production. The main qualifications are:

- High precision metalwork
- Mass reducing processes (several high precision CNC machines)
- Welding
- Press brake bending in sheet metal

Service

The robot should be easy to service and parts should be easy to replace. Students, researchers and technical staff at the institute will carry out the work.

Use

The robot is for research in methods for sustainable weed control and control engineering.
The vehicle should be able to operate at a speed about 0.5 – 1 m/s and drive for about 2 – 4 hours between recharging.
The precision of the vehicle must be within ±5 cm to drive without damaging crops in cornfields. In crops with bigger row distance the demand is not so high. But if the vehicle is driving precisely, then it will be easier to control the tool or maybe it will not even be necessary to control the tool relative to the vehicle.
The dimensions of the vehicle should make it capable of driving in row crops with 500 mm between the rows. The clearance between the vehicle chassis and the ground should be 500 mm to allow corn to pass below the vehicle.
The vehicle should be able to turn with a small turning radius on the headlands. It is not a demand that it should be able to turn as sharply as on Figure 1.9.

-Input

The vehicle should be able to use RTK/GPS position for steering in rows in e.g. cornfields or beet fields. It is also wanted to test the vehicle using vision.

-Output

The vehicle should be capable of carrying different tools for weed removal or data collection in the fields.

-Users

The users are primarily students and researchers with expert knowledge. It should be easy to access the different components and test different tools and sensors. The software should be modular, so that test of different controllers and sensors does not require changing all the software.

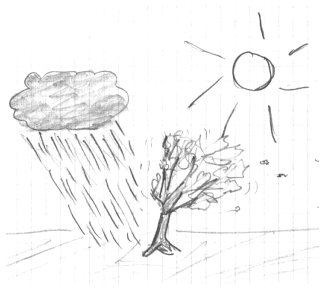


Figure 1.11 The vehicle will be exposed to different weather conditions.

-Environment

The vehicle should be able to operate in fields after sowing. The machine is made for Danish conditions and for test purposes, therefore the fields are assumed nearly level with maximum gradients on about 10%. The ideal field is shown on Figure 1.10, but the conditions can vary a lot.
The vehicle should be able to operate both in rainy and sunny weather. It is therefore important to enclose parts, which can be damaged by water and dust.

-Safety

It is very important, that the vehicle is safe to work with, so that people are not injured. Secondary it is important that the vehicle is not damaged or is causing damage to the surroundings.
There is no special rules concerning the use of mobile robots and since this robot is not for commercial use the general rules do not have to be meet [DEMKO]. We do however want to look at the general safety problems of autonomous vehicles and to make the developed vehicle as safe as possible.

Transport

It should be possible to transport the robot on a car trailer or in a van.

Destruction

Electronics should be removed disposed separately, but the disposal does not give rise to any special demands to the vehicle design.

2. The overall principal concept

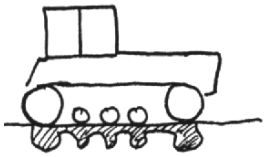


Figure 2.1 Ground pressure of tracked vehicle

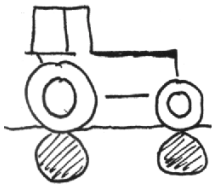


Figure 2.2 Ground pressure of wheeled vehicle.

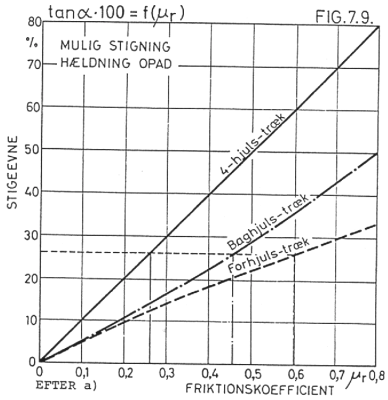


Figure 2.3 Climbing capability of vehicles with FWD, RWD and All-WD.

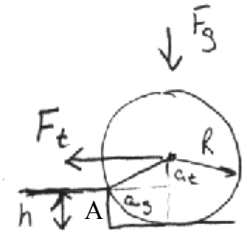


Figure 2.4 A towed wheel passing an obstacle.

This chapter will document the considerations made about the principles of the vehicle and the choice of concept for the mobile robot. The principles considered is: traction, steering, dimensions, power supply and control architecture.

2.1. Traction

The traction methods have big influence on the steering and terrain properties, cost and power consumption of the vehicle.

Wheels, tracks, feet or air cushion

Feet would not be feasible for this robot; hence, the technology is not mature.
An air cushion will damage the crop and raise a lot of dust, which will cover the crop and set very high demands to the dust tightness of the vehicle.
Tracked tractors are quite common on big farms with big fields and little need for road transportation, because of low ground pressure and very good drawbar pull performance on soft and loose soil. These properties are not so important for the mobile robot, where important properties are steering precision, low power consumption and low cost. To develop tracks for the robot would also be too big a task for this project.
Wheels are inexpensive and because of the little need for drawbar pull and load carrying capacity the bigger sinkage and ground pressure is of little importance. The soil conditions in fields after sowing are also normally good for wheeled vehicles and therefore wheels are chosen for traction (appendix 2.1.1).

FWD, RWD or All-WD

The choice of which wheels should be driving depends on the steering strategy, which will be discussed in following passage. In this passage, some general things about front, rear and all wheel drive will be addressed.
The diagram on Figure 2.3 shows the possible climbing versus the friction coefficients and it can be seen that 4-WD gives the best grip. This is because the whole weight of the vehicle creates traction between wheels and road. The second best is rear 2-WD, because a part of the weight on the front wheels is on the rear wheels, when driving up hill. [AB86]
On Figure 2.4 a towed wheel is shown being towed over an obstacle. The momentum equation around the point A shows that it is necessary with a big force F_t to lift the normal force on the wheel F_g. As it can be seen the force will be infinite, when the height of the obstacle is as high as the wheel radius. The maximum force for a driving wheel is equal to the force F_g, when the obstacle is higher than the wheel radius. However, if the obstacle is so high, the wheel will have no grip unless it is pushed against the obstacle. To get the best terrain

performance it is necessary with 4-WD, were all wheels helps the vehicle to overcome obstacles. (Appendix 2.1.3.c and 2.3.3.a)

Figure 2.5 Skid steer loader.

2.2. Steering

It is important for the vehicle to be able to steer very precisely in the crop rows and also to make as little damage as possible to the crop. In the following advantages and disadvantages of different steering strategies will be discussed.

Differential steering

The principle of differential steering is to make a difference between the speeds of the wheels on the left hand side compared to the speeds of the wheels on the right hand side. The difference in speed will make the vehicle turn. It makes a big difference to use this method on an all-wd vehicle compared to a 2WD vehicle, which will be discussed in the following paragraph.

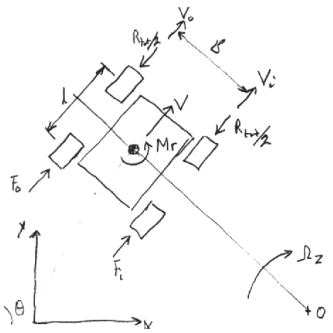


Figure 2.6 Principle of All-WD differential steering.

All-wd

Differential steering is often used for vehicles that should be very maneuverable in difficult terrain. A vehicle using this steering method can turn on the spot and the method is often seen on earth moving equipment like skid steer loaders Figure 2.5. The transmission is also simple to construct, because the wheels in one side is turning at the same speed and the position and orientation of the wheels are fixed. This indicates that it could be a possible solution for a weeding robot, but the method also has some major disadvantages. In a turn the vehicle also turns around a vertical axis through the center of the vehicle, which means that the wheels have to slide sideways. This uses a lot of power to turn the vehicle. The method is also unsuitable for dead reckoning because of the excessive slip, which makes odometer data useless. Damage to the crop is also likely to happen. The wheels will dig out viable weed seeds, which will germinate. Because of these disadvantages we have chosen not to use differential steering all-wd (skid steering). (Appendix 2.1.2.a-d)

Figure 2.7 Robot with 2WD differential steering.

2-wd

When using 2WD differential steering one or more castor wheels support the vehicle. A vehicle using this steering method can turn on the spot. The method is often used in robotics for e.g. AGV´s because of the very good maneuverability. But because of the castor wheels it is not suitable for off-road driving.
When the vehicle is turning, it has to drag the castor wheels. Even a small obstacle will demand a big force from the driving wheels, because of the momentum arm from the turning center to the castor wheel.
We have made some tests with a small trolley with castor wheels on a lawn and on paved ground, see appendix 2.1.3.d. The tests showed that it is necessary with a force 2 – 3 times bigger than the normal force from the driving wheels, to drag the castor wheels in a turn. This steering method is not chosen because of its poor off-road characteristics. (Appendix 2.1.3.a-d)

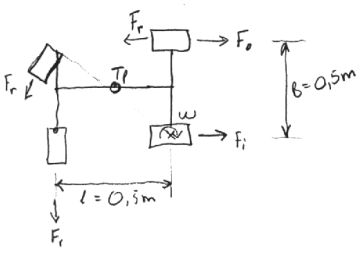


Figure 2.8 Principle of 2WD differential steering.

Figure 2.9 Tractor with articulated steering and the steering principle.

Articulated steering

The principle of articulated steering is to change the angle between the front axle and the rear axle of a vehicle. The disadvantage of this method is that there is little space for tools between the front and rear axle. Poor stability in turns can also be a problem, because of the transversal movement of the center of gravity. An advantage is the simplicity of the construction and good capability of following rows, see appendix 2.1.4.

Changing heading of wheel(s)

This is the principle that most road vehicles use for steering. When the forces on a vehicle are ignored, Ackerman describes the ideal relation between the angles of the outside and inside wheel in a turn. If the relation between the wheel angles follows the Ackerman equation, geometrically induced tire slippage will be eliminated. (Centripetal forces are ignored and therefore this only apply for very low turning speeds in practice) The low geometrically induced tire slippage is important to get good odometer data and steer precisely.
The Ackerman criterion is very difficult to implement mechanically, because it is highly non linear. Therefore on most vehicles the steering angles only approximate the Ackerman criterion for small steering angles. However, deviations from Ackerman do not necessarily mean, that the steering geometry is not optimal under real driving conditions at higher speeds.
If the vehicle is not all-wd, it has the same problems as the model with castor wheels. It is necessary with big forces to turn, when a towed wheel hits an obstacle. If the wheels, which are used for steering, are towed, they can have problems with getting a grip in loose soil for turning the vehicle. This will give wheel slippage a low steering precision.
If the vehicle has all-wd it will have very good off-road properties. The vehicle can steer on just one axle or all axles. If the weeding robot can steer on all axles it will be possible to obtain a very small turning radius. The wheels will in a stationary turn follow the same tracks if the steering of the rear and front wheels are symmetric about the middle of the vehicle (Figure 2.11). Therefore, the vehicle needs very little space between the rows to follow the rows precisely.
The disadvantage of steering on all wheels is that the construction and control become more complex.
Because of the possibility of good odometer data, precise steering and small turning radius, all-wheel Ackerman steering have been chosen.

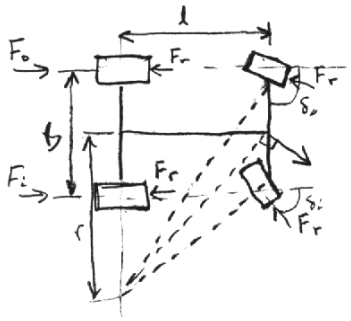


Figure 2.10 Principle of Ackerman steering.

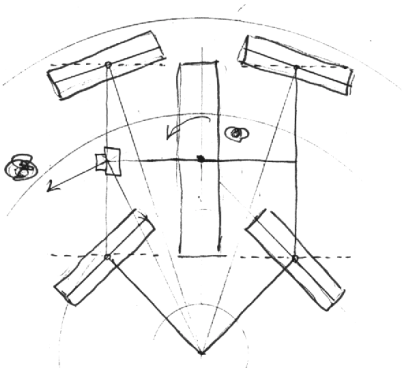


Figure 2.11 Placement of wheels on the robot.

2.3. Dimensions

Four wheels placed in each corner of a rectangle with the length parallel to the crop rows have been chosen for the vehicle. This configuration gives a good stability and the rear wheels can run in the tracks of the front wheels, which decreases the rolling resistance. The configuration can be seen on Figure 2.11.
Other possible wheel configurations are shown in appendix 2.3.1.d.
The symmetric placement of the wheels and the choice of 4WS and 4WD, give the possibility of making transmission and steering gear in four identical modules. This will make the mechanical design faster and especially the construction in the workshop will be faster and less expensive because of fewer different parts. Planning, interpretation of

drawings and adjustment of machines is time consuming in the workshop and therefore it is important to keep the number of different parts low. This modular concept will then consist of a chassis with 4 identical "wheel modules".

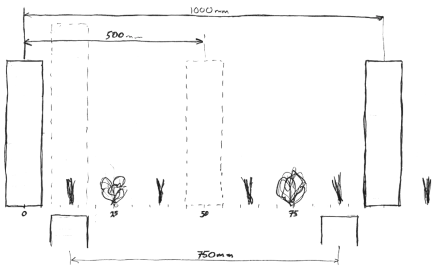


Figure 2.12 Necessary wheel gauge for in-row driving in corn and beet fields.

Gauge

There are several constraints on the dimensions of the vehicle. It should be able to drive in crop rows, with 500 mm between the rows e.g. sugar beet. There is also focus on weeding in cornfields. Weeding is possible when the row distance is increased to 250 mm Therefore it would be advantageous that the vehicle could drive in cornrows, with 250 mm between the rows. Beets are broad-leaved and therefore the wheels should not get too close to the row centre and damage the beets.
This means that the gauge should be either 500 mm or 1000 mm. If the gauge is 750 mm or 1250 mm the wheels will get to close to the row centre in beets, a even bigger gauge will make the vehicle too difficult to transport and handle in general. The ground clearance on 500 mm makes it necessary to make the gauge 1000 mm, because the centre of gravity else will be placed very high compared to the gauge and make the vehicle unstable.
Another possibility is to make the gauge adjustable, but we have judged, that a fixed gauge on 1000 mm. will cover the necessary row widths sufficiently.

Figure 2.13 Possible areas for attaching tools

Placement of tools and loads

The tools should be placed in the most stable position with the best-defined distance to the ground. The best position is in the middle of the wheelbase, where obstacles and irregularities will cause minimum movement of the tool.

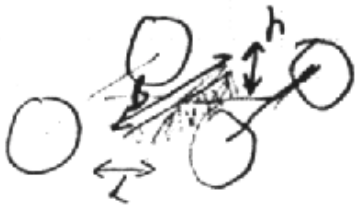


Figure 2.14 Spot sprayer

Tools

This test vehicle should be able to be tested, with a big variety of tools for weeding, spraying, collection of crop data and soil samples, etc. This type of vehicle is not intended for cultivating etc., which is much more efficient with conventional tractors. Therefore this vehicle is not designed for big drawbar pull, but just to be able to transport itself, tool and a load of e.g. herbicides, under the conditions given in the field.
DJF has proposed some tool concepts, which the vehicle should be able to operate with.

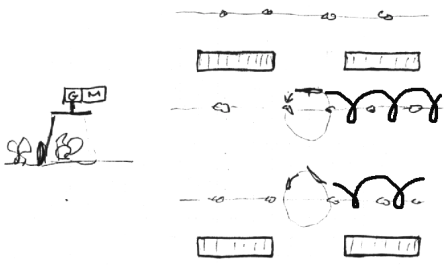


Figure 2.15 Rotating hoe

Spot sprayer

The idea is to identify the individual weeds and spray the weed leaves, without polluting the bare ground or the crop. This principle has also been used for spraying railway tracks for weeds, which is a simpler task, because all vegetation must be sprayed.

Rotating hoe

As shown on Figure 2.15 this tool consist of a rotating hoe, which is synchronized with the forward speed and the position of the crop. The tool will be able to remove weed between the crops without the use of herbicides. The disadvantage is the power consumption of the tool and possible influence on the steering of the vehicle.

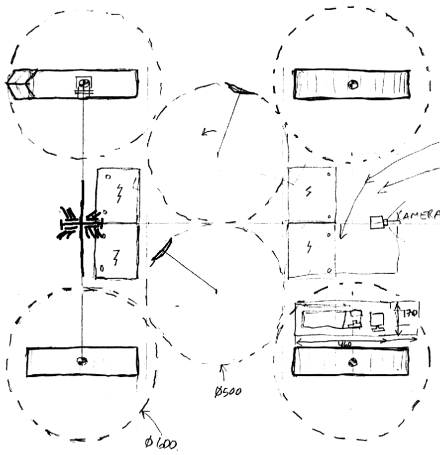


Figure 2.16 Dimensions

Wheelbase

A short wheelbase improves the manoeuvrability of the vehicle, but also makes it harder to control. Our goal is to make the vehicle flexible, therefore the wheelbase is chosen so, that it will just leave room for the tools and give the best manoeuvrability.
The rotating hoe and the space for rotation of the wheels limit the wheelbase to minimum 1000 mm, see Figure 2.16. This wheelbase also allow for leave raisers on the wheels to prevent damage to the crop.

Ensuring ground contact

The wheels should always have contact to the ground to improve traction and steering and to make the tools follow the ground in a better defined way. On road vehicles, different spring-damper systems are used for this purpose. On this low speed off-road vehicle, it is not necessary with more damping than the tyres can provide. However, to make the wheels follow the ground there has to be a longitudinal pivot joint between the front and the rear axle. On Figure 2.16 the front axle can turn about a longitudinal axis, because of the rotational joint in the middle of the front axle.

2.4. Control Architecture

Sensors

GPS system

During our initial investigations, we found that an affordable GPS system was not precise enough (the precision of DGPS is 1-5m). To get a satisfactory precision of 1-5 cm it would require a carrier-phase differential GPS (also called RTK) equipment priced at about DKK 300.000.
Therefore we decided not to use GPS, but later we got the possibility to borrow such equipment from the National Survey and Cadastre Denmark. Read more in “The GPS Module”, page 95.

Orientation

It is very important to know the orientation of the vehicle, not only the heading but also the tilt in both directions. For example, the GPS only returns the position of the GPS antenna. We need to know the roll, pitch, and compass angles to convert the antenna position to the control point. The compass angle (yaw) is also important to steer the vehicle.
To find the orientation of the vehicle different sensors can be used. We have considered various technologies with various technical pros and cons, but as our budget is limited, the price is also an important factor:

Gyroscope.

A gyro can measure the orientation of all three axes. A gyro is sensitive to the rotation of earth (unless they have build-in compensation) and they all have to be initialized in a known orientation. The advantage is that it is not sensitive to accelerations.

The cheapest solution we could found was the electronic dual-axis "Micro gyro 100" from www.gyration.com which in the spring 2000 was priced US$450.

Accelerometers

There are various technologies to measure acceleration. By measuring the angular rate on three axes, it can find the orientation relative to the orientation when it was initialised. This is similar to a gyro and some manufacturers market accelerometers as electronic gyros.

GPS

With a GPS antenna we get the position in three dimensions. Having three GPS antennas it is possible to calculate the orientation of all three axes. It requires several meters between the antennas and high precision GPS such as carrier-phase differential GPS with a precision of 1-5 cm. The price for this solution is extremely high. Results with 4 antennas on an autonomous guided tractor can be found in [GPSTRAC].

Magnetometer

The field of the earth can be measured with a 3-axis magnetometer. The measurement is influenced by the magnetic declination and deviation caused by local magnetic fields. (Read more about these errors on page 100). We were recommended to use products from the companies Billingsley and Bartington by people connected to the "Ørsted" project. Their products were however quite expensive and we found other solutions for example the Vector 2x from www.precisionnav.com at only US$ 50.
To calculate the heading from a magnetic field it is required to have data from an inclinometer (tilt-sensor).

Inclinometer

An inclinometer measures the angle of gravity on two axes. We found an optical inclinometer from www.usdigital.com priced at US$ 75. It is based on a damped pendulum with a quadrature encoder and measures an absolute angle of gravity.

Solution

From Precision Navigation (www.precisionnav.com) we also found another module than the one mentioned above. It is an all-in-one product that perfectly suits our needs. It was priced at a reasonable $345 that with academic discount became $245. It contains a 3-axis magnetometer and a fluid based tilt-sensor. It compensates for tilt internally, and via the RS232 interface we can read compass and 2 axis tilt data. Read more in the "Orientation module", page 100.

Vision

So far the most successful way to find plants it is to use a camera. Usually it would look at the infrared spectrum in which plants show up clearly from the background. There are already systems in use that can detect plants but the technology to separate weed from crop is not yet convincing.
At the "Department of Control and Engineering Design" they already had a cmos camera used in previous vision projects so there was not very much for us to decide. It does however not look at the infrared

spectrum and therefore it can only detect “paper plants” with good contrast to the background.
However, a cmos camera is the best technology for use in control applications. Read more in the “Vision module” on page 113.

Motor feedback

We need to get speed and position feedback from the motors to control them.

Tachometer

A tachometer returns the current speed and is therefore useful for speed control. To get the position of the motor we need to integrate the tachometer output. As the speed output is an analogue signal, it is sensitive to noise. The noise will also be integrated and the result is a position error. The integration will typically be done with a computer sampling the analogue signal through an AD converter. The sample and hold and quantification of the signal will also introduce a position error.

Encoders

An alternative is encoders, which read the current position of the motor. An absolute encoder always knows its position i.e. it does not have to be initialised. An incremental encoder sends pulses to a counter and thus the position is always relative. This can be solved by using an index marking that resets the counter at a certain position. An encoder is excellent to get the position, but deriving speed from counter values introduce a quantification error. Especially at low speed and at high sample rates there might only be a few counter values between two samples to derive the speed from.

For the steering motors, we need to know the absolute position. An absolute encoder from www.usdigital.com is US$ 270 while an incremental encoder with index from the same company is US$ 48. However when including the cost to interface four of them with the PC it is US$275 for the absolute encoder and US$136 for the incremental encoder.

For the drive motors a tachometer could be used, as we are mainly interested in controlling speed and not the position. For the dead reckoning system, we do however need to know the travelled distance and with the integration errors described above we have chosen incremental encoders. From www.usdigital.com we found an incremental encoder that could produce 8192 counts per revolution which is enough to get a good precision when deriving speed. Read more about this in the “Encoders module” on page 89.

Hardware

Distributed control

For data communication in commercial or industrial applications, the CAN-Bus is often used when having a large number of sensors supplying data for a large number of units. The CAN-bus is a distributed network system. All units connected to the bus have access to all data on the bus. A sensor will send data on the bus with a priority and a unique ID. All other receivers can then decide if they

will use this data or not. The advantage is that the cabling is simple as only one set of cables will connect all units. The disadvantage is that you need a microcontroller for all units interfacing to the bus. This makes it expensive unless you buy a large quantity.

Centralized control

A cheap and flexible solution is to wire all control signals to interface cards in a PC. All connections between sensors and the users of sensor data are made by the PC and all calculations are made in the software. In this way it is very easy to make modifications as they only have to be changed in one place. The PC also gives great possibilities to make an informative interface to the user.
We have chosen to use a PC as this vehicle is going to be used as a base for many future projects.

Navigation

Manual control

The vehicle should be easy to manoeuvre manually. A cheap, simple and well-known solution is to use a PC game joystick. The keyboard could also be used, but as the keys only offer on/off position it is difficult to control the speed and turning rate precisely. We have therefore chosen to use a joystick. In addition a modern PC joystick often offers extra sliders and buttons that can be used by the user to control and activate special functions such as turning sensitivity, sideways driving, etc.

Automatic control

Position

We borrow the GPS equipment and therefore the vehicle must be able to drive without that. A typical way of getting the position when no absolute position sensor is available is to use encoder and compass data in a dead reckoning system. The dead reckoning continuously calculates the position of the vehicle by summating wheel distances and vehicle headings. With a well designed dead reckoning system we are able to use and test the vehicle no matter if the GPS is present or not. The only difference will be the actual precision of the position.

Path

The robot should be able to follow a list of waypoints. It could for example be a list of crop positions generated when sowing. The waypoints could however also be generated by the on-board vision system the calculates the position of the plants it sees on the fly.
The vehicle should not necessary drive the direct way between two waypoints. Especially it might not be possible for it to turn sharp enough to go directly to the next waypoint. This could happen at the end of a crop row where it should turn 180° and go back with the next row.

Position control

There are many methods to let a vehicle follow a path. We have chosen to use a simple solution to start with. The method is often used when an AGV should follow a line on the ground using optical sensors.

Motor control

We need to control the absolute position of the steering motors. As the friction to the ground vary a lot there must be an integrator in the controller to be sure that it can reach the wanted position. We have decided to use PI controllers
For the Honda drive motors we need a speed control and as the rolling resistance (including climbing hills) varies a lot there must be an integrator to use its power potential.

Software

The software should be as flexible as possible and it should be easy for other projects to continue our work without having to start from the beginning. Software code made by others is especially difficult to understand and to develop further, because all people have their own programming style and preferred programming language. Especially when considering the risk of hidden errors in the code, few people would therefore wish to develop new code directly in the code from previous projects.
To avoid this problem we decided to make a module based software system. All the software functionality should be divided into a number of logical modules that are each compiled into separate DLL's. The DLL's should have a standardized interface just like all Windows drivers. Each DLL can therefore be replaced by another DLL with the same interface, without affecting the functionality of the rest of the system. A replacement DLL can even be programmed in any programming language.
It is therefore easy to modify and develop the system without needing to consider and understand completely what is going on in the other modules.

Identifying modules

In the first phases of the development we had divided the software system into 5 modules. The overview of the structure is shown in appendix 5.0b modules. We soon realized that this was not enough and we therefore added 7 more modules.
In the following sections we will argue for the choice of the final set of modules.

Hardware

It should be possible to replace the attached hardware (sensors and DA converters) without affecting the other parts of the system. Therefore there should be a module functioning as the driver for each hardware part. This requires the following modules to be made:

Hardware	Module	Input	Output
DA converter IO card	MotorsOut.dll	Control signals (values)	Control voltages and directions to motors
Encoders	Encoders.dll	Counter values from quadrature decoder interface cards	Steering motor angles Drive motor speeds Drive motor distances
GPS	GPS.dll	Data from GPS on the RS232	Global position of vehicle origin.

		port.	
Compass Tilt	Orientation.dll	Data from the electronic compass on the RS232 port.	Orientation and rotation matrix from vehicle frame to global frame.
Camera	Vision.dll	Images from camera	Position of objects in the image.

Position

The current position of the vehicle could be relevant to use by several other parts of the system. Therefore we have made a module that calculates the position of the vehicle based on data from the encoders, compass and GPS modules. The position module handles the dead reckoning calculations and it should also work if GPS is not present:

Module	Input	Output
Position.dll	Sensor data from Encoders.dll, GPS.dll and Orientation.dll	Current position of the vehicle.

Task planning

The vision module needs a task planner to handle what should happen when it no longer sees objects (crops) that the vehicle should follow. This has been placed in a separate module.
Waypoints should be handled by a database; we have placed that in its own module.
The calculation of a path based on waypoints is a task that can be complex and it has therefore also been placed in its own module. The following modules have been identified:

Module	Input	Output
Visiontask.dll	Position of objects on the ground from Vision.dll and the vehicle position from Position.dll	Waypoints.
Waypoints.dll	Waypoints added by the user or by the vision task-planner, or from a database.	Waypoints in sequential order.
Path.dll	Waypoints	Path segments

Controllers

The control of the vehicle can be divided in two areas: the low level control of the motors and the high level control of the vehicle position and direction. Therefore we have placed this functionality in two modules:

Module	Input	Output
Controller.dll	Reference: Path segments that the vehicle should follow	Drive motor speeds and steering motor angles.

	Feedback: Position and orientation of vehicle.	
MotorCTRL.dll	Reference: Drive motor speeds and steering motor angles from Controller.dll Feedback: Drive motor speeds and steering motor angles from Encoders.dll	Control signals for MotorOut.dll

Timing

The timing of the controllers and task planners and the sharing of processor resources should be handles centrally and not by the modules individually. Therefore we have made a module that takes care of this:

Module	Input	Output
Timer.dll	The user specify what modules should be controlled, the rate (Hz) and the priority.	Calls the specified DLL's

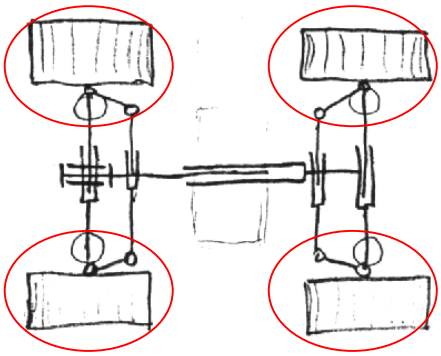


Figure 2.17 The chosen mechanical modular concept with 4-WD and 4-WS. 4 identical wheel modules (red ring).

A graphical overview of all the modules and their connections can be found in appendix 5.0. The detailed description of the modules can be found in chapter 5 (page 82 ff.)

2.5. Motors and power supply

Motors

The motors for the vehicle must be easy to control, supply and to install. It is also preferred, that they can be used indoors for test. DC-motors best meet the above demands. DC-motors can be used indoors and be supplied by battery. Power amplifiers and gears for DC-motors are off-the-shelf products, which makes it a flexible and easy to install solution.

Power supply

The vehicle should be completely autonomous and therefore have its own power supply. The power supply should be able to drive DC-motors and the computer. Batteries where chosen for the vehicle, because it is the easiest to install and should be able to deliver power for 2 – 4 hours of testing, without getting to heavy

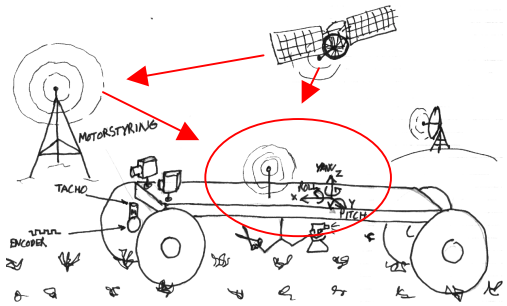


Figure 2.18 Focus will be on RTK/GPS for position data and a 3-axis magnetometer with roll and pitch sensor for attitude data.

2.6. Summary of the overall concept chosen

The modular vehicle concept is based on:

- 4WS
- 4WD
- Capable of overcoming obstacles
- DC motor technology
- Batteries for power supply

The vehicle will be designed with 4 identical corners consisting of a wheel module connected to the chassis.

The control architecture is based on:

- 12 software modules placed in separate DLL's
- RTK/GPS for position data
- 3-Axis magnetometer with roll and pitch sensor
- Vision

3. The mechanical design

In this chapter, the mechanical design will be done on basis of the principal decisions made in the previous chapter. First, the steering and drive gear principal for implementing 4-WD and 4-WS on the vehicle will be determined. In the second paragraph, the wheel module will be designed. Then the chassis will be designed and finally the assembly of the mechanical parts of the vehicle will be discussed.
Overall demands and criteria's for the mechanical design:

- In-row driving in fields with a ground clearance on minimum 500 mm
- Space for different types of tools and sensors
- Possible to implement and design within this project.
- Protection of electronics etc.
- Flexibility

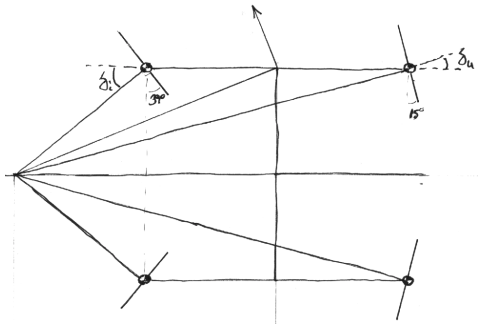


Figure 3.1 Ackerman steering.

3.1. Steering and drive gear

In this paragraph, the principles for mechanically implementing the overall principle of 4-WS and 4-WD will be determined.

Ackerman steering

The principal steering concept chosen is 4-ws and Ackerman steering, because of flexibility, possibility of good odometer data, good steering precision and a small turning radius.
The Ackerman steering geometry is shown on Figure 3.1 and the relation between the wheel angles are given in the formula below.

$$\cot\delta_u - \cot\delta_i = \frac{B}{L} \qquad ,B \approx 1000mm, L \approx 1000mm$$

Formula 3.1

Formula 3.1 shows that the relation between the wheel angles is non-linear and a function of the wheel base (L) and the kingpin gauge (B). The dimensions of the vehicle have been chosen and the very large width compared to the length makes the difference between the wheel angles big even at minor steering angles, see Figure 3.1. This makes it difficult to design the steering gear to avoid geometrically induced tire slippage, which will result in poor odometer data and inaccurate steering.

Figure 3.2 Each wheel is controlled by separate motors (top view).

Implementing 4-WS Ackerman steering

The demands and criteria's for a good solution for the steering gear is:

- Good compliance with the Ackerman theorem.
- Acceptable play or backlash.
- Safe and reliable.

The following principles have been investigated (Appendix 3.1.3 – 3.1.4.b).

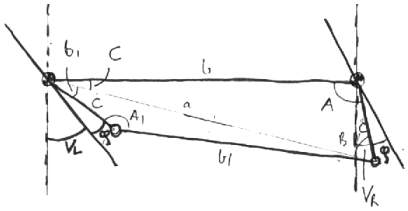


Figure 3.3 Conventional steering mechanism.

Motor to position each wheel

A motor and a gear or a linear actuator can be placed on each wheel to change the wheel heading.

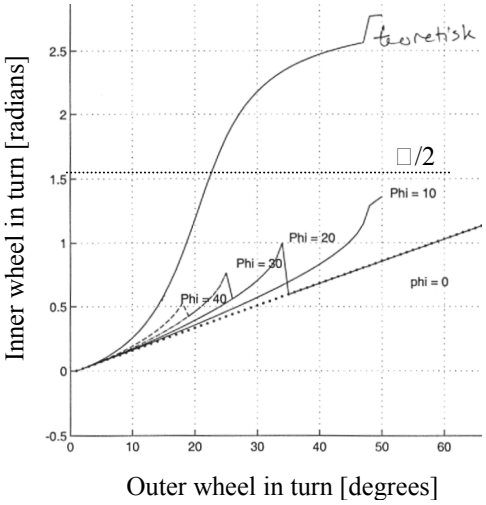


Figure 3.4 Matlab plots showing the results for the conventional steering gear.

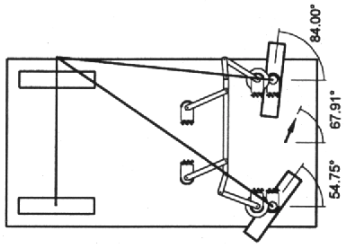


Figure 3.5 Mechanism for Ackerman steering.

Advantages:
With closed loop computer control it should be possible to get very good compliance with Ackerman. Changes of steering strategy e.g. sideways steering or change of dimensions, is easily done in software. If the construction allows it, it will be possible to rotate the steering angle 360^{O}. The flexibility of this method would be a big benefit for research in field steering. The mechanical design will not be very complicated and time can be saved in the workshop. The solution is very compact and connections are made with electrical wires. This makes it less complicated to design the chassis, because there are fewer constraints on the design.

Disadvantages:
There is no mechanical link between the steering wheels, but because it is working in the field at low speed, the safety risk is considered very small. It is expensive to buy all the electronic and mechanical components. There can be problems with getting the wheels to follow the Ackerman in real time and then geometrically induced tire slippage will not be avoided [BERN].

Steering rod

A conventional steering gear, which is used on most road vehicles, where a steering rod connects the steering wheels, Figure 3.3.

Advantages:
Driving forward, the wheels are always well aligned and the geometrically induced slip is very small. Only one input is necessary to control the steering.

Disadvantage:
The steering gear has good compliance with Ackerman for small steering angles on vehicles with a typical length/width ratio around 2. In appendix 3.1.5.a-d different structures of this type of steering gear is investigated. A typical result is shown on Figure 3.4 and it can be seen, that for this vehicle, the steering gear have very poor compliance with Ackerman. The outer wheel angle V_R and the geometrical angle φ is the input and the inner wheel angle V_L is the output. Different geometries have been calculated in Matlab, but a fairly simple solution, with good compliance with Ackerman, have not been found. It is though possible to get good compliance with Ackerman, see Figure 3.5 [MECH99]. These mechanisms are however very complex and takes up a lot of space.

Guide way or special designed guide wheels

A guide way with the wanted path is moved linear by an actuator, so that the wheel angle will be correct, see Figure 3.6; a. This could also be done with a wheel designed to make the steering wheels assume the right angles, see Figure 3.6; b.

Advantages:
It is possible to get good compliance with Ackerman.

Disadvantages:
It is difficult to design and to make the mechanism operate properly with little friction and little play or backlash.

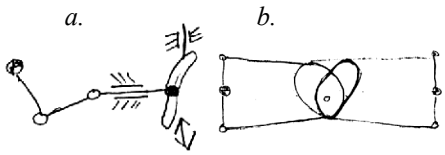


Figure 3.6 Guide way and guide wheel.

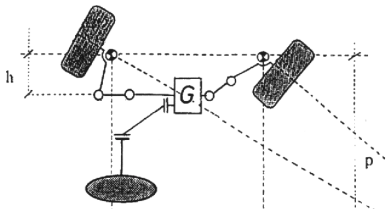


Figure 3.7 Ackerman compliance achieved with differential gear.

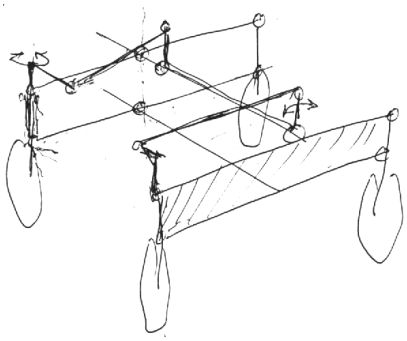


Figure 3.8 Connection of the front and rear wheels with steering rods.

Differential gear

It is possible to design a differential gear, to give the rotation wanted to steer each wheel according to Ackerman, see Figure 3.7 [DIFF98]

Advantages:
Good compliance with Ackerman and good alignment of the wheels. The wheels are also connected mechanically.

Disadvantages:
The mechanical design is very complicated and a project in itself. It will be difficult and time consuming to make in the IKS workshop. This could prevent us from reaching our objective.

Connection of front and rear axle steering gear

The following principles have been investigated.

Connection by steering rods
Rods connect the rotation of the kingpin of the front and rear wheels, see Figure 3.8.

Advantages:
Can be made of standard components.

Disadvantages:
The joints must be placed so that the steering angle is not changed, when the front axle turns a little because of obstacles. The connections will take up a lot of space.

Steering motors for each wheel
Each wheel has its own steering wheel connected to the kingpin.

Advantages:
Sideways movement will make it possible to position and orientate the vehicle more precisely in the rows. The method is very flexible, because the steering strategy is implemented in software. Electrical wires are very flexible to place.

Disadvantages:
The component price will be high and the control of the vehicle will be more complex.

Wire connection
Wires connect the kingpins.

Advantages:
Big flexibility of placement of connection and simple to implement

Disadvantages:
Might give problems with backlash and play.

Hydraulic connection
Hydraulic motors or actuators are used to control the steering.

Advantages:
The play could be very small and the hoses are very flexible to place.

Disadvantages:
Hydraulic components are quite expensive and it will be necessary to have a pump running to give oil to the system.

Choice of 4 wheel steering principle

After having investigated the different possibilities of implementing Ackerman and to connect the steering of the front and rear wheels, we have chosen to use 4 motors for steering. This gives the best possible flexibility, because all the options are in software. This will give opportunity for a large variety of tests of steering methods in the field. It will be possible to test front, rear and all-wheel steering. The point, which the wheels turns around, can be placed anywhere and not just on a line including the front axle, rear axle or the centre point of the vehicle.
This will be a very good possibility for the control department to gain experience with control of autonomous terrain vehicles.

Design of the driving transmission

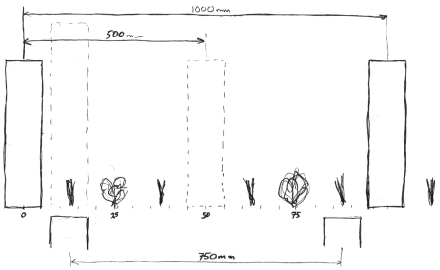


Figure 3.9 The dashed lines indicate the necessary ground clearance.

Transmission and motor

Because of the big ground clearance on 500 mm and the wheel diameter on 350 mm to 500 mm, it is complicated to make a transmission to a common motor, see Figure 3.9. A common motor will also make it necessary with differential gears between the front wheels, the rear wheels and the front and rear axle. This will be very expensive, heavy, complicated and decrease the efficiency of the transmission.
Another possibility is to use a motor to drive each wheel and then, with closed loop computer control, control the speed of each wheel according to its turning radius. This principle has been chosen for transmission and motor.

Wheel module

The wheel module now have its own steering motor and drive motor, which makes the system very flexible. With both drive and steering gear in the same unit, we will have very few constraints on the chassis design and better opportunities for optimising it for experimental use.

3.2. Wheel module

In this part, the mechanical concept of the wheel module will be designed in detail. The overall principal and quantitative structures of the vehicle have been determined. Now the modules (wheel modules and chassis module) should be detailed and the interface between the modules must be determined.

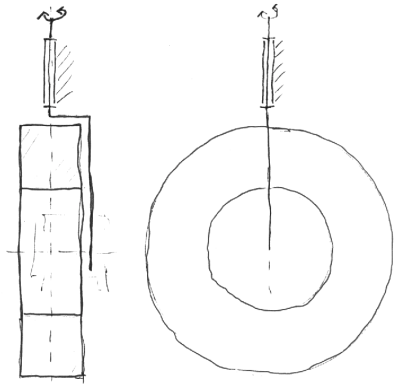


Figure 3.10 Placement of kingpin with zero offset.

Quantitative structure

The wheel module consists of a wheel with drive motor, steering motor and sensors for both motors.
Because of the demand for a big ground clearance and a very slim wheel construction to drive between the crop rows, the kingpin have been placed above the wheel centre, the steering mechanism is said to have zero offset, see Figure 3.10.
This structure has many advantages. There is minimum feedback of forces into the steering system. The steering movement on one wheel have minimum influence on the steering of the other wheels. Zero kingpin offset also leads to minimum power requirement for steering, see page 40. If the kingpin was placed on the side of the wheel, the steering motors would have to deliver a constant moment to hold the direction, which would lead to failure of the motors.
The disadvantage is that the big distance between the bearings and the load attack point will give bigger bearing forces, but this is not a significant problem.

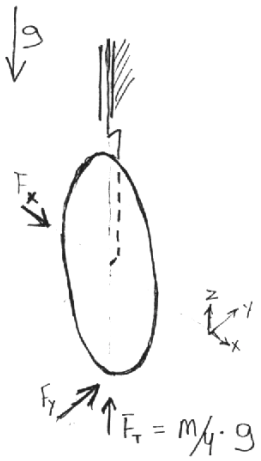


Figure 3.11 Forces reacting on the wheel module.

Forces on the wheel module

When designing a mechanical system, it is necessary to look at two scenarios; the normal operating conditions and the worst-case conditions.

Under normal conditions, the system should be able to perform as specified and it can be important to look at deflections and dynamics. Under worst-case conditions, it is important to realize, that it is impossible or not economical to avoid failure under all conditions. It is important to foresee, what can happen to the system and design the system to fail in the best way. The most important thing is to try to avoid danger to people and secondary to design the system to have inexpensive parts fail, e.g. a screw.

Mechanical design is always a trade-off between conflicting interests, for example low weight and high strength. This vehicle has to have low weight to minimize the energy consumption and cost for motors and batteries.

On Figure 3.11 the forces acting on the wheel module are shown. In the next two paragraphs, the normal and worst-case forces are estimated and then in the third paragraph the forces are updated because the choice of motors limits the allowable forces.

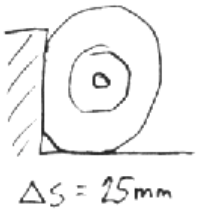


Figure 3.12 Deformation of the tyre, when hitting an obstacle with 1 m/s.

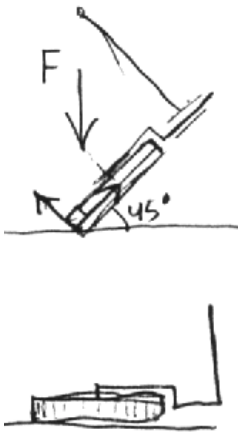


Figure 3.13 Forces if the vehicle is tilted.

Normal working conditions

The forces under normal working conditions are mainly due to the weight of the vehicle, the traction from motors and the steering of the wheels.
The force F_x is mainly due to traction, μ is the estimated rolling coefficient:

$$F_x = \frac{m}{4} \times g \times \mu \approx \frac{300\,kg}{4} \times 9.81\,{}^{m}\!/_{s^2} \times 0.25 \approx 184\,N$$

The force F_y is mainly due to the centripetal force during turning. As the centripetal force increases the resulting force is coming closer to the outside wheels in the turn. Therefore, the whole force will be on the outside wheels just before the vehicle will turn over. Below the forces are assumed to be equally distributed between all 4 wheels:

$$F_y = \frac{m}{4} \times \frac{V^2}{r_{turn}} \approx \frac{312\,kg}{4} \times \frac{(2\,{}^{m}\!/_{s})^2}{1\,m} \approx 312\,N$$

The force F_t is mainly due to the gravity:

$$F_t = \frac{m \times g}{4} \approx \frac{312\,kg \times 9.81\,{}^{m}\!/_{s^2}}{4} \approx 765\,N$$

Worst-case conditions

We have estimated the worst-case condition the vehicle should be able to manage, without damage.

$F_{x,\ worst\ case}$
If the vehicle hits an obstacle with a velocity of 1 m/s, see Figure 3.12. We assume, this will lead to a deformation of the tyre $\Delta s = 25$ mm and a linear decrease of speed, which will give an average speed $V_{ave} = 0.5$ ${}^{m}\!/_{s}$. This is just an estimate and will depend a lot on the inflation rate of the tyre and the tyre material. The time to decelerate the vehicle would then be:

$$t = \frac{\Delta s}{V_{ave}} \approx \frac{0.025\,m}{0.5\,{}^{m}\!/_{s}} \approx 0.05\,s$$

The deceleration:

$$a = \frac{\Delta V}{\Delta t} \approx \frac{1\,{}^{m}\!/_{s}}{0.05\,s} \approx 20\,{}^{m}\!/_{s^2}$$

The resulting force:

$$F_{x,worst\,case} = m \times a \approx 300\,kg \times 20\,{}^{m}\!/_{s^2} \approx 6\,kN$$

(This could also have been found by an energy approach)
This assumption is not very accurate and depends a lot on the inflation rate of the tyres, but it gives a hint about the size of the forces.

$F_{y,\ worst\ case}$
The vehicle is lifted up in one side or tilting, see Figure 3.13. The maximum static force will be:

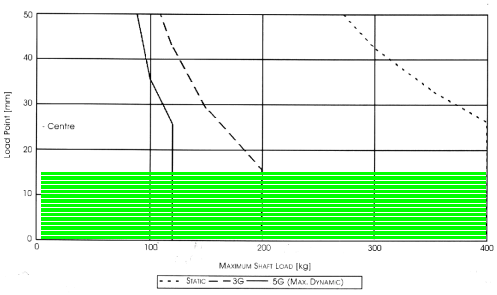


Figure 3.14 Maximum allowable impact shock on the Honda motors

$$F_{y,worst\,case} = \frac{m}{2} \times g \approx \frac{300\,kg}{2} \times 9.81\,{}^{m}\!/_{s^2} \approx 1.5\,kN$$

It is unlikely, that the vehicle will be exposed to big forces in the F_y direction.

$F_{t,\ worst\ case}$

The vehicle is lifted and dropped.
The forces when falling from h = 0.25 m height is calculated using an energy approach, but with the same assumptions as for $F_{x,\ worst\ case}$.

$$F_{t,worstcase} = \frac{\frac{m}{4} \times g \times h}{\Delta s} \approx \frac{\frac{300\,kg}{4} \times 9.81\,{}^{m}\!/_{s^2} \times 0.25\,m}{0.025\,m} \approx 7.4\,kN$$

Honda, maximum impact

Later in the design process, we have chosen to use in-wheel motors from Honda, on page 34. The maximum allowable impact shock on the motors is shown on Figure 3.14 [HONDA]. The motor can nearly handle the forces calculated earlier, but it will be the motor maximum impact shock forces, that the vehicle should be designed for. The working area for this vehicle is in the green field on Figure 3.14. Because the dynamic forces are difficult to estimate, the vehicle will be designed for a worst-case force: $F_{t,\ worst\ case} = F_{x,\ worst\ case} = 4$ kN.

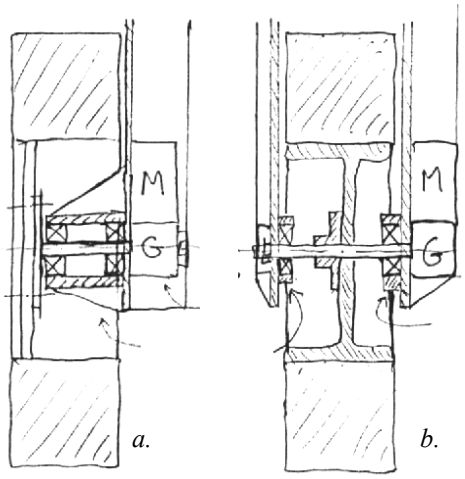


Figure 3.15 Design using motors with worm gears.

Drive transmission

Demands and criteria's for a good drive system:

- DC motor power: the vehicle should be able to drive in a field with loose soil, at 1 m/s.
- Slim construction capable of driving in row crops
- Capable of driving outdoors. Protected against rain and dust, minimum IP54[1]
- Little play or backlash
- Low complexity

A lot of research has been done to investigate the possible drive solutions, which are available on the market and suitable for this vehicle. We have made inquiries to many distributors of drives, but also to users as e.g. wheelchair manufacturers (see "Literature" on page 159). In appendix 3.2.3 a number of different principles are listed. Two principles have been investigated further and are described below. The rest of the principles would require too much work both to design and to produce in the workshop.

DC-motor with worm gear

The motor construction principle is shown on Figure 3.15.
On Figure 3.15; a, the wheel is carried from one side, which makes the bearing forces bigger. However, it is easier to align the bearings.
On Figure 3.15; b, the wheel is carried by a fork, this gives a better distribution of forces in the frame and lower bearing forces. The fork solution will spoil the look of the vehicle.

[1] Appendix 3.2.32

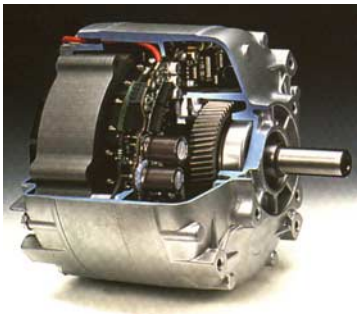


Figure 3.16 Look at the inside of a Honda in-wheel motor.

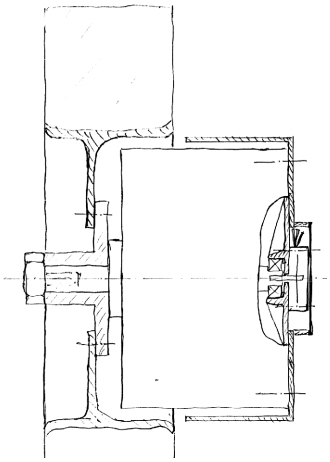


Figure 3.17 Part of the motor is placed inside the wheel rim(top view).

Common for the two solutions are that the gear bearings cannot handle the radial forces and therefore two bearings have to be used. This makes it necessary with a coupling between the wheel axle and the gear axle, because they cannot be aligned perfectly. Misalignment courses big internal forces and leads to failure. Another possibility is a special bracket for the motor, which will allow the motor to move a little.
The dc-motor with worm gear can be supplied by UB-Let, who manufacture wheel chairs and other lightweight transportation equipment (www.ublet.dk). UB-let can also supply the power amplifier to drive the 24V dc-motor, which will make this choice fairly inexpensive.
Worm gears have a preferred direction of rotation, which make them less suitable for the robot, where two motors always will rotate against the preferred direction. Another disadvantage is the very low efficiency of motor and worm gear.
There are no standard velocity sensors for this type of motor and gear, so a bracket must be made for a sensor.

Honda in-wheel motor

From Honda we found an in-wheel brushless dc-motor, with build in gear and brushless dc-motor controller, see Figure 3.16. With the Honda motor, it is possible to design a very slim and compact construction, see Figure 3.17. Below is a list of additional features:

- Speed controlled by a 0-5V signal.
- Overall battery-to-shaft efficiency exceeds 85% (Honda), which is important on a battery-powered vehicle.
- Support loads up to 400 kg.
- Sealed against all kinds of weather.
- Fits into 8" wheel rims

The fewer parts will allow us to design the drive faster and save work in the workshop.
The disadvantages of the Honda solution are the price and that we need to mount an external velocity sensor. The total solution with 4 motors will cost about 8.000 DKK more than the worm gear dc-motor solution, but the work saved in the workshop and the better overall solution can easily justify the choice of the Honda in-wheel motor.

Choice of motor size and gear ratio

To find the necessary motor size we have to estimate the weight of the vehicle and the rolling resistance in the field.
The gross weight of the vehicle is estimated to approximately 312 kg including a tool weight on 25 kg, a 100 kg payload and 60 kg batteries, see appendix 3.3.2.
The rolling resistance is very hard to determine. In the Honda manual a rolling resistance coefficient on $\mu = 0.085$ is suggested for off-road (unpaved surface) applications. Some literature [KØRTEK] suggest rolling resistance coefficients as high as 0.35 for 0.5 m diameter tyres on sand and 0.1 to 0.15 on firm soil. In agricultural notes rolling coefficients for tractors on up to 0.15 to 0.25 under very difficult conditions are found [KVL96].
The shaft torque can be calculated by Formula 3.2, which do not consider gradients. In the table below the maximum rolling coefficients possible, for the different interesting models of the Honda motor, are calculated.

Formula 3.2

$$Shaft\ torque = \frac{0.085 \times weight\,(kg) \times tire\ dia.(m) \times 9.8}{2}$$

The tire diameter used is 0.48 m.

Motor	Shaft torque [Nm]	Weight [kg]	μ_{max}
DDW 2015 J	Rated: 18	312	0.098
	Max: 42	312	0.22
DDW 2020 J	Rated: 50	312	0.27
	Max: 101	312	0.55

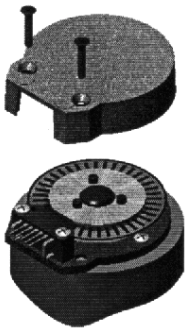


Figure 3.18 Encoder for the Honda motor shaft.

The Honda DDW 2015 should be able to do the job, because the need for a payload on 100 kg is uncertain and could be used only under good conditions. The DDW2020 is also 38 mm longer which will make it more difficult to drive in crop rows. Tima A/S, the importer of the Honda motors, can deliver the DDW 2015 J and it is chosen to buy this model.

Wheel

The wheels for the vehicle should have a good grip under field conditions on bare soil. The wheel rim must be minimum 8" to fit the Honda motor.
With an 8" rim the outside diameter of the wheel of approximately 420 mm and a rated speed of 80 rev/min, we get the following vehicle speed:

Formula 3.3

$$V\left(\frac{km}{h}\right) = \frac{shaft\ speed\ (\min^{-1}) \times tire\ dia.(m) \times 3.14 \times 60}{1000} = 6.3\frac{km}{h}$$

The vehicle speed is 6.3 km/h at rated output shaft speed. This is much more than the wanted 3.6 km/h and therefore the smallest diameter wheels should be chosen to give the biggest tangential force. From Texas, who manufacture garden equipment, we were offered 4 8" rim wheels with tractor tyres at a very low price. The ledges on the tyres make the tyre diameter 480 mm. and the width is 110 mm.

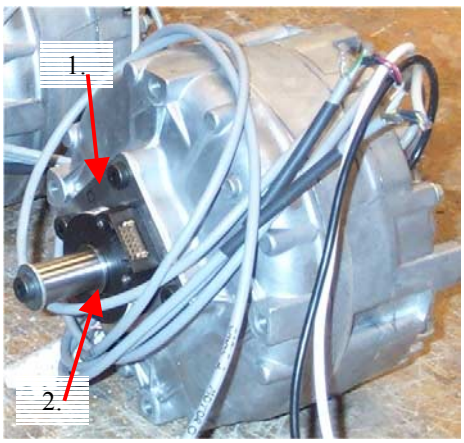


Figure 3.19 Encoder mounted on the output shaft by means of the aluminium bracket (1) and bushing (2).

Sensor for the drive motor

To control the speed of the Honda motors and to get odometer data for the position of the vehicle, we need a sensor on the motors. The Honda motor has a pulse signal, which gives 4 pulses/revolution x reduction ratio. The reduction ratio for the DDW 2020 J is 1:9.3, which leads to 37.2 pulses/revolution. This gives to little position information (40.5 mm/pulse) and far to little information for deriving velocity feedback. Therefore an encoder from US-Digital is chosen, see Figure 3.18. US-Digital has a big selection of low cost encoders with computer interface cards for the ISA-BUS. We cannot get an encoder, which will fit the 20 mm diameter output shaft on the motor and there is no access to the motor shaft on the back of the motor. Therefore a 1" encoder is chosen and a bushing is made for installing the encoder to the output shaft, see Figure 3.19[2]. The bushing has an easy running fit and is normally clamped between a flange on the motor shaft and the hub, see Figure 3.22. It is also secured with a

[2] Appendix 3.2.7.c-d and drawing 1.1.3.3.

screw, so that the bushing with the encoder cannot move if the hub is removed.
The encoder is without index and gives 8192 pulses/rev., which makes it possible to get good position measurements and derive good speed measurements. Read more in “Encoders module” on page 89.
The encoder is fitted to the Honda motor, with an aluminium bracket, see Figure 3.19[3].

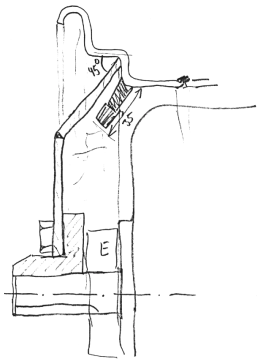


Figure 3.20 New rim plate design.

Wheel rim

The wheel rim delivered by Texas have a centred rim plate, which will make the motor with encoder and hub wide and not make the best possible use of the in-wheel design. We have chosen to make a new rim plate, which will allow us to make a very slim and optimal design for driving in narrow crop rows.

Special demands and criteria's for the wheel rim design:

- Easy access to the encoder and the air valve on the inner tube.
- High stiffness
- Good alignment
- Protection of the encoder

Attaching the wheel rim to the shaft

To get easy access to the encoder and the air valve, we have chosen to fit a hub to the shaft to easily screw the wheel on and off, see Figure 3.20[4].
The motor has a splined shaft with a thread in the end. This makes it simple to attach the hub with a key and a screw in the shaft end, to prevent the wheel from falling off. To use a key length on 33 mm is enough to transmit the maximum shaft torque to the wheel[5]. Because of reversing of the rotation direction, there is a risk that the screws can loosen [Decker]. Putting loctite on the thread before assembly prevents this.
An alternative solution is to use a Taperlock clamp element to attach the wheel instead of using a hub. This solution is good for reversing applications, but is more expensive and difficult to fit..

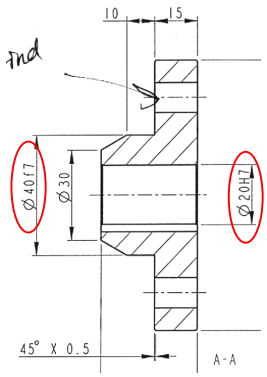


Figure 3.21 Section drawing of the hub.

Hub

The hub hole is designed with an easy running fit (H7/f8) to fit the motor shaft, see Figure 3.21. Tolerances from the Danish Standard [DS96] are used for the key/spline and the spline measure is 5H9.
The wheel rim is aligned with the hub with an easy running fit (H7/f7). The wheel rim is screwed on by M10 x 16 screws[6].
The hub is made out of one piece in st37 by turning, drilling and broaching.

Design of the wheel rim plate

The new wheel rim plate has the shape of hollow truncated cone, see Figure 3.22[7]. This design is chosen to make space for the hub and the encoder and to give the rim good stiffness. In collaboration with the workshop, we have chosen to make the rim plate of two parts, which

[3] Appendix 3.2.7.c-d and drawing 1.1.3.2.
[4] Appendix 3.2.7.a-d
[5] Appendix 3.2.7.b
[6] Appendix 3.2.7.g and drawing 1.1.3.1
[7] Appendix 3.2.7.e-f and drawing 1.1.3.4 – 1.1.3.4.2

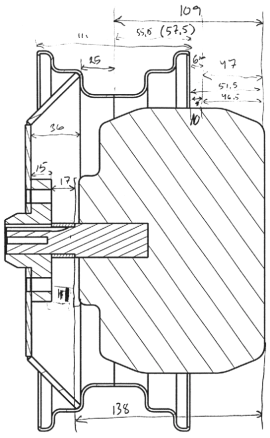


Figure 3.22 The final solution.

are welded together. An alternative solution could be to use a press, but this would require a special tool.
The part with the shape of a truncated cone is made by rolling. The centre hole in the plate has a H7 fit to align the rim and the hub. The 4 holes for screwing the rim to the hub have big clearances, because they are not used for alignment.
The rims were originally made of two parts spot-welded together. Welding the parts on the outside makes the new rim and then the old rim plate is removed by turning. The new rim plate is welded on and surfaced on a lathe to align the wheel.
The new rim plate is 4 mm thick and the material is st37, the plate are know subjected to bigger bending moments, because it is not in the centre of the rim. Therefore, the thickness is increased with one mm and the plate has no holes like the old rim plate. The rolling of the plate welded to the centre plate also increases the strength significantly.
It is necessary to protect the encoder against sand and small stones. When the rim is attached to the hub, it is only necessary to put a strip of foam rubber around the circumference of the motor to make a complete seal, between the motor and the rim.

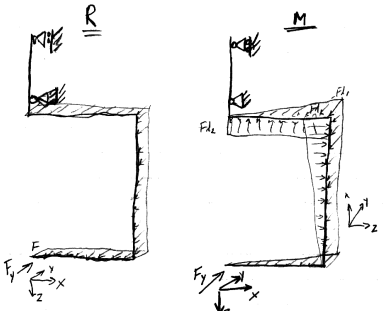


Figure 3.23 Reactions and moments in the motor frame.

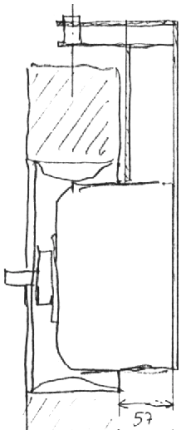


Figure 3.24 Motor attached in a U-profile (cut out side view).

Motor frame, Part design

The motor frame is the part connecting the Honda motor and the kingpin.
Demands and criteria's for the motor frame design in addition to those mentioned on page 33:

- High stiffness
- Little deflection
- Suitable for wiring

Forces

On Figure 3.23 the moments M and the reactions R in the frame, caused by the force F_x, is shown. The force $F_{x,\ worst\text{-}case}$ is the force causing the biggest moments in the frame, because of the distance from the kingpin to the point of attack. Since $F_{x,\ worst\ case}$ is a worst-case force, we assume, that it will not occur at the same time as any of the other worst-case forces. The normal force from the weight of the vehicle is small compared to $F_{x,\ worst\ case}$ and is ignored.
The vertical frame beam is exposed to both torsional loading τ, bending loading M and shear loading R.

Structure

When looking at Figure 3.24 it is obvious to make a U-profile (channel cross-section) to attach the motor to. The U-profile makes the construction very slim and the profile with round edges will not easily damage the crop.

Attaching the motor frame to the kingpin

On Figure 3.26[8] three different solutions for connecting the motor frame to the kingpin are shown.

[8] Appendix 3.2.9.a-c

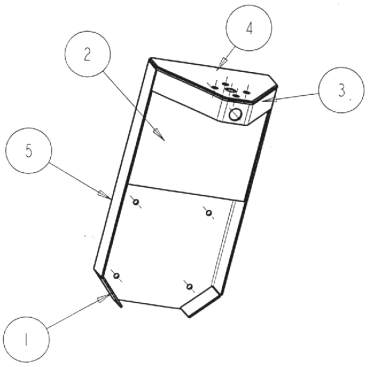


Figure 3.25 Motor frame solution.

Solution (a) uses a taper-lock, which can handle big forces and moments and is good for aligning. The solution will be quite heavy and not leave much room for wiring.
Solution (b) is assembled with screws and a small tap is made for aligning the parts. This solution is not so complicated, easy to assemble and the screws could be dimensioned for overload protection. The solution does not leave room for the wires.
Solution (c) is the final solution, where the motor frame is bolted to a flange on the kingpin. The top of the motor frame is made of a 5 mm plate, which is supported by a bended plate, see Figure 3.25. This solution is easy to assemble and leaves good room for wiring. A hole is made in the centre of the plate to align it with a tap in the centre of

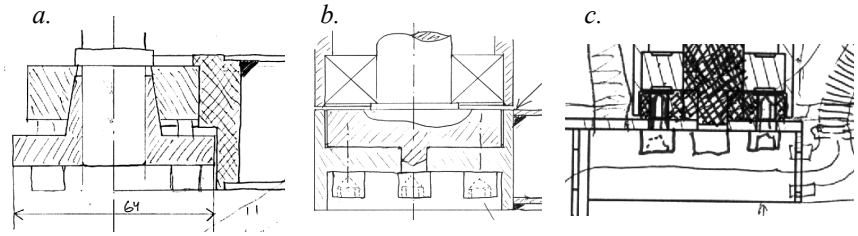


Figure 3.26 Assembly solutions for motor frame and kingpin.

the kingpin.
After welding the motor frame it is machined on a lathe, were it is going to be assembled with the kingpin, so the motor frame can be fitted straight.

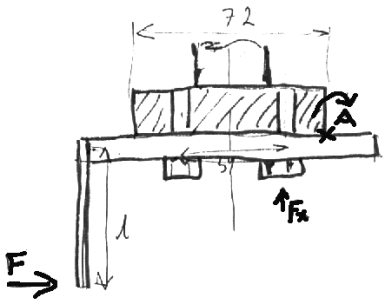


Figure 3.27 Screw connection of kingpin and motor frame.

Screw dimensions

The screws should be able to handle the worst-case forces on the wheel and the pretension of the screws. The force contribution from the worst-case force is calculated on Figure 3.27. By taking the momentum around point A the force F_x can be found. It is found that it is necessary with M10 screws and that the flange on the kingpin has to have a 14 mm long thread to avoid damage to the thread[9].

Dimensions

To find the dimensions of the motor frame, we have used beam theory to estimate the stresses.

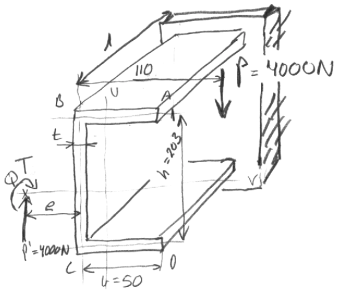


Figure 3.28 Open channel.

First, it was tried to design the U-profile as an open channel, see Figure 3.28. The load P does not go through the elastic axis of the beam and hence the beam is subjected to torsion as well as bending [ROARK].
Resolving the load P into an equal and parallel load P´ and a twisting couple T solve the problem. The load P' passes through the flexural centre Q. The twisting couple T is equal to the moment of P about Q. The flexural stresses due to P´ is found and the stresses due to T are found and superimposed giving the stresses due to the actual loading. The formulas were written in a Matlab *.m file and it was found, that it is necessary with a thickness on 6 mm[10]. An open channel section is not very good, when the beam is subjected to torsion, which gives the biggest contribution to the stresses.

[9] Appendix 3.2.11.d
[10] Appendix 3.2.11.a-c

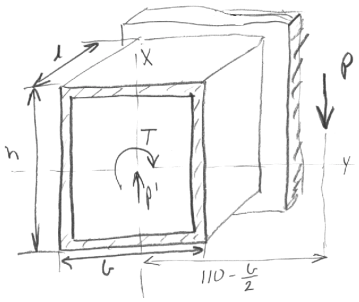


Figure 3.29 Closed channel

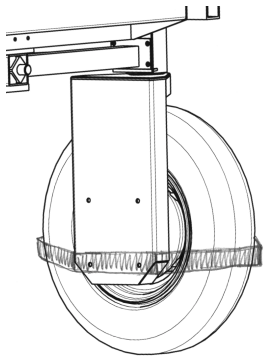


Figure 3.30 A metal tube is screwed to the motor frame and bend around the wheel to push the corn to the sides, to avoid the corn being run over by the wheel.

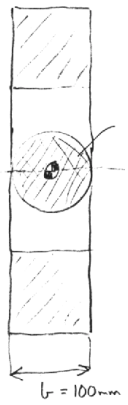


Figure 3.31 Assumed tire print.

The channel section is closed by a plate from the top plate to the motor, this makes the top part a closed section, see Figure 3.25; 2. The beam is calculated as a thin-walled rectangular tube, see Figure 3.29. A thin-walled rectangular tube with 3 mm thick walls is found to have a normal stress on 8 $^{N}/_{mm^2}$ under worst-case conditions. Because the bottom part of the beam still have a channel section and to be safe, it is decided to have a thickness of 3 mm.
At the bottom corners of the frame, some metal plates are welded on to improve the stiffness, make the frame look better and to protect the motor, see Figure 3.25; 1. The frame is open in the bottom to allow dirt to fall out.
The top plate Figure 3.25; 4 is made of 5 mm. steel plate, because the section is not closed at the bottom.
The drawings for the motor frame are 1.1.2 – 1.1.2.5. A drawing with dimensions of the sheet metal plates before bending are generally not made because the machining allowances are not well defined. The dimensions are based on the knowledge of the technicians in the IKS workshop. Drawing 1.1.2.2.1 are however, an attempt to make such a drawing based on data from "Hoischen Technisches Zeichnen" [HOISCHEN], but the drawing were not to much use in the workshop.

Wiring

The wires from the Honda motor and encoder should go up through the closed frame and out through a 21 mm hole, with rubber protection, in the plate to reinforce the channel section, Figure 3.25; 2. The wires will then come out through the hole (3), where the wires will go into a flexible conduit to protect the wires.

Avoid damage to crop leaves

If the vehicle is going to drive in cornrows, a lot of corn will be damaged and driven over by the wheels. This can be avoided if a metal tube is bend around the wheels to push the corn to the sides, see Figure 3.30. Such a device can easily be made and mounted to the motor frame.

Steering mechanism

The principle for the steering mechanism is shown on Figure 3.10. Demands and criteria's for a good implementation of the steering principle:

- DC motor power: the vehicle should be able to steer in a field with loose soil, at 1 m/s.
- Low cost
- Capable of driving outdoors. Protected against rain and dust, minimum IP 54
- Little play or backlash
- Low complexity

Transmission and control loop

In this paragraph the machine elements for implementing the steering mechanism will be chosen and the right dimensions will be calculated.

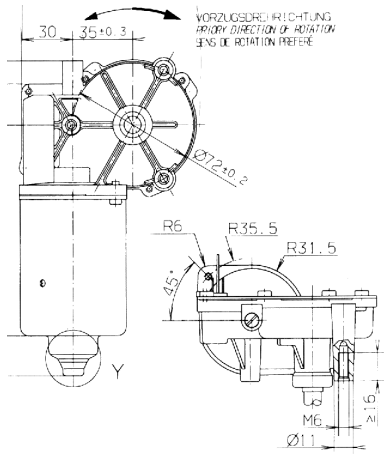


Figure 3.32 Steering motor dimensions

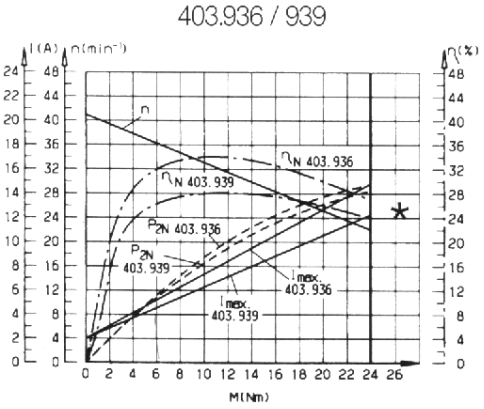


Figure 3.33 Motor characteristics

Power requirements

It is necessary to estimate the torque needed to steer the vehicle. Many parameters influence the power requirements as for example caster angle and camber angle. However, here the kingpin is vertical and just above the wheel centre and this removes many of the normal parameters. A convenient method, for calculating the maximum torque required for steering, is to calculate the torque necessary to steer on dry and clean concrete under stationary conditions [TRACTORS]. This is usually the heaviest steering load.

$$T = Wf\sqrt{\frac{I_0}{A} + e^2} \approx 750\,N \times 0.7\sqrt{\frac{(0.1m)^2}{8}} \approx 19\,Nm$$

f = 0.7, effective friction coefficient
I_0 = polar moment of inertia of tire print
e = 0, kingpin offset
A = tire print area

The exact shape of the tire print is not known and are therefore assumed to be circular, with a diameter equal to the nominal tire width, see Figure 3.31.

Steering motor

Demands and criteria's for choosing steering motors.

- Price
- Power requirement 19 Nm
- Intermittent service
- 24 V DC power supply, because the Honda motors have made it necessary to have 24 V power supply.
- The steering response (velocity) should be minimum 90 $^{\text{degrees}}/_{\text{s}}$ with a 19 Nm load
- Output shaft, which is easy to attach the kingpin to.
- Possibility of attaching encoder to the motor or gear

From NLB we have got at catalogue of inexpensive gear motors used for mechanical engineering in general e.g. in the automotive and agricultural industry. The motors have worm gears and are protected up to IP 40, see Figure 3.32.
The disadvantages of these motors are the low efficiency, preferred direction of rotation and that only torque loads are allowed on the gears.
The worm gears are self-locking, which can be both good and bad for this application. The advantage is that it might make the control less sensitive to terrain irregularities. The disadvantages are, that it can cause stability problems, but the risk should be small for relatively slow control systems [SERVO]. Another problem is, that the strength of the gear is not very high and the gear could be damaged by big torque loads on the kingpin.
To control the motors it is necessary with a power amplifier and a sensor to give position feedback. Some of the motors have a feedback option, but the resolution is only 4.6 degrees/pulse and there is no index to initialise the system.

A list of possible solutions is shown in appendix 3.2.14.
The motor 403 939 AR from the 0277(SW2K) series is chosen; because it meets the above demands, see the motor characteristics on

a:

b:

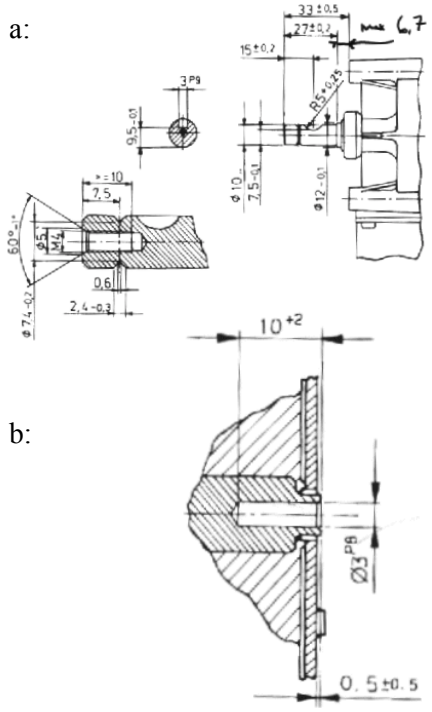


Figure 3.34 The adaptation possibilities.

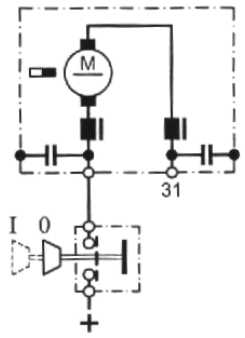


Figure 3.35 Motor diagram.

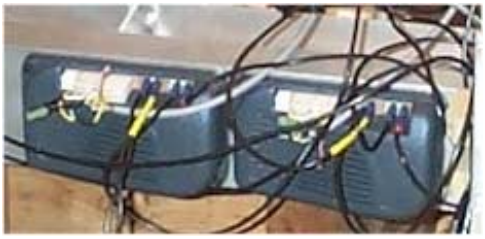


Figure 3.36 Curtis 1228 amplifiers

Figure 3.33. The gear strength of this motor is specified to 25 Nm. This solution have some drawbacks, but have been chosen because of the very low price at 357,- DKK. We will then try to compensate for the drawbacks in the mechanical and electrical design and in the software. The motor should be built in, so that it can be easily replaced if the gear breaks or problems occur with the self-locking worm gear.
The shaft is splined and is easy to make adaptations for, see Figure 3.34; a. Opposite the main drive, there is a additional drive possibility, which can be used for sensors, see Figure 3.34; b.
The wiring diagram for the motor is shown on Figure 3.35. The two capacitors are used as interference suppression filters, but when connected to a pulse width modulated amplifier the capacitors will get very hot and melt. After having melted the capacitors on one motor, they were removed on the 3 other motors.

Steering motor power amplifier

To control the steering motors it is necessary to amplify the control signal from the computers D/A converter to the power level required for driving the motors.
Our demands and criteria's for choosing power amplifier are:

- Linear
- 4-quadrant control
- Min. 12 Amps
- Low power consumption
- Low cost
- Motor protection, current limitation

For power requirements up to a few hundred watts, transistor amplifiers are widely used, but the power losses are significant. To minimize the power loss PWM (Pulse Width Modulation) amplifiers are used, these amplifiers can be used for very high powers. The principal of the PWM method is to switch battery voltage on and off with a high frequency (kHz sampling rate). The average output DC-voltage is controlled by adjusting the pulse width [SERVO].
Because of our requirement for low power consumption, we have chosen to use PWM amplifiers for controlling the steering motors. UB-Let can supply inexpensive 70A PWM 4-quadrant amplifiers from Curtis Instruments Inc., dedicated for wheelchairs, which seem suitable for this vehicle, see Figure 3.36.
The amplifier can be programmed to be linear and to limit the current to max. 20A. The 20A current limit is a little to high, but in the software we also have the capability to check whether the motor is turning or not, when it should. This information can also be used for protection of the motors. The amplifier is easy to build in and is protected by a rugged die-cast housing. The amplifier seems ideal for the vehicle and 4 amplifiers are bought from UB-Let.

Coupling

The gear bearings can only handle torques and can therefore not be used to support the kingpin. Since it is not possible to align the motor shaft and the kingpin perfectly, it is necessary to use a coupling between the motor shaft and the kingpin.
Demands and criteria's for choosing the right type of coupling:

- Little backlash
- Able to transmit a torque on 25 Nm

- High torsional stiffness
- Low price
- Compact
- Parallel misalignments up to 0.25 mm
- Easy to adapt

Different coupling principles are investigated, see appendix 3.2.16.a-c and two types are chosen as the best for this application, see Figure 3.37.

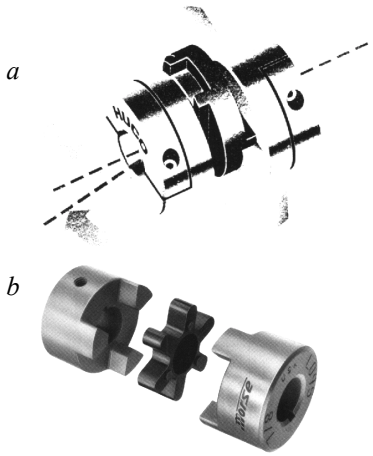


Figure 3.37 a; Oldham coupling, b; jaw coupling

The Oldham coupling is chosen because it has zero backlash, smaller dimensions, higher torsional stiffness and very small restoring forces, which leads to low bearing loads.
Clamp style shaft attachment is chosen because it is best for applications with reversing torque and vibrations. The discs are of acetal, because this gives higher torsional stiffness and long backlash-free life, compared to nylon discs. “Thro' bored” hubs are chosen, because they have 2-3 times longer backlash-free life, due to a protective coating, with lower abrasion factor.

It is difficult to follow the selection guide in the Oldham coupling documentation totally because the duty time and the reversing torque are very difficult to estimate, see appendix 3.2.17. If the maximum torque (19 Nm) and a duty time on 3 hours are estimated, it will be necessary with a size 57 coupling, which is very expensive. Therefore, we have only looked at the static break torque and the peak torque and estimated, that a size 33 will be able to do the job. The properties of a size 33 coupling are static break torque on 53 Nm and a peak torque on 9.0 Nm (The coupling should sustain minimum 10^6 peak torque reversals). This size of coupling might also give a better overload protection of the gear even though the static break torque is twice the strength of the gear. Another possibility of overload protecting the gear is to tighten the clamps so that torques higher than 25 Nm will make the gear shaft slide in the coupling clamp [OLDHAM].

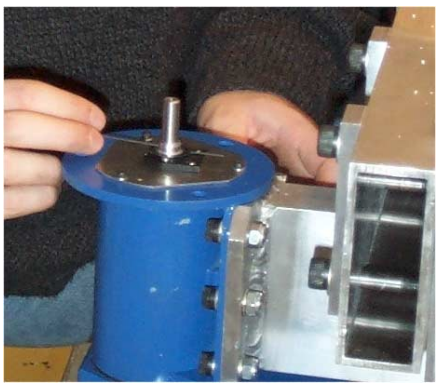


Figure 3.38 Assembly of encoder to the kingpin.

Sensor for the steering motors

To control the steering angle of each wheel, it is necessary to have position feedback on the kingpin, gear or motor. Encoders are optimal for position feedback. The most common are incremental quadrature encoders, which can only give a relative position and the direction. Here we want to measure the absolute steering angle and therefore we need an encoder with an index. US-digital, who also supplies the drive motor encoders, has an encoder with index and a resolution of 2048 counts per revolution. The encoders are quick and easy to assemble and protected by a rugged housing, see Figure 3.38. They can be supplied for shafts diameter from 2 mm. to 1" and are easy to align. They can also supply a PC ISA-BUS interface card for 4 encoders, which will make the implementation simple. We had first made a layout for a prototype card for interfacing the encoders, but this is nearly as expensive and could cause many problems.

The solution from US-Digital is the best and cheapest solution we can find and the encoders and interface cards are chosen. See “Encoders module” on page 89.

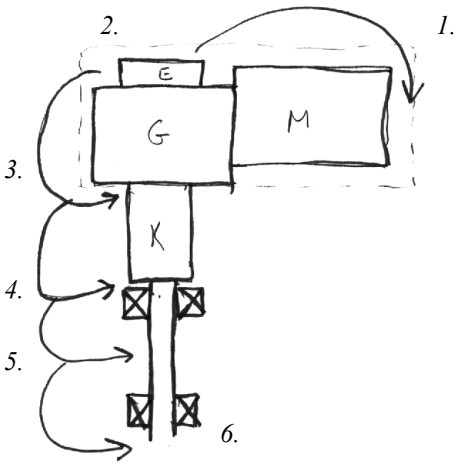


Figure 3.39 Placement of encoders

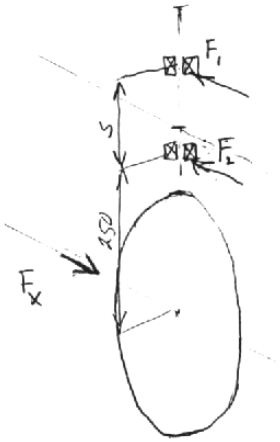


Figure 3.40 Bearing forces

Figure 3.41 Self lubricating slide bearings.

Placement of the encoders

The placement of motor, gear, coupling and kingpin with bearings are given by the nature of the machine elements, we are using. There are several possibilities for building in the encoders, which can be seen on Figure 3.39. Position 1, 2 and 3 have some common disadvantages.

- The possibilities of replacing the motor and gear are constrained.
- Because of backlash in the gear and elasticity in the gear and the coupling, there will be errors on the steering angle feedback to the controller.

The advantages of position 1 and 2 are easy access to the encoder and simple wiring.
Positions 5 and 6 also have common disadvantages:

- Difficult access to the encoder.
- It is necessary with a very big diameter encoder, which will be more expensive.

Position 4 has the advantage, that an inexpensive encoder can be used with only 10 mm diameter through hole. The encoder will be measuring directly on the kingpin, which will give the best measurement. A drawback is that the construction has to be designed to give good access to the encoder and possibility of suitable wiring.

The encoders are built in at position 4 and are ordered with a 10 mm diameter through shaft hole, 2048 counts/rev, index and centring tool.

Bearings

The kingpin has to be fitted in two bearings.

Forces

The radial forces acting on the kingpin bearings are shown on Figure 3.40. The worst case loads must be transmitted to the chassis through the kingpin bearings.

The worst-case horizontal load on the wheel module is $F_{x,\text{ worst-case}} = 4$ kN and the worst case axial load is $F_{t,\text{ worst-case}} = 4$ kN.
The radial bearing forces is calculated by using the formulas for momentum equilibrium. The length from the wheel centre to the lower bearing is estimated to l = 250 mm. and the length between the bearings is s = 100 mm. The sign of the horizontal reactions F_1 and F_2 is not important.

$$F_1 = \frac{F_x \times l}{s} \approx \frac{4kN \times 0.25m}{0.1m} \approx 10kN$$

$$F_2 = \frac{F_x \times (l+s)}{s} \approx \frac{4kN \times (0.25m + 0.1m)}{0.1m} \approx 14kN$$

The vertical load $F_{t,\text{ worst-case}}$ is only going to be carried as axial load of one of the bearings.

$$F_a = F_{t,worst-case}$$

If the vehicle is lifted, the axial load on the bearings, from the weight of a wheel module, must be carried.

$$F_{a,weight} = m_{wheel\,module} \times g \approx 23kg \times 9.81 \frac{m}{s^2} \approx 0.23kN$$

Bearing type

Different types of bearings can be used as kingpin bearings. We have found two suitable types of bearings for closer examination: self-lubricating slide bearings and ball/roller bearings.

The inexpensive slide bearings can take big static loads and can easily handle the small velocities, see Figure 3.41[11]. The only drawback is the relatively high static friction coefficient, which is about 6 times bigger than for ball or roller bearings. This kind of non-linearity should be avoided or minimized in control systems. The friction in the bearings is though of minor importance compared to the wheel – ground friction.

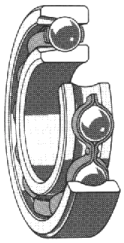


Figure 3.42 Deep grove ball bearing

Ball or roller bearings have very low friction coefficient and good carrying capacity. It is a very common and durable solution. We made inquiries about bearing properties and prices to SKF and found some bearings, which could be used. The kingpin is loaded both axial and radial. Many different bearing configurations can handle this, but if deep grove ball bearings are used both the upper and the lower bearing can be the same type and size.

The best way to ensure the alignment of the bearings is to make the bearing house out of one piece on a lathe or milling machine. To get the lowest weight, good strength and encapsulation of the bearings it is best for this application to make a thin-walled part, which is symmetric around its axis of rotation.

On Figure 3.43 two solutions for the bearing design is shown. On the left half a deep grove ball bearing is used for carrying the axial ($F_{a,\,worst\text{-}case}$) and the radial load ($F_2$) a smaller diameter roller bearing is used for the radial force (F_1) and the small axial force ($F_{wheel\,module}$). The roller bearing has a smaller diameter because it can carry bigger radial forces and the actual axial force ($F_{wheel\,module}$) is small. On the right solution two deep grove ball bearings are used and therefore the blank for machining can have a bigger inner diameter. Therefore, the part can be lighter and less material must be removed.

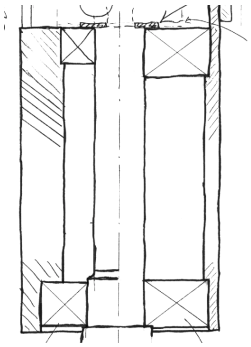


Figure 3.43 Two ways (left and right half) of implementing the kingpin bearings.

For the kingpin bearing house it is chosen to use two deep grove ball bearings and make a thin walled part, which can be machined from a blank tube.

Bearing size

Before searching for a suitable bearing the diameter of the kingpin must be calculated. The criteria are that yielding must not occur, when the part is subjected to the worst-case load $F_{x,\,worst\text{-}case}$. This load is causing a maximum bending moment M_b just below the lower bearing.

$$d = \sqrt[3]{\frac{32 \times M_b}{\pi \times \sigma_{0.2}}} \approx \sqrt[3]{\frac{32 \times 4kN \times 0.25m}{\pi \times 235MPa}} \approx 0.035m$$

- d is the diameter of the kingpin just below the bearings
- $\sigma_{0.2}$ is the yield stress

A trade-off is made between the bearing cost and weight and the criteria above and the kingpin shaft diameter is chosen to 30 mm.

[11] Appendix 3.2.18

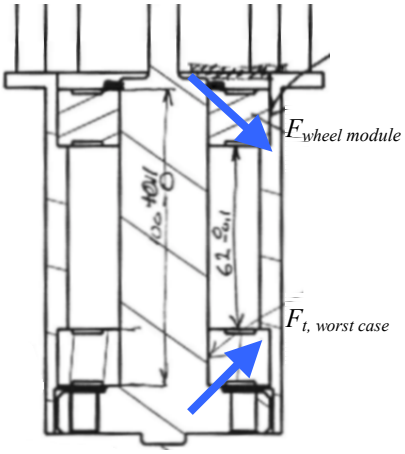


Figure 3.44 Bearing arrangement and force flow

The bearing 6306-RS1 is chosen from the SKF General Catalogue, this particular bearing is made in big numbers and is therefore quite inexpensive. The bearing has one seal and should be built in so the seals prevent dirt from damaging the bearings. The dynamic load rating is C = 28.1 kN and the static load rating is C_0 = 16 kN. C_0 is higher than the radial worst-case force F_2 = 14 kN, which can occur when the kingpin is not turning, and can therefore cope with the worst case static conditions radial. The axial load carrying rate of deep grove ball bearings are: $F_a = 0.5 \times F_r$ and therefore can the maximum axial force $F_a = F_{t,\ worst\text{-}case}$ = 4 kN easily be coped with.
A calculation of the life time of the bearing shows, that the bearing even under the worst normal conditions will have a very long life, see appendix 3.2.20.

Radial location of the bearings

The bearing arrangement is cross-locating, which means, that the shaft is located with a Seeger ring in the top and a flange on the shaft in the bottom. The axial force $F_{wheel\ module}$ is carried by the top bearing and the axial force $F_{t,\ worst\text{-}case}$ is carried by the lower bearing, see Figure 3.44
The load on the inner ring of the bearings is stationary, because normally the radial force will be perpendicular to the wheel shaft and the inner ring of the bearing rotates with the wheel. SKF recommends a clearance fit g6 for the kingpin and the inner ring of the bearing, which makes axial displacement possible, see Figure 3.45.
The load on the outer ring is rotating and SKF recommends a N7 interference fit for the housing. Because the loads normally are small and to make the mounting easier, the fit is changed to N6 in co-operation with the workshop.

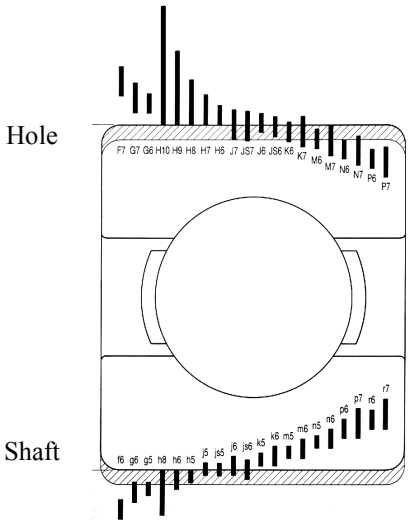


Figure 3.45 Bearing tolerances

Steering mechanism, Part design

The function of the housing is to get a good flow of the forces from the wheel frame to the chassis and to make it possible to give a rotational steering input to the kingpin.
Demands and criteria's for the housing:

- Adequate strength and stiffness
- Protection of the electrical and mechanical components, minimum IP54
- Easy to assemble and do service
- Low price
- Easy wiring

Installation of motor, coupling and encoder

To install the motor to the kingpin, it is necessary to make a special rack, which can support the torque load from the motor and protect the coupling and encoder from dirt and moisture.
The rack can either be made as a solid on a lathe (Figure 3.46) or as a stay solution with a separate shield as protection (Figure 3.47)

Solid part symmetric about its axis of rotation

The benefits of this solution are, that the part to support the motor also protects the coupling and the encoder. After having worked with the solution, some major drawbacks are found and the solution is rejected. The problems are to get access to the encoder and coupling in an easy way, which can be very useful on test equipment like this vehicle. The

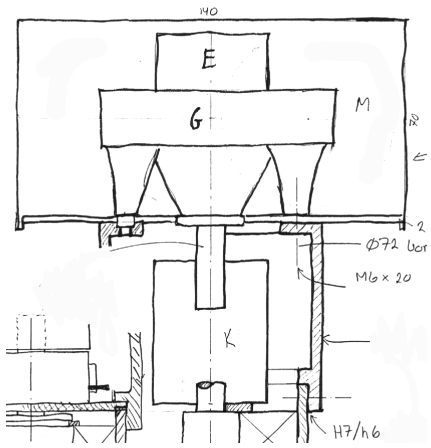


Figure 3.46 Solid alu. part, symmetric about its axis of rotation.

assembly will also be difficult and require that holes are made to get access to the screws, for tightening the coupling clamp. The part would also be very difficult to machine and require the purchase of new tools in the workshop, see Figure 3.46[12].

Stay solution

A very simple time saving solution to make in the workshop is to make some stays to carry the motor, see Figure 3.47[13]. If the motor bracket has countersunk holes for the screws to assemble the motor bracket and the stays, the parts will be automatically aligned. If, countersunk screws are also used for assembling the stays to the kingpin bearing housing, the kingpin and the gear shaft will be easily aligned. Good alignment of the kingpin and the gear shaft will give less wear and friction in the coupling. Another advantage of the stay solution is very easy assembly and access to watch and adjust the encoder and the coupling.
The disadvantage of the solution is that a separate shield must be made.
Three stays are made for each wheel module. The stays are made in aluminium with a relatively big cross-section to give a good surface of support and to give the construction good stiffness.

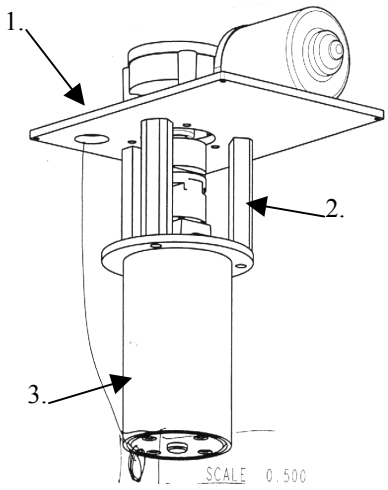


Figure 3.47 Stay solution: 1; motor bracket, 2; Stay, 3; Kingpin bearing housing.

Motor bracket

The motor bracket is aligned with the kingpin, because of the countersunk screws in each end of the stays, see Figure 3.47. The three stays on the steering motors are machined at the ends for alignment with the application.
The motors are the same and we want the parts from each wheel module to be interchangeable, but because the left and the right wheel modules are mirrored, the motor brackets have to fit stay arrangements, which also are mirrored, see Figure 3.48[14].
Therefore, there is made 6 holes in the motor bracket with 60° between the holes, so that in the left side of the vehicle 3 holes are used and in the other side the other 3 holes. A reason for this is that we want the vehicle to look symmetrically about its longitudinal axis and not have the motors in one side point forward and backward on the opposite side.
The centre hole for the gear shaft is placed on the centreline of the motor bracket to have the steering motors with enclosure look the same on each side. This placement of the stays is also chosen to make it easy to screw the stays to the flange on the kingpin housing.

Wiring

The wires coming from the drive motor and out of the hole in the motor frame is guided to the motor bracket. The idea is that all the wires from the wheel module are going to be gathered in the enclosure for the steering motors. All the wires from one wheel module is then put into a flexible conduit and connected to the power supply and electronics on the chassis. The purpose of the conduit is to protect the wires from damage and to prevent the steering motor from turning more than about 145° to each side. If an error happens which will make the steering motor continue to turn the wires would be damaged,

[12] Appendix 3.2.22-24
[13] Appendix 2.2.25-26 and drawing 1.1.1 and 1.1.1.1
[14] Appendix 3.2.26-27 and drawing 1.1.1 and 1.1.1.4

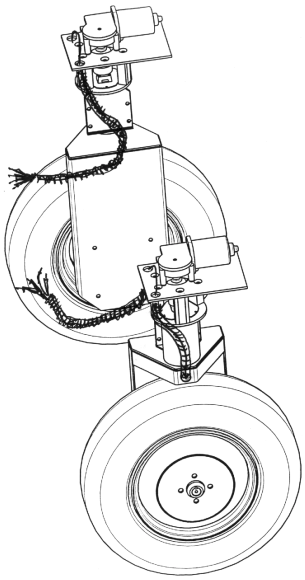


Figure 3.48 The wheel modules are mirrored on the vehicle.

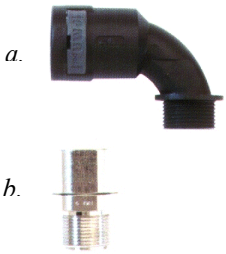


Figure 3.49 a; 90° elbow, b; swivel.

Figure 3.50 The wires are going through a conduit to the motor bracket.

if precautions are not made. From the company Bagger-Nielsen we found a conduit, which should offer excellent flexibility and be ideal for robotics. A inside diameter on 16.4 mm. should be enough for the wires. At the end where the conduit is connected to the motor frame, it would be best with a 90° elbow and a swivel to allow the elbow to turn a little. This would make it easier for the steering motors to turn and reduce the necessary bending radius of the conduit.

When we received the conduit and connectors, we found that the inside diameter of the 90° elbow and the swivel were decreased to only 10 mm., which is too little for the wires. Therefore the conduit is used from the motor frame to the motor bracket, without the swivel and the 90° elbow, see Figure 3.50. From the motor bracket to the front and rear module it was found that the inside diameter of the conduit was too small and the cables are only protected here, with a rubber ring in the motor bracket. This can accepted because the cables is in a position, were they are well protected.

Protection of motor, coupling and encoder

We have decided not to cover all the parts with one shield or enclosure, because the wheel module and the chassis should be independent modules, which could be replaced or changed. Therefore, the wheel module should have its own enclosure. The motor, coupling and encoder could either be in one box or the motor could have its own box and a cylindrical enclosure could be made for the encoder and the coupling.
We have chosen the last solution. The solution with the motor bracket has made it easy to make an enclosure to fit to the motor bracket. The coupling and encoder are enclosed with a thin aluminium plate, rolled to fit around the stays[15].

Interface between the wheel module and the chassis

The kingpin bearing housing is bolted to the chassis. A plate with 6 low clearance holes are welded to the bearing housing, which is used for assembling and aligning the wheel module and the chassis. Different methods for assembling the wheel module and the chassis are sketched in appendix 3.2.28, but the method chosen was estimated to be the simplest and easiest method, see Figure 3.50.

Kingpin bearing housing

The bearing housing is made out of cylindrical blank with a flange welded to it and then machined. Then the plate for bolting it to the chassis is welded to it. The flange has three countersunk holes for assembly with the stays. To protect the bearings from water and to make the wheel modules look better, the housing is covering for the flange on the kingpin, see Figure 3.52[16]. The strength is ensured by calculating stresses in critical areas like screw connections and bearing supports[17].
To prevent corrosion and to get a good finish of the vehicle all the iron parts must be painted. The parts are painted blue, because we think,

[15] Appendix 3.2.21-22 and 3.2.31
[16] Appendix 3.2.19-29 and drawing 1.1.1.5 –1.1.1.5.4
[17] Appendix 3.2.29

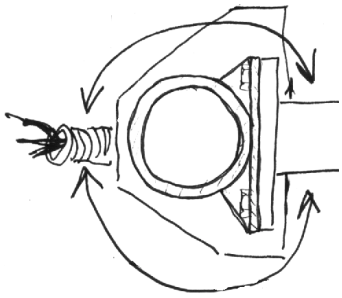


Figure 3.51 Wheel module and chassis mechanical interface (top view)

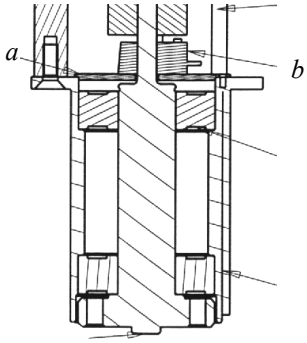


Figure 3.52 Cut-away projection of the kingpin bearing housing. a; encoder plate, b; encoder.

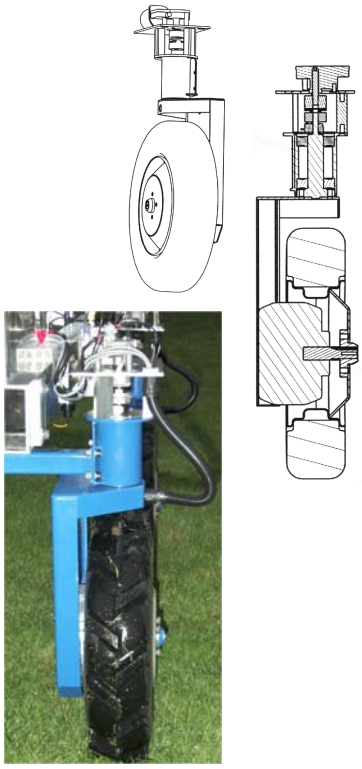


Figure 3.53 The wheel module

that the blue colour and the aluminium surface match very well, see Figure 3.50.

Bracket for the encoder

An aluminium plate is made for mounting the encoder to the bearing housing, see Figure 3.52[18]. A recess in the plate is aligning it radial with the kingpin. Ensuring that the encoder is mounted so that the shaft eccentricity and the radial play are below 0.10 mm.
Originally, the bearing housing with flange should have been made out of one blank, but the workshop did not have a suitable blank and made it by welding a flange to a cylindrical blank. This caused a welding where the threaded holes for the encoder plate were going to be made. This caused some difficulties with making the holes precisely and the plates had to be corrected a little to be assembled.

The kingpin

The tap on the kingpin is made to make it easy to align and assembly the motor frame and the kingpin, see page 37. The diameter of the flange is limited by the diameter of the bearing housing, which is made to fit the bearings chosen.
The screws and the thickness of the flange are dimensioned to handle the worst-case force. The screws can be prestressed to also have the function of overload protection. The screws will then be overload and break.
The kingpin has a little collar above the flange to support the bearing.

Summary of the wheel module

With the design of the wheel module, we have achieved our objectives.

- Narrow design optimal for driving in row crops.
- Making it possible to have a ground clearance between the wheels on 500 mm
- Allows the wheels to be turned ± 145° making it possible to drive the vehicle sideways and make very narrow turns.
- The protection rate of the steering motors, encoders and couplings are not achieved yet. The enclosures for these parts have not been designed.
- Very stiff frame construction, with nearly no play in the bearing arrangement.
- Little play or backlash in the gear and coupling arrangement.
- During testing the couplings have slipped from time to time, but because the steering encoders are placed on the king pins it is not necessary to adjust the encoders again.

[18] Appendix 3.2.19 and drawing 1.1.1.2

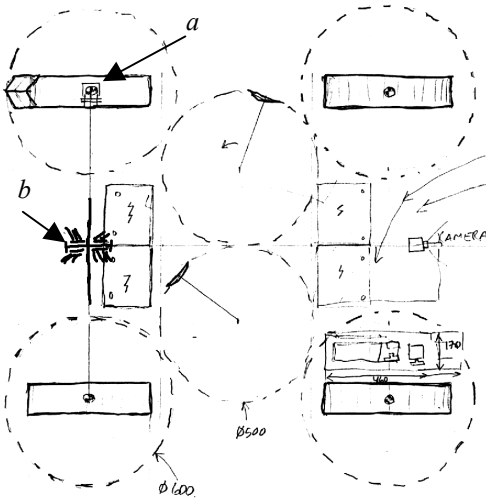


Figure 3.54 Overall principle and quantitative structure, a; kingpin joint, b; longitudinal pivot joint

Figure 3.55 Dryfit batteries from Exide, red ring around the chosen type.

3.3. Chassis

In this part, the mechanical concept for the chassis will be designed in detail.

Summary of overall principle and overall quantitative structure

On Figure 3.54 the principal structure of the vehicle is shown. The kingpin (Figure 3.54.a) is just above the wheel centre and the distance between the kingpin centres is 1000 mm both lengthwise and crosswise. Either the front or the rear axle should be attached to the chassis with a longitudinal pivot joint (Figure 3.54.b) to allow rotation about the longitudinal centre axis of the vehicle. This is made to make all the wheels follow the ground. In the middle there is made room for different types of tools. The ground clearance should be 500 mm between the wheel modules.

Quantitative structures

As summarized above the overall principles, structures and dimensions have been determined and here the placement and shape of the parts for the chassis are going to be determined.

Components which must be placed

The first step is to find the dimensions and special requirements regarding the placement of the different components.

Batteries

The batteries for this application have to be able to deliver a nearly constant current for 2 to 4 hours. Normal car batteries can only give a big current for a very short time. Therefore, we need to use traction batteries, which can deliver a constant current for a longer period.

The batteries should be able to supply the vehicle for about 2 to 4 hours, which should be long enough for research and test conditions.

Consumption:

Equipment	Comments	Consumption
Honda motors	4 x 6A (Unpaved)	24 A
Steering motors	4 x 2A (Min. current)	8 A
Computer and peripherals	1 x 4A	4A
Total		**36A**

The consumption, especially for the steering motors, is very difficult to estimate, because the consumption depends a lot, on how active the motors have to be to eliminate ground irregularities and other disturbances.
To operate, for 2 to 4 hours the vehicle will need about 70Ah to 140Ah. This is for driving on unpaved surfaces under changing conditions.

A trade off is made between the weight and cost of batteries and the estimated power consumption and two dryfit batteries 12V, 70Ah/5h

are chosen from Exide after having investigated other types and makes, see page 158 in “Literature”.
Dryfit batteries are chosen because they do not require maintenance, which is good because the vehicle is going to be operated by different people, who then do not have to be concerned about the batteries, see Figure 3.55. Other advantages are no waste of acid and nearly no exit gasses. The disadvantages are that the batteries can only be discharged 70% and the volume of the batteries is bigger than for standard batteries. The lifetime is also shorter than for standard batteries. The lifetime is 7 - 800 cycles at 70 – 60% discharge. If the batteries are not used for a longer period, they must be fully charged. The self-discharging time of the batteries is about 1 – 1½ year.
The dimensions of the chosen battery are: 330 x 171 x 236 mm and the weight is 30 kg.

The batteries are the heaviest part of the robot and have big influence on the weight distribution of the vehicle and should be placed near the vehicle centre.

Figure 3.56 High frequency battery charger from Exide.

Battery charger

To charge the batteries we have a high frequency charger for dryfit batteries, see Figure 3.56. The charger is of the constant current type, which has the advantage that the battery is nearly fully charged when the gas point is reached. This leads to shorter charging time. It was initially considered to place the charger on the vehicle, so that it would be very easy just to plug the robot to 220V AC. However, this was later given up to make room for a monitor.
The dimensions of the battery charger: 245 x 160 x 135 and the weight is 3.1 kg.

Figure 3.57 GPS Total Station 4700

RTK/GPS

The RTK/GPS equipment from Trimble to be placed on the rover (the robot) is made for outdoor use and has good protection from water, dust, etc.
The equipment for the rover is:

- GPS Total Station 4700, integrated GPS receiver and radio modem, see Figure 3.57.
- GPS antenna without ground plane
- Radio antenna
- Stand for the GPS antenna made of a 500 mm long aluminium tube with a diameter on 1" equal to 25.4 mm. The stand can be extended with more tubes.

This equipment has only been borrowed from KMS and we cannot be sure to get the same equipment again. Therefore, the 4700 will not be built in, but fastened to the vehicle in a convenient position.

The GPS antenna should be placed so that no part of the vehicle can shadow for the GPS signal. The antenna must be placed in a stable position to avoid problems with the dynamics of the vehicle.

Compass

The compass is very sensitive to hard-iron distortion and magnetic fields from electrical equipment (Compass on page 61).

Figure 3.58 Computer cabinet placed on the vehicle.

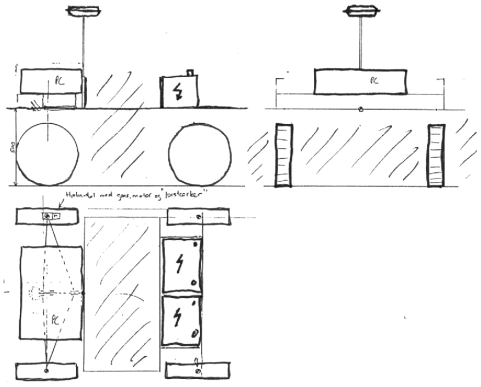


Figure 3.59 Quantitative structure; constraints

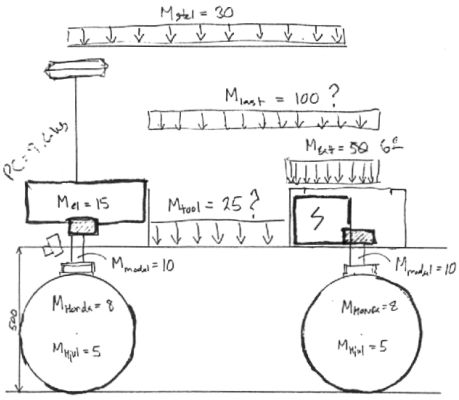


Figure 3.60 Estimated weight distribution

Initially the difficulties with finding a proper placement of the compass where underestimated and it was placed on the GPS antenna stand.

Computer

The used computer provided by the department has a big cabinet with room for several disc drives. There is good room in the cabinet for both the computer, power supply, error handling and interface, because the AC-to-DC power supply, cd-rom drive and floppy disk drive is removed, see Figure 3.58. The outside dimensions of the cabinet are: 520 x 390 x 185 and the weight is 9.6 kg. In addition, it is necessary with approximately 80 mm free space where the wires are going to be connected to the computer to get access to connecting and disconnecting cables.

The computer takes up most space and should be placed centrally to avoid problems with the wiring, because all the control signals are connected to the computer.

Monitor

It was decided to buy a small monitor to build into the robot, because it would make it easier to work with the robot. Earlier we had decided to use a Notebook and a programme like PC-AnyWhere for controlling the robot computer. A monochrome 5" monitor was found. The dimensions of the monitor are 224 x 110 x 153 mm and the weight is 1.5 kg.

It is necessary to find a good position for the monitor, where it is easy to work with the keyboard and the mouse during tests.

Camera

The camera should be placed near the front axle and in the middle of the wheels. The camera is very small and the dimensions are about 40 x 40 x 50 mm.

Placement of components

To place the components on the chassis some drawings is made with the constraints given. The constraints are the dimensions, wheel placement and functional areas for crops and tools, see Figure 3.59. The drawing is copied and different quantitative structures are made, see appendix 3.3.1.

The quantitative structure on Figure 3.59 is chosen after having calculated the weight distribution of the structure; see Figure 3.60[19]. The weight distribution is without tools and payload, which should be placed at the centre of the vehicle:

- The front axle load is about 96 kg
- The rear axle load is about 135 kg

The weight distribution is acceptable, because the biggest load is on the rear axle. It is an advantage, that the front wheels have the lowest axle load, because they will compact the soil and reduce the rolling resistance for the rear wheels. The batteries are placed in front of the rear axle to put the biggest load on the rear axle and to place the batteries as low as possible, but as close to the vehicle centre as the

[19] Appendix 3.3.2.a-c

tool area can allow. The computer and the electronics are then placed above the front axle to distribute the weight more even. The longitudinal pivot joint is made on the front axle, because the computer can be raised without increasing the height of the centre of gravity much.

Part design

Now the main components of the vehicle is roughly in position and we can start to design the parts of the chassis.

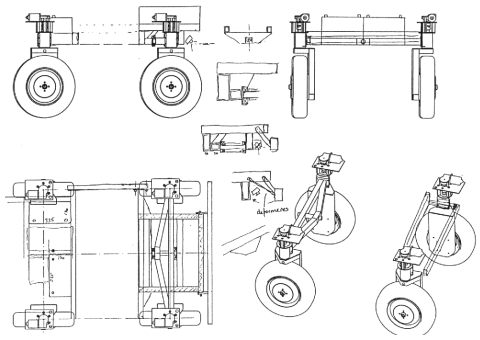


Figure 3.61 Concept with clear tool area

Structure of the frame

Two principles or concepts for the structure of the frame will be discussed. The first concept is that the functional area for the tools should not be crossed by any disturbing elements and the second concept has side members crossing the tool area.
At this stage of the design process the wheel module has been drawn in Pro-Engineer and some drawings like the one on Figure 3.61[20] is printed to support the sketching of alternative solutions.

Concept with clear tool area

It is tried to make a frame structure to support the concept of a clear tool area, see Figure 3.61[20]. This constraint makes it necessary to have beams lengthwise on the outside of tool area. To have long beams passing the wheel modules will give a fairly stiff construction. However, because of the wiring on the wheel module, it is not a good solution to have the beams pass the wheel modules on the outside. When looking at the force flow it is realized, that this concept will give poor stiffness or the vehicle will become very heavy.

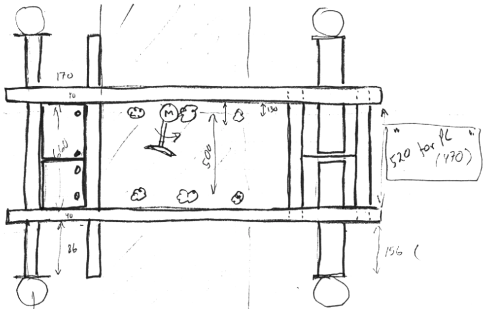


Figure 3.62 Concept with side members. (Rotating hoe tool attached to the left side member)

Concept with side members

This concept has side members (beams lengthwise) going all the way through the construction and tools, front part and rear part can be bolted to it, see Figure 3.62[21]. In this way the side members can be replaced or modified if it should be required by a future tool solution. The solution with the side members makes it possible to make both a stiff and light construction.
If the distance crosswise between the side members is 660 mm, there will be room for the batteries and good room for the computer and electronics. This distance is also convenient for attaching e.g. the rotating hoe directly to the side members. The side members is placed quite high to make it possible for the front axle to rotate enough about the longitudinal pivot joint to let the vehicle pass small obstacles, see Figure 3.63. This height also makes it possible to place e.g. a boom for a spot sprayer below the side members and still have 500 mm ground clearance.

Materials

It is chosen to use beams with rectangular thin-walled sections to build the chassis. Open-walled sections give very bad strength properties compared to close-walled beams, when the beams are subjected to torsion. To use plates to build the chassis will be very time consuming to design and make in the workshop.

[20] Appendix 3.3.3
[21] Appendix 3.2.4-5

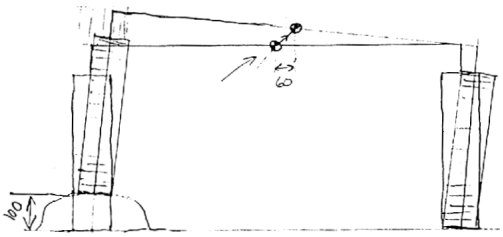


Figure 3.63 Passing an obstacle. Notice the rotation and movement of the front axle

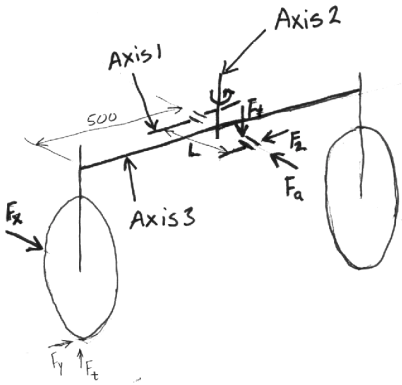


Figure 3.64 The forces on the front axle.

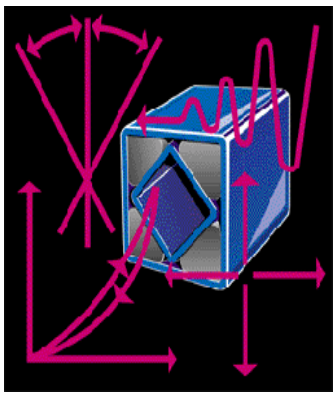


Figure 3.65 Rosta bearing properties; axial, radial and torsional spring with damping

It is chosen to use aluminium to reduce the weight of the chassis. The workshop has big experience in working with aluminium and the material is good for beams. An aluminium beam can be made with a higher cross section and therefore have a bigger EI (modulus of elasticity x moment of inertia) modulus than a steel beam with the same weight.

Front axle

The design of the front axle is an iterative process, were both the front axle, the front module and the rear module is changed, until a satisfactory solution is found. This means that results, which have not yet been explained, will be shown because they have influence on the actual problem being explained.
The front axle should be able to rotate about the longitudinal centre axis of the vehicle to make all the wheels follow the ground, when passing irregularities and obstacles. Figure 3.63 shows the movement of the front axle, when passing a 100 mm. high obstacle, see appendix 3.3.7. The vehicle should be able to pass obstacles, which is about 150 mm high, with all wheels following the ground.

Load analysis

It is assumed that both wheels are not subjected to the worst-case force $F_{x\ worst\text{-}case}$ at the same time. To get acceptable bearing forces it is necessary not to have too small a distance (L) between the bearings. The force F_x is causing both an axial and a radial bearing force. The radial force is both due to the moment about the $axis_1$ and the $axis_2$, see Figure 3.64. The force of gravity is small and hence neglected and therefore the total worst-case bearing forces are:

$$F_a = F_{x\,worstcase}$$

$$F_t = \frac{F_{x\,worstcase} \times 0.25m}{L}$$

$$F_2 = \frac{F_{x\,worstcase} \times 0.50m}{L}$$

$$F_r = \sqrt{F_t^2 + F_2^2}$$

Choice of bearings

To minimize vibrations and shock loads it would be good to use bearings with damping. Normal ball bearings and slide bearings have very little damping, but it is possible to make bearings with rubber as spring and damping material, see appendix 3.3.9. From Jens-S we have information about Rosta bearings, see Figure 3.65.
The Rosta elements are chosen because they are very easy to built in and do not need very fine fits and provides damping.

Choice of Rosta bearing

One of the described ways to install the Rosta bearings is to use friction [ROSTA]. Installing the bearings in this way means that they can be used as overload protection. The screws should be tightened to a level where the bearings will slide if the vehicle is subjected to the worst-case force.
To be able to use as small and lightweight bearings as possible it is chosen to make the distance (*L*) between the bearings as big as

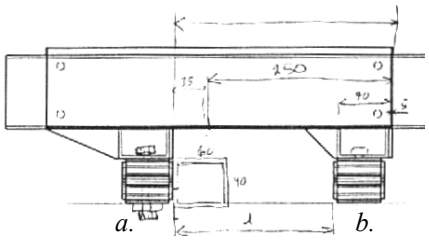


Figure 3.66 Placement of the Rosta bearings (Side view).

possible, see Figure 3.66. Bearing (*a*) is placed as close to the front axle as possible to get the camera close to the front axle. Bearing (*b*) is placed as close to the tool area as possible to increase the distance (L) between the bearings to decrease the bearing forces. The big distance between the bearings will also make the unwanted deflections due to rotation about $axis_2$ and $axis_3$ smaller, see also Figure 3.64.
Other types of Rosta bearing arrangement were also investigated, see appendix 3.3.10. The arrangement with two individual bearings was found to give the smallest deflections under normal conditions, to be easiest to install and to give the lightest construction. We have chosen to use the bearing DR-S 27 x 40:

Force direction	Max. load [N]	max. pitch [mm]
Radial	1300	0.5
Axial	300	0.5

The max axial load on 3 x 300 N can easily be exceeded, which will damage the bearing. Therefore, some collars are placed on the front axle to make direct metal-to-metal contact if the 0.5 mm pitch is exceeded, see Figure 3.70;3;6. Radial the design of the Rosta bearing prevents overload by internal metal-to-metal contact.

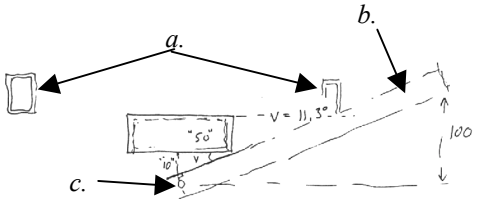


Figure 3.68 Constraints on the turning angle of the front axle. a; side members, b; front axle, c; longitudinal pivot axis.

Design of the front axle

For the vehicle to be able to pass obstacles the front axle have been chosen to be pivotally mounted. We have estimated, that it would be reasonable if the vehicle can pass 150 mm high obstacles. This means that the front axle should be able to turn 8.6°.
The concept chosen puts some geometrically constrains on the way this 8.6° turn can be realized, see Figure 3.68. The distance between the side members is constrained by the room needed for the batteries. The longitudinal pivot axis lowest position is constrained by the demand for 500 mm ground clearance and the height of the front axle. The parameters that can be changed are the height of the side members above the longitudinal pivot axis and the height of the front axle.

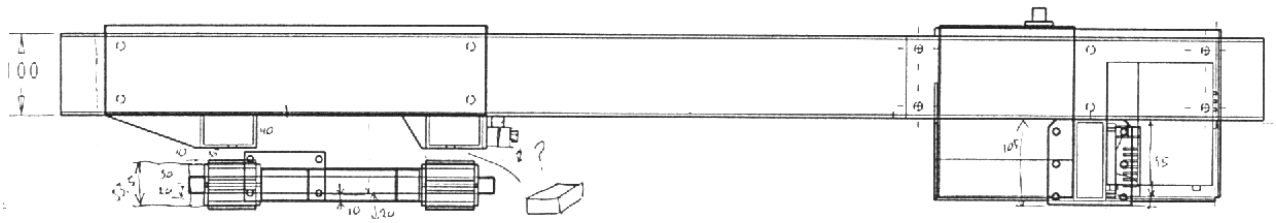


Figure 3.67 Placement of the front axle on the chassis

It has been chosen to use rectangular aluminium beams for the side members with the dimensions 100 x 40 x 4. This beam is also used for the rear axle, which is placed below the side members, see Figure 3.67. This structure gives good possibilities for the front axle to turn. If the height of the front axle can be limited to 40 mm., there will be 60 mm from the top of the front axle to the bottom of the side members. To make the front axle turn 8.6° it is necessary with minimum 58 mm., so this seems to be a good geometrical solution.

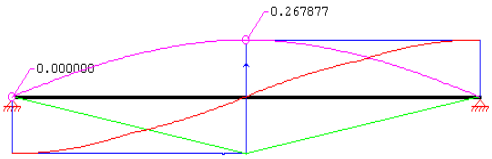


Figure 3.69 Diagram from WinBeam, showing shear (blue), Moment (green), rotation (red) and deflection (purple)

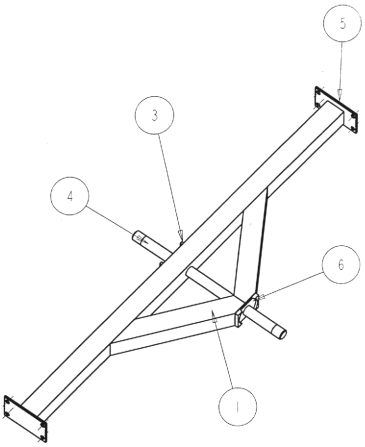


Figure 3.70 The final design of the front axle.

Figure 3.71 Bolting the Rosta bearings to the front module.

It is preferred to use some of the different beams available in the workshop and the available beams with a height of 40 mm. is: 40 x 40 x 2, 40 x 40 x 3 and 70 x 40 x 2. With a small dimension used for the main beam (see Figure 3.70) the stresses due to bending and torsion are high and therefore support beams are added. The support beams make the construction much stiffer and increase the moment of inertia where the bending and torsional moments are biggest.

Strength of the front axle

The cross-section properties and stresses are calculated in Matlab and in the finite element programme WinBeam, the deflections and rotations are calculated.
In appendix 3.3.11.a-c it can be seen that the stresses without support beams are on the limit of failure under the load $F_{x,\ worst\text{-}case}$, when a 40 x 40 x 3 beam is used. The 40 x 40 x 2 beam is used for support beams, which give a better force flow to the bearings. This also reduces the deflection and rotation due to the force $F_{gravity}$ about the longitudinal pivot axis (Figure 3.68) to:

- $Rot_{gravity} = 0.05^{\circ}$
- $Def_{gravity} = 0.26$ mm

The rotation causes the gauge between the front wheels to be 0.9 mm longer at the ground, which is acceptable.
The rotation about $axis_2$ could cause an error on the encoder measurement of the steering angle. However, the rotation must be smaller than rot_{tyngde}, which is acceptable, because the force $F_{x,\ normal}$ is smaller than F_{tyngde}. The moment of inertia is also much bigger due to the support beams.

Bolting the wheel modules to the front axle

The wheel modules are bolted to some flanges, which are welded to the ends of the front axle, see Figure 3.70. The flanges are 4 mm thick like the flanges on the wheel modules. In appendix 3.3.13, it is shown that the screw connection can easily handle the worst-case forces and that the screws will break before the flanges are damaged.

Bolting the Rosta bearings to the front axle

The Rosta bearings have been chosen with a 20 mm through hole, because an axle with threads in the ends is the easiest way to assemble the front axle with the bearings, see Figure 3.70. Some 4.5 mm collars are welded to the front axle to prevent the Rosta bearings from being damaged by axial displacements bigger than 0.5 mm.

Bolting the Rosta bearings to the front module

The Rosta bearings are held in place by friction. Two bolts through a U-profile and the front module give the necessary pressure on the bearings, see Figure 3.71. In appendix 3.3.14**,** the necessary tightening torque is calculated to be 9 Nm. The strength of the U-profile is shown in appendix 3.3.14-15. When the screws are tightened with 9 Nm, the overload protection can handle F_x forces up to about 4 kN. These calculations are only estimates, because the friction coefficients are very uncertain.
A 10 mm thick aluminium plate is placed between the bearing and the front module to give the right distance and to reinforce the front module and front axle assembly when tightening the screws.

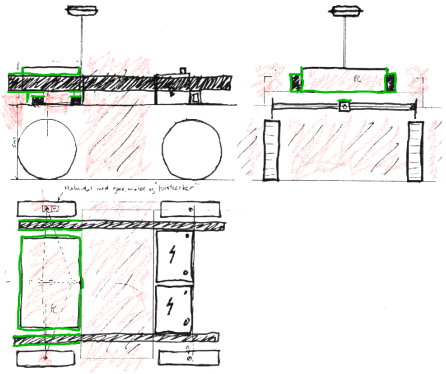


Figure 3.72 Functional surfaces are marked with green and forbidden areas are marked with red.

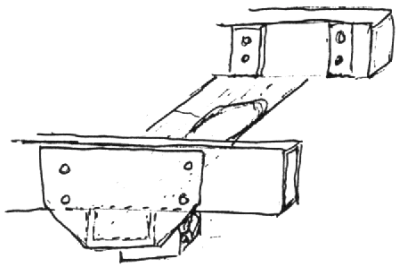


Figure 3.73 Structure to connect the front axle and the side members.

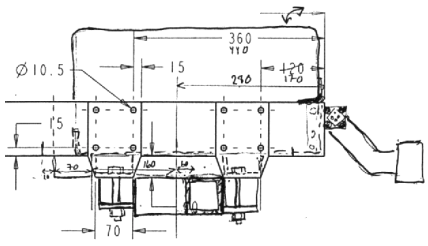


Figure 3.74 The front module with computer enclosure and front axle.

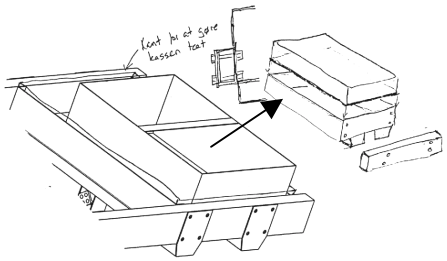


Figure 3.75 The front module as an independent box.

Design of the front module

The front module should connect the front axle with the chassis, contain the computer and electronics and have a retainer for the compass and GPS antenna.
Demands and criteria's

- Enclosure for computer and electronics IP 54
- Easy access to computer and electronics
- Strength to handle the forces

Structure

During the design of the front axle, more dimensions and structures have been determined for the side member concept. On Figure 3.72 the functional surfaces and forbidden areas with relation to the front module are sketched. The red colour marks the space, which is occupied with crops, tools, the wheel modules, the box with computer and electronics and the front axle. The surfaces marked with green are the functional surfaces, which needs to be connected by the front module.
The front module should connect the front axle bearings to the side members to support the weight of the vehicle and to support the front axle when loaded by driving and worst-case forces. It should enclose the computer and electronics and have mounting possibilities for the camera and the retainer for the GPS antenna and the compass.

It is decided to use two beams with a channel cross-section as the load carrying members of the front module. The Rosta bearings are mounted to the beams and the beams are mounted to the side members with a sheet metal bracket, see Figure 3.73 and Figure 3.74.

The computer and the electronics must be protected from rain and dust; therefore, it is decided to make an enclosure in sheet metal. The dimensions of the computer is 520 x 390 x 185 mm and it is necessary with to add 80 mm to the 390 mm to make room for wiring and to connect cables to the computer. There is 680 mm between the side members so the computer can be placed with its long side across the vehicle, with the connectors pointing backwards.

The sheet metal plate is welded to the two beams to increase the stiffness and the strength of the front module. To make a cover for the computer box, it is necessary that the sides of the box are higher than the side members, to prevent water from seeping into the enclosure, see Figure 3.75. To do this the front module is made as a box, which the side members are screwed to the sides of; see Figure 3.75 and the final result on Figure 3.76[22].

Force analysis and dimensions

The box structure with the two beams (Figure 3.76; 3) welded to the box gives a good force flow from the front axle bearings to the side members.
An aluminium beam with channel cross-section and the dimensions 70 x 40 x 4 mm is on stock in the workshop. Because the beam is going to be welded to the bottom plate (Figure 3.76; 1) and the side plates (Figure 3.76; 2) it is calculated in WinBeam as a rectangular fixed beam. In appendix 3.3.19 the results are shown, the beam is loaded as

[22] Appendix 3.3.17-18 and drawing 1.2.3 – 1.2.3.3

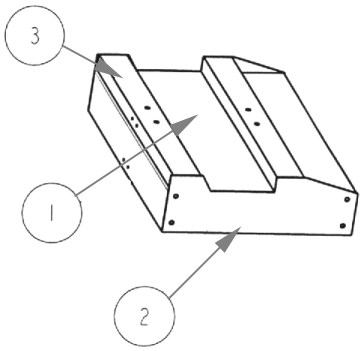


Figure 3.76 The final design of the front module. 1; bottom plate, 2; side plate, 3; channel cross-section beam.

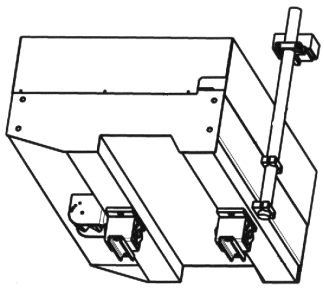


Figure 3.77 Front module with cover and with camera, Rosta bearings and antenna stand mounted.

Figure 3.78 Cables entering the front module.

if it is subjected to the total worst-case load. The load is assumed to go through the flexural centre of the beam and it is therefore only subjected to bending, which can be calculated in WinBeam. Under the given conditions shear can be disregarded and the stress due to the bending moment is about 50 N/mm^2, which is below the yielding stress for aluminium. The beams on stock are chosen for the front module.
The bottom plate is made of 2 mm thick aluminium and the side plates of 3 mm thick aluminium. The dimensions are chosen so it will not be too difficult to weld the plates and to give the plates adequate strength and stiffness.

Front module details

The front module has 4 holes for mounting the retainer for the compass and GPS antenna in the back, see Figure 3.76 and page 61. On the front beam of the front module there is two holes for mounting the camera, see page 60[23].
To make it possible to access the screws for mounting the retainer above, the camera and the front axle to the front module, 4 holes with 50 mm. diameter is made in the bottom plate[23].
In the front a button to reset the system and 3 LED's are placed. The green LED is on when the system is OK. The red LED is on until the controller has started. The yellow LED is not yet in use (see page 74 in “Computer and interface”).
It has later been decided to place the man machine interface at the rear module and therefore, it would be a good idea to place the LED's and the enable button or an extra set at the rear module.

Front module cover

The cover for the front module should provide IP54 protection of the electronics inside the front module and give easy access to the electronics. It was in dialogue with the workshop decided to make cover of one aluminium plate, which was bended and welded in the corners to make it strong and tight. The cover is carried by the side members and can be secured with screws.
The main problem is to get the wires to the electronics and still have IP54 protection. At the stage were the front module was designed, 2 conduits with all the wires were going into the front module. Now the wires from the wheel modules are not in a conduit and the wires from the rear module are coming out of a cable trunking on top of the side members. This makes it difficult to make the entry of the cables tight, when it should be easy to remove the cover for servicing. A cutout is made in each side of the cover for the wires, see Figure 3.78. The entries with the bundles of wires going through are going to be tightened with a rubber plate glued to the outside of the cover. The edges will also be shielded with rubber to protect the cables[24].
The best solution would have been if the side plates on the front module had been higher so that a tight entry could have been made in the side plates and not in the sides of the cover, which is often removed.

[23] Appendix 3.3.18
[24] Appendix 3.3.18.a and drawing 1.2.3.4

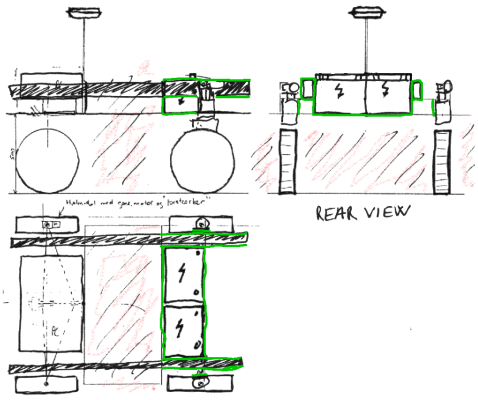


Figure 3.79 Functional surfaces are marked with green and forbidden areas are marked with red.

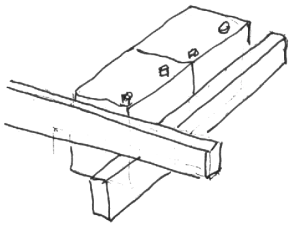


Figure 3.80 The basic structure of the rear module.

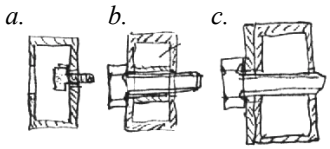


Figure 3.81 3 ways of mounting a thin-walled rectangular beam.

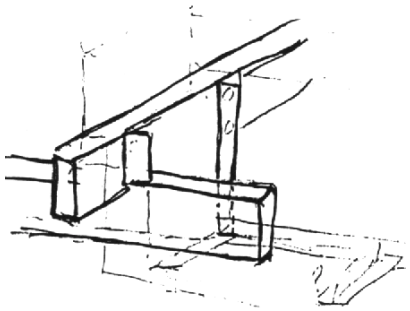


Figure 3.82 Plate welded to the rear axle for carrying the batteries and to be assembled with the left side member.

Details

A stop switch is placed on top of the cover in the centreline of the vehicle as close to the middle of the vehicle as possible. The switch is connected to the error system and will stop the vehicle totally.

Design of the rear module

The rear module should connect the two rear wheel modules to the side members, carry the batteries and protect a few electronic components from water and dust.
Demands and criteria's.

- Enclosure for electronics IP54
- Lightweight
- Easy access to batteries and electronics

Structure

On Figure 3.79 the functional surfaces and forbidden areas are sketched with relation to the rear module. As mentioned on page 54, aluminium beams with the dimension 100 x 40 x 4 mm have been chosen for side members and rigid rear axle. Placing the side members on top of the rear axle gives room for tools and for the front axle to turn around the longitudinal pivot axis. This structure is also a convenient way to make a strong, stiff and lightweight rear module construction.
Now we need to design a construction to carry the batteries placed in front of the rear axle, to enclose the electronics that are going to be placed on the rear module and to mount the rear axle to the side members, see Figure 3.80.

In appendix 3.3.21.a-b there is sketched some different ways to assemble two rectangular beams with e.g. screws. It should be possible to disassemble the side members from the rear module to make the vehicle flexible. It was found that the best solution is to make some kind of mounting plate to assemble the beams.

Thin-walled rectangular beams cannot just be bolted together with big pressure, because the sides of the beams will be deflect and can easily be overloaded. On Figure 3.81 three different ways of making screw assemblies are sketched. Solution (*a*) weakens the beam, (*b*) is not easy to use in long beams and therefore it is chosen to use solution (*c*), where the beam is reinforced with an extra plate.

It is found that the best solution would be to integrate both the construction to carry the batteries and the mounting plates necessary to assemble the rear module to the side members. On Figure 3.82 a mounting plate is shown, which is welded to the rear axle and can be used to carry a battery. On Figure 3.83[25] the rear module is shown with one of the batteries in place and with the two rear steering motor amplifiers and the battery charger placed on the back of the rear axle. A cover is also sketched and it can be seen, like on the front module that the sides have to be higher than the side members to prevent water from seeping into the electronics.
The mounting plates are changed so that they will be the sides in a box enclosing the electronics on the back of the rear axle and the batteries

[25] Appendix 3.3.22-23 and drawing 1.2.2.1 – 1.2.2.7

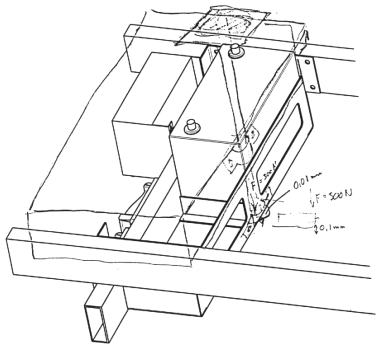


Figure 3.83 Draft of the rear module.

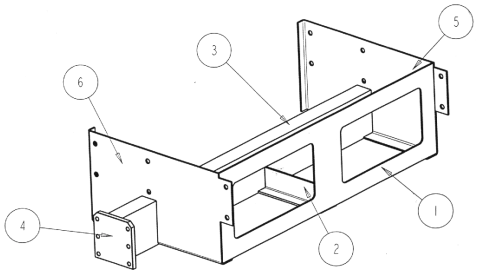


Figure 3.84 The final design of the rear module.

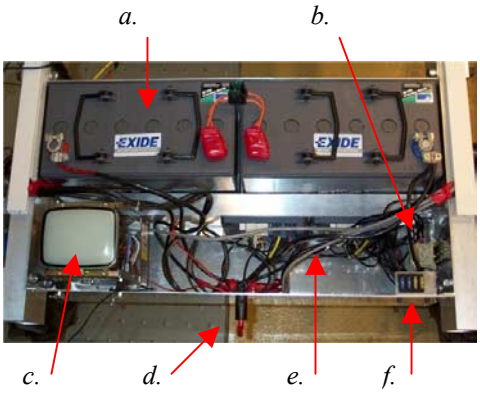


Figure 3.85 Placement of components in the rear module.

in the front. The batteries are carried by the mounting plates (Figure 3.84; 6) and a beam with a T cross-section (2) welded to the rear axle (3) and the front plate (1) welded between the mounting plates[26]. There is free access to the batteries from the bottom and the front to save weight and to give air-circulation around the batteries. When a cover is on the front module it is not possible for water to get pass the batteries to the electronics.

Force analysis and dimensions

The weight of the rear module is going through the rear axle to the wheel modules. The load due to the gravity is about 0.75 kN on the rear axle, but because of the upright position of the beam the moment of inertia is 146 cm^4, which causes the beam to deflect only 0.04 mm and the stresses due to bending is only $\sigma_M \approx 4$ N/mm^2. The interesting part is the stresses and deflections due to the worst-case forces. In appendix 3.3.25-29, the Matlab and the WinBeam results can be seen. The stresses are well below the yield stress for aluminium.

To mount the wheel modules, 10 mm thick aluminium plates are welded to the ends of the rear axle. Here all the 6 holes in the mounting plate on the wheel module are used. It is found that with the maximum tightening torque of the screws the friction between the aluminium plates and the wheel module mounting plates cannot hold the worst-case load. However, with the 10 mm aluminium plates and 8 mm screws the screws and mounting plates can hold the worst-case load, see appendix 3.3.24.

The beam with the T cross-section and the front plate are also tested in WinBeam. The moment of inertia of the T-beam is found to 7.1 cm^2 and the deflection is found to be 0.1 mm, if the beam is assumed to be a cantilever beam loaded with 300 N at the free end, see appendix 3.3.28.a. The T-beam can easily handle the weight of the batteries, but the dimension is not decreased, because this dimension is available in the workshop and easy to weld.
The front plate can also easily handle the vertical load of the batteries, because of its upright position and the resulting big moment of inertia about the longitudinal axis of the vehicle. The T-beam is supporting the front plate in the direction of the longitudinal axis of the vehicle and prevents the front plate from deflecting due to torsional loading from the T-beam.
The holes in the mounting plates are placed in each end to minimize the transversal forces on the screw connections.

Rear module details

The rear module is in the back closed with a bended plate screwed to the side plates. A power switch is placed in the middle of the back plate, see Figure 3.85; d[27].

On Figure 3.85 the placement of components can be seen. A DIN rail terminal (b) is placed in the right hand side for connecting the power from the batteries to the amplifiers for the rear steering motors and rear Honda motors. A fuse carrier for the fuseholders is fastened to the back plate (f). The steering motor amplifiers are screwed to the rear axle (e).

[26] Drawing 1.2.2 – 1.2.2.7
[27] Appendix 3.3.29 and drawing 1.2.2.8

Figure 3.86 The monitor is fitted to the back plate with 3 rubber vibration dampers.

Monitor

The monitor is placed vertically in the left hand side of the rear module. A cutout is made in the cover and a transparent plastic plate is fitted to the cutout to make the cover tight, so that the monitor can be seen with the cover in place.
The monitor is mounted in three rubber vibration dampers to protect the monitor, see Figure 3.86. The weight is 1.5 kg and the maximum load carrying capability of the dampers are 3 kg (3 x 1 kg/damper). A special bracket is made for attaching the damper in the middle at the bottom.
Since it has been decided to place the monitor on the vehicle instead of the battery charger, a proposal for safe and easy connection of the charger to the batteries can be seen on page 152 of "Recommendations".

Rear module cover

The rear module cover is in principle similar to the front module cover, see Figure 3.87.

Figure 3.87 Rear module with back plate and cover.

Side members

The height of the side members on 100 mm. have been chosen because it also could be used for the rear axle and it gives the chassis a big load carrying capability.

Load analysis and strength

The width has been chosen to 40 mm, because it gives the rear axle and the side member's adequate strength and good stiffness to the vehicle. In appendix 3.3.18 a rough estimate of the bending moment can be seen, it is assumed that the construction is stiff. With this assumption and a thickness of 4 mm, the beam will exactly have the right properties to avoid yielding. In reality, the forces will be smaller due to the elasticity of the construction and it is chosen to use the beam with the dimensions: 100 x 40 x 4 mm.

Bolting the front and rear modules to the side members

The front and rear modules are screwed to the side member's using 10 mm screws and nuts. To avoid damage to the side member's extra plates are used to reinforce the side member's. 5 of the plates for reinforcement can be seen on the right side member on Figure 3.88[28].

Figure 3.88 The chassis with the partly assembled wheel modules.

Camera mounting base

The camera consists of a print with the CMOS camera chip and a lens system. It is necessary to make a mounting base for the camera, which will provide adequate protection from water, dust and impact from minor objects. The mounting base should make it possible to position the camera in different angles. The camera should be placed on the front module just in front of the front axle.

Mounting base

The lens system is the heaviest part and therefore it is chosen to clamp the round lens system to the mounting base. A small box (50 x 50 x 30 mm) is used for enclosing the print. The camera angle can be adjusted in 5 angles: 20°, 45°, 60°, 70° and 80°, which makes it possible for the

[28] , Appendix 3.3.29 and drawing 1.2.6-7

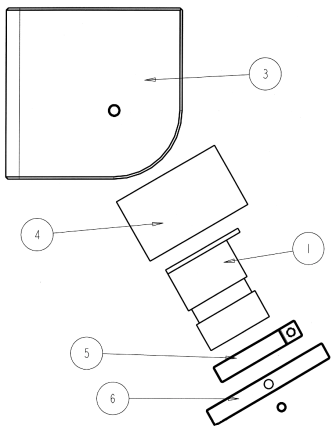


Figure 3.89 Camera mounting base (exploded view).

Figure 3.90 Vision camera in mounting base.

Figure 3.91 On-the-fly photo of RTK/GPS testing.

camera to monitor the area from vertically down to straight ahead. The numbers of degrees between the angles above are smallest between the last angles to give the best possibilities to adjust the camera angle, when looking nearly vertically down, see Figure 3.89 and Figure 3.90[29].
A recess is made in the mounting plate to give room for the nut to assemble the front axle to the Rosta bearings. The front bottom corner on the mounting plate is rounded off so that it will not be in the view of the camera, but still give some protection of the camera lens.

RTK/GPS equipment

The GPS antenna is delivered with a 500 mm long stand with a 25.4 mm diameter. The GPS antenna must be placed so that other equipment cannot shade for the signals from the satellites to the antenna. It is decided to mount the stand to the backside of the front module, because the antenna is best protected in the middle of the vehicle, see Figure 3.92.
Because the RTK/GPS equipment is borrowed, it is decided to just attach the radio antenna with straps to one of the side members. The 4700 GPS Total Station and battery are just placed on top of the front module cover while doing RTK/GPS tests, Figure 3.91.

Attachment

The stand is clamped to the front module, see Figure 3.92, but is later moved to the right side member, see Figure 3.94 and next paragraph. We will also recommend making a possibility for attaching experimental equipment on the top of the front module cover with flexible straps or rubber bands.

Compass

Precision Navigation, Inc. delivers the compass called TCM2. There are several things, which must be considered before installing the compass.

- The magnetometers should not saturate.
- The TCM2 should be located away from local sources of changing magnetic fields.
- The TCM2 should be mounted in a physically stable location.
- The TCM2 should be mounted as close to level as possible.

The compass should not be placed near large masses of ferrous metals such as transformers and vehicle chassis, large electric currents, permanent magnets such as electric motors, etc.
Considering the design of the robot the above demands limits the possible locations of the compass considerably. The compass is however capable of making a deviation table for compensating for hard-iron distortion fields, but the fields have to be static. (page 100 in "Orientation module")

Location

It is decided to place the compass on the stand for the GPS antenna, because it is assumed that it will not be disturbed by electronics in that position, see Figure 3.92.

[29] Appendix 3.3.31-32 and drawing 1.3 – 1.3.4

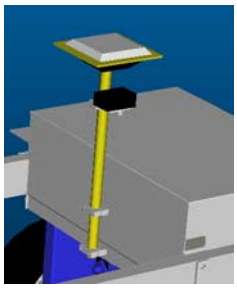


Figure 3.92 Stand placed on the back of the front module.

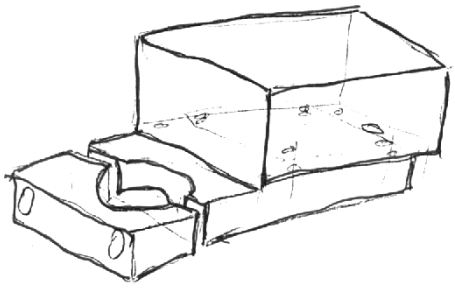


Figure 3.93 Compass sketch showing the clamps for mounting the compass to the 1" tube.

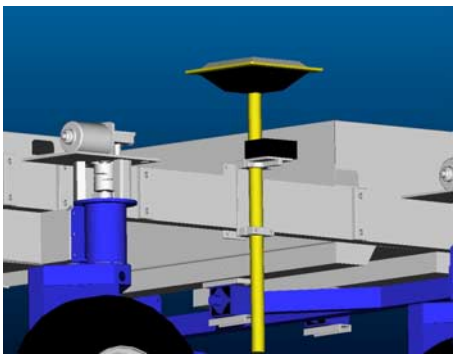


Figure 3.94 GPS antenna and compass clamped to the right side member.

The bottom plate on the front module, which the stand is mounted to, should be reinforced with a plate perpendicular to it, to make the construction stiff enough and minimize vibrations in the stand.

Relocation

During tests of the vehicle, it was found that the compass had a deviation up to about 10° even after calibration of the compass. The deviations are due to magnetic fields from the steering motor amplifiers in the front module.
The stand could also start vibrating at certain speeds on paved or hard surfaces, which leads to tilt and roll overflow and compass readings with big fluctuating errors. The compass can be tilted up to about 24°, so the tilt overflow must be because the natural frequency on 20Hz of the fluid based tilt sensor is hit.
It was found that a normal magnetic compass does not deviate on the right side member and therefore two plates are made for clamping the compass and the GPS stand to the right side member, see Figure 3.94. With the compass and GPS antenna clamped to the side member, it can easily be moved to the best position.

Mounting and Enclosure

The stand is clamped to the front module with special made aluminium clamps, which are made for 1" tubes.
A little plastic box has been bought to protect the compass against damage during the initial testing. The plastic box for the compass can be screwed on a aluminium mounting plate, which can be clamped to a 1" tube with one of the above clamps, see Figure 3.93[30].

Summary of the chassis design

With the design of the chassis, we have achieved our main objectives.

- The chassis is very flexible and give opportunity for fitting different sensors and tools. Tools can be fitted to the side members between the rear and front module. If there is too little space for the tool, the side members can easily be replaced with longer side members.
- The vehicle has been easy to assembly and it is easy to get access to the different parts, see next chapter.
- The front and rear covers protect the electronics. However, the protection rate is not IP54, as it should because the entries for cables and wires are not sealed yet.

[30] Appendix 3.3.30 and drawing 1.4 – 1.4.4

Figure 3.95 Chassis and wheel modules assembled by the workshop technicians.

3.4. Assembly

We have assembled most of the robot, on Figure 3.95 it can be seen how much was initially done by the workshop technicians, in appendix 3.4.1 pictures of the assembly and comments can be studied. Later the technicians mounted the monitor and the front and rear covers.

Some details as holes for fuse carriers, amplifiers, etc. where not designed on CAD, even though we had sketched and discussed possible placement solutions. Therefore, we had to use some time with placing components, wires, etc.

We had chosen to find the exact positions for the electronic components, when the vehicle was assembled. Even though we were using a 3D CAD tool, it can be difficult and time consuming to find the optimal positions, because wires and cables are flexible. It is a lot easier to discuss the placement standing beside the vehicle rather than sitting in front of the computer or making sketches. The drawbacks are that the optimal position cannot always be used because other components have been placed there, or it might be difficult to drill the hole were it is wanted.

Figure 3.96 Wiring the vehicle.

Mechanical

We had very few problems with assembling the parts from the workshop. The hole in the hubs for the motor shaft had only been rough machined. One of the threaded holes for fitting the steering motor encoders were not in position and the hole in the encoder plate was made bigger (page 48).
The rest of the parts were very easy to assemble.

Mounting of the computer

The computer is so far only placed on foam rubber, which give insufficient damping and the shocks can cause the computer to freeze. The computer should be mounted with vibration dampers like the monitor is. The temporary mounting of the computer do also cause the wiring in the front module enclosing to be a bit disorderly.

Wiring

Wiring of the wheel modules

It was found that the centre hole in the motor bracket was not suitable for the steering motor encoder cable. Therefore, an additional hole was drilled in each motor bracket for the encoder cable.
To wire the wheel modules were easy and the wires from the Honda motor and encoders fitted well into the 21 mm diameter holes with rubber strips on the edges.

Wiring of the chassis

The cable trunking makes it easy to arrange the cables well. The wiring inside the front and rear module are not as well organized as it could be. Especially in the front module were the computer is just temporarily mounted on foam rubber.

Vehicle

Gross weight

After assembly of the vehicle, it is weighed with a dynamometer hanging in a crane. The weight is measured to 200 kg, but the resolution of the dynamometer is 25 kg. We have earlier estimated the weight to 230 kg (Appendix 3.3.2.c), but the charger have been removed, the chassis weighs less than 30 kg and the wiring etc. do probably not weigh the earlier estimated 5 kg.
Now we will estimate the weight to about 215 kg, which is very satisfactory considering the weight of wheels, motors and batteries. The low weight does also improve the performance of the vehicle.

3.5. Summary

With the use of in-wheel drive motors, it has been possible to make an ideal design for in-row driving in typical row distances in corn, maize and beet fields. The ground clearance is 500 mm and makes it possible to drive in cornfields to the stage of earing.

The concept of making a modular design with 4 identical wheel modules and chassis has been very efficient. It has simplified the production, as fewer different parts are needed. The assembly of the vehicle has also been easier. The wiring can look very complex, but when dividing it all by four (four wheel modules) it becomes much easier to understand how the vehicle works and faster to start working with it.

With 4WD the off road capabilities are good (in the field) and with the relatively low weight the overall performance have shown to be satisfactory during tests of the vehicle.

Using DC motors to steer each wheel (4WS) makes it possible to make very sharp turns and still fulfill the Ackerman equation. This would not have been possible with a rack-and-pinion steering. The implementation of the steering control in software gives high flexibility for research.

All the connections to the wheel modules are electrical which makes it very flexible to move a module or changing the wheel base. The side members can be replaced with longer or differently designed side members if it should be wanted to change the wheel base. This is fairly easy to do because the wires have a little extra length and it is only necessary to enter the new positions of the of the wheels in the software to make the steering follow Ackerman.

4. Computer and interface

Figure 4.1 For maximum safety the main fuses are placed between the batteries.

4.1. Electrical power wiring

The wiring of the power lines is shown on Figure 4.2.
Batteries of this size can supply a frightening current that can melt any conducting material. Therefore we have placed a 60 A fuse (2x30 A) between the two 12 V batteries, see Figure 4.1. In addition, the two middle terminals of the batteries that are connected to the fuse are wrapped up with adhesive insulation tape. This solution makes it impossible to make a short-circuit without having a fuse that can blow.
The computer supply and all motors also have their own fuse.
The 10mm^2 main power cable is connected to DIN rail terminals that are interconnected and mounted on a DIN rail. This is the 24 V terminal that supplies all other electrical equipment through 2.5mm^2 cables. We have placed a DIN rail both in the front and rear compartment to make it easy to attach extra equipment.
The Honda motors are denoted as H0, H1, H2 and H3 and the steering motors supplied from the Curtis motor power amplifier are denoted as S0, S1, S2 and S3.

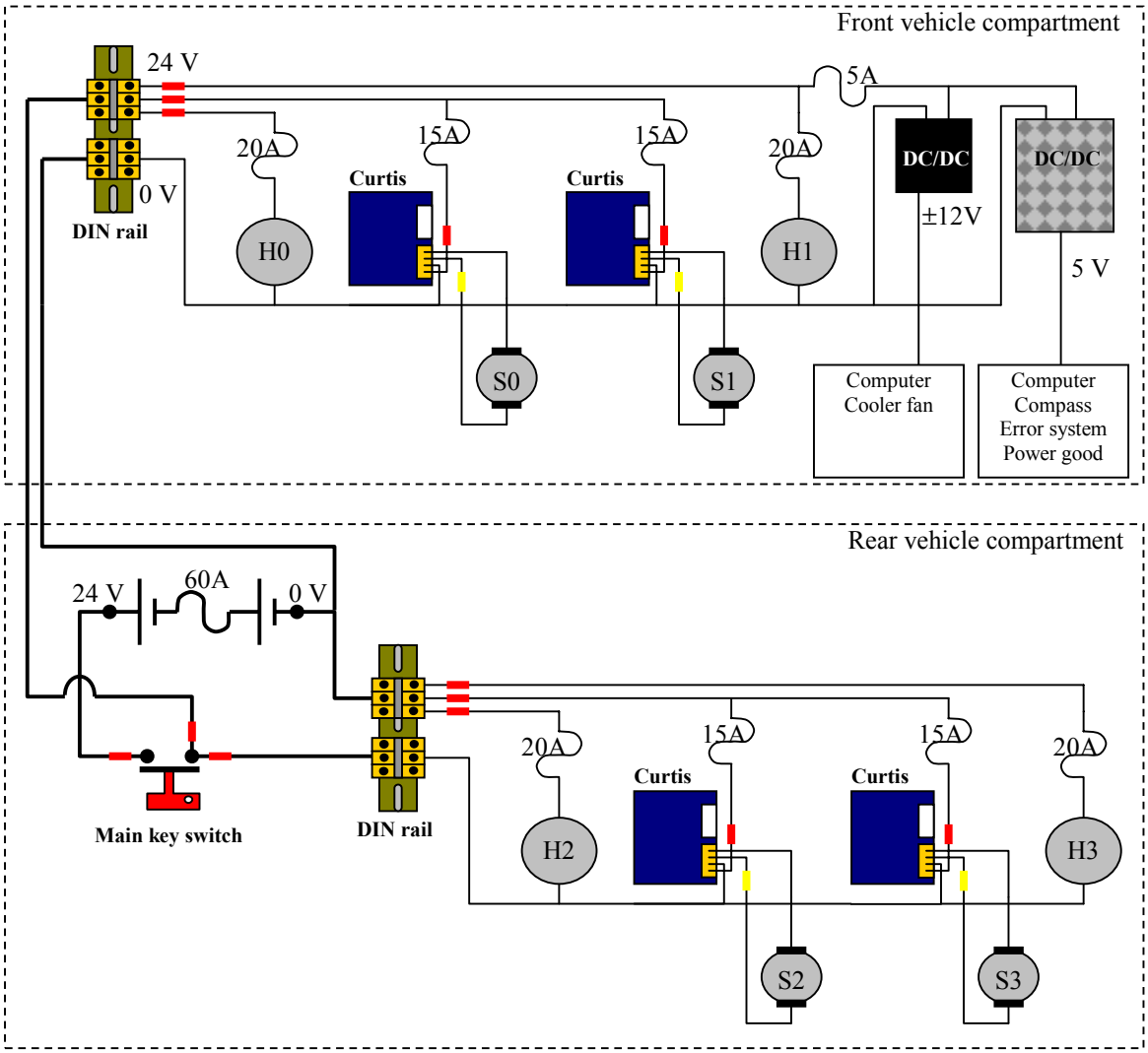
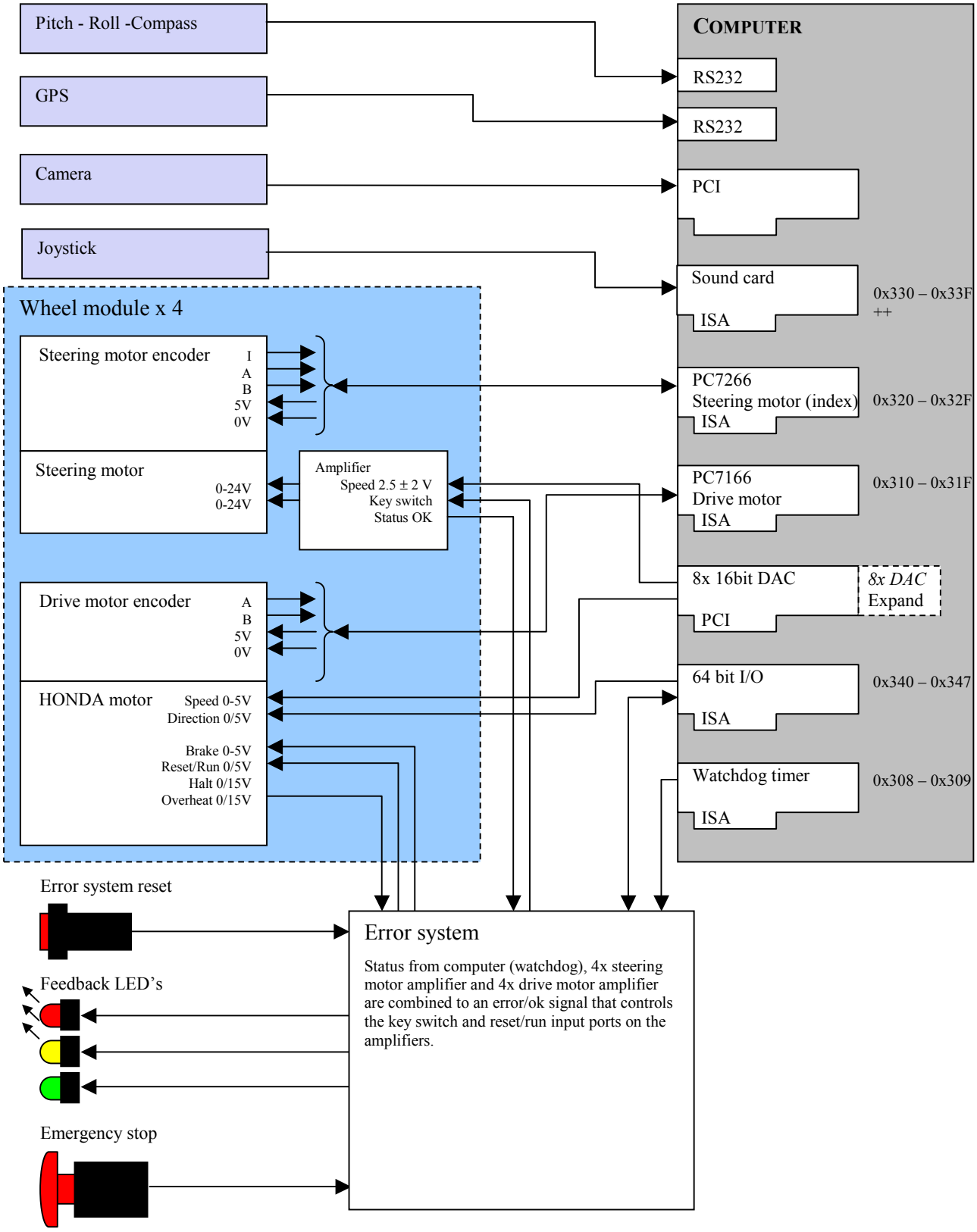


Figure 4.2 Wiring of power electrics.

4.2. Control connections

An overview of the control connections is shown below:



Honda motors

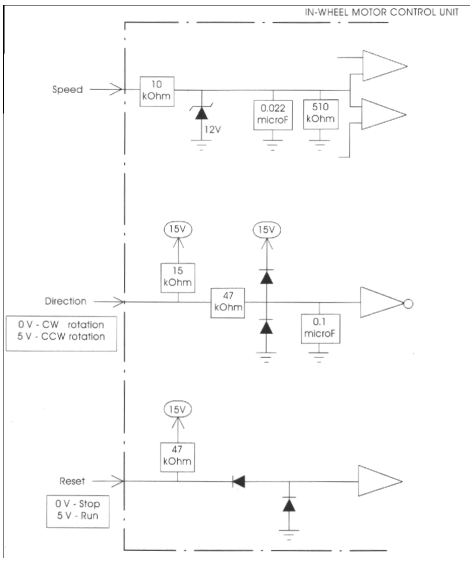


Figure 4.3 Electrical input interface for Honda motor

Speed input

The analogue voltage output from the DAC is connected with the Honda motor speed control input. The Honda electrical input specifications are shown on Figure 4.3.
The Honda motor speed input resistance is 10 kΩ. As the DAC range is –10V to 10 V and the Honda motor internal control input voltage is secured by diodes to be 0-12 V then maximum possible current drawn is 22 V / 10 kΩ = 2.2 mA.
The DAC has a voltage output driving capability of ± 5 mA max.

Direction input

The direction input port of the Honda motor is internally pulled up to 15V as shown on Figure 4.3. In the specification for the 64 bit Digital IO card we could not find any data indicating a maximum sink current at high level (5V) i.e. how much current it will draw to try to keep the level at 5V even though the Honda motor tries to pull it to 15V.
To avoid the risk of damaging the IO card we placed a transistor in between the IO card and the direction input. The transistor network is shown on Figure 4.4. The IO output is connected to V_s.
When the transistor is saturated the collector current is 15V/15kΩ = 1mA. To be sure that the transistor will be saturated in any situation we set its current amplification h_{fe} = 10 (the specification for a BC547 transistor is h_{fe}= 100 – 400). We can calculate the basis resistor using the standard formulas for a saturated transistor:

$$R_b = \frac{V_s - V_{be}}{i_b} = \frac{V_s - V_{be}}{\frac{i_c}{h_{fe}}} = \frac{5V - 0.7V}{\frac{0.001A}{10}} = 43k\Omega$$

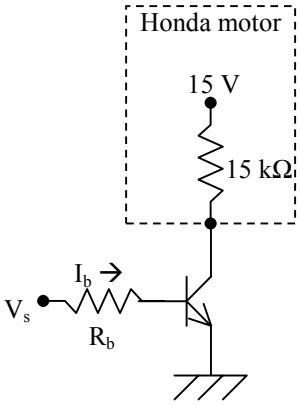


Figure 4.4Interface between computer digital output and direction port on Honda motor

Reset input

When the reset input is pulled down to 0V the motor will stop.
It is not clear form the specifications why a diode has been placed in the input without a pull-up on the right side of the diode. There must be some kind of pull-up to make it work and this would affect how much current we need to sink to stop the motor. Since no supply has been shown on the right side, we expect that contribution to be insignificant. The sink current through the pull up on the left side of the diode is I=15V/47 kΩ = 0.32 mA.

Brake input

The brake is regenerative and charges the batteries while activated. Braking 50% at the rated speed will produce a 6A charging current. There is no specification on the brake input in the Honda manual except that it is in the range 0 to 5 V. They do however have a suggested wiring chart indicating that the brake specifications are identical to the speed input specifications.

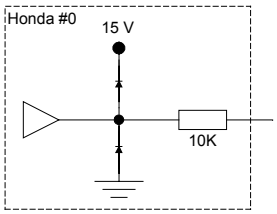


Figure 4.5 Overheat signal from the Honda..

Overheat output

The overheat goes low (0V) when the temperature of the core is close to become critical (80°C). When OK the voltage is 12.9 V and the output resistance is 10kΩ (see Figure 4.5)

Halt output

The halt output goes high whenever the motor is stopped either if the overheat of the core is critical (90°C) or if the speed signal is set to 0. The latter is a problem and prevent us from use it directly in the external error system. It has however been wired to the error system so it is easy to implement if necessary.
The specifications are similar to the overheat output but internal it is pulled up to 15V through another 10kΩ resistor.

Current output

The current output is an analog signal that indicates how much current the motor draws. 1V = 10A and with an A/D converter it would be possible for the computer to log the current together with all other data and to avoid overheat.

Wire colours

The control wires from the Honda motor are gathered in two cables. The cables were not shielded and therefore we have cut them off close to the motor to extend them with shielded cables. There are 5 wires in each cable from the motor and we have connected them all except for the control power supply that is only used if you want to generate the speed signal with a variable resistor.
The cables are connected to the outside end of the terminal strip in the computer housing. They are labelled “Motor x cable y” (x = 0, 1, 2, 3 and y = 1 or 2).

Description Cable 1	Wire color From motor	Wire color Extension cable
Input - Speed	Orange	Pink
Input - Brake	Yellow	Yellow
Input - Direction	Brown	Brown
Input - Run/Reset	Blue	Grey
Output - Halt	Green	Green
Cable 2		
Output - Ground	Black	Brown
Output - Overheat	Red	Yellow
Output - Current	Grey	Grey
Output – Pulse	Purple	Pink

Curtis power amplifier

We do not have any detailed data for the electrical interface to the Curtis power amplifiers, but we have a suggested wiring diagram for use on electrical wheelchairs. From this diagram, shown on Figure 4.6, we can estimate the specifications.
In addition to the power input and output connections there is an 18 pin connector with a lot of functionality. We only need to use the

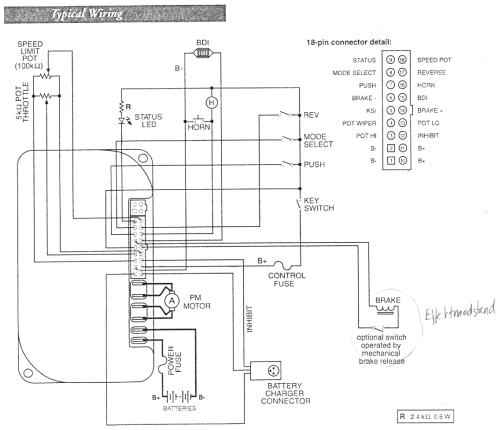


Figure 4.6 Suggested wiring when using the Curtis amplifier with an electric wheelchair.

speed input, the key input and the status output. Unfortunately the Curtis unit includes a lot of security features made for wheel chairs:
a) There must be a brake connected. If it has "fallen off" then the amplifier will not work. Therefore we had to place a 470Ω power resistor from pin 6 to pin 14.
b) The speed input is intended to be made as a variable voltage division between "Pot hi" (pin 3) and "Pot lo" (pin 13). If these connections have "fallen off" then the amplifier will not work. Therefore we had to place a 5kΩ resistor between these two pins.
c) The maximum allowed speed is intended to be adjusted with a potentiometer. We want the computer controller to decide what should be the maximum speed. Therefore we had to connect the "Speed pot" (pin 18) with "Pot hi" (pin 3) to set the speed limit to maximum.

For the connections that we do use, we have got (or calculated) the following specifications:

Speed input

The speed input recognize voltages in the range 0 to 5 V. From Figure 4.6 we can see that in a wheelchair it would be generated from a voltage division in a 5kΩ potentiometer. The current through the resistor is 5V/5kΩ = 1mA. The input port must use much less than this to avoid affecting the voltage division. The DA converter has a voltage output driving capability of ± 5 mA max.

Key input

From Figure 4.6 we can see that 24 V = Run and 0 V = Stop. The input appears to be internal pulled down to 0V. By measurements we found that the voltage must be at least 19V to activate it.
We tried to measure the key input resistance. It appeared to be about 650Ω but it was not constant. This did cause some problems later (see "Reed relay welds" on page 74)

Status output

From Figure 4.6 we can see that the status output is capable of driving a LED with a current about 10 mA. It can therefore drive the CMos logic with no problems. Later we also added a LED and it has not caused any problems.

Wire colors

Out of 18 pins in the amplifier plug we have connected 5 that we either use or think could be useful in the future. The direction pin is not used, but the amplifier can be programmed to use it. The battery indicator could also be useful as we currently have no indication of the battery condition. The cables are connected to the outside end of the terminal strip in the computer housing.

Description	Wire color
Status	Green
Direction	Yellow
Key switch	Grey
Speed	White
Battery indicator	Brown

D/A converter

The DA PCI card has 8 channels of analogue output, 4 bits digital output and 4 bit digital input. We have divided this into 4 cables – one for each wheel module. From the plug in the DA PCI card goes 4 cables that are numbered 0, 1, 2 and 3. There are one bit of digital input and output in each cable. The bit number is the same as the cable number. The speed signal for the Honda drive motors use channel 0 to 3 in the DA i.e. the same as the cable number. The speed signal for the steering motor use channel 4 to 7 in the DA i.e. the cable number+4. The four cables are connected to the inside end of the terminal strips.

Wire colours

In each of the four cables, the following wires are connected:

Description	Wire color
DO - direction out (not used)	Yellow
DI - digital in (not used)	Green
Va - Speed drive	Pink
Ga - Ground drive	Grey
Vb - Speed steer	White
Gb - Ground steer	Brown

Drive Encoder

The cables that connect the Honda drive motor encoder with the PC7166 encoder interface has been extended with shielded cables. The length varies with the position of the wheel (1½m to 3½ m). The plug in both ends have one set of cable colours which did not exactly match the colours in the cable we used to extend it.

Wire colours

Description	Wire color Plug (both ends)	Wire color Extension cable
1 - Ground	Brown	Brown
2 - not connected	Purple	-
3 - Quad input A	Blue	Grey
4 - 5V supply to encoder	Orange	Pink
5 - Quad input B	Yellow	Yellow

4.3. Hardware handling of errors

To help prevent accidents and damage to the vehicle, the vehicle is equipped with an error handling system. It is important to make the robot start up, run and shut down in a controlled way.
Computers can lock or become unstable and the outputs during start up and shut down is not well defined. Because of these problems it is important to make an error system, which do not rely on the computer. Windows 98, which we are using as operating system, is not made for real time control. This fact makes it especially important to design a good error system.

Demands to the error handling system

- Action on errors, which could cause safety problems or damage to the vehicle.
- Must detect if the computer is locked or unstable
- React on errors from the motors and amplifiers
- React on output from obstacle detection
- Have a stop button
- The system should take a safe position if wires breaks or the system is damaged.
- The computer must be able to react on the errors
- Human machine interface, which make it possible for the operator to see the state of the system and reset it.
- Additional inputs available for future developments

All the above errors should make the vehicle stop and set the computer in a safe mode. Output from the obstacle detection system is on this level just an on/off signal. Obstacle detection could be developed to make the computer react intelligent to obstacles near the vehicle, but still it would be necessary with an input to the error handling system outside the computer.

Inputs to the system

- Honda motors: Overheat
- Steering motor amplifiers: Status
- Red emergency stop switch, placed on the outside of the vehicle. (Brake switch type).
- Obstacle The system will be prepared for input for obstacle detection (on/off).
- Watchdog. Timer, which the computer has to reset at a preset time interval. If the computer fails to do so a relay will be closed.
- Error clearing. Reset button, which will enable the system after errors have been corrected.

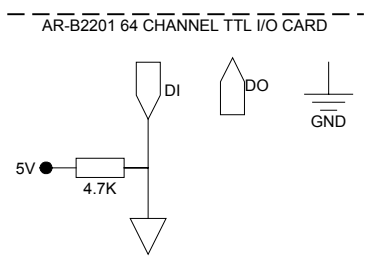


Figure 4.7 Interface to the I/O card.

Outputs from the system

- Honda motors: Reset
- Steering motor amplifiers: Keyswitch
- Computer: All input and output signals should be readable for the computer. A 64 channel TTL I/O card with 5V pull up have been chosen to interface to the computer, see Figure 4.7.

- Human interface: LED to indicate the state of the system.

Design

As it can be seen above a lot of the inputs and outputs are specified and the task is to react on the errors and make an output to the reset functions on the motors and amplifiers. It is also necessary to specify ways to connect to the error system, because tools and other features will be developed for the robot in the future.

There are 2 possibilities for reaction on errors:

1. Make an error system that includes the logic for all features and completely independent of the computer.
2. Make the computer handle all errors. The only situation where the computer cannot handle an error is when it is locked or is instable. The external error system therefore only needs to check if the computer is running as it should.

Solution 1 has been chosen, because it will be the safest and it is necessary to some logic for the watchdog anyway.
The error system can be seen in appendix 4.7.

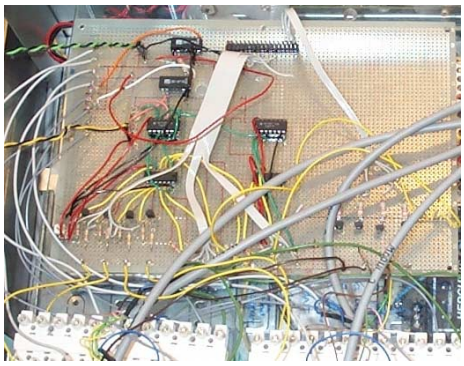


Figure 4.8 Error system.

Logic

Because of the many inputs the system has to handle, we have chosen to use integrated circuits to implement the logic. CMOS logic from the HC74 family have been chosen, because they are capable of sourcing or sinking up to 25mA and have low power consumption at low switching frequencies. Because of the high sourcing capability the devices can be used to drive relays and LED's.

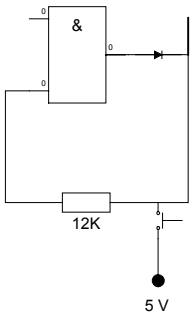


Figure 4.9 AND gate used as a latch. The pushbutton will enable the system.

All the signals from the motors and amplifiers are 0V when okay and therefore NOR gates are chosen. AND gates are used to AND the 5V signals from the NOR gates to a 5V OK signal from the logic system. The last AND gate is used as a latch, where the output is feedback as one of the two inputs of the gate, see Figure 4.9. A pushbutton is connected to the gate to set the input high and enable the system, when the errors have been corrected. To protect the output on the gate when the enable button is pushed a diode is used. The voltage after the diode is still above the 3,7V necessary to drive the reed relay on Figure 4.10; b.

Input interface

The Honda overheat signal is interfaced by inverting NPN transistors, which mean that 0V is OK. The steering motor amplifiers are pulled down to 0V, when the amplifier is OK. If a connection breaks the signal will be pulled up to 5V.
RS-Components could only deliver quadruple 2-input positive-and gates and triple 3-input NOR gates, therefore it was necessary with 2 of each IC to have additional inputs for future features.

Output interface

Reed relays were used to control the 4 Honda reset inputs and the 4 steering motor amplifier keyswitches, these relays have low power consumption and give us high flexibility compared to a solution using transistors. The CMOS logic can source up to 25mA and the reed

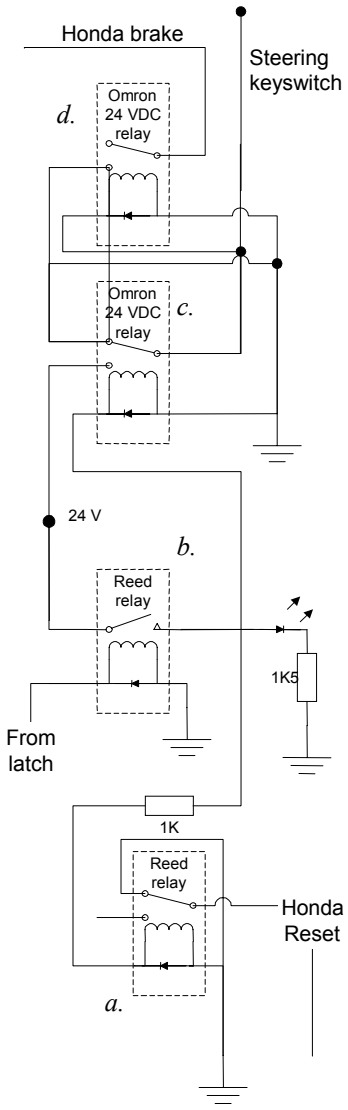


Figure 4.10 Output system

Figure 4.11 Green, yellow and red LED and pushbutton to enable the system.

relay has a resistance on 500Ω, which gives a current consumption on only 10mA at 5V.
Relay (b) on Figure 4.10 can handle a cut-in current on 0.5A and relay (a) can handle a cut-in current on 0.3A. The current consumption from the 4 steering motors has been calculated to 0.15A and from the 4 Honda motors to 1.3mA. Therefore the relays should be able to be coupled to more devices. It is likely that future features can use one of the outputs from one of the 2 relays to reset/operation functions.

Changes to the output system

During testing of the error system the changes below were made.

Reed relay welds

Before dimensioning the relays for supplying 24V to the steering motor keyswitch, we had measured the resistance in the steering motor amplifiers to 650Ω and calculated the current consumption to 0.15A. The max. cut-in current for the 1-pole relay is 0.5A and therefore we believed it to be working. But it showed that the cut-in current welded the switch in the relay and prevented the relay from switching back. The current could not be limited enough by a resistor, without getting to big a voltage drop, which would prevent the amplifiers from being enabled. Therefore we choose to let the 1-pole relay control a bigger mechanical relay Figure 4.10; c.
Instead of the 1-pole reed relay a bigger solid-state relay should have been used. To let the reed relay control the mechanical relay is working, but it is not so nice a solution.

Connection of Honda brake signal

Even though the Honda motor brake signal was wired to the error system we did not use it in the beginning. We did however soon discover that if an error occurred on a slope, the vehicle would begin to roll on free wheel. By activating the electrical regenerative brakes, the speed is significantly reduced when the controller is not working. A mechanical relay was available and easy to connect to the output to the steering motor amplifiers. The relay connects the brake signal to 0V, when everything is working, and to 5V (max. brake signal) if an error occurs, see Figure 4.10; d.

Computer interface

The computer reads all the input error signals and the state of the error system. The error system is connected to four 8 bit I/O ports on the I/O interface card with a single multi-wire ribbon, see appendix 4.8. Port 2 and 3 are used for monitoring the error system. 4 bits on port 0 are used for setting the direction of the Honda motors and 2 bits on port 1 are used for driving status LED's.

Man Machine Interface

The red LED (Figure 4.11) is turned on and off by the motor controller software module. When all errors has been cleared and the controller starts to run it will turn off and then the reset push button can be released.
The yellow LED is not yet in use.
The green LED indicates that the error system is in OK state and that the steering and drive motors are enabled.

Driving LED's

The LED's are driven from the 5 V and 24 V supply through a current limiting resistor. To get a good reliability the current limiting resistors are designed for 15mA.
The green LED has a voltage drop on 2.1V and is supplied with 24 V. Thus the resistor must be at:

$$R = \frac{24V - 2.1V}{0.015A} = 1460\Omega \approx 1.5k\Omega$$

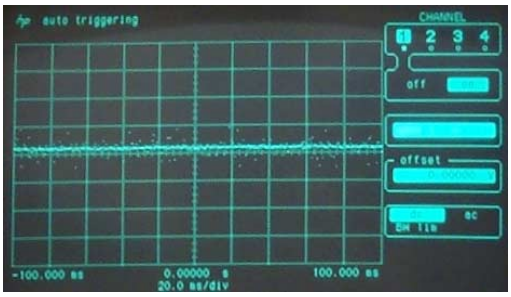


Figure 4.12 Noise on the error signal from the steering motor amplifiers. The vertical scale is 2V/div

The red LED has a voltage drop on 1.7 V requiring a resistor of:

$$R = \frac{5V - 1.7V}{0.015A} = 220\Omega$$

The yellow LED has a voltage drop on 2.0 V requiring a resistor of 200Ω.

Noise problems

The error system was not stable and random errors occurred in the system. A digital scope was used to find the error and we systematically monitored the signals in the error system, starting with the latch. We found that the error signals from the steering motor amplifiers were very noisy and had transient voltage peaks up to about 2V, which were likely to cause the random errors, see Figure 4.12. The peaks could be even higher, because the digital scope samples the signal and between two samples we get no indication of the signal level.
As we already use shielded cables we could not think of any easy way to eliminate the noise at the source. Instead we chose to use capacitors to reduce the peaks. The impedance of a capacitor varies with frequency. A DC voltage cannot pass through while for high frequencies it will work as a short-circuit. Since the signals are on/off DC-signal this approach does not cause any problems except for a minor delay (phase shift) of the signal. It was initially found that 10 µF capacitors could solve the problem. Later during tests in the field random problems occurred again and we changed the 10 µF capacitors, with 47 µF capacitors and since we have not had noise problems, see Figure 4.13.

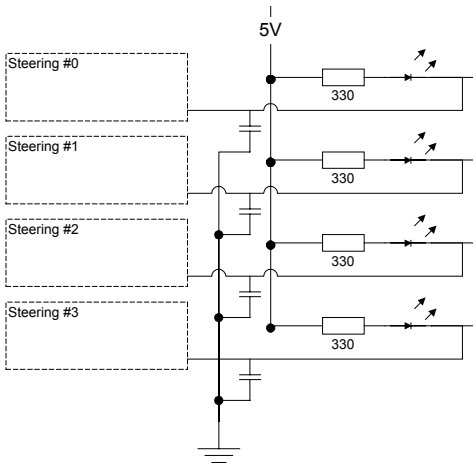


Figure 4.13 Interface of steering motor error signal.

Steering motor amplifier errors

The steering motor amplifiers are very sophisticated and have a lot of special functions, which make them ideal for wheelchairs. A lot of fault and warning codes can be signaled by the amplifiers, see appendix 4.10. Normally the status signal from the amplifier is connected to a LED, which is showing the error or warning with a blinking code.
This feature has caused some problems for us, because the error system detects an error as one instantaneous logic signal and therefore it sees the blinking code as 'error', 'not error', 'error', 'not error', etc. It simply stops the vehicle every time an amplifier tries to signal something, even though it is a non-critical warning.
To enable us to get the warning we have connected a red LED to each amplifier error signal, see Figure 4.13. When an error occurs the error system stop the amplifier immediately. Unfortunately the amplifier also stops sending the error code when its key switch is turned off. To

read the error we need to keep pushing the enable button, which activates the amplifier and makes it send the codes.

Recommendations

The error system is working satisfactory, but the system could benefit from a redesign.
The soldered connections on the prototype board can easily be damaged and course random errors, which can be difficult to locate.
The connectors to connect the wheel modules to the error system can probably cause noise problems and it is also very time consuming to connect all the wires.
If big problems with the error system occurs we recommend designing a new error system with background in the experiences gained from the current system.
It is recommended to make the error system in a dedicated box, which will provide noise protection of the system. The connections to the box should be made with appropriate connectors as e.g. sub-D connectors for easy connection of the cables. Instead of the prototype board a dedicated print board should be made. The 1-pole reed relay and the mechanical relay should be replaced with a solid-state relay and the mechanical relay for the Honda motor brake signal should be replaced with a transistor or reed relay.
It might also be interesting to remove the steering motor amplifiers from the computer box and see how this will affect the noise level. The amplifiers are pulse width modulated and can therefore give a lot of noise. Other noise sources are the 5V and 12V dc-to-dc converters and the hard disk drive motor.

4.4. Computer interface

The following equipment is connected to the computer:

Type	Description	Interface
Network	As the computer has no CD or FD drive, there is no other way to install programs to the computer or to get log-files from it.	PCI
Vision	The vision card is used to connect the camera to the computer	PCI
D/A converter.	This 8-channel DAC is used to send the speed control voltages to the 8 motor power amplifiers. Another 8 channels can be added with a plug-in.	PCI
PC7266 encoder interface	This is a quadrature decoder card for the encoders connected to the 4 steering motors.	ISA
PC7166 encoder interface	This is a quadrature decoder card for the encoders connected to the 4 Honda drive motors.	ISA
Graphics	The graphics card drives the internal 640x480 VGA monochrome monitor and the external 800x600 SVGA monitor when the vehicle is docked.	ISA

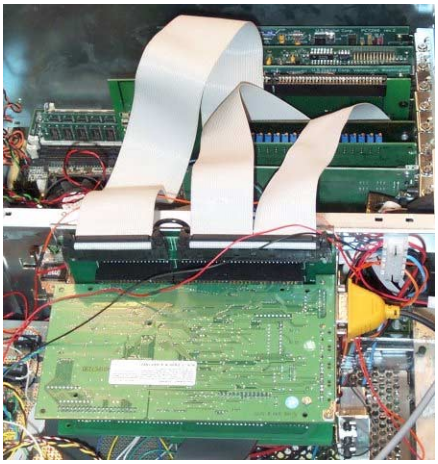


Figure 4.14 The ISA extension gives 3 extra slots.

ISA extension	The ISA extension card is a normal ISA card plugged into the ISA bus. With cables it connects another 3-slot ISA bus to the computer ISA bus.	ISA
The following 3 ISA cards are plugged into the external ISA bus.		
Watchdog timer	The watchdog timer is an indispensable safety precaution. It activates the external error system if the computer locks up.	ISA
Digital IO card	The 64 bit digital IO card is used to control the direction of the Honda motors and to read the status of all error input to the external error system.	ISA
Sound card	We do not use sound, only the Joystick connector.	ISA
Serial ports		
Compass module	The compass module sends 3-axis orientation data at 20 Hz, using its magnetometer and fluid-based tilt sensors.	COM 1
GPS	The GPS sends position data at 5 Hz.	COM 2

All slots in the computer are filled out. If more connections need to be made there are two solutions:
1) Extend the number of ISA slots. Either by adding another extension board or use a computer with passive bus.
The problem is however that the number of available IO addresses on the ISA bus is limited.
2) Use a modern ATX motherboard with USB connection. The USB interface allows up to 127 items to be connected at the same time. Many of the cards we use are specialised an might not be available for USB in the near future, but the Sound card, Graphics card and Network card could be replaced by USB connected versions.

4.5. Computer power supply

The power supply converts the power source voltage (here 24 V) to the power input voltages required by the computer.

Power consumption

A computer based on the AT standard requires the following connections:
+ 5 V: Is used to supply the processor, motherboard and other logic boards in the attached equipment (i.e. hard drive, interface cards, etc)
- 5 V: Is not used by the motherboard but is only routed to the ISA bus. It was used in older floppy disc controllers but is not used any longer by any device. The –5V supply has been omitted on this system.
+ 12 V: Is used by all drive motor and fans and is routed to the ISA bus. It is also used by the serial port driver when signalling a logical '0' on the port.
- 12 V: Is only used by the serial port driver when signalling a logical '1' on the port. On newer computers it is generated internally by the serial port driver and is therefore not required from the power supply. We were not able to test the vehicle computer at the time of ordering the power modules. Therefore we have made – 12 V available to make the system as general as possible.
PG : The Power Good input is a logical input that must go high when all voltages in the supply are stable. Detailed description later.

In addition a computer based on the ATX form factor requires 3.3 V. If the motherboard and processor should be replaced by a more powerful one in the future it will be necessary to include a 3.3V supply as all computers newer than the one we have included are based on the ATX standard.

The power consumption by the computer and attachments are:

Description	+5 V	+12V	-12V	
DAC	580 mA	70 mA		Typical
PC7266 encoder interface	310 mA	80 mA		Max
PC7166 encoder interface	310 mA	80 mA		Max
8 x Optical encoders	0 mA	80 mA (8 x 10)		Estimated, no data available
Vision interface PCI	?	?		
Compass module	16 mA	0 mA		Typical
Serial ports on computer		8 mA (2 x 4)	8 mA (2 x 4)	*
Computer + processor fan + graphics card + net card	5100 mA	70 mA		**
Hard drive	400 mA	270 mA***		

Total	6716 mA	660 mA	10 mA	

* The RS232 specifications requires a signalling output in the range ± 5-15V with a 3-7kΩ load. With our 12V supply, the maximum current drawn by one port is 12V/3kΩ (min) = 4 mA
** Initially we estimated a consumption of 5A @ 5V and 100 mA @ 12V based on typical specification of other motherboards. The actual consumption found later by measuring the system was 5100 mA @ 5V and 70 mA @ 12 V.
*** 900 mA while spinning up (4-5 sec.)

The GPS unit requires a supply of (typical) 500 mA @ 12V. It has got its own battery supply but we intended to supply it from the computer supply to keep the idea of having only one set of batteries to supply everything. Unfortunately, we did not include the extra load from the hard drive while spinning up. If the GPS is supplied from the computer supply then it must not be turned on when the computer is turned on. The human factor will ensure that this happens someday and therefore we decided to use a separate supply for the GPS.

After installing the computer with all the ISA and PCI cards we found that the heat generation required a fan to be installed. The fan requires 240 mA @ 12V. It has been connected to the –12V supply to preserve the +12V supply.

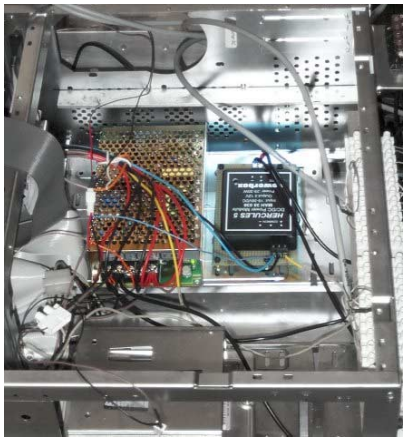


Figure 4.15 The computer supply DC/DC converters mounted in the computer box.

Power modules

With guidance from Powerbox we found the following modules that runs on a 24V supply:

Name	Output Voltage	Output Current (maxload)	Input current (maxload)	Specified Efficiency
Saturn SD-50B-5	5V*	10A	3A	73%
Hercules 5 – MAH 35036	±12V	2 x 1250mA	1500mA	77 - 85%

*can be adusted in the range 4.5 - 5.5V

The two modules are mounted in the computer cabinet below the error system as shown on Figure 4.15. The Saturn 5 V supply is to the left and the Hercules +/- 12 V supply is to the right.

The Power Good signal

The Power Good input (PG) reset the computer. When PG is low (<0.8V) the computer continuously reset to avoid starting up under unsafe conditions. When the PG goes high the computer will begin to initialise and load the cmos.
The computer works even though the power good signal is not connected. It is however not a safe solution as the PG is not intentionally pulled up internal. We measured this by connecting PG with 0V through a µ-ammeter and there was no deflection at all.

We have delayed the PG input from the power supply with a simple RC circuit. A capacitor is charged by the 5 V power supply through a resistor.

The voltage over the capacitor is given by the equation:

$$V(t) = V_{\text{sup }ply} + (V_{capacitor,init} - V_{\text{sup }ply})e^{-t/RC}$$

With a voltage limit of 0.8V we can solve for t/RC:

$$0.8V = 5V + (0 - 5V)e^{-t/RC}$$

$$\Updownarrow$$

$$0.16V = 1 - e^{-t/RC}$$

$$\Updownarrow$$

$$t/RC = -\ln 0.84 = 0.174$$

With a 150 µF capacitor and a 120 kΩ resistor this equivalents to a delay of 3 seconds.
When turning on the power with the main switch the computer will wait 3 seconds before it begins to initialise.

Wire colors

The power wires from the motherboard and hard drive have the following colours:

Colour	Voltage
Red	+5 V
Black	0 V
White	-5 V
Yellow	+12 V
Blue	-12 V
Orange	Power Good

Figure 4.16 The 5" monochrome VGA monitor protected by plexiglas

4.6. Computer monitor

Initially we did not intend to place a monitor on the vehicle. Windows should start automatically when the power was turned on and it should automatically run our controller software.
However, as the software became more and more complex we realised that we had to be able to follow the status of the controller and make changes to the parameters while being in the field.
We placed a small monitor on the vehicle. It is a 5" 640x480 VGA size monitor with a 4bit greyscale colour depth (actually it is green not grey…). The image is very small and therefore it is virtually impossible to see anything while the vehicle is driving. But when standing still we can use it to change all the parameters offered by the modules and to see all control signals and status messages.

Unfortunately we discovered that our programming language Delphi4 would not run on a 16-colour monitor. Therefore we could not make any changes or debug the code while in the field. We always had to drive the vehicle to the garage and connect it with a larger external monitor.

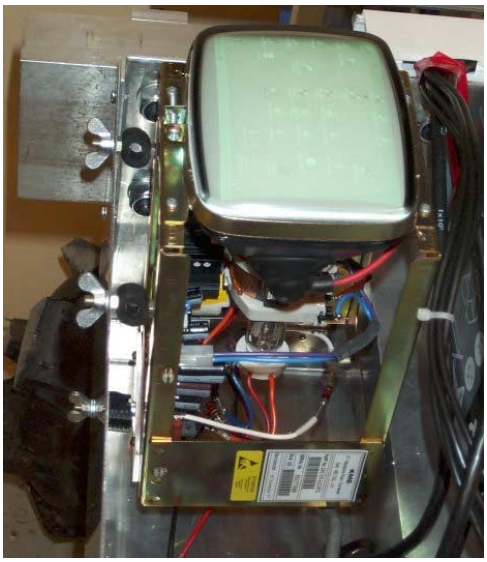


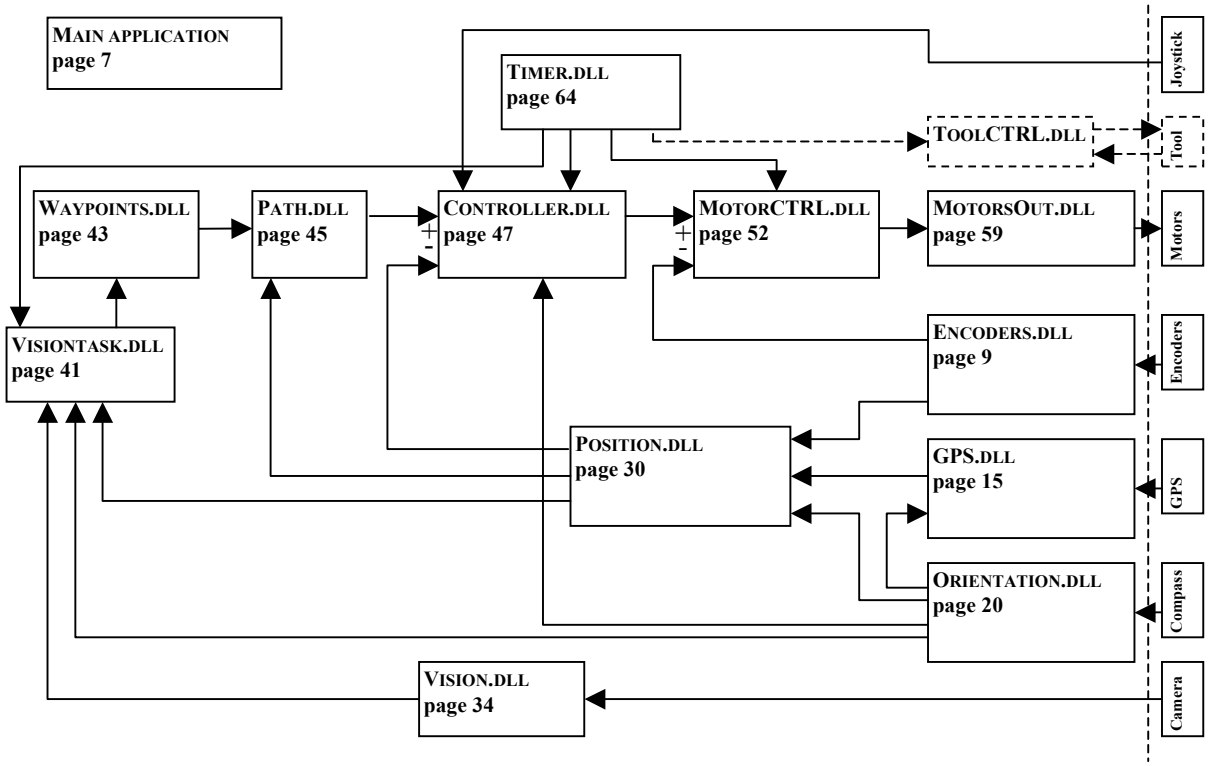
Figure 4.17 The 5" VGA 'open chassis' monitor mounted with rubber-dampers in the back of the robot

The monitor needs a 12V DC supply and use about 1.1 A. As the decision to include the monitor was made after making the computer power supply it is not included in the "Power supply" calculations and thus we got a 24-12 V DC/DC converter just for the monitor to supply it from our 24 V main source.

5. Software modules

To get maximum flexibility we have made a module based software system. The argumentation for the selection of modules can be found in "Identifying modules" on page 23. On Figure 5.1 below an overview of the modules can be found, but the detailed edition with dataflow descriptions is in appendix 5.0a.
The Delphi source code for all the modules is in appendix 5.5.
In the following 57 pages the theory and communication in each of the modules will be described.

Figure 5.1 An overview of all software modules and their connections.



5.1. Software modules overview

The software system consists of 12 DLL's and a main program. The DLL's have a standardized interface and they can therefore be replaced by other DLL's with the same interface, without affecting the functionality of the rest of the system. A DLL can be made by any Windows programming language.

The modules are:
Timer.dll: handles the timing of the whole system.
Motorsout.dll: interface to the DA converter. It converts the control signals to control voltages.
Encoders.dll: interface to the motor encoders. Calculates speeds, distances and angles.
GPS.dll: interface to the GPS receiver. Calculates vehicle center position from GPS antenna position and rotation matrix.

Orientation.dll: interface to the compass and tilt sensors. Calculates the rotation matrix from the vehicle frame to the world coordinate system (Easting, Northing, Height).
Position.dll: Calculates current position based on Encoder, GPS and orientation data.
Vision.dll: Interface to the camera. Analyse images for objects and and calculates their position on the ground.
Visiontask.dll: Converts vision objects to waypoints and if no objects are found, it will generate some waypoints to position the vehicle at the next crop row.
Waypoints.dll: Handles a database of waypoints loaded from file or added by the vision task planner.
Path.dll: Calculates a path the vehicle should follow between the waypoints.
Controller.dll: Controls the position and heading of the vehicle based on data read from the path, position and orientation modules.
MotorCTRL.dll: Controls the speed of the drive motors and the angular position of the steering motors.

Programmers interface

We have specified an interface to each of the DLL's. The interface is a set of functions that can be called in the DLL. In appendix 5.2 the interfaces for all DLL's are listed with a description for each of the functions. If a new DLL is made to replace an existing DLL, then it must contain the functions with the same names and definitions as specified in the interface. There is however no problem in adding more functions than specified. A DLL can also easily be used in other applications, as it is described in appendix 5.2 how it is used.
Some common functions have been added to all DLL's. For example they all have a "Showwindow" function that shows a settings window with parameters for this DLL.

Timestamp definition

We assign a timestamp to all measured data and calculated data based on measured data. This is very important because there are delays through the system (both hardware and software). The delays are different in the GPS, encoders and heading measurements. Before we can combine the measured data, we must therefore be sure that they describe the state of the real world at the same time.
As described in the Timer module section we use hardware timers on the motherboard to get maximum precision. The hardware timer counts how many ms have passed since the computer was turned on. The actual precision is 1 ms and therefore we use this counter value as the timestamp.
The access to the hardware timers goes through the Windows multimedia extensions in MMsystem.dll. The function *TimeGetTime* from this DLL returns the counter value as an integer value.

Data logging

It is important to be able to analyse and plot what is happening everywhere in the system when performing a test. Data logs are mainly used as documentation of test and to optimise the control system, but also to solve problems.

We have implemented a logging system in the three kernel module DLL's: Controller.dll, MotorCTRL.dll and Position.dll.
When the *SignalLog* function is called in the module it creates a new text file in 'c:\logfiles' and give it a name that includes the name of the module and the date and time of the log activation.
For each sample when new data is available in the module it writes a text line to the file with data.
The format is compatible with Matlab and after a test the data can be imported as a matrix in Matlab with a fixed number of columns and a variable number of rows depending on the duration of the test.
The data written to the file are:

Written by Controller.dll (at 20 Hz):
* Timestamp of data
* Seeker position (Easting, Northing, Height)
* Line definition of path segment (Ax + By + C = 0)
* Intersection position (Easting, Northing, Height)
* Orientation data (Roll, Pitch, Compass)
* Heading error

A total of 14 floating-point values per line (sample) gives a line size of about 135 bytes and a dataflow of about 2.7 kb/s.

Written by MotorCTRL.dll (at 50 Hz)
* Timestamp of data
* Honda motor reference (Motor 0, 1, 2 and 3 in rev/min)
* Honda motor error signal (Motor 0, 1, 2 and 3 in rev/min)
* Honda motor control signal (Motor 0, 1, 2 and 3)
* Steering motor reference (Motor 0, 1, 2 and 3 in degrees)
* Steering motor error signal (Motor 0, 1, 2 and 3 in degrees)
* Steering motor control signal (Motor 0, 1, 2 and 3)

A total of 25 floating-point values per line (sample) gives a line size of about 170 bytes and a dataflow of about 8.5 kb/s

Written by Position.dll (called by controller.dll at 20 Hz)
* Timestamp of last integrated position
* Last integrated position (Easting, Northing, Height)
* Timestamp of current estimated position
* Current estimated position (Easting, Northing, Height)
* Timestamp of GPS position
* GPS position (Easting, Northing, Height)

A total of 12 floating-point values per line (sample) gives a line size of about 115 bytes and a dataflow of about 2.3 kb/s

The data logging has also helped us to discover numerous problems that we would never have discovered in any other way (wrong calculations and assignment of timestamps, wrong calculation of path, etc.). As a real-time system is impossible to debug we would not have been able to solve the problems either if we could not se what was happening from the log files.

Status and error messages

Every time the state of a module changes it sends a message to the main application that shows it to the user. Internal in the module each state has an errorcode (number) while the main application only receives the text string assigned to the state.

Sending the message

Status and error messages are sent from the modules to the main application using the Windows API function Postmessage. When a message is sent it is placed in the message queue of the main application. The handle to the main application used to send the message is given to the module when calling the Init procedure.
The Windows message identifier constant is defined in RoboConst.pas to be WM_User + 2

Errorcode - State machine

The internal error code is used to control a state machine inside the DLL.
All functions and parts of the module use the error code (state) to check if the module is in a state that allows this part of the code to be executed. For example, the encoder module can return wheel angles even though it is in an error state indicating that the absolute position indexes have not been found. However if it is in any other error state it will not return wheel angles.
The state machine ensures that successive detections of the same error will generate only one error message and it also makes it possible to send an OK message when the error has been cleared.

Module loading

The main application loads all software modules (DLL's) and calls the Init function in each of them. Many of the modules also need to access other modules. They load other modules in the same way as the main application does. Windows handles this in a smart way:
A DLL can only be loaded once under the same application and every time a DLL request to load another DLL that has already been loaded by the main application it will just receive a handle to the same DLL. The Orientation.dll is for example used by 4 other DLL's (in addition to the main application) that all request to load it, but there is only one instance of Orientation.dll that they all share. Saying this it is also obvious that the Init and Close functions must not be called by other DLL's. They must only be called by the main application.

Dynamic loading

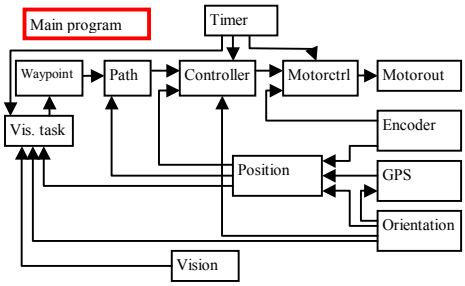
The most typical way of loading and using a DLL from another program is with static loading. The functions that are used from the DLL are linked into the program and can be called as if they were defined inside the program. The program automatically loads the DLL when the program loads and if it cannot be found an exception will occur.
To get maximum flexibility we decided to use dynamic loading. This means that we load the DLL manually when we want to. With dynamic loading we can decide what should happen if a DLL is not there and thus we can, for example, let the main application run even if some of the DLL's are not there. This would enable future projects

to remove DLL's that are no longer used, but still use modules or applications that want to load them.
We have however prioritised other things higher than this and therefore it has not been implemented this completely in all modules.

Compatibility with other programming languages

The DLL's we have made are compatible with any Windows programming language (C++, Visual Basic, Java, Delphi, etc)
All function that can be called in the DLL's are defined with the *standard call* convention[31]. This calling convention is not the default convention in either Delphi or C++ but it used by Microsoft in all Windows system DLL's and thus all API[32] calls use this convention. As all Windows programming languages can call the Windows API they can also call the DLL's we have made. This ensures that any future projects with this vehicle can reuse our DLL's no matter what programming language is used.

[31] The calling convention defines how the function parameters are transferred and the order of the parameters data bytes.
[32] API = Application Programmers Interface



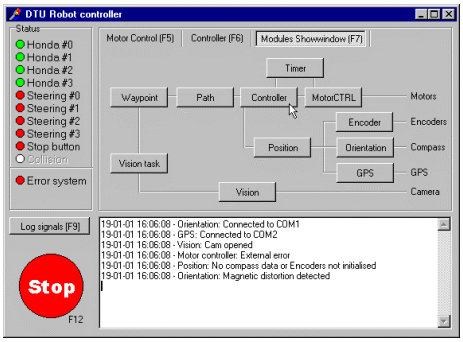


Figure 5.2 The main program handling all the software modules.

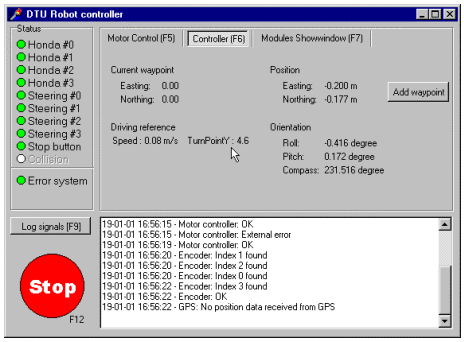


Figure 5.3 The "Controller" tab in the main program shows navigation data from the modules.

5.2. Main application

The main application handles the overall control of all the software module DLL's and the main interface to the user.

Initialisation

When running the program it loads all 12 DLL's used in the whole system. If a DLL was not found or it does not conform to the standard we have specified for this DLL, then a message will be shown to the user.
If the DLL was loaded successfully then it is initialised by calling its "Init" function as described in appendix 5.2.

Feedback to the user

Status and error messages

All the module-DLLs send a message to the main program when their state changes. This can be both error messages and status messages. When the program receives a message it writes a line to the white message box at the bottom of the window (see Figure 5.2). The line contains the date and time of receiving the message, the name of the module it received the message from and finally the text message sent from the module.

Interesting values and control signals

The program gives the user a chance to follow what is going on in the modules by showing many interesting values to the user.
From Controller.dll it receives the values shown on the "Controller" tab (see Figure 5.3):
* Current waypoint
* Current estimated position
* Current orientation (pitch, roll and compass)
* Current high level speed reference and turning point

From MotorCTRL.dll it receives the values shown on the "Motor Control" tab (see Figure 5.4):
* Current reference and error for the 4 drive motors.
* Current reference and error for the 4 steering motors.

External error system

In the left side of the program window the status of the external error system is shown. All inputs to the error system are listed except for the watchdog. If the "LED" is green (white on the monochrome monitor) then it is OK and if it is red (black) then there is an error on this item. The output of the error system is also shown at the bottom. If an error has occurred then the output of the error system will report an error even though the input error is no longer there. The error system will remain in the error state until the reset button in the front of the vehicle has been pressed.

User functions

Module settings

All module-DLL's offers a settings window that enables the user to change the parameters in the module. The window is shown by calling the showwindow function we have implemented in all DLLs. When clicking on the buttons on the "Modules showwindow" tab on Figure 5.2, the main program calls the showwindow function in the DLL. The main program does not know what happens and that a window is shown – it is all handled by the module.

Log files

Click the "Log signals" button at the left side of the window to start and stop detailed logging of all values and signals to a file. Read more about data logging in "Software modules overview" on page 83.

Adding waypoints

Click on the "Add waypoint" in the right side of the window in Figure 5.3 to add the current position to the list of waypoints in Waypoints.dll. This makes it easy to make the waypoints for a path by driving through it once and clicking this button at the wanted positions.

Stop button

The red stop button at the bottom-right of the window will immediately cut the power to all motors. It can be activated with a mouse-click or by pressing F12 on the keyboard.

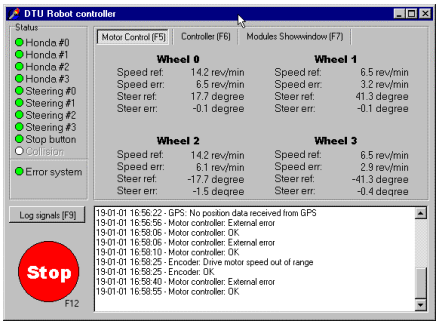
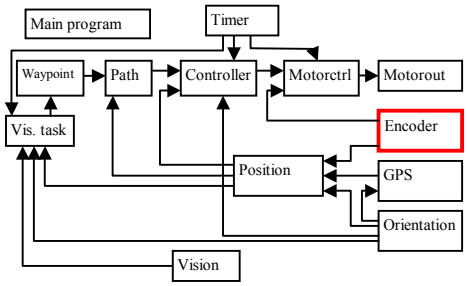


Figure 5.4 The "Motor control" tab in the main program shown references and errors for all 8 motors.

Safe closing down

When the user click the X button in the upper right corner of the window to close the application then it will first call the motor controller to verify that all speeds and angles are zero. If this is not the case then the motor controller will begin to set the speeds and angles to zero. The main program will not close until the motor controller reports that it has been done successfully. However if there is an error somewhere in the system preventing the motor controller from doing it then the program is allowed to close.



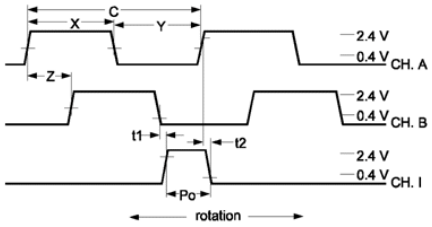


Figure 5.5 The quadrature signals

Figure 5.6 The quadrature decoder interface card.

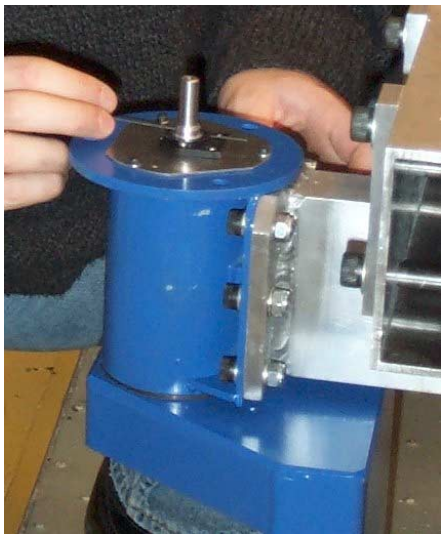


Figure 5.7 Adjusting index position on a steering motor encoder.

5.3. Encoders module

Hardware interface

We use optical incremental encoders (bought from www.usdigital.com) to determine the position and speed of four drive motors and four steering motors. The encoders do not have a shaft, but are made to fit on a shaft with a specific diameter. The encoders transmit a quadrature-encoded signal that by definition also includes the direction of the current rotation. A quadrature signal consists of two square signals (A and B) with a phase difference (see Figure 5.5). By detecting which of the signals goes high first it can determine the rotation direction. One cycle is a complete period of both signals i.e. a cycle contains 4 sharp edges. This enables the quadrature decoder to get up to 4 counter values for each cycle.
To decode the quadrature encoded signal from all 8 motors we need a decoder with a PC interface. To save money we intended to build a decoder on a PC-AT prototype card using the LS7266R1 decoder ICs from USdigital. The diagram for the card is shown in appendix 5.4. Later we decided that as this robot should be used by many other projects it would not be a durable solution and we bought complete decoder cards from USdigital.
The quadrature pulses are collected by two PC interface cards that can each handle 4 encoders.. Each card has 4 plugs on the backside where the cables from the encoders are connected (See Figure 5.6). The cards are plugged into ISA slots on the PC.

Steering motor encoders

The encoder give 512 cycles per revolutions. The encoder PC card can decode a cycle in up to 4 counts. We have chosen the maximum resolution and thus we have 2048 counts per revolution.
Each counter value corresponds to a wheel angle change of 360°/2048 = 0.18°

When the vehicle computer starts and the controller is initialised, we do not know in which direction the wheels points.
Therefore, the steering motor encoders have been ordered with an index marking. When the encoder is in the position of the index marking the index port goes high. We have programmed the 7266 PC encoder interface to load a preset value to its pulse counters when the index goes high.
The position of the index has been adjusted as close to the neutral wheel position as possible during the assembly. I.e. the counter is close to “zero” when the wheel is in the neutral position (forward direction). This is however not precise enough so we have also implemented a software-offset that can be set by the user.
The counters are 24 bit counters but we only use a very small part of this range as the encoders can send a maximum of 2048 counts per revolution. To avoid damaging the cables it will not be possible to turn the wheel more then 140 degree in each direction. We are then using a maximum range of +/-800 counts.

Figure 5.8 Encoder mounted on the Honda motor

Drive motor encoders

The encoder has 2048 cycles per revolutions. With 4x multiplication in the PC card, it becomes 8192 counts per revolution. The resolution in the measured distance with wheels having a 1.5 meters circumference is 1.5m/8192 = 0.18 mm.

Software

The Encoders.dll module is the interface to the encoders. It reads the counter values from the PC cards, convert them to angles for the steering motors and speeds for the drive motors and make them available for the other software modules.

Communication with PC cards

We made a Delphi interface unit to the two Quadrature encoder PC cards. The unit offers functions to initialise and test the cards and to read from or reset the counters on the card.
Each of the two cards contains 4 quadrature decoders with counters, that works independent. The counters are accessed by their own IO address within the address space assigned to the card.
All 8 decoders are initialised to use 4x multiplication in the quadrature decoding for maximum resolution.

Diagnostics

The ISA bus is an old fashioned system. There is no direct way of detecting if the card you are writing to is actually there or if other cards have been set to the same IO address. Therefore, we have to manually test that all 8 counters work properly:
After the initialisation, we write a counter value to each of the 8 counters. When reading the counters afterwards we can verify the value is the same as we wrote to it. By choosing a test value with "ramdom" bit changes i.e. it must not be only 0's or 1's, there is a very high probability that if the numbers are identical then the communication to this counter is working.

Card facilities

The PC7166 card is used for the drive motors. It only offers basic functionality to read and write to the counters as no more is needed for the drive motors.
The PC7266 card (see Figure 5.6) is used for the steering motors. It can handle encoders with an index marking and contains a programmable PAL to configure what should happen when the index goes high or when a counter overrun (CARRY) is detected. It can be programmed to load a preset value (this is the mode we use), to reset the counter or to generate an interrupt.

Steering motor encoders

The steering motor encoders are used to make a closed loop control of the wheel directions i.e. the position of the steering shaft. The resolution is 0.18°.

Initialising – finding the wheel positions

As we only use a very small part of the 24 bit counter range there is enough space to use two different zero bases to distinguish whether

the index has been found or not. The two zero bases are loaded to the interface card before and after searching for the index:
1) The “SteeringCounterInit” value (= 0x0FFFFF) is first loaded to the counter. When reading a counter value during the initialisation we calculate the angle of the wheel relative to this counter value i.e. relative to the position of the wheel when the vehicle computer started.
2) The “SteeringCounterZero” value (= 0x000FFF) is loaded to the preset latch, which is loaded to the counter when the index port goes high. When this value is loaded to the counter, we know the absolute position of the wheel and can calculate the exact direction in degrees.

When initialising the steering motor counters we need to search for the index. The easiest solution would be to turn the wheels one revolution, but this would damage the control cables to the drive motor.
We cannot be sure in which position the wheels are when starting to search for the index. Either because they have been rotated manually while the computer has been turned off, or because the computer might have frozen - we use Windows (!) – without being able to position the wheel safely before “closing” down. Therefore, we can only search in an area of +/- a few (20) degrees from the initial position of the wheel.
When reading counter values during this process we need to determine which zero base is closer. If the “SteeringCounterZero”is closer, then the index has been found and the initialisation has finished for this wheel.
The Motor controller software module handles the initialisation. The Encoder module only takes care of calculating the wheel angles and reporting for what wheels the index has been found.

Calculating wheel angles

The Steering angle for wheel #i is calculated using the formula:

Formula 5.1

$$Steeringangle_i = Compensation_i - \frac{360°}{2048}(Counter_i - CounterZero)$$

Where
i : The wheel number [0..3]
Compensation$_i$: is the user-defined wheel alignment for wheel #i
CounterZero : is the zero base. During the initialisation it is SteeringCounterInit = 0x0FFFFF and after the initialisation it is SteeringCounterZero = 0x000FFF

Notice the sign change from Counter to Angle: We have defined a CW rotation with positive angles, which is opposite to the encoders definition.

Drive motor encoders

The drive motor encoders are used to make a closed loop control of the wheel speeds and to track the distance travelled by each wheel.

Deriving travelled distance

For the position integration we also calculate the travelled distance for each wheel. With 24 bit counters there will be a counter overrun every 1.5m* 2^24 / 8192 meters = 3072 meters. To avoid this problem we do instead calculate the delta distance for each sample:

Formula 5.2

$$\Delta Distance_i = Sign(i) \cdot Circumference_i \frac{\Delta Counter_{i,n}}{8192 \frac{counts}{rev}} \quad [m]$$

Where
i : The wheel number [0..3]
Sign(i) : For odd wheel numbers (wheels in the right side of the vehicle, I∈[1,3]) this value is +1 and for even wheel numbers (wheel in the left side of the vehicle, I∈ [0,2]) this value is –1.
Circumference$_i$: is the circumference for wheel #i
ΔCounters : is calculated from Counter$_{i,n}$ and Counter$_{i,n-1}$ including the necessary precautions when a counter overrun or "underun" happens.

The distance change since last sample is added to an Extended floating point variable in the encoder module that counts the total travelled distance.
A spinning wheel (Figure 5.9) will currently not be detected and if it happens it will cause a considerable error in the calculated distance.

Deriving speed

The maximum resolution is 8192 counts per revolution from which we derive the speed at the rate of the motor control loop (50 Hz).
The speed is derived using the formula:

Formula 5.3

$$Speed_i = Sign(i) \frac{60000 \frac{ms}{\min}}{Timestamp_{i,n} - Timestamp_{i,n-1}} \frac{\Delta Counter_{i,n}}{8192 \frac{counts}{rev}} \quad [rev/min]$$

Where
Sign(i) : For odd wheel numbers (I∈[1,3]) this value is +1 and for even wheel numbers (I∈ [0,2]) this value is –1.
Counter$_{i,n}$: the counter value for motor #i at sample #n
Timestamp$_{i,n}$: the timestamp (in ms) for motor #i at sample #n

Figure 5.9 Wheel spin will not be discovered when calculating travelled distance.

Quantification errors

The precision with which we can derive the speed depends on the actual speed. The tires have a circumference of about 1.5 meters and as the controller runs at 50 Hz we can calculate the quantification error. At 2 m/s it is:
2 m/s = 1.33 rev/s = 10922 counts/s = 218 counts/sample = 0.5 %

In Figure 5.10 the quantification error at different vehicle speeds are shown. The average error will be half of these values.
The precision is significantly reduced at lower speeds. It is therefore necessary to average samples at lower speeds. Another way of seeing the error is to calculate the contribution by on count to the rotation speed:

$$\Delta\omega = \frac{60 \frac{s}{\min}}{TC} = \frac{60 \frac{s}{\min}}{\frac{1}{50} \cdot 8192} = 0.37 \frac{rev}{\min \cdot count}$$

where
T is the time between two controller samples.
C is the resolution of the encoder.

Speed [m/s]	Counts/sample	Error
2	218	0.5 %
1	109	0.9 %
0.5	55	1.8 %
0.25	27	3.7 %
0.1	11	9 %

Figure 5.10 Quantification errors when deriving speed from counter values.

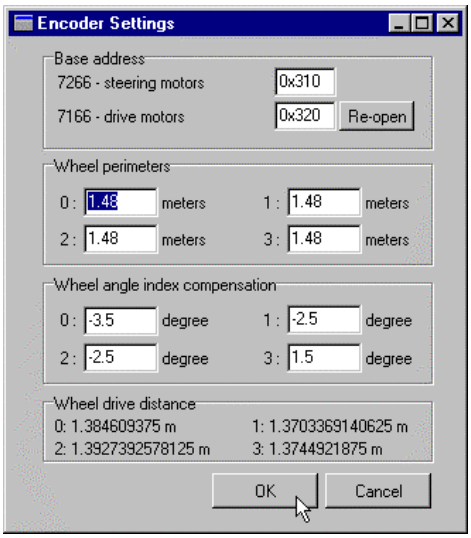


Figure 5.11 The settings window in the Encoders module

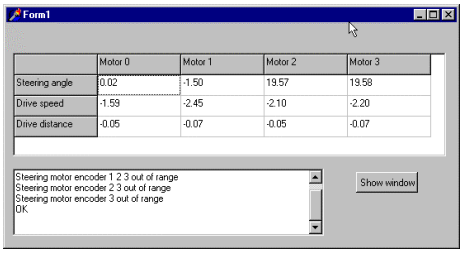


Figure 5.12 Test program made to test the Encoder module

Timestamp and Publication

Just after reading the counter values from the PC cards – before the calculations begin - we assign a timestamp to the data.
For each sample we save the following data in one record: timestamp, speed and distance for each of the drive motors and the angle for each of the steering motors.
The last 32 records are saved in a small database so that other modules can find encoder data that matches specific timestamps. When a module calls the "GetMotorSpeedAndAngle" or "GetMotorDistAndAngle" functions in the Encoders.dll they request a timestamp and they will get the data nearest to the requested timestamp.

User interface

When calling the "Showwindow" function in the DLL a window will show up (Figure 5.11) with the following elements:
Base address: Set the IO address on the ISA bus for the PC7166 and PC7266 encoder interface card. Both cards use 16 consecutive addresses from the base address. The prefix '0x' is used when it is a hexadecimal number.
Changes to the PC7166 base address can be executed by pressing the "Re-open" button. Changes to the PC7266 base address would require the index marking to be found again and therefore the controller needs to be restarted before the change is executed.
Wheel perimeters: As the tyre pressure, load and tyre wear can be different for each wheel it is possible to enter the perimeter for each wheel to obtain the maximum precision in the controller and the travelled distance for the dead reckon system.
Wheel angle index compensation: The index marking has been adjusted as close to the neutral position of the wheel as possible. The final adjustment can be made here.
The value is added to the calculated angle cf. Formula 5.1.
Increasing the value turns the wheel CCW.
Wheel drive distance: The distance travelled by each of the wheels since the module was loaded.

Tests

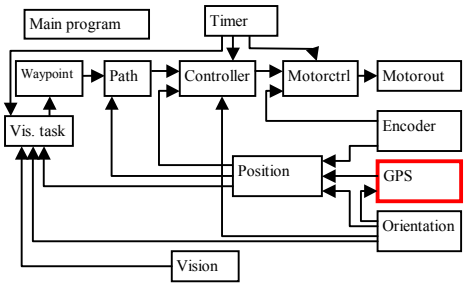
A program (shown on Figure 5.12) has been made to test the Encoder module alone. It loads and initialises the Encoder.dll and shows the calculated values for the user. It can however not test index marking searching for the steering encoders as it happens in a loop with the Motor controller software module.

Status and error messages

The encoder module can detect the following errors:

"Encoder PC card … not found or conflict."
If it is impossible to read from and write to the encoder PC interface cards. This happens if the cards were not found or there is a conflict with another card or there is an error on the card.

“Steering motor encoder x out of range” When the “GetMotorSpeedAndAngle” and “GetMotorDistAndAngle” functions are called in the encoder module it verifies the status of the steering motor position. If the angle is outside the range [-180°;180°] degree or the index of the encoder has not been found it will send this message. It will only be sent once when there is a change in the status.
“Drive motor speed out of range” If the speed calculated from the counters is more than ±200 rev/min then there must be an error somewhere. The maximum no load speed of the Honda motors is measured to 130 rev/min (125 according to the Honda manual).
“OK” If an error has been cleared.



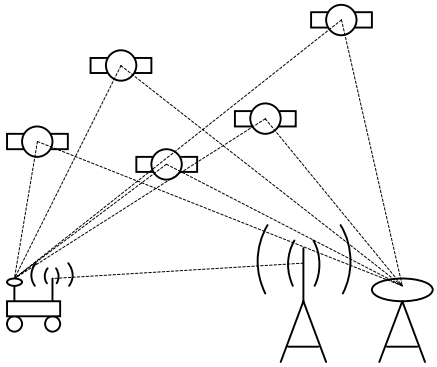


Figure 5.13 RTK requires a base station and contact to at least 5 satellites.

Figure 5.14 RTK base station. GPS antenna to the right, computer in the middle and radio transmitter to the left.

5.4. The GPS Module

The GPS system

The Global Position System is a position system with navigation satellites covering the whole world. Each satellite has a very precise atomic clock and continuously sends out a code that contains information about the UTC time (Universal Time Coordinated). When the GPS receiver receives the time code it can calculate the distance to the satellite based on information about the signal travel time. Three satellites should be enough to calculate the position but it would require an atomic clock in the GPS receiver too. As an atomic clock is extremely expensive a cheaper quartz clock is used. Therefore it is required to be in contact with four satellites to get enough equations to solve the position in 3 dimensions.
There are many types of errors that reduce the precision and several ways to compensate for these errors (see [GPS])
To get a higher precision some system also look at the phase of the signals sent from the satellites. This will increase the precision in the identification of the distance to the satellite.

DGPS

Another way of increasing the precision is with differential GPS (DGPS). A master station in a known position continually compares the data received from the satellites with its own calculations based on its known position. The difference is transmitted as corrections to the client GPS receiver with a radio signal. The client subtract the correction received by radio from its own measurements to increase the precision. The error will however not be exactly the same at the position of the client GPS as it is at the master station limiting the position accuracy to 1-5 meters.
It is possible to subscribe to corrections sent out national on the FM band.

RTK

A combination of DGPS and phase measurements is the Real Time Kinematic (RTK). It requires a maximum distance of 10-15 km between the master and client. In return we get a precision of 1-5 cm. Until now there has not been a national network of master stations. Instead you had to bring your own master station that is setup in the area you want to work in. The position of the master station must be determined before the RTK survey can begin. The system we have used includes a base station. However, recently a company has started to offer a subscription to RTK base stations through a mobile phone. RTK requires continuous contact with 5 satellites, which is only possible if no trees and buildings limit the “sight” to the sky.
When it begins (or if you loose track of a satellite) there is an initialisation delay where the client GPS receiver counts periods and tracks the phase of the signals.

Figure 5.15 The Trimble RTK base station computer and controller.

World positions

World co-ordinates are given as longitude and latitude in degrees that specifies the position on an ideal mathematical model of the earth. One of the most used ellipsoidal models of the earth is WGS 84 (World Geodetic System 1984). The surface of the ellipsoid is not necessary coincident with the surface of the earth or the mean sea level since the surface of the earth is irregular with mountains, hills, etc..
It is not easy to work with positions given as longitude and latitude as they cannot be perceived as coordinates in a rectangular coordinate system. Therefore a map projection is often used, that transforms a small area (zone) of the earth unto a unified rectangular coordinate system. There are a huge number of earth models and projections.
A common used projection is the Universal Transverse Mercator (UTM). It divides the earth in 120 zones (1 North, 1 South, 2 North, 2 South, ..., 60 North, 60 South).
In Denmark zone 32 North is used for mapping and surveying purposes even though parts of Denmark (east of 12°E) is in zone 33. After the UTM projection the coordinates are given as Northing and Easting in meters.

With the Trimble GPS module it is possible to write the UTM projection algorithm to the flash memory and make it perform the projection of the coordinates. Then we get coordinates that can be used directly in the position calculations.

Hardware interface

To test our vehicle with precision GPS we borrowed an RTK GPS system from the National survey and Cadastre Denmark. The Trimble GPS system consist of a Master station and a Rover (client). The base station is shown on Figure 5.14 and Figure 5.15.
The rover is connected to the computer with a 9-pin serial plug and is configured to send the GPS measurements at its fastest possible rate, 5 Hz.
The Trimble GPS data conforms to the NMEA 0183 specification (National Maritime Electronics Association), which describes a standard RS232 bus format for exchange of a variety of navigation information (GPS, radar, compass, etc). However, when sending UTM projected positions the message used by Trimble is longer than the NMEA 0183 standard of 80 characters.

Figure 5.16 The Trimble RTK rover on the vehicle.

Software

The GPS.dll reads the positions from the GPS and makes them available to the other software modules.

Communication

A small communication object runs in a separate thread and receives the characters from the serial port. The characters are appended to a string until a carriage return character is received i.e. the complete NMEA string has been read from the serial port.
The position string is sent from the GPS at 5 Hz.

Timestamp

We have used two different ways to determine the timestamp to assign to GPS data:
1) The UTC timestamp of the data is converted to the computer timestamp by subtracting a time correction value. We calculate the correction value from a special UTC NMEA string that the GPS module has been configured to send out at a rate of 1 Hz. The difference between the UTC and the computer timestamp at the receipt of the UTC data is the correction value.
The advantage is that we use the real UTC timestamp assigned to the GPS data. We just convert it to our time scale.
The disadvantage is that we do not know if the special UTC NMEA string that we base the correction value on, has the same calculation delay as the position.

2) There is a time delay from the position has been determined in the GPS until we can give it a timestamp:

a) The latency in the GPS is 0.1 s according to the manual.
b) The transmission of the about 80 characters through the serial port. The serial port speed is 19200 bit/s = 2133 byte/s (including stop bit). The transmission time is 80 / 2133 = 37.5 ms

The total delay is about 137 ms that we subtract from the timestamp at the time of receipt.
The advantage is that we know that the timestamp will be reasonable precise.
The disadvantage is that the precision is only reasonable. A latency specification of 0.1 s can be anything between 50ms and 150 ms.

Calculations

The position read from the NMEA string is the global position of the antenna. After decoding the string it is transformed to the position of the vehicle centre using the rotation matrix read from the Orientation.dll module.
We want to know the global position of the origin of the vehicle coordinate system. This depends on the position of the antenna in the vehicle coordinate system and the orientation (pitch, roll and compass) of the vehicle system.
If we denote the global projected coordinate system A and the vehicle coordinate system B then we have:

$$^{A}\underline{q} = {}^{A}\underline{p} - {}^{A}_{B}\underline{\underline{R}} \cdot {}^{B}\underline{v}$$

Where
$^{A}\underline{q}$: is the global position of the vehicle origin (which we want to find)
$^{A}\underline{p}$: is the global position of the antenna (the position returned by the GPS unit)
$^{A}_{B}\underline{\underline{R}}$: is the rotation matrix from the B coordinate system to A. This is read from the Orientation.dll
$^{B}\underline{v}$: is the local position of the antenna in the vehicle coordinate system. This is entered once for all by the user.

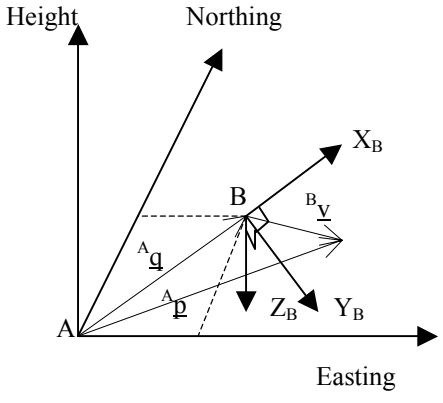


Figure 5.17 The vehicle coordinate system {B} in the global coordinate system {A}.

User interface

When calling the "Showwindow" function in the DLL a window will show up (Figure 5.18) with the following elements:

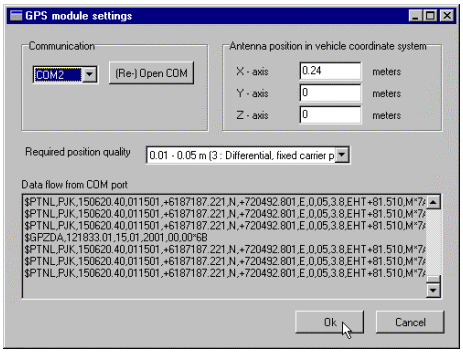


Figure 5.18 The settings window in the GPS module.

Communication: Specify what communication port the GPS is attached to. If you change the port you can open the new port by clicking the "Open COM" button.
Antenna position: It is required to enter the position of the GPS antenna in the vehicle coordinate system. The global position found by the GPS is at the antenna position. When we know the local coordinates of the antenna we can calculate the global position of the vehicle center using the rotation matrix read from Orientation.dll.
Required position quality: Depending on the number of satellites the GPS module is in contact with, and the type of radio corrections received, the precision of the position returned by the module will vary. With this option you can set a required precision and if this requirement is not met, the GPS module will report an error. The options are:

1 : Autonomous GPS fix (precision 5 - 10 m)
2 : Differential, floating carrier phase (precision 0.1 - 1 m)
3 : Differential, fixed carrier phase (0.01 - 0.05 m)
4 : Differential, code phase only – DGPS (precision 1 - 5 m)

Precision identifier #3 is required for maximum precision – which we need to get usable results.

Dataflow: In this window you can view the NMEA strings read from the GPS via the COM port.

Tests

A program has been made to test the GPS module alone. It loads and initialises the GPS.dll and regularly reads the position from it. The position and timestamp is shown to the user.
In the beginning we thought that the rotation matrix should be calculated by the GPS module as it was the only module that should use it. Therefore this test program was also used to test the calculation of the rotation matrix. It calculates 100.000 rotation matrices when pressing a button. The time on an 500MHz PC was 715 ms but this might give the wrong impression as the calculation loop probably is small enough to be in the internal processor cache and therefore is calculated much faster than in normal situations. It does however give the information that we don't have to worry about the processor power used in our situation with a rate of 20Hz.
This test program was also used to verify the calculation of the rotation matrix, enabling the user to enter an orientation and a vector that should be multiplied to the matrix and verify that the result is correct.

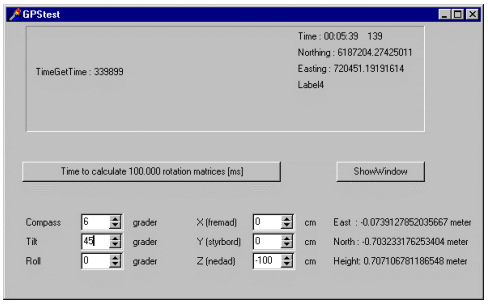


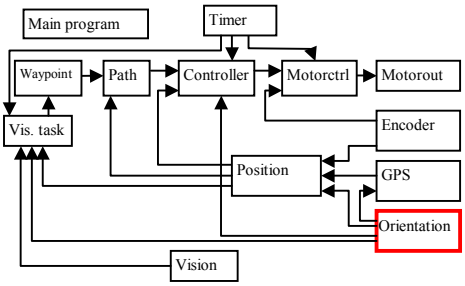
Figure 5.19 The test program for the GPS module and for the rotation matrix calculation.

Status and error messages

The GPS module can send the following messages:

"COM: …"
There is an error in the communication with the serial port. It can however not detect if there is no external hardware connected to the port.

“Connected to COMx” If the communication with the serial port has been initialised OK. It can however not detect if there is no external hardware connected to the port.
“Error converting string from GPS” If the NMEA string read from the GPS module is invalid or not complete. Our implementation of the NMEA string decoding is based on the actual output from the GPS and it does not conform to the complete NMEA standard.
“GPS data not valid (missing satellites)” If the precision of the GPS data is lower than the required precision set by the user..
“No position data received from GPS” If the GetPosition function is called and there is no position available.



5.5. Orientation module

Hardware

We bought a combined electronic compass and orientation module from Precision Navigation (see Figure 5.20). It is a 3-axis magneto-inductive magnetometer with a build-in fluid based tilt sensor and tilt compensation. The interface is a simple RS-232 serial connection through which it can output the orientation (compass + pitch + roll) at a rate up to 30 Hz. It is very compact in size (6,35 cm x 5,08 cm x 2,7 cm) and consumes only about 70 mW.
The compass module can output measurements at a rate up to 16Hz in Standard sampling mode. In this mode we get the highest precision with a repeatability on the compass reading at ±0.1°. If we enter the Fast sampling mode we can get measurements at a rate up to 30 Hz, but this will also increase the random noise with ±0.3°.
The tilt sensors can measure angles on both axes up to ±24° with a precision of ±0.2°. The natural frequency is 20Hz and the settling time is about 300 ms.

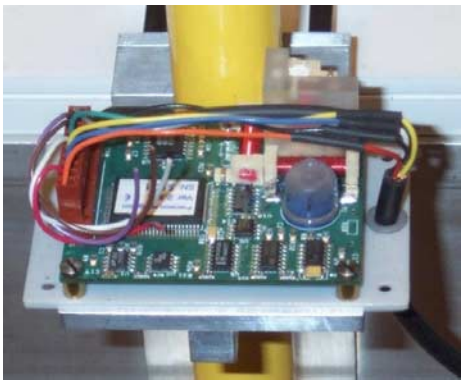


Figure 5.20 The compass module. The fluid based tilt sensor is the blue bell-shaped glass in the right side.

There are a large number of configuration facilities in the compass module. The programming of the module happens through a command line interface with a modem program like Hyperterminal, which is included in Windows. The manual describes each command and the answer.

Errors in measurement

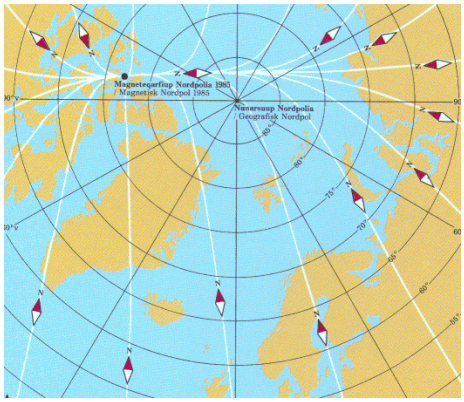


Figure 5.21 Magnetic declination happens because the magnetic north pole is not at the true north pole.

Magnetic declination

The magnetic declination is the angle between the magnetic north and the true north. Its value depends on where you are on the earth (see Figure 5.21) but it also vary over time.
An interactive map of magnetic declination can be found at:
http://www.rummet.dk/3_undervisning/uv_omorsted/uv_kompas/body_uv_kompas.html
In year 2000 the declination is about 1° on Sjælland. The software module compensates for this.

Magnetic distortion (deviation)

Magnetic fields that do not come from the earth will distort the measured field and cause a wrong compass heading to be measured. Local magnetic fields are made by iron and electronics and therefore the compass should be placed as far away from these error sources as possible. It is however not possible to entirely avoid the influence from local magnetic fields. If the distortion is mainly caused by a constant field (for example from iron) then it can be solved by making a deviation table. The influence from the iron depends on the actual heading. A deviation table lists the angle between the measured north and the actual magnetic north. This declination angle must be measured for as many headings as possible. There must be enough entries in the deviation table to be able to make a linear interpolation between two entries.

Manual deviation table

To make a deviation table the vehicle is placed in a known position. In addition you must know a number of positions in various directions around the vehicle. The vehicle is placed with heading towards each of the known positions and the measured compass heading is written down. It is now possible to calculate the actual heading between the two known positions and compare them with the measured compass heading. When having enough entries in the deviation table it is possible to calculate the actual heading for any heading (0-360°) measured by the compass.

Automatic deviation table

The compass module we have can make deviation compensation in a much easier way:
1) The compass is set in a recording mode
2) The vehicle is driven slowly in circles while varying the pitch and roll as much as possible. The vehicle must spend at least one minute to complete each circle.
3) The compass analyses the recorded compass and tilt data and by comparing the magnetic field of samples with same heading but different pitch, it can compensate for the local field.
4) The compass returns a score of how detailed a "deviation table" it could generate from the data.

We completed this test on a small hill (½m high and 2m wide) and got a good score. After this we could record improved results but they were still not good enough. The heading was still up to 20 degree wrong (before it was up to 45 degree) see "Test results" page 143.
If the problem had been the automatic method of creating the deviation table, then we could solve it by making a manual table and implement it in the Orientation module. We often get the "Magnetic distortion detected" error (see description in the "Status and error messages" section, page 105), which indicates that the distortion is not from a permanent local magnetic field but is generated by the nearby power amplifier. We moved the compass to a better place (see "Relocation and calibration of the compass" page 145) but the result did not improve.

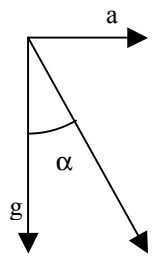


Figure 5.22 Horizontal acceleration of the compass cause the error α.

Accelerations

The compass module measures the resulting gravity vector with its fluid based tilt sensors. Therefore, it is very sensitive to accelerations of the module.
The tilt data are used both by our software to calculate the rotation matrix and by the compass it self to calculate the tilt compensated compass heading. The heading calculation is highly sensitive to tilt errors, as a tilt error of 1° will cause a heading error of about 2° (depending on where the compass is on the earth).
The tilt errors calculated below can therefore have a significantly influence on the heading error.

If the acceleration (a) is perpendicular to the gravity (g) as shown on Figure 5.22 then the resulting tilt error is

$$\alpha = \tan^{-1}\frac{a}{g}$$

Formula 5.4

Accelerations of the compass module can happen in 3 situations:

1) When the speed of the vehicle is changed.
A typical acceleration of a low speed vehicle could be 0.35 m/s^2 giving a tilt error at 2°.

2) When the vehicle is turning (centripetal acceleration)
If the vehicle turns in a circle with a radius of 1 m at a speed of 1 m/s the acceleration is

$$a = \frac{v^2}{r} = \frac{1^2}{1} = 1\frac{m}{s^2}$$

Yielding a tilt error of 6°. This is however a very high speed for such a sharp turn. If we reduce the speed to 0.5 m/s the tilt error is 1.5°

3) When a wheel hit an obstacle
The situation when a wheel hit an obstacle is shown in Figure 5.23. To pass the obstacle the wheel must be lifted the distance h during the time it takes to travel the distance d. The reality is much more complex due to the dynamics of a rubber wheel and the vehicle. The distance d can be calculated from h and r using phytagoras:

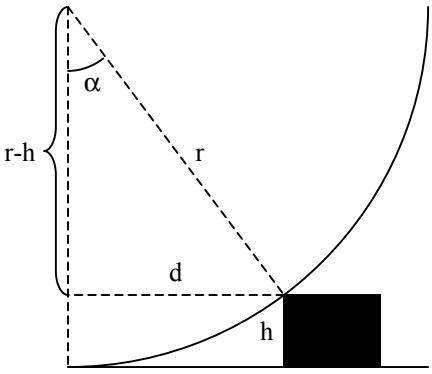


Figure 5.23 A wheel hits an obstacle.

$$r^2 = d^2 + (r-h)^2$$
$$\Updownarrow$$
$$d = \sqrt{h(2r-h)}$$

If the vehicle moves at a constant speed V and we assume that the acceleration is constant while moving the vertical distance h, we can find the acceleration:

$$h = \tfrac{1}{2}a't^2$$
$$\Updownarrow$$
$$a' = \frac{2h}{t^2} = \frac{2h}{\left(\frac{d}{V}\right)^2} = \frac{2h}{\left(\frac{\sqrt{h(2r-h)}}{V}\right)^2} = \frac{2V^2}{2r-h}$$

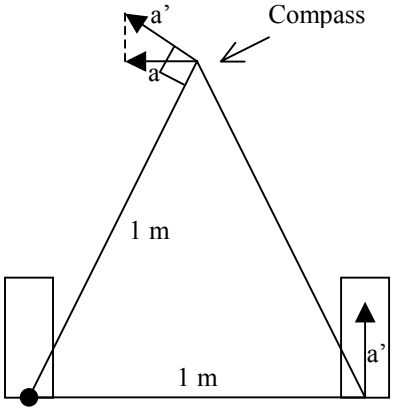


Figure 5.24 The right wheel hits an obstacle and accelerates the compass.

On Figure 5.24 we can see the influence of the acceleration a' on the compass. The acceleration will occur when the right wheel on the drawing hits an obstacle. The vehicle will rotate around the left wheel and the acceleration of the compass will be as shown (a'). The angle between the a' vector and the a vector can be found by simple geometric rules to be 30° and thus we can find the acceleration perpendicular to gravity:

$$a = a'\cos 30° = \frac{V^2}{2r-h}\sqrt{3}$$

The wheel radius is 0.24m. If the vehicle runs at a speed of 1 m/s and hits an obstacle with a height of 0.05m then the acceleration is about 4 m/s^2. From Formula 5.4 we then get a tilt error of 22°!!!

Solutions

The errors in point 1 and 2 are acceptable and can be reduced further without significant affect on the functionality of the vehicle. In normal operation in the field it will follow almost straight lines (crop rows) at low and constant speed. Only at the beginning and end of the rows these errors will influence the precision and in these situations the precision demands are reduced.

The error in point 3 can however only be reduced by lowering the operating speed of the vehicle or by placing the compass module as close to the ground plane as possible. The latter is however not possible because of the requirements for the clearance below the vehicle and because of the risk of magnetic distortion by metal objects.
We are not interested in lowering the speed either as this would reduce the functionality of the vehicle significantly.
Several solutions are possible:
1) Damping the heading data calculated by the compass module.
The damping should however be minimized while turning the vehicle to maintain maximum precision in the dead reckon system.
This would require the Orientation.dll or Position.dll module to continuously get the turning or wheel angle references set in the controllers.
2) Compensate the tilt readings before calculating the heading.
It is possible to read the raw magnetometer data from the compass. This would enable us to ignore the heading calculated by the compass module and instead calculate the heading in our own software. It would then be possible to compensate the tilt data before calculating the heading.
a) With a 3-axis acceleratometer, the tilt readings can be compensated to find the true direction of the gravity vector.
b) Based on the knowledge from the motor encoders we can calculate the current acceleration and rotation of the vehicle. This would enable us to compensate for the errors mentioned in point 1 and 2 above, but not the most critical error in point 3.

Damping data

A simple way of damping the dataflow from the compass and tilt sensors are by using the digital damping formula:

Formula 5.5

$$V_{damped}(T) = W(n) \cdot V(T) + \left(1 - W(n)\right) \cdot V_{damped}(T-1)$$

Where

$$W(n) = \frac{1}{2^n} \quad , n = 0,1,2,...$$

V(T) : is the measurement of either tilt or compass
T : is the sample number.
n is a constant set by the user to specify the weight of a new measurement.

We dampen pitch, roll and compass data using Formula 5.5. The pitch and roll data are damped using n=3 and currently it can not be changed by the user.
The compass damping can be changed by the user as described in the “User interface” section on page 104.

Software

The Orientation.dll reads the orientation data from the compass and makes them available to the other software modules.

Communication

A small communication object runs in a separate thread and receives the characters from the serial port. The characters are appended to a string until a carriage return character is received i.e. the complete data string has been read from the serial port.
The orientation string is sent from the module at 20Hz

Timestamp

There is a delay from the actual measurements are made in the compass module until we get it in the software module. The delay depends on the baud rate and whether the Fast sampling option in the compass module is enabled or not:

Measurement time	50 ms (30 ms in Fast sampling mode)
Computation time	20 ms
Serial data transfer	7.5 ms (30 byte at 38400 baud)
Total	77.5 ms (57.5 ms in Fast sampling mode)

When the software module has read and decoded the orientation string it assigns a timestamp to the orientation data. The timestamp is the current time minus the delay found above i.e. the orientation data we have is always at least 77.5 ms old.

Calculation and publication

For each set of orientation data received, the rotation matrix is calculated.
The timestamp, orientation and rotation matrix are stored as a record in a small database. The last 32 samples are stored so that other modules can find orientation data that matches specific timestamps. For example the GPS module need to search in "old" orientation data to compensate for the antenna position at the time of the position measurement.
When a module calls the "GetOrientation" or "GetRotationMatrix" functions in the Orientation.dll they request a timestamp and they will get the data nearest to the requested timestamp.
In addition it is possible to let the Orientation.dll notify another module every time new data is available. This is used by the position integrator in Position.dll to update the position estimate for each sample of orientation data.

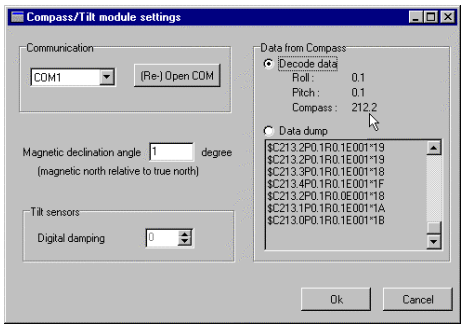


Figure 5.25 The settings window for the orientation module

User interface

When calling the "Showwindow" function in the DLL a window will show up with the following elements:
Communication: Specify what communication port the compass is attached to. If you change the port you can open the new port by clicking the "Open COM" button.

Magnetic declination: Enter the magnetic declination angle relative to true north. This angle will be subtracted from the angle read by the compass.
Data from compass: The raw text data flow from the compass can be seen or it can be decoded for easy viewing.

Tests

A program has been made to test the Orientation module alone. It loads and initialises the Orientation.dll
The orientation and timestamp is shown to the user as it is received from the compass.

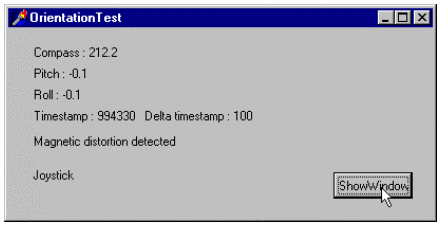


Figure 5.26 The test program made to test the Orientation module

Status and error messages

The orientation module can send the following messages:

"COM: …" There is an error in the communication with the serial port. We can however not detect if there is no external hardware connected to the port until data is actually received from the module.
"Connected to COMx" If the communication with the serial port has been initialised OK. It can however not detect if there is no external hardware connected to the port.
"No data received from compass" If a module request orientation data and data has not yet been received from the compass.
"Error calculating Rotation matrix" If an exception occurred while calculating the rotation matrix.
"Magnetic distortion detected" When the automatic deviation compensation is activated in the compass module it continuously compares the recorded field with the expected field. If the difference is to big it will set a bit in the string sent to the software module. Unfortunately we often get this message which could be due to the large electromagnetic fields generated by the nearby power amplifiers.
"Magnetometer out of range" The 3-axis magnetometer can measure magnetic fields up to ±0.8 μT on each axis. If the magnetic field is stronger than that, the compass module will set a bit in the string sent to the software module to indicate that the heading data will be invalid.
"Tilt out of range" The fluid based tilt sensors can measure a pitch and roll angle up to ±20° (according to the specifications, our measurements say about ±24°). If the tilt angle is greater than this, the compass module will set a bit in the string sent to the software module to indicate that the tilt data is invalid.
"OK - receiving data" When the Orientation.dll software module begins to receive orientation data from the compass for the first time or after an error.
"Error converting string" If the string read from the compass is invalid or not complete.

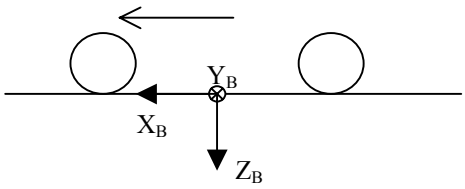


Figure 5.27 The vehicle coordinate system.

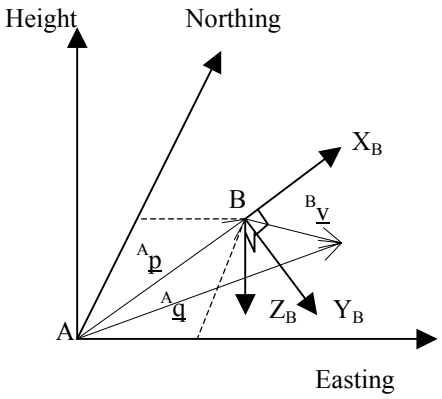


Figure 5.28 The vehicle coordinate system {B} in the global coordinate system {A}.

Rotation matrix

The vehicle coordinate system is placed like shown in Figure 5.27. The definition of the axes follow the typical definition of a vehicle coordinate system as used in the field of vehicle technique.
The definition of the axes are related to the definition of the orientation of the vehicle. All three orientation angles (compass, pitch and roll) are defined as right hand rotations around the axes as they are defined on Figure 5.27.

The rotation matrix is used by many modules (GPS, Position, Vision, Controller) to convert vectors given in the local vehicle system to vectors given in the global system.

In Figure 5.28 the vehicle coordinate system is placed in the global coordinate system. We know the global position of the vehicle origin (^{A}p) and a local vector in the vehicle coordinate system (^{B}v). The local vector could for example be the position of a tool on the vehicle.
To find the global position of the tool we need to convert the local vector to the global coordinate system before we can add it to the vehicle position. This depends on the orientation (pitch, roll and compass) of the vehicle system.
If we denote the global projected coordinate system A and the vehicle coordinate system B then we have:

$$^A\underline{q} = {}^A\underline{p} + {}^A_B\underline{\underline{R}} \cdot {}^B\underline{v}$$

Where
$^A\underline{p}$: is the global position of the vehicle origin.
^{A}q : is the global position of the tool.
$^A_B\underline{\underline{R}}$: is the rotation matrix from the B coordinate system to A.
$^B\underline{v}$: is the local position of the tool in B.

The rotation matrix from {A} to {B}

The transformation from the global world coordinate system {A} to the vehicle coordinate system {B} can be split up in 3 rotations:

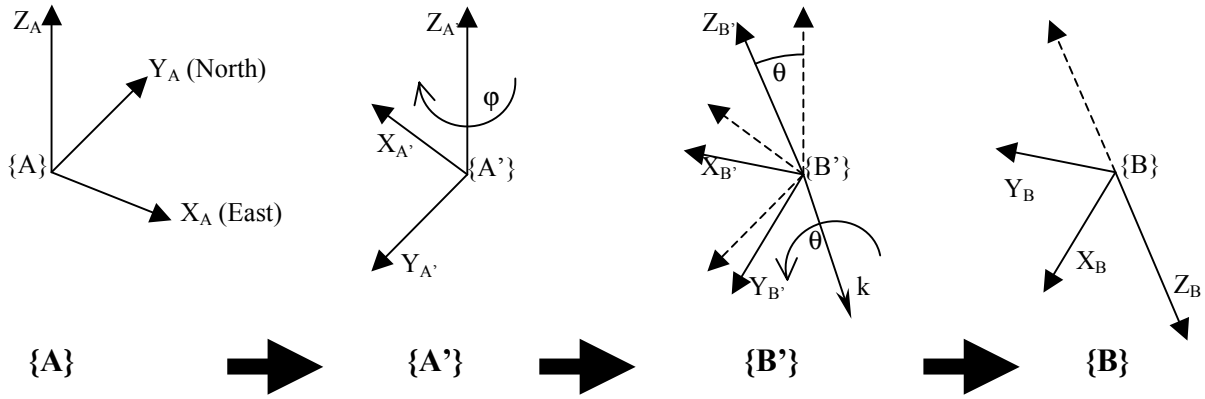


Figure 5.29 The rotation from frame A to frame B in three steps.

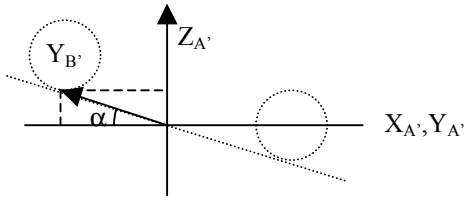


Figure 5.30 The vehicle on a slope (pitch).

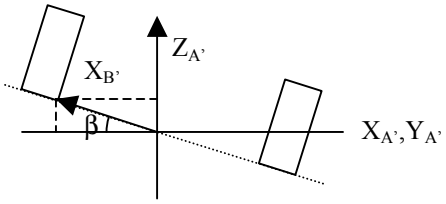


Figure 5.31 The vehicle on a slope (roll)

From {A} to {A'} the system is rotated with the compass angle φ which defines the angle between North (Y_A) and the heading of the vehicle ($Y_{A'}$).
From {A'} to {B'} the system is rotated the angle θ around the k vector. We do not know θ and k, but they can be found from the Pitch (α) and Roll (β) angles. The Pitch angle is the angle of the slope when the vehicle goes up or down a hill as shown in Figure 5.30.
The Roll angle is the angle of the slope when the vehicle goes across a hill i.e. is rotated sideways as shown in Figure 5.31.

From {B'} to {B} the axes are reordered.

A rotation matrix defines a coordinate system (e.g. {B}) relative to another coordinate system (e.g. {A}). It is defined as the unity vectors of {B} given in {A}:

$$ {}^{A}_{B}\underline{\underline{R}} = [\, {}^{A}\underline{\hat{X}}_{B} \quad {}^{A}\underline{\hat{Y}}_{B} \quad {}^{A}\underline{\hat{Z}}_{B} \,] $$

In the following sections we will develop the rotation matrix for each of the 3 rotation steps shown in Figure 5.29, from the global frame {A} (coordinate system) to the vehicle frame {B}. Finally they are multiplied to get one rotation matrix from {A} to {B}.

Rotation from {A} to {A'}

This is a simple rotation about the Z-axis. However as it is a left hand rotation around the defined Z-axis, the sign of the Sin elements has changed compared to traditional representations:

$$ {}^{A}_{A'}R = R_Z(\varphi) = \begin{bmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} $$

Formula 5.6

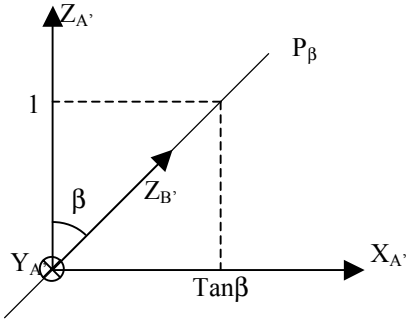


Figure 5.32 The roll angle contribution.

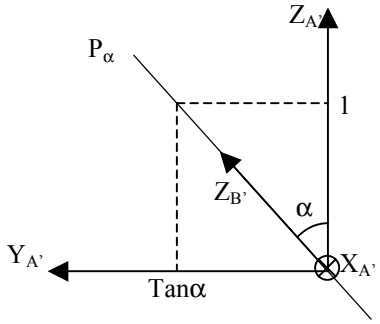


Figure 5.33 The pitch angle contribution.

Rotation from {A'} to {B'}

We cannot find the rotation matrix directly from α (pitch angle) and β (roll angle), because they are not defined as traditional Euler rotations around an axis. The two angles are totally independent and are defined relative to the same axis (gravity). Therefore we cannot order one of them before the other and make a rotation matrix for each of them that are just multiplied together.
Instead we can combine the two rotations α and β to an equivalent rotation θ about a vector k. We can directly use θ and k to find the rotation matrix using formula 2.80 in [CRAIG89]
In the following sections we will first find θ then k and finally use the formula from [CRAIG89] to calculate the rotation matrix.

Based on the Pitch and Roll angles we are going to find where $Z_{B'}$ is in $\{Z_{A'}\}$ i.e. how much it is rotated (θ) and in which direction (k vector).

The roll angle β is the angle between the gravity vector (coincident with $Z_{A'}$) and a plane P_β that contains $Z_{B'}$ as shown on Figure 5.32. The plane contains and rotates around the $Y_{A'}$ axis, however we do not know where $Z_{B'}$ is in that plane as it also depends on the pitch.

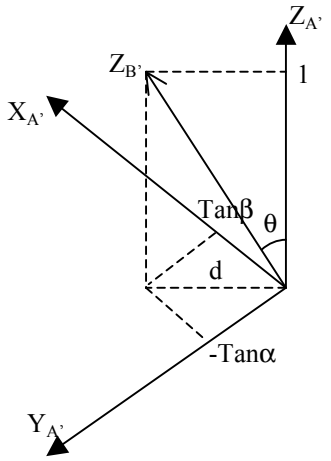


Figure 5.34 The resulting tilt vector.

The pitch angle α is the angle between the gravity vector (coincident with $Z_{A'}$) and a plane (P_α) that contains $Z_{B'}$. This plane contains and rotates around the $X_{A'}$ axis. The pitch angle is defined as the right hand rotation around the X-axis, therefore the angle in Figure 5.33 is negative.

The $Z_{B'}$ axis is coincident with the intersection of the planes P_α and P_β. The combination of Figure 5.32 and Figure 5.33 is shown in Figure 5.34. The $Z_{B'}$ vector is the intersection of the two planes P_α and P_β.

The distance d is:

$$d = \sqrt{(\tan\alpha)^2 + (\tan\beta)^2}$$

and the relation to the angle γ is:

$$d = \tan\theta$$
$$\Updownarrow$$
$$\theta = \arctan d = \arctan\sqrt{(\tan\alpha)^2 + (\tan\beta)^2}$$

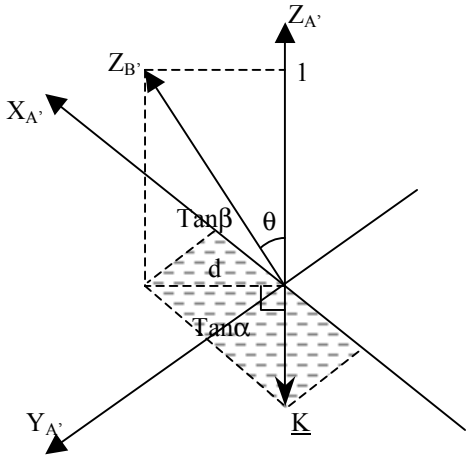


Figure 5.35 The equivalent rotation.

The vector K about which the $Z_{B'}$ is rotated, is in the $X_{A'}$, $Y_{A'}$ plane and is perpendicular to d as shown in Figure 5.35.
Figure 5.36 shows the scene from the top:

The angle γ of the K vector to the $X_{A'}$ axis can be found with the atan function. However as it must work in all 4 quadrants we must use the extended atan2 function:

$$\gamma = \mathrm{Atan2}(\tan\beta, \tan\alpha)$$

The K vector coordinates are:

$$\underline{K} = \begin{bmatrix} \cos\gamma \\ \sin\gamma \\ 0 \end{bmatrix}$$

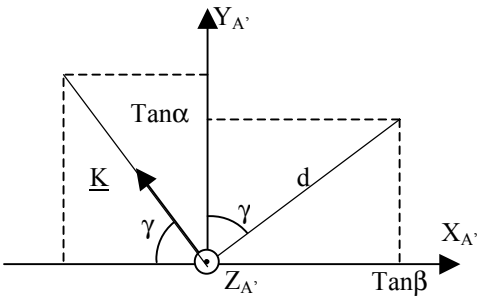


Figure 5.36 The K-vector seen from the top.

From [CRAIG89] we get the general formula for the equivalent rotation:

$$R(K,\theta) = \begin{bmatrix} k_x{}^2(1-\cos\theta)+\cos\theta & k_xk_y(1-\cos\theta)-k_z\sin\theta & k_xk_z(1-\cos\theta)+k_y\sin\theta \\ k_xk_y(1-\cos\theta)+k_z\sin\theta & k_yk_y(1-\cos\theta)+\cos\theta & k_yk_z(1-\cos\theta)-k_x\sin\theta \\ k_xk_z(1-\cos\theta)-k_y\sin\theta & k_yk_z(1-\cos\theta)+k_x\sin\theta & k_zk_z(1-\cos\theta)+\cos\theta \end{bmatrix}$$

Finally we can write the rotation matrix from {A'} to {B'} by inserting K:

Formula 5.7

$$ {}^{A'}_{B'}R(\gamma,\theta) = \begin{bmatrix} (\cos\gamma)^2(1-\cos\theta)+\cos\theta & \cos\gamma\sin\gamma(1-\cos\theta) & \sin\gamma\sin\theta \\ \cos\gamma\sin\gamma(1-\cos\theta) & (\sin\gamma)^2(1-\cos\theta)+\cos\theta & -\cos\gamma\sin\theta \\ -\sin\gamma\sin\theta & \cos\gamma\sin\theta & \cos\theta \end{bmatrix} $$

where

$$ \gamma = \text{Atan}\,2(\tan\beta, \tan\alpha) $$

$$ \theta = \text{Atan}\sqrt{(\tan\alpha)^2 + (\tan\beta)^2} $$

Rotation from {B'} to {B}

The last rotation is a simple reordering of the axes. This is done to follow the convention and rotate it to the defined vehicle coordinate system. The Z-axis goes in the opposite direction and the X and Y axes switch place.

Formula 5.8

$$ {}^{B'}_{B}R = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} $$

Rotation from {A} to {B}

The final rotation matrix is the multiplication of the three rotation matrices in Formula 5.6, Formula 5.7 and Formula 5.8:

$$ {}^{A}_{B}R = {}^{A}_{A'}R \cdot {}^{A'}_{B'}R \cdot {}^{B'}_{B}R $$

$$ = \begin{bmatrix} c\varphi c^2\gamma s\gamma v\theta + s\varphi s^2\gamma v\theta + s\varphi c\theta & c\varphi c^2\gamma v\theta + c\varphi c\theta + s\varphi c\gamma s\gamma v\theta & -c\varphi s\gamma s\theta + s\varphi c\gamma s\theta \\ -s\varphi c^2\gamma s\gamma v\theta + c\varphi s^2\gamma v\theta + c\varphi c\theta & -s\varphi c^2\gamma v\theta - s\varphi c\theta + c\varphi c\gamma s\gamma v\theta & s\varphi s\gamma s\theta - c\varphi c\gamma s\theta \\ c\gamma s\theta & -s\gamma s\theta & -c\theta \end{bmatrix} $$

where

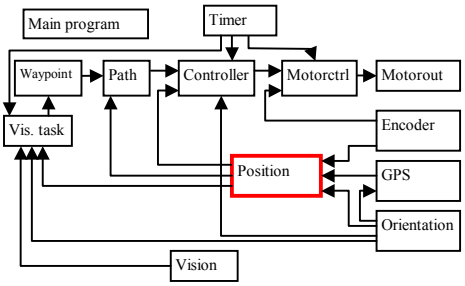
$$ \gamma = \text{Atan}\,2(\tan\beta, \tan\alpha) $$

$$ \theta = \text{Atan}\sqrt{(\tan\alpha)^2 + (\tan\beta)^2} $$

cγ = cosγ, sγ = sinγ, cφ = cosφ, sφ = sinφ, cθ =cosθ, sθ = sinθ, vθ = 1-cosθ

The rotation matrix must be re-calculated every time new orientation data is available i.e. at a rate of 15-30 Hz. Therefore some effort have been put in the implementation to make the calculation as processor efficient as possible.

The current implementation spends 690 ms to calculate 100.000 rotation matrices with a 500 MHz K6-2 processor.



5.6. Position module

The position module gives the absolute position of the vehicle as its output. It is used by the controller and by the vision task planner.
The position module consist of two parts:
1) Position integrator (dead reckon)
2) Position estimator

Position integrator

The position integrator first requests the newest orientation data (rotation matrix) available. From the encoder module it reads the direction and the total travelled distance for each wheel.
To increase the precision we want the orientation data to be in the middle of the linearization of the encoder data. On Figure 5.37 the orientation data at the time t_2 is in the centre of the line between the encoder data at t_1 and t_3. To do that we need to read encoder data in the "future", i.e. we request encoder data $½\Delta t_{orientation}$ later than the timestamp of the available orientation data. This is not a problem as the orientation data is delayed so the requested encoder timestamp will still have passed.

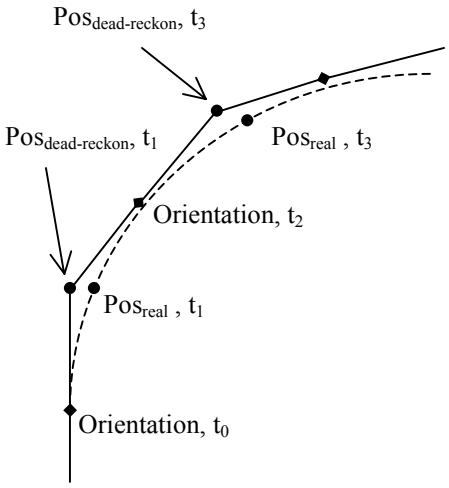


Figure 5.37 The position integrator

From the encoder data we calculate a distance vector that indicate the direction and distance of the vehicle origin - in its own local coordinate system - since the last encoder data was read. When we use symmetric 4 wheel steering and the wheel movement follow the Ackerman steering equations then the distance vector will always be coincident with the vehicle x-axis (forward direction) i.e. it will be on the form (d, 0, 0). As the compass directly shows the direction of the vehicle x-axis the globally moved distance will in this case be the distance d in the compass direction.
However this advanced way of calculating the position-change is very relevant when using alternative steering axes. When using front or rear wheel steering or especially if it drives sideways, the vehicle will not move in the same direction as the compass shows.

When multiplying the local position-change vector to the rotation matrix given by the orientation module we get the change in the global coordinate system (Easting, Northing, Height). This global position change is summated for each integration.

Timestamp

The calculated position is based on the timestamp of the rotation matrix plus $½\Delta t_{orientation}$ as described above and therefore this timestamp is assigned to the calculated position. The compass module currently runs at 20Hz an thus $½\Delta t_{orientation}$ = 25 ms.

Publication

The last 32 positions are saved in a small circular database. This database is used internal by the Position module to lookup old position as described in the "When GPS is available" section below.

The orientation module controls the sample rate, as we will have little benefit from integrating more often than orientation data are available. The Position module receives a notification from the orientation module every time new data are available and it then performs the integration.

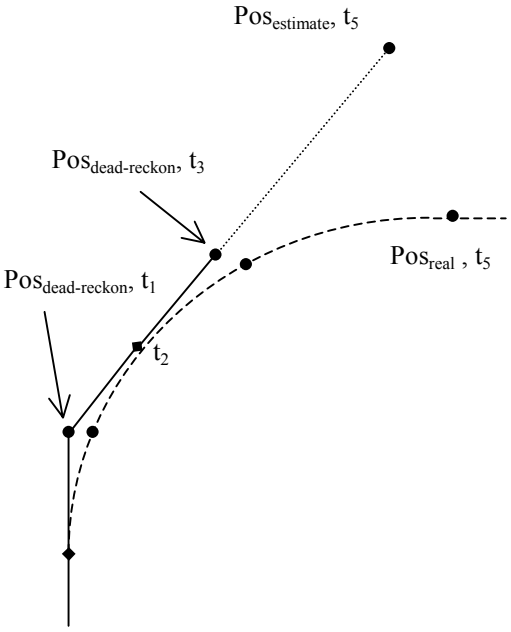


Figure 5.38 The position estimator

Position estimator

The integrated position is always delayed because of a delay in the compass module. The delay in the compass module is 58 ms when using the fast sampling mode (78 ms in standard mode). On Figure 5.38 the present time is at t_5 and the last orientation data available is at t_2 i.e. $t_5 - t_2 = 58$ ms. As described in the "Timestamp" section above, the timestamp of the position integration is 25 ms after this i.e. t_3 is 25 ms after t_2. The integrated position is therefore delayed 33 ms (t_5-t_3). At a speed of 1 m/s the vehicle will then be 3.4 cm further than the position given by the integrator. We therefore need to estimate the current position based on previous known positions. Due to the definition of Ackerman steering formula the vehicle will always drive in circles (forward movement is just a circle with infinite radius). A natural solution could therefore be to use the last three integrated positions to calculate the equation of a circle and find the position at the present time. Another and more general solution could be to solve an n-order polynomial of the time for each axis based on the last m positions [CRAIG89].

We have chosen to use the simplest solution as a beginning, which is a linear extrapolation from the last two positions (first order polynomial). As shown on Figure 5.38 there is a difference between the real position of the vehicle (pos_{real},t_5) and the estimated position based on the last two integrated positions ($pos_{estimate}$, t_5).

When GPS is available

The position integration and estimation is still very relevant when the GPS is attached to the system. The maximum sample rate with our GPS equipment is 5 Hz. In addition there is a delay from the GPS measurement until we have the position decoded in the software. The position module therefore needs to calculate the position change since the last GPS position.
We use the timestamp of the most current GPS data available to lookup in the position database (the last 32 integrated positions are saved). We can then find the change in position since the last GPS data and add this change to the GPS position.

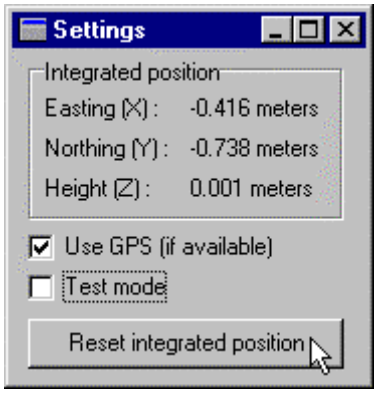


Figure 5.39 The position module settings window

The position module will return a position no matter if the GPS is available or not. The only difference is the origin of the global coordinate system in which the position is given.
When the GPS is available it use the GPS origin - a fictive distance to Greenwich and Equator (it is fictive because of the projection). When the GPS is not available the origin is the position of the vehicle when the integrator was initiated.

User interface

When calling the "Showwindow" function in the DLL a window (Figure 5.39) will show up with the following elements:

Integrated position : this is the latest integrated position.
Use GPS : If GPS data is available then they will be used to find the position if this option is enabled. If we want to test the precision of the dead-reckon system then we still want the GPS data to be logged in the logfile as a reference but we do not want it to be used for the position calculation. In that case this option should not be enabled.
Test mode : Test mode was made for the situation where we do not have encoder data available i.e. when testing the module on a desktop computer. When test mode is enabled it will generate encoder data as if the vehicle was running at a constant speed.
Reset integrated position : Reset the integrated position to calculate future position from the current position. It is important to do before starting a dead-reckon test but has no effect on the resulting positions when using GPS.

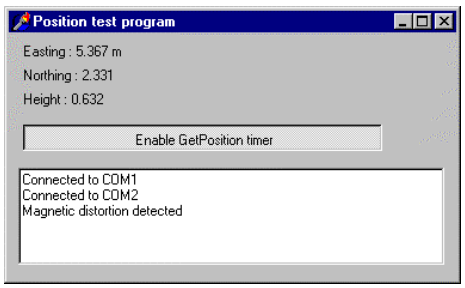


Figure 5.40 Test program used to test the Position module

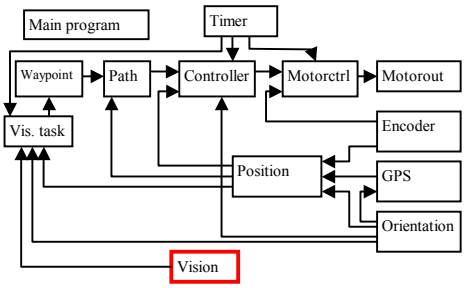
Tests

A program has been made to test the Position module alone. It loads and initialises the Encoders.dll, Orientation.dll and Position.dll.
The position is shown to the user.
This test program was however only used for the initial tests. The real position based on data from all 8 encoders can only be calculated when all motors are running with the controllers.
The final application that handles all the software modules and that offers the main interface to the user, shows the position to the user. Therefore this has been used to make the final verifications of the position module.
Initially we tested that driving 1 meter in a compass direction (e.g. south or north) would actually produce a position change of 1 meter in that direction. Further tests happened by driving the vehicle with dead reckoning (see “The position integrator” on page 143)

Status and error messages

The position module can send the following messages:

“Integrator not called yet by Orientation module” If a module calls the GetPosition function, but no position is available because the integration has not begun yet.
“Integrator started” When the integrator has been called for the first time.



5.7. Vision module

Hardware - the FUGA 15 PCI Camera

The camera used in this project is based on cmos technology. This gives exciting possibilities for use in high-speed control loops that is not found in the CCD technology. It is possible to include functionality based on cmos components directly in the photo sensor chip. In the Fuga 15 camera there are gain control and A/D conversion in the chip. The pixels are read from the chip as integer values just like an ordinary memory module. The pixel rate is specified to be 5 MHz, but due to limitations in the PCI card the actual obtainable pixel rate is about 1.5 MHz for images larger than about 1000 pixels. An image at 32x32 pixels (about 1000 pixels) can be read at 1500 Hz!

Figure 5.41 The camera installed in the special designed mounting base.

Image data from the Ccam

The Ccam has a resolution of 512x512 pixels with 8-bit color depth in gray scale. Each image therefore has a size of 256 KB and with an expected control loop speed at 25 Hz it would require 6400 KB of image data to be read and analyzed every second.
It is possible to tell the Ccam only to return a smaller part of the image (a window) and to omit every second pixel (or other fraction).
We need to use the maximum image width offered by the camera. It is therefore initialized to return every fourth pixel in both x and y direction yielding an image at 128x128 pixels. The maximum frame rate is then 92 Hz. Running the control loop at 25 Hz with this smaller image leaves processor power for all image analyzes and control calculations as the image data flow is reduced to 400 KB/s.
In commercial applications the vision module would be implemented with a DSP to lower the processor requirements.

Software

The Vision.dll module reads images from the camera, analyzes them and calculates positions of interesting targets in the image.

Threshold

To reduce the data flow the gray image is thresholded to a 1 bit monochrome image. If the gray value (intensity) is greater than the threshold value it becomes a white pixel in the monochrome image and if the value is smaller it becomes a black pixel.

Adjusting brightness

The 8-bit resolution of gray values is not absolute due to the on-chip gain control. This enables us to run the vision system under a wide range of natural light conditions. We can change the gain so that the interesting parts of the image always have intensities around 127, no matter if the scene is exposed to intensive sunlight or it is a gray rainy day.
The threshold value is therefore a constant set to 127. The image brightness (gain) is controlled by the “AutoBrightness” function that is called for each image sample.

Experimentally it is found that good results can be obtained when adjusting the brightness relative to the intensity of the target. The "AutoBrightness" function averages the intensity of 5 pixels around the center of mass of the currently selected target. This spot intensity should be a certain number of gray levels above the threshold value. If it is different, the image brightness gain is adjusted accordingly.

When initializing the vision loop, this method is however not enough because there must be a target chosen before it works. Choosing a random target would not work because it might hit the darkest area of the image which would cause the whole image to become white. Therefore the brightness is initially controlled by another function ("InitBrightness"). This function adjusts the brightness so that about 5% of the image becomes white in the thresholded image. Doing it this way however requires extra precautions to be taken when analyzing the image for a target to follow:
1) If there is no target in the image, it will still make the brightest 5% of the image white.
2) If there are many components (possible targets), it will only show the brightest 5% of the pixels belonging to the components and their intensity will be very close to the threshold value.

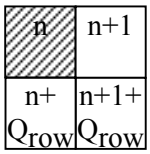


Figure 5.42 The filtering mask.

Removing noise

The thresholded image will contain noise i.e. stand-alone white pixels that will only confuse the image analyzes. The source can either be defective pixels in the photosensitive chip or components in the scene that are not of interest (small stones or other items laying on the ground). The noise is removed with a filter. A pixel becomes white only if all of the 3 neighbor pixels n+1, n+Q_{row} and n+Q_{row}+1 are white. If the number of pixels in a row is Q_{row}, then the logical value of the pixel is:

Formula 5.9

$$P_n = P_n \text{ AND } P_{n+1} \text{ AND } P_{n+Q_{row}} \text{ AND } P_{n+Q_{row}+1}$$

Extracting components

To analyze the tresholded image and find components (white areas) in it, we use the connected component-labeling (CCL) algorithm shortly described in [JMC98].
The result of the CCL is a partitioning of all white pixels into a set of numbered components. Each of the components consists of a number of pixels that are connected according to the choice of connectivity method (4-connectivity or 8-connectivity).

We have extended the functionality to calculate the size in pixels, the center of mass and the average intensity for each component. This is useful information when choosing one of the components as the target.

Choosing a target

Depending on the environment there will be none, one or more real targets (objects or plants that we want the vehicle to follow). There will however always be several components in the image due to the automatic brightness control. The problem is to sort out the components that do not represent a real target.

For a component to qualify to be a possible target it must fulfill the following requirements:
1) The size in pixels must be greater than the minimum component size.
2) The size in pixels must be smaller than the maximum component size.
3) The average intensity of the component must be a certain number of gray values ("Minimum target intensity") higher than average intensity of the whole image.

The parameters for the above selections can be set by the user in the configuration window included in the vision module.

The components that qualified to be a possible target are then weighted according to their size and position in the image. The two components with the highest weight are chosen as the current targets. When a new target is found it is assigned a new unique ID which will be the same for this target until it disappears at the bottom of the image. The image sample rate must therefore be high enough to ensure that the target only moves in small steps between each image. If the sample rate is not high enough there is a possible risk that it will choose a wrong (new) target believing that it is the same target (same ID) as on the last image.
When a target is close to disappearing at the bottom of the screen, it will be sent on to the vision task planer as a local waypoint.

Converting image plane to world ground plane coordinates

When we have the pixel coordinate (x_p, y_p) of the target, we want to convert this to coordinates in the real world. To do this we use the pinhole model of the camera.
The pinhole model is an ideal camera in which all light rays passes through a distinct point in space (projection center) as straight unbroken lines. In reality this is not the case due to lens distortion and other deformations. It is possible to compensate the image to remove the distortions by taking a picture of a constructed scene with components in well-known positions and analyze where they show up in the image plane. We have chosen not to do this because the errors we get by using the pinhole model are small compared to the errors that occur in the 2D to 3D conversion.

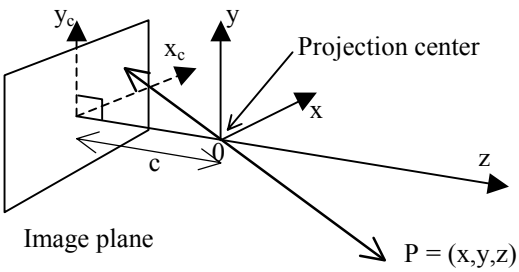


Figure 5.43 Projection of a point (P) on the image plane.

The camera constant (focal length, denoted as c) and the image plane (size of photo sensitive chip) specifies the geometry of the camera. The camera constant is specified to be 8 mm on the lens, but previous projects have measured it to be 8.61 mm [KFV99].

The camera coordinate system is defined with origin in the projection center as shown in Figure 5.43. The x-axis and y-axis are parallel to the image plane while the z-axis (*optical axis*) points away from the image plane (towards the scene). The image plane is where the photosensitive chip resides. The image coordinate system (x_c, y_c) is a displacement of the camera coordinate system, -c along the x-axis.

An image point in the camera coordinate system P = (x,y,z) will be projected trough the projection center to the image plane. The relation between camera coordinates (x_c,y_c) and image coordinates (x,y,z) are defined by the scaling factor c/z:

Formula 5.10

$$x_c = -x \cdot \frac{c}{z} \quad \Leftrightarrow \quad x = -\frac{z \cdot x_c}{c}$$

Formula 5.11

$$y_c = -y \cdot \frac{c}{z} \quad \Leftrightarrow \quad y = -\frac{z \cdot y_c}{c}$$

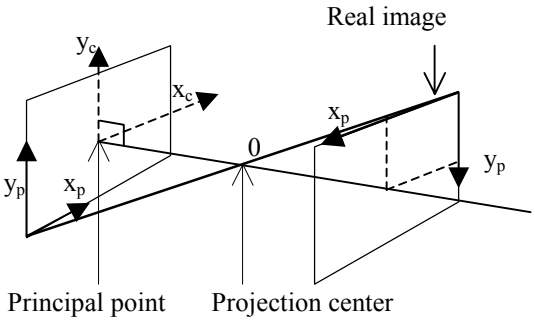


Figure 5.44 Pixel coordinate system in the real image.

When the software reads pixels from the camera the coordinate system is placed in the top left corner with the x_p-axis pointing to the right and the y_p-axis pointing downwards. This is shown in Figure 5.44 (imagine standing in the projection center looking at the real image). This is the usual convention when showing and editing images on the computer screen.
The projection of this coordinate system on the image plane is also shown in the figure.

The relation between the (y_p, x_p) coordinate system in pixels and the (y_c, x_c) coordinate system in meters can be found from the left side of Figure 5.44:

Formula 5.12

$$\frac{y_c}{d_{chip}} = \frac{y_p}{n_{p,y}} - \tfrac{1}{2}$$

$$\Updownarrow$$

$$y_c = d_{chip,y}\left(\frac{y_p}{n_{p,y}} - \tfrac{1}{2}\right) = 0{,}0064m \cdot \left(\frac{y_p}{128} - \tfrac{1}{2}\right)$$

$$= 50 \cdot 10^{-6} \cdot y_p - 3{,}2 \cdot 10^{-3}[m]$$

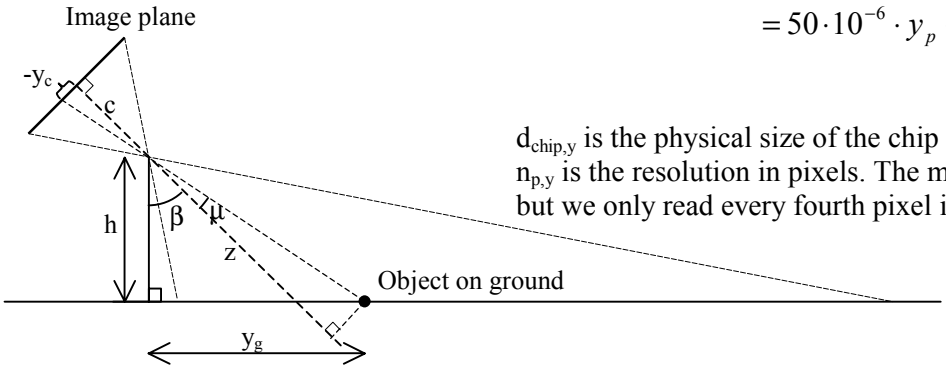


Figure 5.45 An object on the ground is projected on the image plane.

$d_{chip,y}$ is the physical size of the chip = 6,4 mm
$n_{p,y}$ is the resolution in pixels. The maximum resolution is 512 pixels, but we only read every fourth pixel i.e. our resolution is 128 pixels.

The direction, physical chip size and resolution are the same on the x-axes as the y-axes:

Formula 5.13

$$x_c = d_{chip,x}\left(\frac{x_p}{n_{p,x}} - \tfrac{1}{2}\right) = 50 \cdot 10^{-6} \cdot x_p - 3{,}2 \cdot 10^{-3}[m]$$

To calculate the position of an object in the real world, we need to find the angle μ. This is done from Figure 5.45:

$$\tan\mu = \frac{-y_c}{c}$$

$$\Updownarrow$$

$$\mu = \tan^{-1}\left(\frac{-y_c}{c}\right)$$

Formula 5.14

We also need to find the z coordinate of the object. The z coordinate is shown on Figure 5.45 as the projection of "Object on ground" to the camera coordinate system:

Formula 5.15

$$\cos(\beta+\mu)=\frac{h}{\frac{z}{\cos\mu}}$$

$$\Updownarrow$$

$$z=\frac{h\cdot\cos\mu}{\cos(\beta+\mu)}$$

Finally we can calculate the (x_g, y_g) coordinate of the object in the ground coordinate system. The x-axis of the ground system is coincident with the x-axis of the camera coordinate system, but points in the opposite direction. Combining Formula 5.10 and Formula 5.15 yields:

Formula 5.16

$$x_g=\frac{h\cdot\cos\mu}{c\cdot\cos(\beta+\mu)}x_c$$

Where the variables are as shown on Figure 5.45:
h : the height of the focal point above the ground
c : the camera constant (8.61 mm)
μ : the angle from the optical axis to the object
β : the angle from the optical axis to vertical

From Figure 5.45 we get:

Formula 5.17

$$\tan(\beta+\mu)=\frac{y_g}{h}$$

$$\Updownarrow$$

$$y_g=h\tan(\beta+\mu)$$

We now have the needed formulas to calculate the position of an object on the ground in meters when we have its pixel position in the image. With Formula 5.12 and Formula 5.13, the (x_p, y_p) pixel coordinates are converted to (x_c, y_c) coordinates on the photo sensitive chip in meters. Then we can calculate μ from Formula 5.14, which is used in Formula 5.16 and Formula 5.17 to calculate the ground coordinates (x_g, y_g) of the objects.

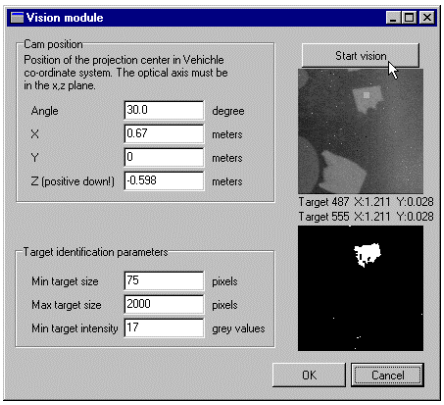


Figure 5.46 The settings window in the vision module.

Users interface

When calling the "Showwindow" function in the DLL a window will show up (Figure 5.46) with the following elements:
Cam position: The position of the camera projection centre in the vehicle coordinate system. The angle is between vertical (down) and the optical axis i.e. in the range [0;90] degrees.
Target identification parameters: Specify some requirements for a target to be valid. The minimum and maximum size in pixels will ensure that it only chooses targets with the size of the actual targets. The minimum target intensity is the difference in grey values between the average intensity of the target and the average intensity of the whole image. A typical value could be 20 grey values. Setting a higher value requires a higher contrast between the object and the background.
[Images]: Two images are shown to the right.

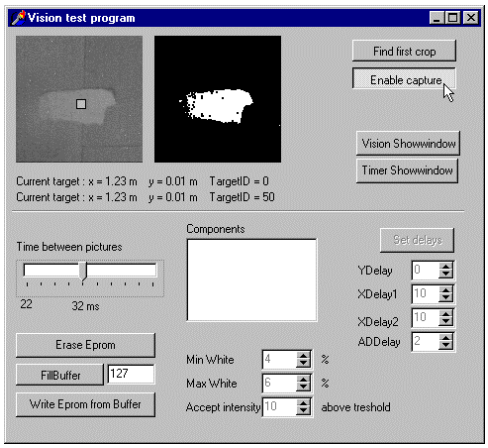


Figure 5.47 The program made to test the vision module and camera.

1) The latest unmodified grey image read from the camera. The current target is marked with a small grey square as shown on Figure 5.49.
2) The tresholded image based on image 1.
Between the images the position and target id of the targets are shown. The position is given in the vehicle coordinate system.

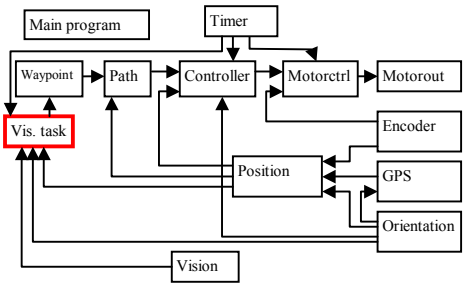
Tests

A program has been made to test the camera and the vision module. It has been used in several phases of the project and therefore it is quite a mess of features. Many of the features do not work any longer after the vision functionality was encapsulated in the Vision.dll. In fact all features below the line in Figure 5.47 are obsolete.
The current edition of the program loads and initializes the Vision module and shows the images and targets (if any) to the user.

Status and error messages

The vision module can send the following messages:

"Cam opened" The connection to the PCI interface card has been opened.
"Error communicating with cam" The configuration of the camera (image size, brightness, etc) has failed.



5.8. Vision task planner

The vision task planner evaluates the targets found by the vision system and makes global waypoints that are added to the waypoint database.

The vision task planner follows the targets returned from the vision system. Each target is assigned a unique id by the vision system so the vision task planner can follow the change of the specific target. When the target that is currently followed by the task planner is no longer seen by the vision system, it is add to the waypoint database as a global waypoint.

When there are no more targets we expect to be at the end of the row. The task planner generates a waypoint at the position where the next targets are expected to be found, i.e. the position of the beginning of the next row.

The vision task planner is not very intelligent at the moment. If it sees a target while going towards the automatically generated "target searching waypoint" then it will be added to the end of the list. When it has reached the new row it will continue the circle to get to the waypoint that was found on the way. On the way to this waypoint it will se a target from the previous row which is added to the end of the list and it will drive in a circle.

Users interface

There are currently no settings that can be changed by the user. Calling the "showwindow" function in this DLL will just show an empty window.

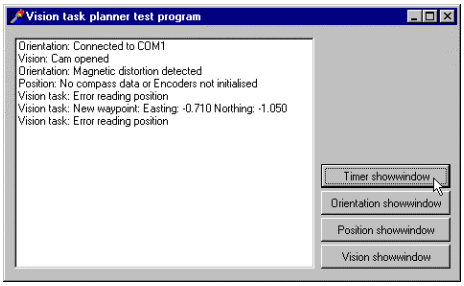


Figure 5.48 Program made to test the vision task planner.

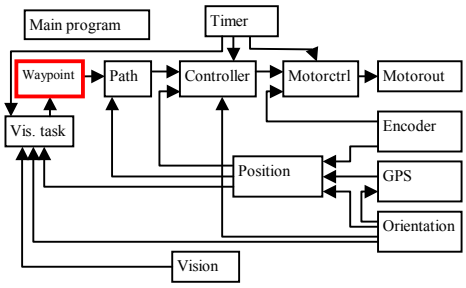
Tests

A program has been made to test the vision taskplanner and vision module (see Figure 5.48). It loads and initialises the vision, taskplanner, orientation and position module.
We have tested it by moving the camera forward by hand. It was tested on a table with small pieces of paper to simulate plants. The test program shows the waypoints that was created by the vision task planner.

Status and error messages

The vision task planner can send the following messages:

"… DLL not loaded - Vision task planner disabled The vision task planner needs the Orientation, Position, vision and waypoint modules to be loaded to work properly. If any of these fail to load then the task planner will be disabled.
"Error reading position" If it could not read a valid position from the Position module then the task planner will be disabled.
"Error reading orientation" If it could not read a valid rotation matrix from the orientation module then the task planner will be disabled.



5.9. Waypoint module

The Waypoint.dll software module handles a database of waypoints. The waypoints are used by the Path.dll module that calls Waypoint.dll every time a new waypoint is needed.
The waypoints are stored in a text file with a format compatible with Matlab. This enables us to easily plot the waypoints together with the actual path recorded by the Position module.
The last saved file with waypoints is automatically loaded when Waypoint.dll is initialised.

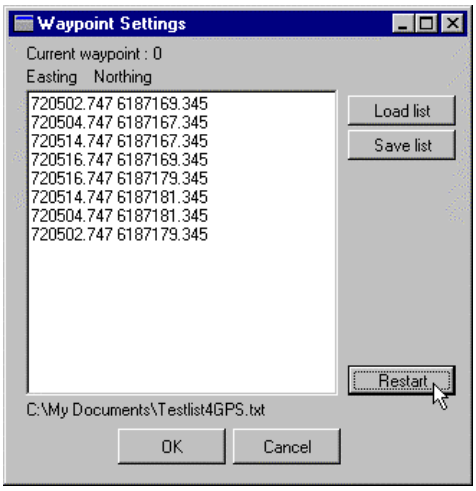


Figure 5.49The waypoint settings window

User interface

When calling the "Showwindow" function in the DLL a window will show up as shown in Figure 5.49.
Current waypoint : The number shows the waypoint line that was returned at the last call to Getwaypoint i.e. the vehicle is currently driving towards this waypoint. If the value is 0 then there has not been called a waypoint since it was restarted.
Waypoints : This is a simple text editor in which the waypoint are written. There must be one and only one waypoint on each line and the waypoints are written in the order East, North with one or more spaces in between.
Load/Save list : Press these buttons to load and save the waypoints to a text file. The last saved file is automatically loaded next time the module is initialised.
Restart : Press this button to reset the waypoint pointer i.e. the next call to GetWaypoint returns the first waypoint in the list.

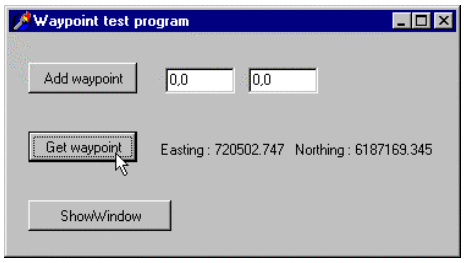


Figure 5.50 The test program made to test the waypoint module

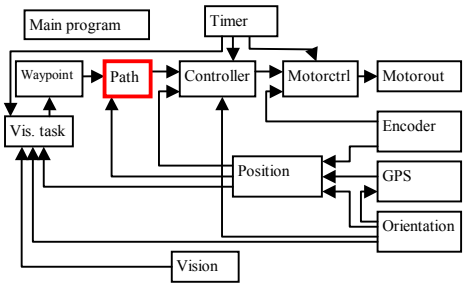
Tests

A program (shown on Figure 5.50) has been made to test the waypoint module. It loads and initialises Waypoint.dll and gives the user access to the three key functions in the module: Add a waypoint, Get next waypoint and ShowWindow.

Status and error messages

The waypoint module can send the following messages:

"No waypoints loaded" If no waypoints were loaded when initialising the module i.e. no files have been saved or the file is empty.
"Error in waypoint list" If there were errors in the waypoint list loaded when initialising the module.
"No more waypoints" If the GetWaypoint function is called and there is no more waypoints in the list.
"New waypoint(s) received" If the GetWaypoint function is called after the module has entered the "No more waypoints" state and new waypoints have been added after entering this state.



5.10. Path module

The path module reads waypoints from the waypoint module and generates a path the vehicle should follow through the waypoints. The current implementation is very simple. It only makes straight lines from one waypoint to another. This is a simplification of the original intention with this module. It should use circles or splines to make a smooth path to avoid generating step inputs for the controller, but to save time and reach our goal we had to do it this way. With this simple solution, the user gets the job to make waypoints that describe a reasonable smooth path.
To make the first line to the first waypoint it calls the position module and uses the current position as the "*from*" waypoint.

Line equation

We cannot use the typical line definition y=ax+b as it is not defined for lines of the type x=c (vertical lines). Therefore, we use the line definition:

$$Ax + By + C = 0$$

$$\Updownarrow$$

Formula 5.18

Formula 5.19

$$y = -\frac{A}{B}x - \frac{C}{B} \qquad ,B \neq 0$$

If we have to points P_{from}= (x_{from}, y_{from}) and P_{to} = (x_{to}, y_{to}) then we can use Formula 5.19 to find the constants A, B and C.

x_{to}-x_{from} = 0:

This is a vertical line and thus B=0. From Formula 5.18 we have:

$$x_{from} = x_{to} = -\frac{C}{A}$$

If we set A=1 then C=-x_{to}.

$$\begin{Bmatrix} A \\ B \\ C \end{Bmatrix} = \begin{Bmatrix} 1 \\ 0 \\ -x_{to} \end{Bmatrix}$$

x_{to}-x_{from} ≠ 0:

This is a line with a slope<>∞ i.e. we can use the line definition in Formula 5.19. If we set B=1 then we have:

$$y = -Ax - C$$

The slope of the line is:

$$-A = \frac{y_{to} - y_{from}}{x_{to} - x_{from}}$$

Finally we can find C from Formula 5.18:

$$C = -Ax - By = \frac{y_{to} - y_{from}}{x_{to} - x_{from}} x_{to} - y_{to}$$

The line constants are:

$$\begin{Bmatrix} A \\ B \\ C \end{Bmatrix} = \begin{Bmatrix} -\frac{y_{to} - y_{from}}{x_{to} - x_{from}} \\ 1 \\ \frac{y_{to} - y_{from}}{x_{to} - x_{from}} x_{to} - y_{to} \end{Bmatrix}$$

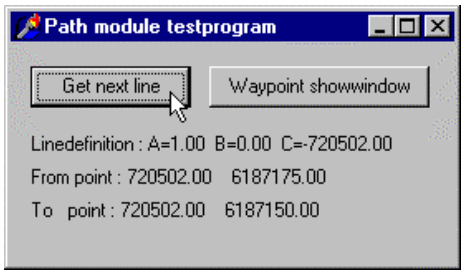


Figure 5.51 Test program for the path module.

User interface

When calling the "Showwindow" function in the DLL nothing will happen. The function has been implemented but it does not open any window because there are currently no parameters that need to be changed in this module.

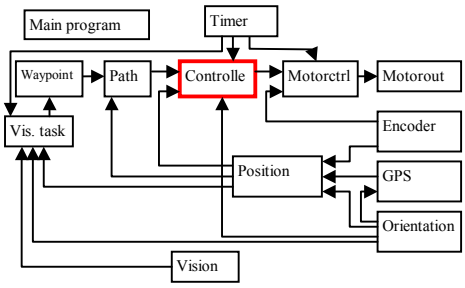
Tests

A program has been made to test the path module (Figure 5.51). It loads and initialises Path.dll and Waypoint.dll. The user can call the *GetNextPathSegment* function in the DLL by clicking the 'Get next line' button.

Status and error messages

The path module can send the following messages:

"Waypoint DLL not loaded – path planner disabled" If the waypoint module cannot be found or loaded then the path planner will not work.
"Position DLL not loaded – path planner disabled" If the position module cannot be found or loaded then the path planner cannot make a path from the current position to the first waypoint. Therefore it will be disabled.



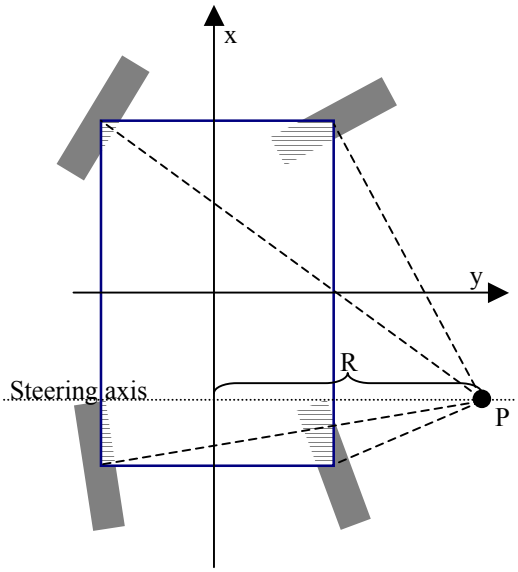


Figure 5.52 The vehicle turns about a point (P) on the steering axis.

Figure 5.53 The "Logitech Wingman" joystick gives many possibilities.

Formula 5.20

5.11. Controller

The main controller controls the direction and speed of the vehicle. It outputs the speed and angle references for the motors controlled by the motor controller.

Controller modes

The controller can be in 3 different modes: Manual control, automatic control and no action. The mode currently depends on what button is pressed on the joystick. For maximum safety when making the first tests with the system, the controller was in no action mode if no button was pressed on the joystick or if no joystick is present.
Holding button one on the joystick (Figure 5.53:1) activates the manual control mode in which the speed and turning of the vehicle is controlled by the joystick.
Holding button four (Figure 5.53:4) on the joystick activates the automatic control mode in which the speed and turning of the vehicle is controlled automatically based on the path it tries to follow.
Later we felt confident that it would behave well when in automatic mode, so we changed the functionality. Now a single click on button two will activate the automatic mode and it will remain in this mode until the manual control button is pressed.
The two active controller modes output a vehicle speed and a rotation radius (R on Figure 5.52). The rotation radius is combined with the steering axis set by the user (see the "Users interface" section on page 127) to get a rotation point (P on Figure 5.52). From the vehicle speed and rotation point, the speed and position of each wheel can be calculated using the Ackerman formula on page 131 in "Motor controller module".

Manual control

When button one is pressed on the joystick the controller enters manual mode. The position of the joystick is read as (x ,y) coordinates. For both x and y axes the controller adds a dead band to the centre position and converts the joystick position to a floating point value in the range [-1;1].
The maximum manual control speed is set by the user in the controller.dll settings window. We discovered very soon that it had to be changed very often depending on the environment. When driving fast in the field the speed is set to 2 m/s (close to the maximum possible speed) and when needing precise positioning of the vehicle the maximum speed had to be reduced to about 0.8 m/s.
Therefore we implemented a maximum speed adjustment with the extra slider on the base of the joystick (Figure 5.53). With this slider the maximum speed is adjusted in the range 0.15 m/s - 2 m/s.
The maximum speed is multiplied with the joystick value to give the resulting speed.
The turning radius R (shown on Figure 5.52) is calculated from:

$$R = \frac{r_{\min}}{x_{joystick}}$$

where

r_{min}: is the minimum turning radius (sharpest turn) allowed when twisting the joystick maximum to a side. This value is adjusted with the extra joystick slider (which also sets the maximum speed) in the range 0.2 m – 2.5 m. When setting a higher maximum speed, the minimum turning radius is also increased to give a more stable steering.
$x_{joystic}$: is the position of the joystick sideways. The value is between –1.0 and 1.0. If the value is 0 (driving forward) then R is set to +infinity.

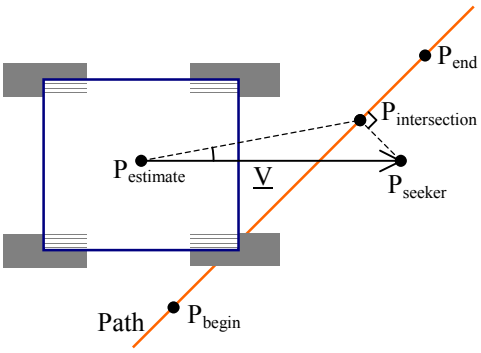


Figure 5.54 The vehicle follows the path by calculating a heading error to a target point ($P_{intersection}$) on the path.

Automatic control

When clicking button four on the joystick, the controller will enter the automatic mode.
The controller calls the path module to get a segment of the path it should follow. The segment is a line definition (Ax+Bx+C=0) and two points on the line specifying the beginning and end point of the segment (P_{begin} and P_{end} on Figure 5.54).

Control method

The method we use to make the vehicle follow the path is based on methods used to make AGV's follow a line on the ground using an optical sensor.
The line we follow is just not visible and our "optical sensor" is a calculated ahead seeking. The seeker position (P_{seeker} on Figure 5.54) is a user-defined distance in the vehicle forward direction. It is used to find a point on the line ($P_{intersection}$) that we want the vehicle to drive towards. This yields a direction error that controls the motor speeds and wheel angle references.

To calculate P_{seeker}, the controller gets the current position ($P_{estimate}$ on Figure 5.54) from the position module and the rotation matrix ($\underline{\underline{R}}$) from the orientation module. If the user-defined ahead seeking is $\underline{V}$ = ($d_{aheadseeking}$, 0, 0) then we get:

Formula 5.21

$$\underline{P}_{\text{seeker}} = \underline{P}_{estimate} + \underline{\underline{R}}^{global}_{vehicle}\,\underline{V}^{vehicle}$$

Finding the intersection point

To find the intersection point we need to find a line perpendicular to the path and containing the point P_{seeker}. The line Ax + By + C =0 can be written on the form:

Formula 5.22

$$y = ax + b$$

where

$$a = -\frac{A}{B} \qquad ,B \neq 0$$

$$b = -\frac{C}{B} \qquad ,B \neq 0$$

The orthogonal line is:

Formula 5.23

$$y = -\frac{1}{a}x + b'$$

When inserting a point (x_0, y_0) in Formula 5.23 we can find b':

$$b' = \frac{1}{a}x_0 + y_0$$

and thus the the equation for the orthogonal line is:

Formula 5.24

$$y = -\frac{1}{a}x + \frac{1}{a}x_0 + y_0$$

To find the intersection between the original line and the orthogonal line we solve for x:

$$ax + b = -\frac{1}{a}x + \frac{1}{a}x_0 + y_0$$

$$\Updownarrow$$

$$x = \frac{ay_0 + x_0 - ab}{a^2 + 1}$$

We can now insert the definitions of a and b from Formula 5.22. Though they are not defined for B=0, it can be proven that the following equation is also valid for B=0:

Formula 5.25

$$x_{\text{intersection}} = \frac{-ABy_0 + B^2x_0 - AC}{A^2 + B^2}$$

To find the corresponding $y_{intersection}$ we can insert $x_{intersection}$ in Formula 5.22:

Formula 5.26

$$y_{\text{intersection}} = -\frac{A}{B}x_{\text{intersection}} - \frac{C}{B} \qquad , B \neq 0$$

If B=0 we need to insert $x_{intersection}$ in the orthogonal line definition instead (Formula 5.24):

Formula 5.27

$$\begin{aligned} y_{\text{intersection}} &= -\frac{1}{a}x + \frac{1}{a}x_0 + y_0 \\ &= -\frac{B}{A}x + \frac{B}{A}x_o + y_0 \\ &= y_0 \end{aligned}$$

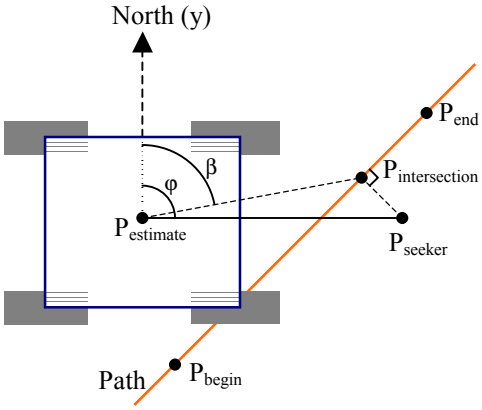


Figure 5.55 The heading error is calculated from the compass (φ) and the direction to a point on the path (β).

We can now calculate $P_{intersection}$ using $P_{seeker}=(x_0, y_0, z_0)$ in Formula 5.25, Formula 5.26 and Formula 5.27.

Valid path segment

If $P_{intersection}$ has passed the end point of this path segment (P_{end}) then the segment is no longer valid. Then the controller requests a new path segment from the path module and the intersection point has to be calculated again with this new segment.
It can happen that the intersection point is before the beginning point (P_{begin}) of the new line segment because we only use straight lines. When this happens we still consider the segment to be valid.

Heading error

From the current position ($P_{estimate}$) and the target point on the path ($P_{intersection}$) we can calculate the heading reference (β on Figure 5.55) We use a modified edition of the Atan2 function that we have called Atan2ex. It returns the compass direction from one point to another i.e. the angle is in the range [0;360[and it is measured from the y-axis (direction north).

$$\beta = \text{Atan2ex}(\Delta y, \Delta x)$$

where

$\Delta x = P_{intersection,\ x} - P_{estimate,\ x}$

$\Delta y = P_{intersection,\ y} - P_{estimate,\ y}$

Having the current compass heading of the vehicle (φ) we get the heading error (γ):

$$\gamma = \beta - \varphi$$

It does however need to be normalized to the range [-180;180] before it can be used in the controller. This is done with a modulus:

$$\gamma' = ((360 + 180 + \gamma) \bmod 360) - 180$$

Turning point

The heading error is minimized using a P controller i.e. the heading error is multiplied with the P gain to get a control value. From the control value we calculate the turning radius (R on Figure 5.52). This is done all in one formula:

$$R = \frac{45}{K_p \gamma} \quad [m]$$

where

γ : is the heading error.

K_p : is the P gain.

45: is a scaling factor so that the maximum possible heading error of 90° yields a maximum turning radius of 0.5m if K_p=1.

Limitations

The control method described in the above sections has some limitations:

1. It follows a line but not necessary in the correct direction.

This is not a problem if the angle between the vehicle heading and the line direction never exceeds 90 degree. With a stable controller this can only happen in two situations:

a) If the vehicle points in the opposite direction of the first two waypoints when entering automatic control mode.

b) If two adjacent path segments are perpendicular or almost perpendicular to each other.

Both situations can easily be avoided.

2. It is very sensitive to errors in the compass heading.

The compass heading is a direct part of the heading error. When the controller has minimised the heading error to zero there will still be an actual heading error of the same size as the error in the compass. This error is multiplied with the distance to the intersection point resulting in a steady-state position error.

For example, if the compass shows a heading that is 10° wrong and the ahead seeking distance is 2 m then there will be a constant position offset of about 2m*sin(10°) = 0.35m !!!

This is a serious problem as the compass can actually have a heading error of 10°. Read more about heading errors on page 100 ff. in "Orientation module".

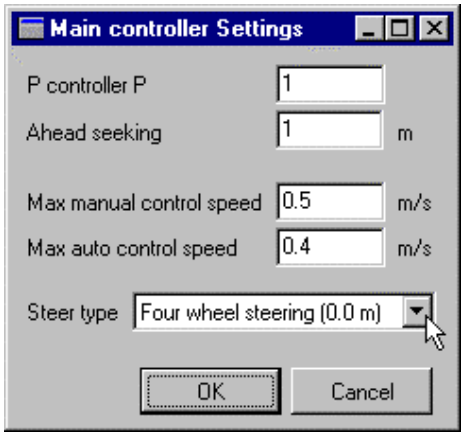


Figure 5.56 The settings window in the controller module

Users interface

When calling the "Showwindow" function in the DLL a window (Figure 5.56) will show up with the following elements:
P controller P: The P gain for the position (heading) controller.
Ahead seeking: The distance, in the vehicle forward direction, from the centre of the vehicle to the *seekerposition*. See description in the "Control method" section on page 124.
Max manual control speed: When controlling the vehicle manually with the joystick, the speed can be controlled continuously variable between zero and the value entered in this field.
Max auto control speed: This is the speed reference when in the automatic control mode.
Steer type: Specify where the Ackerman steering axis should be. For example when set to 0 m, the point around which the vehicle turns will always be on a line through the centre of the vehicle. See Figure 5.52.

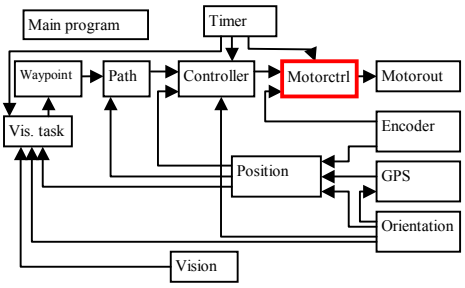
Tests

This module requires all other modules to be loaded. The program we made to test this module was later refined to become the main application that controls the whole system. The main application is described on page 87 ff.

Status and error messages

The main controller can send the following messages:

"… DLL not loaded - Controller disabled" The controller needs the Orientation, Position, MotorCTRL and Path modules to be loaded to work properly. If any of these fail to load then the controller will be disabled.
"Joystick not connected to computer" The joystick is used to control the main modes in the controller. If the joystick has fallen of, then the controller will be disabled.



5.12. Motor controller module

The motor controller module implements four position-controllers for the steering wheel angles, and four speed-controllers for the drive motors.
It also handles the initialisation of the steering wheel positions.

Control loop

The motor controllers are called by the timer module. The controllers run at 50 Hz. Initially we made a rough calculated of the time constant for the Honda motor in the system at 0.3 s. This would give a recommended control rate of 30 Hz. The motor constants for both steering and Honda motor are not well defined in the documentation. Therefore we set the rate to 50 Hz. We experienced that it worked well for both steering motors and Honda motors and the controllers could easily be made stable. Later we calculated a time constant (based on estimated motor constants) for the steering motors to be about 0.02 s. This would require a control loop rate of at least 200 Hz. We have not investigated this further as it works, but we expect this to be caused by the Curtis power amplifiers that transforms step inputs to ramp inputs.

Honda motor controllers

We have implemented four individual controllers for the four wheels. This is a simplification that causes some problems because the wheels are not independent in the real system. If all wheels have proper friction to the ground then their rotation will be locked together with the Ackerman formulas.
If we for example set one of the wheels to drive 1 m/s, and the other three controllers were disabled, then all wheels would drive 1 m/s. If all controllers are enabled then three wheel will try to go 0 m/s while the one wheel will try to go 1 m/s i.e. they will work against each other.
We will not get as obvious deviations in the references as in the above example because we calculate the correct speed referenced based on the Ackerman formula. However deviations will occur due to:

- Wheel positions not measured precisely or they vary under different load conditions.
- Wheel perimeters not measured precisely and because the perimeter of the air filled tires will vary under different load conditions.
- Steering motors not following their reference (steady-state errors)

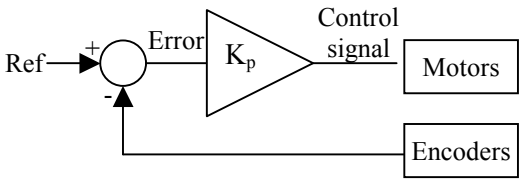


Figure 5.57 A P-controller.

P controller

With a P controller (Figure 5.57) having a gain that causes the system to be over-damped, these small deviations will not cause big problems. If a wheel runs a little faster than it should, then the power to this wheel would just be reduced slightly.
However when trying to reach a P-gain optimised for high speed and low steady-state error, these deviations will become more significant. If one wheel runs slightly faster than expected then the error

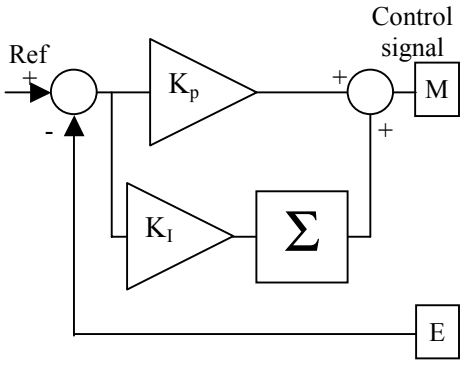


Figure 5.58 A PI-controller.

(reference-measurement), and thus the power sent to the motor, can end up being close to 0 or even negative i.e. the wheel will not contribute pulling the load of the vehicle or maybe it will even work against the other motors.

PI controller

With a basic PI controller (Figure 5.58) these deviation will cause much bigger problems. Small constant errors can wind up in the integrator and can end up causing the wheels to work against each other at full power.

First choice of controller

Due to the above considerations we chose to implement a P controller to start with. When trying the controller in reality we discovered a big problems that appears to be caused by backlash in the Honda gear (see page 139 in "Test results".

Second choice of controller

We implemented a PI controller because the integrator would dampen these sudden direction changes. As expected, it could not be used in its standard form. The wheels began to work against each others resulting in sudden jumps and other unexpected behaviour.
We implemented some practical ways to reduce these problems:
1) A ceiling to the integrator sum
2) Set the integrator sum to zero when the controller is activated after an error or stop, and when the reference is set to 0
3) Only to integrate errors of importance (>1 rev/min) i.e. small error will not be added to the sum.

add. 3: If one wheel has a little smaller circumference than the other wheels then it will be forced to constantly rotate a little faster than its reference because the other wheels will pull it. This will cause a wind up in the integrator and make the controller reverse the direction of the motor to brake the wheel. The value of 1 rev/min is chosen by experiments.

With these practical modifications the controllers worked acceptable. It is however obvious that this is not a general solution to the problem.

An I controller is not good for the drive wheels. From the log files we could see one of the problems: When the vehicle makes a sharp turn to the right (1m radius), the left wheels should spin almost 3 times as fast as the right wheels, but the I controller is much to slow and therefore the inner wheels can easily run at full power while the out wheels doesn't contribute at all.

Future choice of controller

A general solution could be to implement a MIMO controller that includes the Ackerman formula and the speed of the other wheels in the control of each wheel.

Another thing to consider is if the vehicle runs in heavy terrain and one wheel loose track and spins. This will give a significant change of the time constant and probably make the controller unstable.

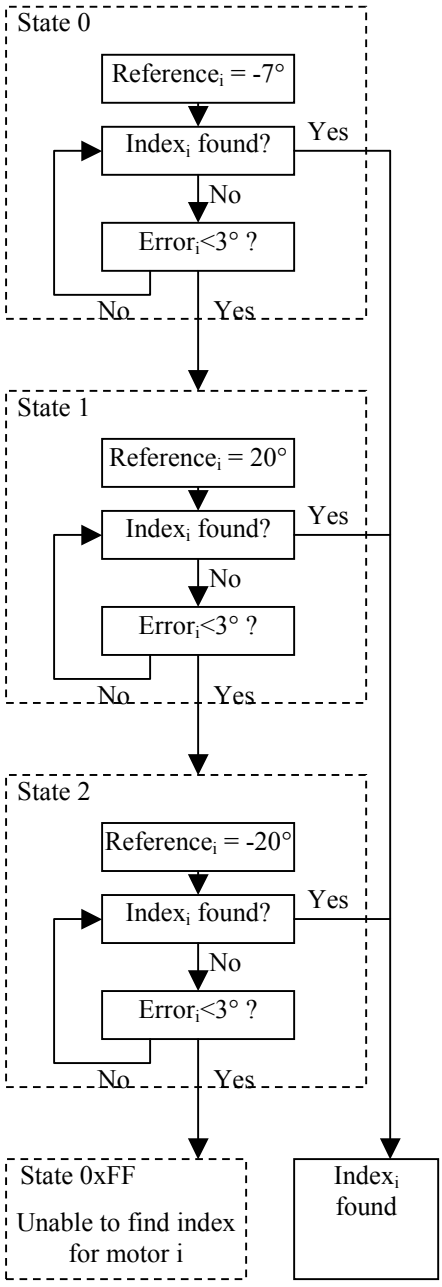


Figure 5.59 State machine used to find the index of the steering motor encoders.

Steering motor controllers

We have implemented four individual controllers for the four wheels. This is a simplification that can cause problems if the reference changes in large steps. The reference for each motor will always be calculated using the Ackerman formula but during the transient response, when going from one steering position to another, the wheels do not follow the Ackerman formula. This is both because the needed angular change in the reference is different for each wheel and because the disturbances are different for each wheel.
As the vehicle is mainly intended to be used on loose ground it is not a big problem if the wheel alignment is not always perfect, but a solution could be to ensure that the input to the Ackerman calculations is a ramp input and not a step input.

P controller

A P controller is not very good in this situation because the dynamics vary too much. If we adjust the gain to avoid too much overshoot under condition with low friction to the ground then it is no able to move the wheel under conditions with high friction.
Especially during the initialisation, when searching for the encoder index, the friction will be high as the vehicle stands still.

PI controller

Adding an integrator to the controller solves the above problems. The proportional part will still ensure a fast response while the integrator part will remove the steady-state error.

MIMO controller

As mentioned above there are problems when using four totally individual controllers as the position of the wheels should always follow the Ackerman equation. Modelling a MIMO controller that also tries to minimize the deviation from the steering formulas could solve this.

Initialising the steering motors

Before the controller can run properly it needs to know the absolute angular position of the wheels i.e. it needs to search for the index marking on the encoders.
The initialisation process is a state machine for each wheel module as shown on Figure 5.59. It will search for the index marking in an area of ±20° from the initial position. Please find the "Initialising – finding the wheel positions" section page 90 to get more details.
During the initialisation-process, the controller reads the angle from the Encoders.dll using a special function *InitializingSteering* that calculates angles relative to the initial position instead of to the index marking.
If the state machine reaches state 0xFF for wheel$_i$ then it was not possible to find the index marking automatically. The user must then turn the wheel "manually" using the test program for the MotorOut.dll module. This test program allows the user to directly adjust the control signal sent to the motor amplifiers.

NOTICE: The wheel must be turned using motor power. If you try to turn it directly by hand, the nylon gear in the steering motor will be damaged!

If the friction to the ground is very high for example if the wheels are sinking down into the ground then the motors might not be able to turn the wheels while standing still. Currently the controller cannot detect if this happens and it just waits for the wheels to reach the search positions which will never happen in this situation.
There should be a recovery feature implemented that could take care of this problem. If a wheel is not responding properly then the vehicle should begin to drive (slowly) forward and backwards in small steps to generate enough slippage for the wheels to turn.

Detecting external errors

For each sample the motor controller will read all status bits from the external error system. If an error is detected the controller will be disabled immediately and all control values will be set to 0. It will also send a message to the main application to inform the user that errors were detected.
When all external errors have been cleared the controller will wait 1.5 second before enabling the controller. If new errors are set during this time it will not enable the controller.
This feature was made because of the Curtis power amplifiers. They have a status signal that is not just OK or NOT OK. Instead they blink a code, telling what kind of error they have detected. It can therefore happen that all four controllers send OK signals to the error system even though it was only a part of a blinking procedure.
We do not want the controller to accidentally start and therefore we wait until we are sure that the OK signal is not just a blink.

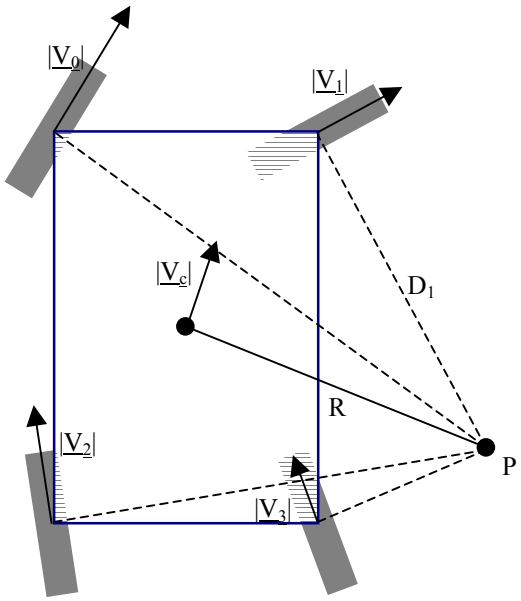


Figure 5.60 Wheel speeds when turning about P

Ackerman calculations

The result of the high level control of the vehicle is a vehicle speed and a rotation point around which the vehicle should turn. From these values we can calculate the wheel speed and angle for each wheel module using the Ackerman formula.

Wheel speeds

The wanted speed at the position of each wheel is:

$$|V_i| = \frac{D_i}{R}|V_c|$$

Formula 5.28

Where
V_i : is the speed at wheel i [m/s]
D_i : is the distance from the centre of wheel i to the turn point P [m]
R : is the rotation radius, the distance from the centre of the vehicle to P [m]
V_c : is the wanted speed of the vehicle centre [m/s]

From this we can calculate the rotation speed of the wheel:

$$\omega_i = \frac{60\frac{\text{sec}}{\text{min}}}{O_i}|V_i| \quad \text{[rev/min]}$$

Formula 5.29

Where
O_i : is the circumference of wheel i

V_i : is the speed at wheel i as found in Formula 5.28

Wheel angles

As shown on Figure 5.61 the angular position of each wheel is:

$$\varphi_i = \text{Atan}\,2(-(P_{t,x} - P_{0,x}), P_{t,y} - P_{0,y})$$

If the turning point is to the left of the vehicle then the signs are different:

$$\varphi_i = \text{Atan}\,2(P_{t,x} - P_{0,x}, -(P_{t,y} - P_{0,y}))$$

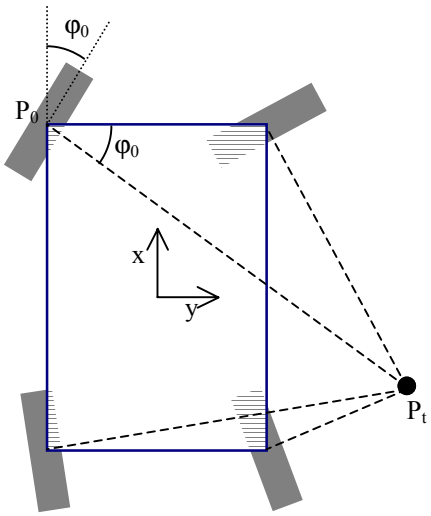


Figure 5.61 Wheel angles when turning about Pt

Steering the vehicle

The steering of the vehicle happens by specifying a point in the x,y plane around which the vehicle should turn. The point can be anywhere in the plane - also at the position of one of the wheels. In that case there is an infinite number of solution for that wheel using the Ackerman formula, but the function will choose one of them.
To make the vehicle go straight forward, the Y coordinate should be set to infinite to follow the theory, however setting a high value is enough. It don't even have to be very high - setting it to 1000 m would yield an angle reference of 0.03° i.e. below what can be measured by the encoder.

Closing down

As mentioned above in "Initialising the steering motors" the wheels must be in a position less than 20° from the index marking when starting the controller. If this is not the case then the vehicle cannot run as it doesn't know the actual direction of the wheels. Therefore it is important that the wheels are positioned in their forward direction before closing the controller down. This is handled by the *ReadyToClosedown* function in this DLL. Please see appendix 5.2 for further details.

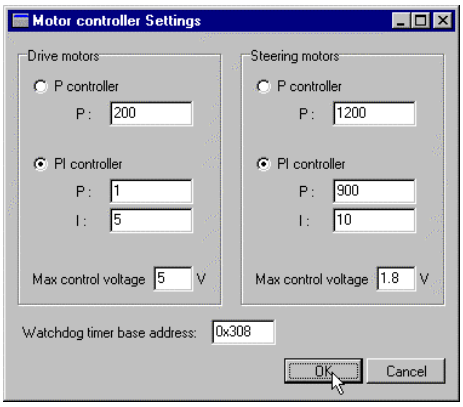


Figure 5.62 The motor controller settings window

User interface

When calling the "Showwindow" function in the DLL a window will show up (Figure 5.62).
Drive motors and steering motors : Various parameters can be set for the two controller areas:

- Specify if it should be a P controller or a PI controller.
- Specify the gain of the P and I parts.
- Specify a maximum control voltage. In normal operation this is set to the maximum possible i.e. 5 V for the Honda motors and 1.8V for the steering motors.

Watchdog timer : Set the base address of the Watchdog timer on the ISA bus.

Test program

A program has been made to test the Motor controller module. It loads and initialises the MotorCTRL.dll and also Encoders.dll that is needed to run the controller.
The test program initialises the steering motors (search for the index) when started. After that, the user can use the sliders to set the angle

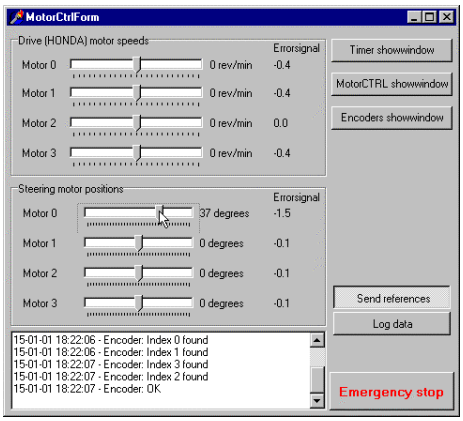


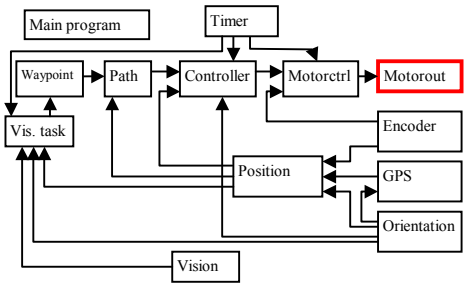
Figure 5.63 The test program for the motor controller module

references for each of the 4 steering motors and the speed reference for each of the Honda motors.
To the right of each slider, the controller error is shown (= reference - measurement). This enables the user to follow what is actually going on in the controller. The test program gets the error signals by calling the *GetErrorsignals* function in the MotorCTRL.dll as described in the ”Programmers interface” in appendix 5.2.
At the bottom left of the window the message log window will show all messages sent from the used modules.
By pressing the buttons to the right, the user can call the showwindow function in the Timer module to change the control rate, the Motor controller module to change controller parameters and the Encoder module to change the parameters in the encoder calculations.
Press the emergency stop button to stop the controller. The button calls the *SetEmergencystop* function in the MotorCTRL.dll. See a description in the ”Programmers interface”, appendix 5.2.

Status and error messages

The motor controller module can send the following messages:

“… DLL not loaded - Motorcontroller disabled” If the input (Encoders.dll) and output (Motorout.dll) DLL’s are inaccessible then the controller is disabled.
“External error: …” The external error system has reported one or more errors. The controller will be disabled until all error has been cleared and the reactivation button has been pressed on the error system print card.
“Index not found at steering motor x …” The controller has searched for the index by turning the steering motors but unsuccessful. The wheel have to be turned “manually” to the base position for the controller to find the index.
“D/A error: … “ An error has occurred in the initialisation of or communication with the D/A converter PCI card.



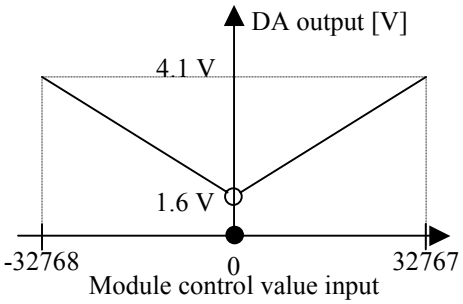
5.13. The Motor out module

The D/A converter (DAC) that controls the motor speeds contains 8 channels in the same card. The resolution is 16 bit covering a voltage range of –10 V to 10 V.
A driver DLL was included with the DAC. We had the options to:
a) Write a Motor DLL to interface our system with the D/A driver. This would however, just be a waste of system resources as the new DLL would not contain any important functionality.
b) Write a Motor DLL interfacing our system directly with the D/A card, i.e. rewrite the functionality that is contained in the DAC driver. Our tight schedule does however not offer time to reinvent the PCI card communication.
c) Add extra functionality to the interface file beyond just mapping functions to the DLL (the interface file is included in the program that uses the DLL and gives easy access to the functions in the DLL).

We have chosen option c. This is the easiest and most efficient solution. The drawbacks are however that:

- we cannot implement a Showwindow function like the one we have included in all other modules (see appendix 5.2).
- The MotorOut module must not be included in more than one module at the same time.

The MotorOut.dll module must therefore only be handled by the MotorCTRL.dll module i.e. it takes care of all communication with and initialisation of the MotorOut module.



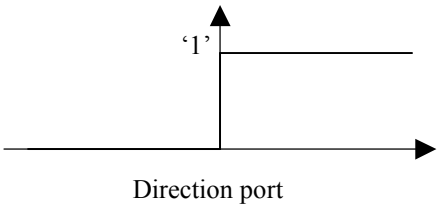


Figure 5.64 Conversion of control value to control voltage for Honda motors

Honda motor

The in-wheel Honda motors are used to drive the wheels. It contains gear, brush-less DC motor and a motor amplifier in one compact unit.

Electrical interface

The Honda motor speed input voltage is in the interval 0-5 V while the direction is controlled through another port. Due to a build-in threshold the effective speed control voltage interval is 1.6 V (Speed = 0%) to 4.1 V (Speed = 100%) i.e. an effective area of 2.5 V. The resolution in the effective voltage interval is then 13 bit.
We remove the Honda dead-band (threshold) because we want the control to be as linear as possible. The speed is given to the module as a signed 16bit integer (–32768..32767) that represents a linear speed from –100% to 100%. As shown on Figure 5.64 The absolute value is converted to a voltage for the Honda motor speed input port while the sign of the speed is send to a digital output port connected to the direction input port of the Honda motor. The motor threshold is removed so that speed 0 = 0 V and speed 1 = 1.60008 V.

Steering motor power amplifier

The steering motor amplifier converts the control signal to a power current driving the steering motors. It is a Curtis controller bought from the electrical wheelchair manufacturer UBlet.

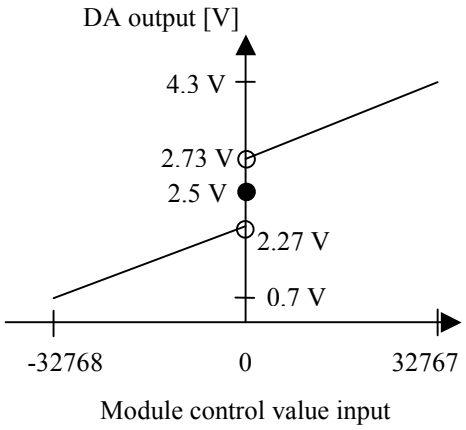


Figure 5.65 Conversion of control value to control voltage for steering motors

Electrical interface

The Curtis power amplifier can only handle positive speed control voltages in the range [0;5] V. It can be used in the same way as the Honda motors with the direction and speed input on separate ports. We have however programmed the controller to include the direction in the 0 to 5 V range i.e. 2.5 V ± 2.1 V where a voltage below 2.5 V is backward direction. We have chosen this solution because it is easy to control. This method does however introduce a possible initialisation hazard, as the control voltage will be 0V (= max speed backward) when the computer starts. This hazard is prevented by two different error systems:

1) The robot error system will be in error state when the computer is turned on. It will remain in this state until the error-clearing button is pressed in the front of the vehicle. The amplifiers will only be activated when the error system reports OK. If the error is cleared before the controller is ready and has adjusted the speed signal to 2.5 V then the other error system will take care of this:
2) The amplifier includes its own error system (in fact it is a bit more sensitive than we like). We have enabled the option "High pedal disable" (as described in the "Programming of the power amplifier" section below), which makes the amplifier require that the speed signal is adjusted to zero speed i.e. 2.5 V before it is activated.
   In addition there is a dead-band at the voltage edges, which means that the controller will always stop if the speed control voltages is below 0.4V (or above 4.6V)

The direction input port is not used but to get maximum flexibility for future changes we wired it anyway. The direction port on the amplifier has been connected with a wire in the bundle and is therefore accessible in the computer box.

The amplifier dead band has been found experimentally to be 0.23V. This includes the power needed to overcome the stiction in the motor and gears i.e. the motor runs forward when the amplifier receives a speed voltage of 2.5+0.23 = 2.73 V. The internal safety functions in the amplifier stop the motor when the voltage goes above 4.3 V so the effective work area is 1.57 V.

We remove the amplifier dead-band because we want the control to be as linear as possible. The speed is given to the module as a signed 16bit integer (–32768..32767) that represents a linear speed from –100% to 100% as shown on Figure 5.65.

Programming of the power amplifier

The power amplifier has a build-in microprocessor that allows a lot of advanced features to be set directly in the controller. This does however require a programming unit, which we borrowed from UBlet. Many of the features in the amplifier are made for user-friendly wheelchair use, but as we want our own computer to take care of the advanced functions we have programmed the unit to be as linear and non-automatic as possible. The most important parameters are described in appendix 5.1.

It is for example possible to set the acceleration rate – but not as we want it: the amplifier will always give a ramp output even when it receives a step input!

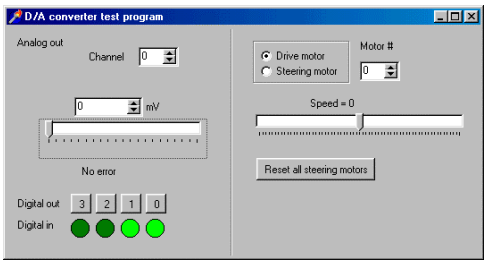


Figure 5.66 Test program made to test the Motorout module and DA converter

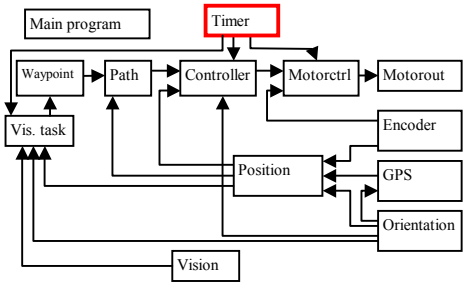
Test program

A program has been made to test the DA converter through the MotorOut.dll module and the extra functionality in the interface file. As shown on Figure 5.66 it is divided into two sections:

- The left-hand section is used for direct access to the DA converter. Choose which one of the 8 channels to modify (0..7) and adjust the voltage either by the slide for quick changes or type a value in the "up-down textbox".
  The left section also offers direct access to the digital input and output ports of the DA converter. They are however not used any longer by our system.
- The right-hand section uses the functions provided by the "programmers interface" (see appendix 5.2) with control signals in the range –32767 to 32767. To change the speed of a motor, choose the motor type (drive or steering), choose a motor number (0..3) and adjust the speed with the slider.
  All steering motors must be set to speed=0 before the external error system can enter its OK state. This is done quickly with the "Reset all steering motors" button.

Error and status messages

When calling functions in the Motorout.dll module (the driver included with the DAC) a return value indicates the result of the call. In the interface file we map these status numbers to a description text and post it to the main application just like all the other modules. There are 18 different messages and they all begin with "**D/A error: …**". The description that comes after, is self-explaining. The most important are:
"**Unknown card type**", "**Function not supported**", "**Open driver failed**", "**Unable to allocate memory**" and "**DA voltage out of range**".



5.14. Timer module

The Timer module handles the timing for all modules. The user can configure the timer module to call other modules at a rate up to 1 kHz (1 ms/sample).

Standard Windows timers

The standard Windows timers (API functions SetTimer and KillTimer) are far from precise enough. Though you can specify an interval down to 1 ms, the actual precision is only about 55 ms. This is one of the reasons why Windows has got a reputation of being extremely imprecise.

Multimedia timers

Instead, we use the multimedia timing functions offered by the MMsystem.dll included in Windows. The multimedia timers encapsulate one of the Intel 8254 counters/timers on the Motherboard. The counter chip generates a hardware interrupt every time the specified time has passed. The interrupt has priority over almost all other Windows activities. It is therefore very precise, not only in the interrupt generation but also in the execution of the code at each interrupt.
Microsoft warns strongly against putting processor-consuming code in the multimedia timer callback function and it is obvious that due to the high priority it can totally lock up the operating system.
We do put more code in the callback function than recommended, but we do not mind Windows being slow as long as our code is completed in time.
The timer is started with the following code:

```
1) IF LastTimeID<>0 then StopTimer;
2) TimeGetDevCaps(@TimerCap, Sizeof(TimerCap));
   IF TimerCap.wPeriodMin<=TimerRes then begin
     …
3)   TimeBeginPeriod(TimerRes);
4)   LastTimeID := TimeSetEvent(TimerRes,…,@CallBack,…,TIME_PERIODIC);
   end else begin
5)   SetError('Requested resolution not supported');
   end;
```

1) If the timer is already started then it must be stopped.
2) The capabilities of the timer (MM function *timegetdevcaps*) is compared with the base resolution requested by the user. If the requested resolution is not possible an error will be set 5).
3) Calling *TimeBeginPeriod* programs the hardware timer to generate an interrupt at the specified interval (ms).
4) The timer event is set by calling *TimeSetEvent.* It is specified that the timer should be periodic and thus continue to call the Callback function every time the 'TimerRes' period has passed.

User interface

When calling the "ShowWindow" function in the DLL a window will show up with the following elements:
Base interval: Set the resolution of the timer. The range is 1 ms to 500 ms.

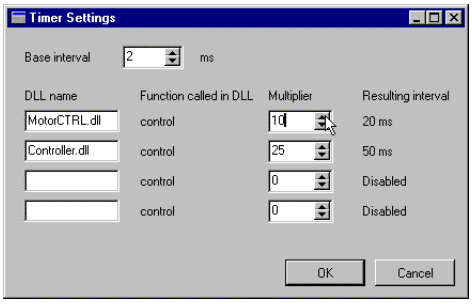


Figure 5.67 The timer settings window

DLL name: type the name of the DLL in which a function should be called at the specified rate. If no path is specified it will search for it in the directory from which the application loaded.
Function called: The name of the function called in the specified DLL cannot be changed in the current edition. The DLL must contain a function called (or exported as) 'control'.
Multiplier: Specify how many times the base interval event should happen between the specified DLL is called. The resulting resolution for this DLL is Multiplier*Base interval.

Test of module

The timer module was tested with the Vision.dll module. A small test DLL was made that only contained a "control" function that would call the sampling function in Vision.dll

Status and error messages

The timer module can send the following messages:

"Timer faster than computer" If the functions called in the DLL's have not completed before the next call by the timer then this message will be sent and the timer will be disabled. The solution can be to lower the timer rate, execute less code in the timer call or to use a faster computer.
"Unable to load DLL: …" If it could not find the DLL you want to be called by the timer.
"Unable to locate function "control" in DLL: … " If the timer could not find a function called control in the specified DLL.
"Requested resolution not supported" If the specified Base interval is smaller than supported by the computer (this should not happen on today's computers).

6. Test results

6.1. Low level controller

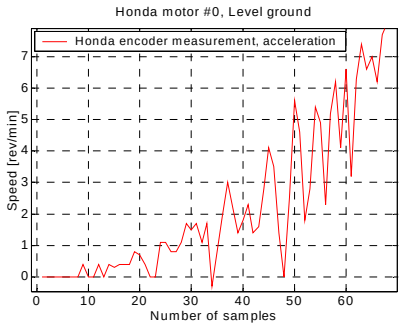


Figure 6.1 Encoder signal.

Drive motors

Encoder signals

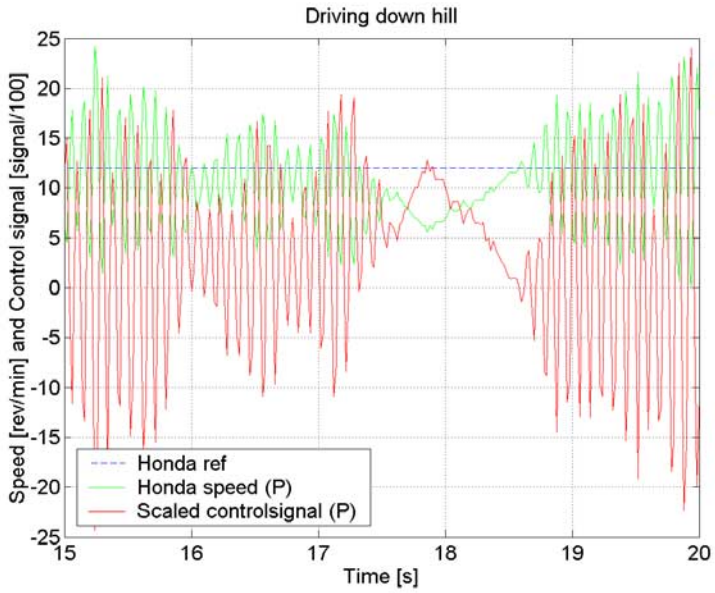
The errors on the speed signal derived from the encoder count get relatively big at low speeds. On page 90 in "Encoders module" it has been calculated that each counter value change will contribute with 0.37 rev/min to the speed. This can be clearly seen on Figure 6.1. With large P gains this could have a negative influence on the stability at low speeds and it would be a good idea to filter the signal.

Dynamics

Backlash

From our first test set-up of the Honda motor we discovered that a P controller did not work well. There is backlash in the gear between the motor shaft and the output shaft. As the encoder is placed on the output shaft there will be a delay from the motor reverses its rotation direction until the encoder and thus the controller register it. This delay caused the P controller to become unstable even at very low gains. The maximum stable gain we could use under normal conditions was K_p=200 (0.6% of maximum power). With a speed error of 40 rev/s (1 m/s) the control signal will be 25% of maximum power. Therefore, there is a huge steady-state error. Giving a reference at 1.5 m/s the actual speed would be 0.1 – 0.3 m/s.

Figure 6.2 Driving down hill with a P controller (Kp=200).



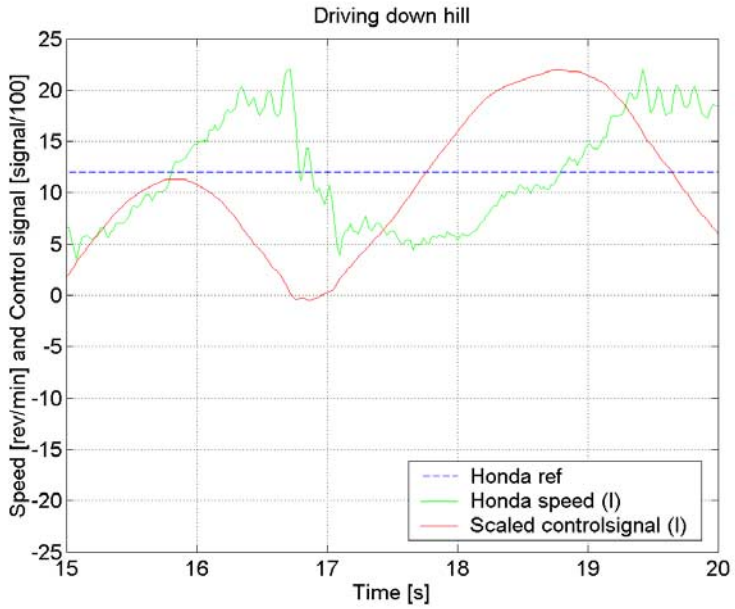
Driving down hill

We decided to initially use the P controller and accept the steady-state error. It worked acceptable until we tried to drive downhill. It became

unstable again as shown on Figure 6.2 with an oscillating frequency of about 11 Hz, which comes from the delay caused by the backlash.

Therefore, we tried the same hill with a pure integrator which works as a low-pass filter. The results are shown on Figure 6.3. It is clear that the control signal has been damped and that the driving is much more smooth. It is still oscillating, but this time it is due to the phase shift introduced by the integrator. We decided that this was acceptable because we wanted to move on and prepare for the GPS tests.

Figure 6.3 Driving downhill with only an integrator (KI=5)



To use pure integration on the drive motors does however introduce other practical problems as described on page 130 in the "Motor controller module".

Step response

On Figure 6.6 a step response for the Honda motors is shown. The controller is an I controller with $K_I = 5.0$. The speed is oscillating a bit, but it can be seen that the stationary error is about zero and that the rise time is quite long. This is characteristic for an I controller.

Figure 6.4 Step response on lawn.

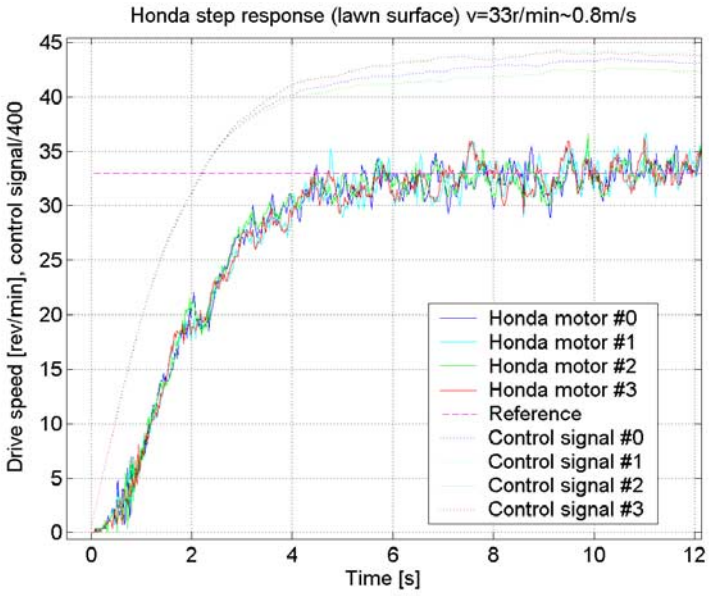


Figure 6.5 Deep lug tractor tires.

Oscillating speed

On Figure 6.3 and Figure 6.4 it can be seen that the speed of the wheels are oscillating. There can be several reasons why the speed of a wheel can oscillate:

1) The four wheels a mechanical coupled and if one wheel is pushing then the other wheels will rotate too. Therefore, the rotation speed recorded for one wheel can be a result of what happens with the other wheels.

2) We use deep lug tractor tires (see Figure 6.5). There are 12 lugs on each side. The 24 lugs will cause the rolling resistance to oscillate.

3) The connection from the rim to the ground is through an air-filled rubber tire. This will work as a spring/damper system

Steering motors

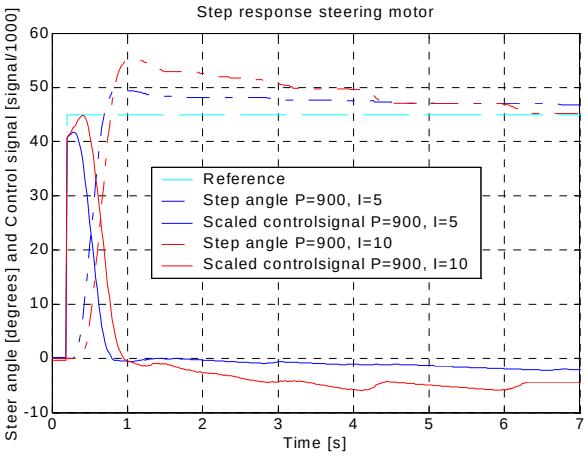
Dynamics

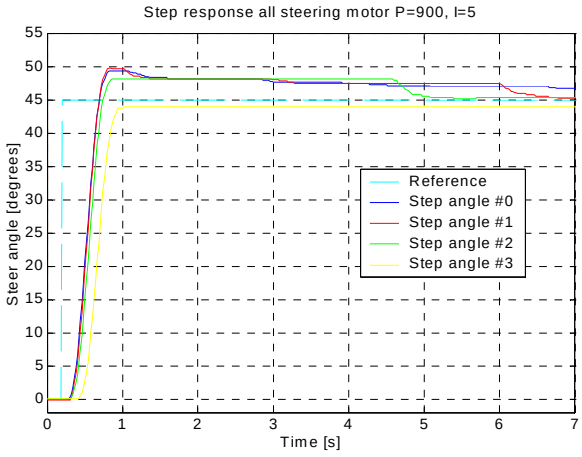
We have tuned the steering motor controllers by hand to find a P gain that gave a good response. The best solution would have been to build a model and simulate the response to find an optimum controller; that would however take a few days. The second best solution would have been to use the Ziegler-Nichols handtuning rules for closed loop systems; that would have taken a few minutes. The first outdoor tests were made only 3 weeks before our dead-line and therefore we were so much in a hurry to get results with GPS that we forgot all about Ziegler-Nichols and thought building a model would be too time consuming.

Figure 6.6 Step responses with the steering motors.

Step response

To investigate the dynamics of the steering motors with our hand-





tuned controllers, we made some step responses. It is a “worst-case” step at 45° while the vehicle stand still on a painted concrete surface. The results are shown on Figure 6.6.

There is a delay from the step input until the wheels begin to move. This is probably a combination of several factors:

1) The power amplifier cannot give a step output even when it receives a step input. This is because they are designed for smooth driving with electric wheel chairs. We have programmed them to give the steepest ramp possible output, which is 0.2 s to go from 0 to maximum output (read about amplifier programming in appendix 5.1).

2) Coulomb friction

There is an overshoot and it takes a long time for the integrator to remove the steady state error. The steady state error is reduced in steps because of the coulomb friction. An obvious thought would be to increase the integrator gain.

On the left plot of Figure 6.6 it can be seen that the overshoot is increased when increasing the integrator gain and therefore this is not a solution alone.
We can therefore conclude that the controller is not good enough to handle steps of this size under these conditions. It takes too long time before the wheel alignment is correct.
It does however work well under normal conditions and therefore we have used the settings (K_p=900, K_I=5) in all other tests.

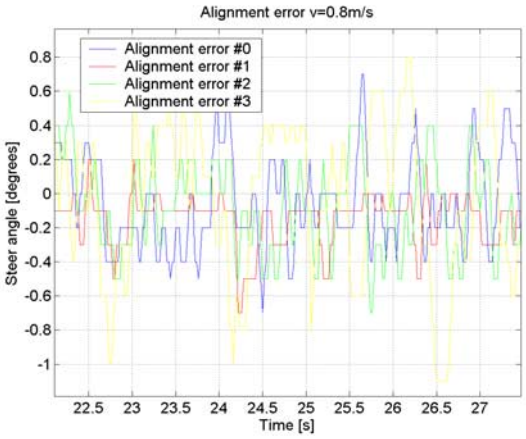


Figure 6.7 Errors due to ground irregularities.

Wheel alignment

To test the variations of the wheel alignment, we have tested the vehicle driving with a fixed steering angle reference on 0° at a speed v = 33 r/min □ 0.81 m/s. The results for each wheel module are listed in the table below and in Figure 6.7. It can be seen that the mean values are very close to zero and the standard deviation is acceptable. The test was carried out on a fairly level lawn.

Wheel module No.	Mean [°]	Std [°]
#0	0.023	0.32
#1	-0.032	0.24
#2	-0.016	0.32
#3	0.011	0.45

The resolution of the encoders is 0.18°/counter-value. It can be seen on Figure 6.7 that the signal is cut in steps of 0.1°. This is because the value is rounded to one decimal when writing it to the log file.

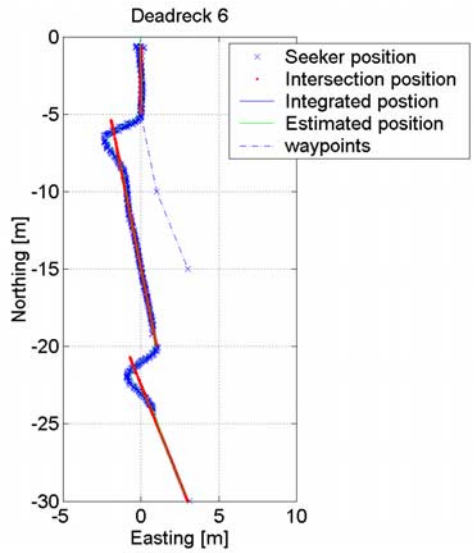


Figure 6.8 Sign error in the controller.

6.2. High level controller

Identification of software errors

The possibility of logging all control data is very efficient for discovering software errors. The plot on Figure 6.1 helped us to identify a sign error in the software. The path it follows (red dots) is not the same as the waypoints. The vehicle is driving based on encoder and compass data (dead reckoning). The positions do not show exactly where the vehicle has driven because of errors accumulating.

The position integrator

The position integrator is continuously using the distance data from the drive motor encoders, the angle data from the steering motor encoders and the compass data to calculate the position of the vehicle. The calculated position is relative to the position where the integrator (summation) was last reset.
To see how well the position integrator is working, the data can be plotted together with the GPS positions, see Figure 6.9.

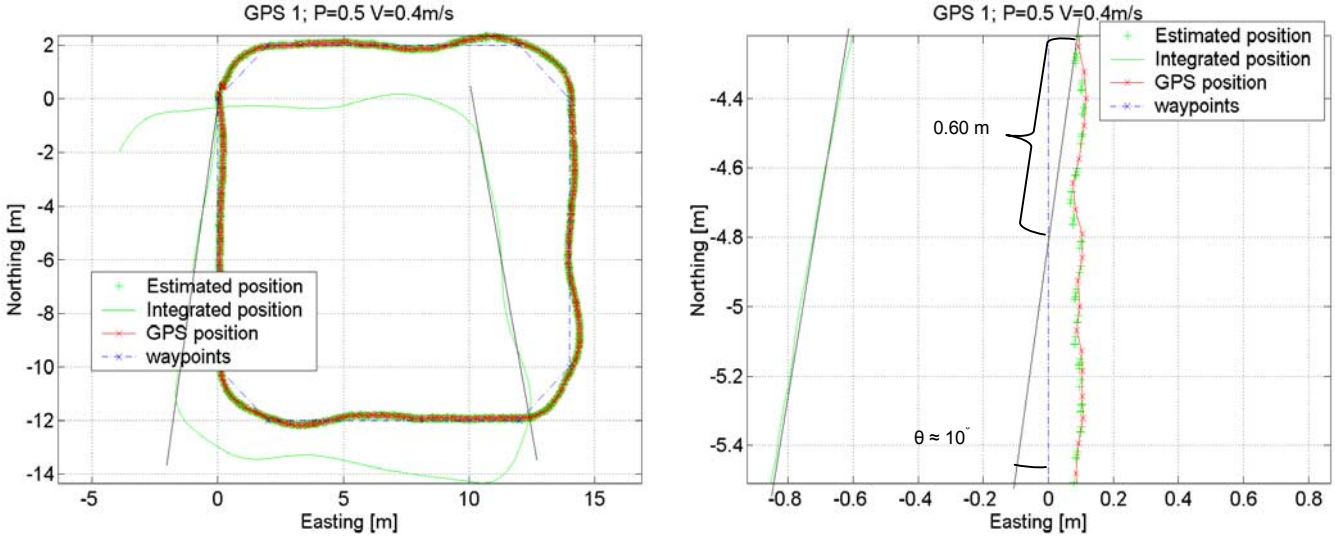


Figure 6.9 Integrated position and GPS position. (The right plot is a section enlargement of the left.)

The vehicle follows the square using the GPS but the green line shows where the integrator thinks it is driving. As it can be seen on the plots above the integrated position is quite different from the GPS positions. This is mainly due to deviation on the compass; the deviation driving north and south can be seen directly on the plot as the angle between the GPS path and the integrated position path on the left figure.

On the right figure, it can be seen how the direction of the estimated positions is the same as the direction of the integrated track. This is because it uses the integrated position to calculate its position between the GPS updates.
The odometer data from the wheels will also add an error to the integrated position. Without an absolute reference small errors accumulates and due to the slippage, this can happen very fast for off

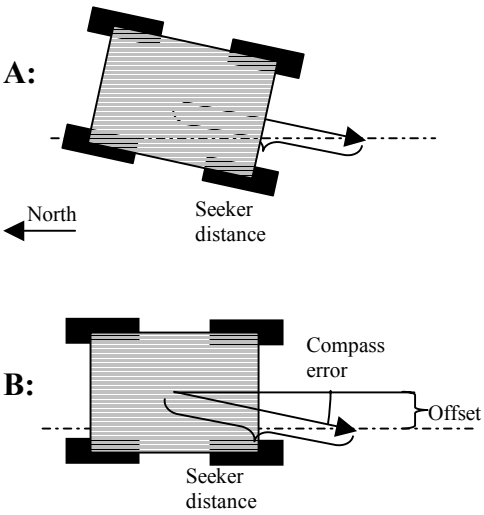


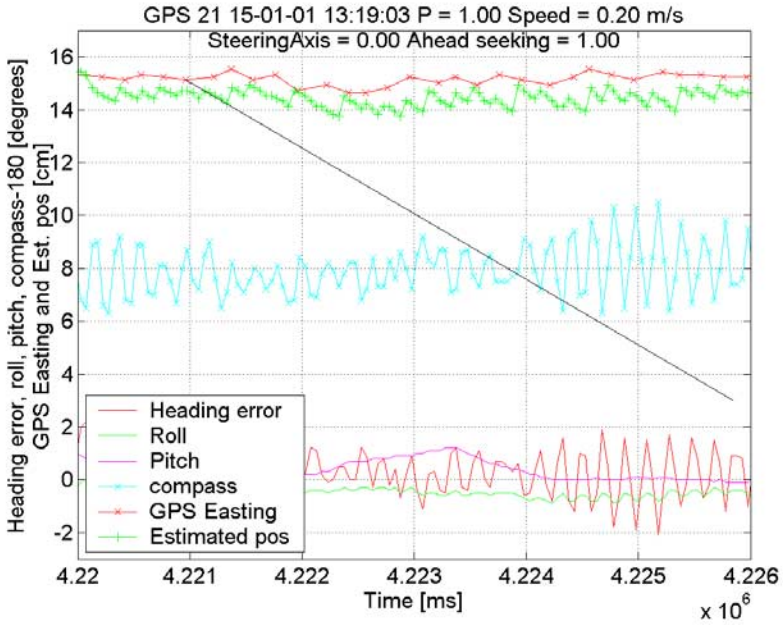
Figure 6.10 Error on the compass causes a position offset.

Figure 6.11 Position and compass data plotted versus time.

road vehicles. The odometer error is however small compared to the error caused by the compass deviation.

GPS (RTK)

The deviation has also influence on the actual path followed by the vehicle, when using GPS positions. It can be seen that the vehicle is driving with an offset on about 10 cm from the waypoint path. On Figure 6.9 the vehicle is using a point (seeker position) 0.6 m in front of the vehicle origin to find the error to the path specified by the waypoints. Because of the error on the compass, it thinks that it is now driving towards the path to remove the offset (Figure 6.10;A); actually, it is driving approximately 10° more towards east (Figure 6.10;B). The controller therefore calculates that the heading error (direction to path - compass) is about 0°, which makes the vehicle go straight forward (see Figure 6.11).
On Figure 6.11, the easting coordinates are plotted versus time while driving south. It can be seen clearly how the estimated positions are estimated in the wrong direction and corrected every time a new GPS position is available (see the black line).

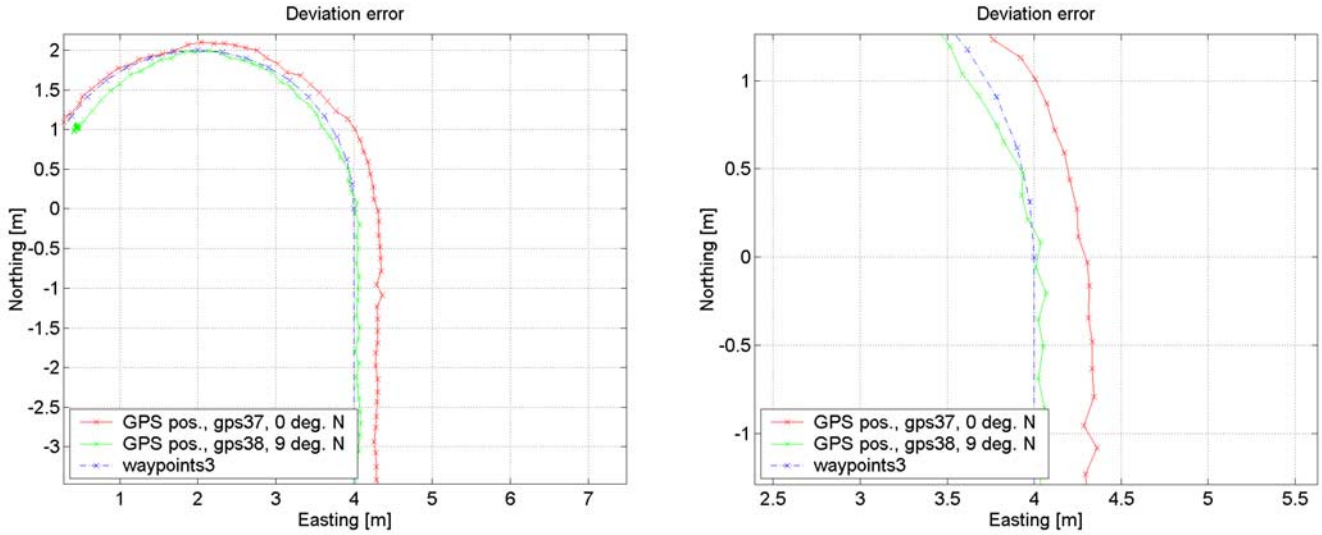


It is seen that the compass has a mean error on about 8° and that the readings are oscillating at 5 Hz, up to 2° to each side. As it is seen, the oscillating compass is directly transferred to the heading error.
On page 101 in "Orientation module" it is described that the compass calculation is very sensitive to tilt errors. The compass oscillation in Figure 6.11 is probably due to physical oscillations measured by the tilt sensors. The green plot of the Roll angle does actually oscillate at the same frequency. Unfortunately, the figure does not show the raw pitch and roll data because they are damped by the orientation software module. The actual angles will oscillate with much larger amplitude. The compass calculates its heading based on the raw undamped tilt data.

The compass error is not eliminated just by adding a constant to the compass reading, because the error is different at different headings.

Figure 6.12 The effect of the compass deviation on the position offset. (The right plot is a section enlargement of the left.)

However, when the vehicle is driving in a certain direction the magnetic field is nearly constant. We had found that the mean error on the compass, when driving north, was about 9°. Therefore, we tried to add 9° to the compass reading and see how this would affect the position offset.



As it can be seen on Figure 6.12, the main reason for the position offset must be due to the deviation on the compass. Without correction, the mean offset is now about 30 cm and with the 9° correction, the mean offset is about 4 cm. That the offset now is about 30 cm is because the seeker position has been adjusted to 1.5 m, from this the deviation angle can be calculated to about 11.5°, see Figure 6.12. We have not tried to test or find a better correction, because it would be better to eliminate the deviation.

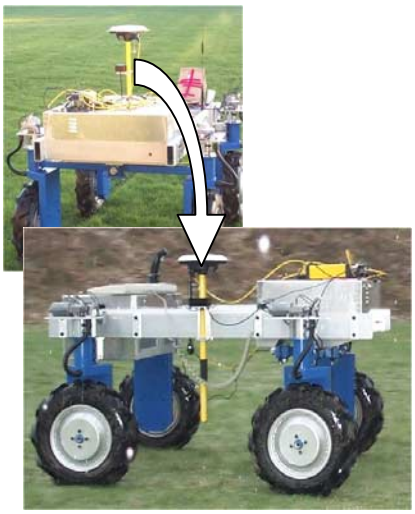


Figure 6.13 New compass and GPS antenna position.

Relocation and calibration of the compass

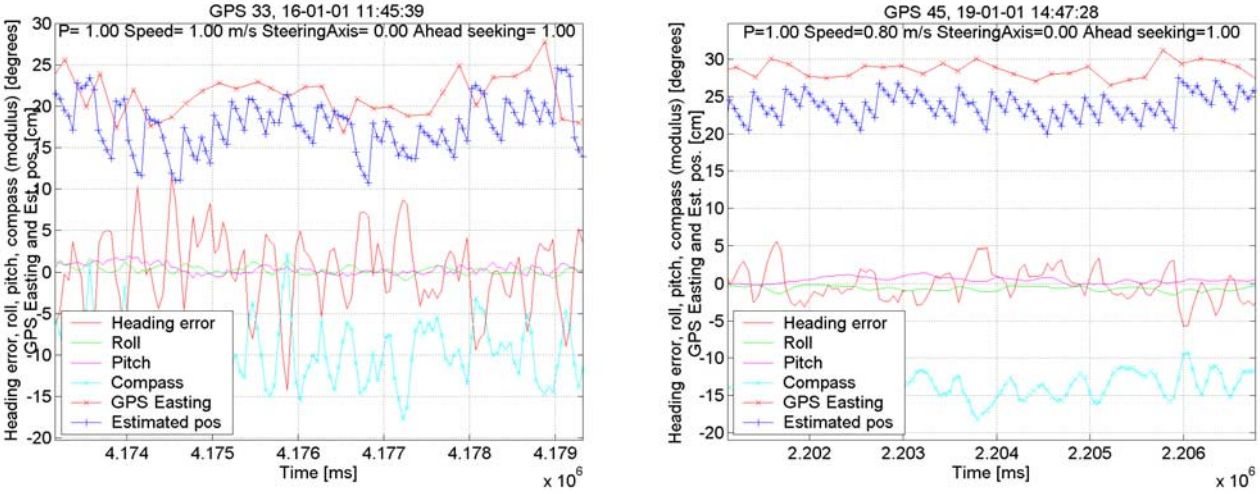
With a needle compass, it was found that the steering motor amplifiers in the left side of the vehicle were disturbing the vehicle compass. The compass was therefore moved to the middle of the right side member, see Figure 6.13.

The compass was then again calibrated for hard-iron distortion fields and then a series of tests were made.

The test shows that the error on the compass is even bigger after the relocation and calibration. On Figure 6.14 it can be seen that the error on the compass has increased from about 10° to about 13° - 14°, which also leads to a bigger offset.

Unfortunately, we do not have time to find a better position for the compass and do more tests. The compass should probably just be raised higher to get away from the distortion fields. This can be tested by temporarily mounting the compass to the vehicle and watch the X, Y and Z magnetometer outputs [TCM2].

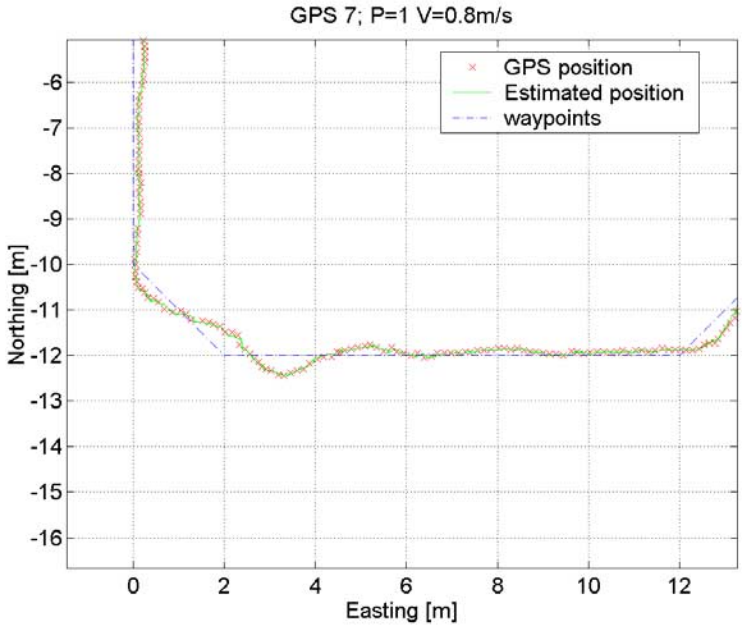
Figure 6.14 The left plot is made with the GPS antenna in the centre and the right with GPS antenna on the side member



Timing

After some changes in the software, we could not get the vehicle to become stable at speeds higher than 0.2 m/s. We had shown that the vehicle could be stable at speeds up to 0.8 m/s, see Figure 6.15.

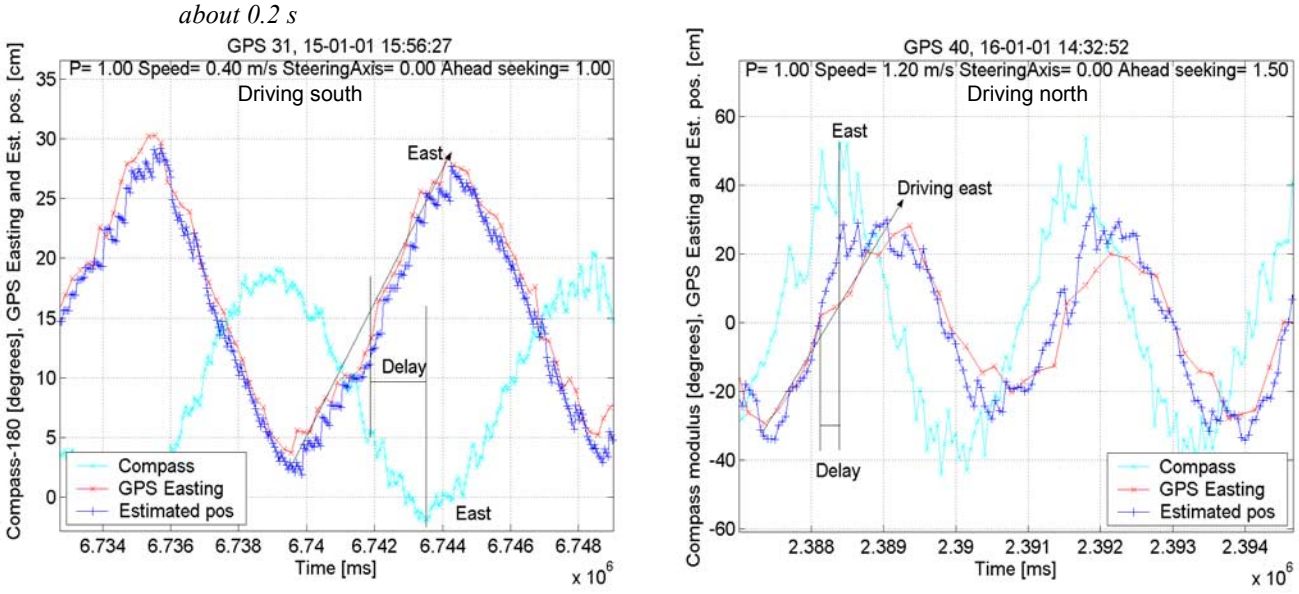
Figure 6.15 The vehicle is stable at v = 0.8 m/s.



It was likely that the problems were related to the different delays in the system and these were looked at without success. However, a plot of the vehicle oscillating from side to side showed that the compass data was delayed, see Figure 6.16; left.
In the middle between two turns, the maximum compass reading should occur, but it is delayed approximately 1.5 s. This negative phase shift was introduced by an error, which made the controller use the oldest available orientation matrix from the orientation module. On Figure 6.16; right a normal situation can be seen, the delay is

approximately 0.25 s due to the damping of the compass with a factor 1, see “Damping data” on page 103 in the Orientation module. This damping function was introduced in the mean time. The right plot is damped and the left is not. The right compass reading oscillates more because of the much higher speed.

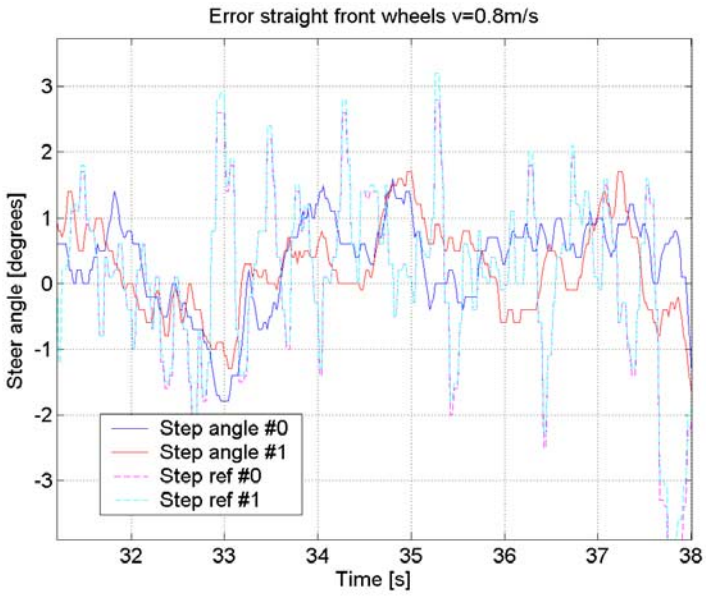
Figure 6.16 Left plot; delay because of software error. Right plot; normal view with delay on about 0.2 s



Damping of the compass

The compass is used in the calculation of the vehicle heading error to the path. As it can be seen on several of the previous plots the compass oscillates, mainly because it use undamped tilt data to calculate its heading. The controller heading error and the wanted speed are giving the references to the drive and steer motor controllers. Therefore it is unwanted that the heading error is oscillating because this increases the power used for steering, see Figure 6.17.

Figure 6.17 The oscillating heading errors influence on the steering of the front wheels.



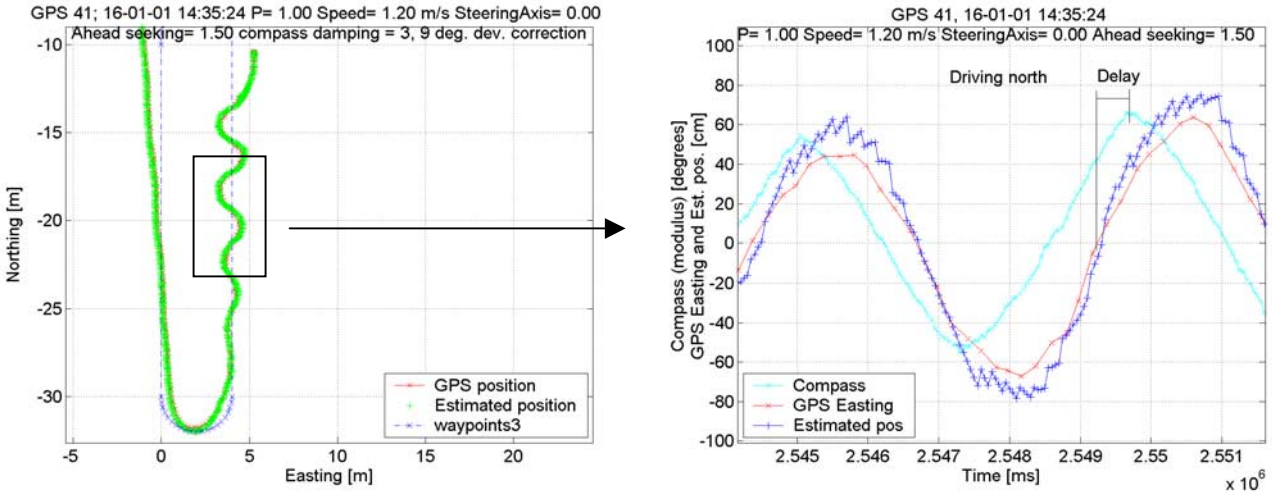


Figure 6.18 Test with a damping coefficient on 3.

Therefore, we introduced a damping of the compass. Tests have been made with damping coefficients from 0 – 3, see "Damping data" on page 103 in the Orientation module. On Figure 6.18 shows that the vehicle becomes unstable with a damping coefficient on 3 at v = 1.2 m/s, this can be compared with Figure 6.16; right where the vehicle is marginally stable at the same speed but with a damping coefficient on 1.

Three tests were also made at v = 0.8 m/s with damping coefficients 0, 1 and 2. The tests shows that the standard deviation is about 43% smaller with damping 2 than with no damping. The results can be seen on Figure 6.19 and in the table below, the standard error and mean position are calculated for the last 12 m before the last turn.

In the table the best results are first. If gps37, gps 45 and gps36 are compared it is clearly seen that the precision is improved if the oscillating compass data are damped.

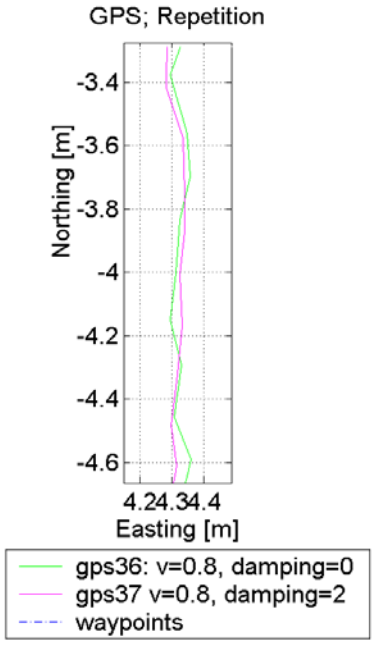


Figure 6.19 Test with different damping coefficients.

No.	Damp. coef.	V [m/s]	Mean [m]	Std. [m]
gps47	1	0.4	720506.565	0.02
gps37	2	0.8	720506.620	0.021
gps45	1	0.8	720506.588	0.024
gps36	0	0.8	720506.628	0.037
gps34	1	1.0	720506.514	0.293

It can also be seen that increasing speed decreases the precision of the vehicle.

Damping of pitch and roll data

On Figure 6.20 it can be seen how the centripetal force influence the sideways tilt measurement (Roll). The phase difference is due to the damping of the tilt data.

Figure 6.20 Effect of the centripetal force on roll.

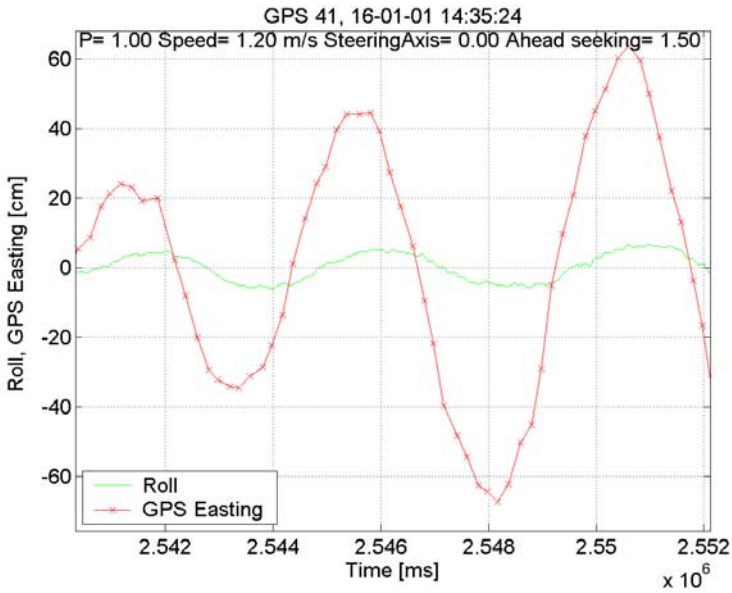


Figure 6.21 The camera detects "crop"

Vision

We have tested the vision system on the desktop by moving the camera over a line paper pieces. When the vehicle finally was ready for driving we focused on getting results with GPS as this equipment was borrowed. Therefore, the final vision test was pushed until it was last chance. The first vision test went reasonable well. It identified waypoints for the first paper pieces and the vehicle drove after them (see Figure 6.21). The compass was unfortunately very unstable because it was surrounded by iron and electronics (not from the vehicle) and it soon began to drive away from the line of paper (after about 0.7 m). We could see from the log files that the compass error varied from 10° to 60° during this short movement. The next day the vehicle monitor did not work and as this was 5 days before our deadline we decided not to waste time to find a solution for that. From the result we did get we are convinced that the vision system, maybe with a few adjustments, would have worked as we intended it to.

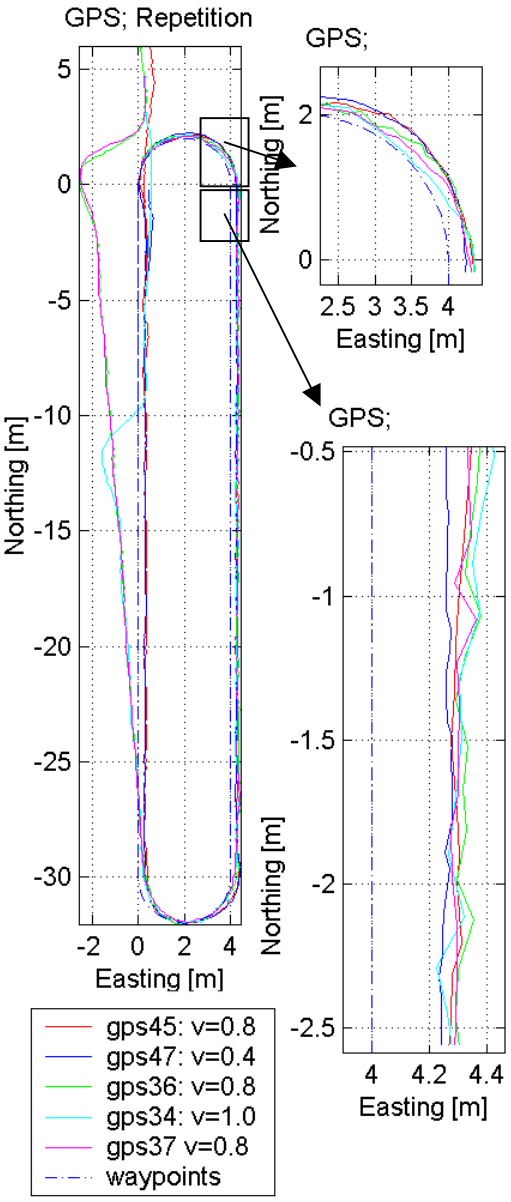


Figure 6.22

Repeatability

In the above sections, we have described several problems that influence the precision of the vehicle. As shown on Figure 6.22, changes to the parameters always give different position errors and offsets. What cannot be seen from the plots is that when the vehicle drives a route several times without changes to the parameters, it would go exactly in its own tracks (see picture below). We are convinced that the control of this vehicle can be improved in a way that gives the same good results under normal field conditions. Thus, the precision exceeds what would be required for an autonomous agricultural robot.

7. Recommendations

During the development and test of the vehicle, we have found several things that could or should be improved. Because of the constant time pressure to get the vehicle driving, we have had to downgrade some parts. In addition, during the tests we discovered things that should be improved. They are listed in the following sections.

7.1. Hardware

Hardware handling of errors

The external error system has caused us many problems. Though all control cables are shielded, there is no connection.
Redesign of the error system, see page 76 in “Hardware handling of errors”.

Possibility for attaching external equipment

The GPS receiver, the computer mouse and keyboard was constantly about to fall off when driving off road. We suggest a flexible rack to be mounted on the covers for the front and rear compartments of the vehicle.

Protection of electronics

For the vehicle to run in field conditions it must have IP54 protections. A few parts still need extra protection to reach this requirement:
The steering motors are not protected from rain and dust. The protection specification for the motor is IP40. They should be protected by a close-fitting cover. We have made a sketch of a suggested solution in appendix 3.2.31.
The steering motor couplings and encoders should be sealed by a cover. A suggestion can be found in appendix 3.2.31.
The covers for the front and rear compartments have holes in both sides for the wires to come in and out. These holes should be sealed to get a properly protection from dust and water (IP54)

Obstacle detection

Currently the vehicle will not detect a collision with humans or other objects. A mechanical sensor in the front of the vehicle should stop the vehicle to avoid damage to people and material.

Computer mounting

Because of the lack of suspension in the vehicle, the computer receives many shocks that will damage it. We experienced that the computer could suddenly restart when it received a shock.
The computer must therefore be mounted with a very flexible suspension to avoid these problems and to ensure a long life of the electronics.

Figure 7.1 4 mm 10A plug and panel socket.

Connecting the battery charger

The batteries must not be charged when the power is on. Now it is quite tiresome to connect the battery charger to the batteries and therefore it is recommended to make the following connection possibility.
Connect the batteries with e.g. two 4 mm 10A plugs (see Figure 7.1) to two appropriate panel sockets placed in the back plate. The negative socket should be connected to the negative pole on the batteries. The positive socket should be connected to the positive pole on the batteries through a relay. The relay should disconnect the charger when the vehicle main power switch is on.
This solution will make it safe and easy to charge the batteries.

7.2. Software

Diagnostics software

It can be quite difficult to discover if an encoder, motor or other electronics does not work. Therefore, we suggest special diagnostic software to be made. For each of the 8 motors it should test if the loop is closed. It should send a control voltage and verify that the encoder is moving and in the correct direction.
If a loop is not closed then it can be either the motor or the encoder that does not work. If it is a drive motor loop that does not work. We can find which of them that is not working by looking at the other encoder signals. If they give a signal, then the motor in the defect loop works. Therefore, the encoder must be out of order.

Orientation module

The compass heading should be calculated by the software module and not by the compass it self. The software should read the raw magnetic field measurements (X, Y, Z) and use the raw tilt data to calculate the compass heading. In this way it can reduce the error on the tilt data before using them. The tilt data errors are described in the "Accelerations" section of the Orientation module (page 101).

Path module

Currently the path module only makes a straight line from on waypoint to the next waypoint. It should find a smooth path between two waypoints that the vehicle can actually follow.

7.3. Control engineering

Model

The documentation on both the Honda motors, the Valeo motors and the Curtis amplifiers are not sufficient for making a good model. We recommend that test are made to identify the transfer functions for the different systems.

Now the mechanical and electrical system are specified and a model of the system should be made to develop, simulate and optimize

controllers for improving the performance and precision of the vehicle. In the design we have tried to avoid or minimize nonlinearities in the system, hence nonlinearities makes it very difficult to design an optimal controller.

Controller

The controller for the drive motors should include a feed forward connection with a proper gain. In this way the motor will get the correct control signal under normal conditions. Thus the PI controller only needs to take care of the errors when the conditions are varying from normal. This will give a faster, more stable and correct control signal for the motors.

We also recommend the implementation of a MIMO controller for both the drive motors and the steering motors. This could ensure that they follow the Ackerman equation more precisely. If one motor has problems following the reference then the other motors should wait for it so that the Ackerman equation can be fulfilled.

8. Conclusion

Figure 8.1 Designed for in-row driving with slim wheel modules and good ground clearance.

We have successfully built and tested an autonomous mobile robot, prepared for use with various chemical and mechanical tools.
With in-wheel drive motors, the wheel and transmission design is very slim, ideal for in-row driving in typical row distances in corn, maize and beet fields. The 500 mm ground clearance makes it usefull for driving in cornfields to the stage of earing.

Modular system

Mechanical

- A wheel module has been developed that is identical for all four wheel positions. This has simplified the production, as fewer different parts are needed.
- All connections to a wheel module are electrical which makes it very flexible to move. In combination with the flexible chassis it is easy to change the wheelbase or if necessary, the wheel modules can be mounted on another chassis.
- Even though the vehicle is very complex the modular mechanical and electrical design makes it fairly easy to work with electrics and software.
- With 4WD the off road capabilities are sufficient for working in the field.
- Using DC motors to steer each wheel (4WS) makes it possible to make very sharp turns and still fulfill the Ackerman equation. This would not have been possible with a rack-and-pinion steering.
- The implementation of the steering control in software gives high flexibility for research.

Figure 8.2 A DC motor is used to steer each wheel.

Software

- We have implemented as much as possible of the control system in the software. This makes the system very flexible and easy to modify.
- All the software is divided into 12 modules that have been compiled into separate DLL's. This makes it easy to make changes to one module without changing other modules.
- The modular software system makes it possible to use different sensor technologies:
    - Global position: A separate module provides the global position in three dimensions (currently from GPS). If another sensor for global positioning should be used, it would only require rewriting the GPS module.
    - Orientation: The compass module provides the orientation in three dimensions. It can easily be replaced like the GPS.
- The modular software system makes it possible to easily change the high and low level controllers without affecting the rest of the software system.

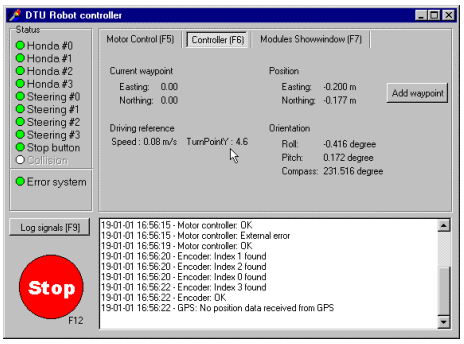


Figure 8.3 The software user interface on the robot.

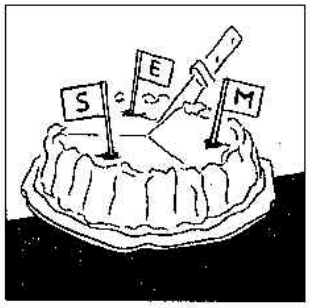


Figure 8.4 Cutting up the mechatronic cake. How to divide the functionality between mechanics, electronics and software.

Figure 8.5 Flexible steering and off road capabilities with 4WD and 4WS.

- We have used Windows for real-time control. The motor control loop runs at 50 Hz and the timing is correct. This is possible by using the hardware timers on the motherboard, which has higher priority than (most) other Window activities.

Cutting up the cake

With the modular design and the extensive use of software, we believe that we have cut up the "cake" right for a flexible off road research vehicle for the Department of Control Engineering and Design. The possibilities for research and development projects are many.

Performance

- Aim 0.5 – 1 m/s. Achieved: 2 m/s at rated speed with a rolling coefficient on 0.14 (firm soil appendix 2.3.3.b)
- Down to 0 m turning radius.
- High efficiency drive. It is possible to run tests for minimum 2 hours and more batteries can be added if needed.

Low level controllers

The wheel alignment is good. There are only minor angular deviations from Ackerman even though the steering of each wheel works totally independent.
The motor controllers have not been optimized. We have mainly focused on getting results with GPS and therefore, we have not prioritized to model, simulate and optimize the low-level controllers.

High level controllers

We have developed a vision system to control the vehicle following “plants” (objects) with good contrast. The initial tests performed well, but the focus on GPS and due to technical problems, we did not have time to make a final test of the vision system.
We have reached promising results with use of GPS to control a vehicle. We have shown that the position-offset error in our results is mainly due to deviation on the compass. It is vital with a very good orientation sensor.

Recommendations

From our results we are convinced that correction of the compass error and an improved position control strategy will produce very good results and that this vehicle can be used for autonomous tasks in the field.
We have made a platform for further research in the field and there is basis for many exciting master thesis projects.

9. Literature

9.1. Primary literature

[CRAIG89] “Robotics, Mechanics and control”, John J. Craig 1989
[DECKER] Decker Maschinen elemente, 12. Auflage, K. Decker, Carl Hanser Verlag 1995
[DEMKO] Demko Checklist for the Machinery Directive, Demko (Included on the cd)
[DIFF]A function generating differential mechanism for an exact solution of the steering problem, Carcaterra and D'Ambrogio, Elsevier 98
[DIGIEL] Introduktion til digitalelektronik – del 1 og 2, 1.udgave 1988, E.V.Sørensen, Den private ingeniørfond ved DTH
[GPS] “GPS” Keld Dueholm og Mikkel Laurentzious, Teknisk forlag 1999
[GPSIMP] A driver's steering aid for an agricultural implement, based on an electronic map and Real Time Kinematic DGPS, R.P. Van Zuydam, Computers and Electronics in Agriculture 24, 153-163, 1999
[GPSTRAC] “Automatic tractor guidance using carrier-phase differential GPS”, Thomas Bell, Elsevier 2000.
[HIGHTECH] High-tech in-row weeding Vehicle by Svend Christensen, Flakkebjerg 1999
[HOISCHEN] "Hoischen Technisches Zeichnen" 26. Auflage, Cornelsen Girardet 1997
[JMC98] „Digital Image Processing“. Jens Michael Carstensen, Technical University of Denmark, Lyngby 1998.
[KFV99] ”Udvikling af visionmodul til åben robotstyring”. Karsten Fischer-Vig, Feb 1999.
[KVL96] Notes for the course Machinery in Plant Production, KVL 96
[MECH] A 180^0 steering interval mechanism, E. Chicurel, Mechanism and Machine Theory 99
[ROARK] Roark's formulas for stress and strain
[SERVO] Servomekanismer 1, Niels Leth, Servolaboratoriet 1981
[TJALVE] Systematisk udformning af industriprodukter, E. Tjalve, Akademisk Forlag 1983
[TOMATO] Robotic Weed Control Systems for Tomatoes, W.S. Lee, D.C. Slaughter and D.K. Giles, Precision Agriculture, 1, 95-113, 1999
[TRACTORS] Tractors and their power units, 4^{th} edition, J.B. Liljedahl, P.K. Turnquist, D.W. Smith and M. Hoki, AVI 1989
[VEHICLE] Notes for the course Vehicle Technology, DTU part 1.

9.2. Secondary literature

- An all-terrain intelligent autonomous vehicle with sensor-fusion-based navigation capabilities, R. Jarvis, Control Eng. Practice, Vol. 4, No. 4, pp. 481-486, 1996
- An introduction to the Kalman Filter, G. Welch and G. Bishop, University of North Carolina, 1997
- Automating Agricultural Vehicles, V. Callaghan, P. Chernett, M. Colley, T. Lawson, J. Standeven, M. Carr-West and M.

Ragget, Industrial Robot, Volume 24, Number 5, pp. 364-369, 1997

- Automatisk styrede maskinanlæg, inklusive industrianlæg, At-anvisning Nr. 2.2.0.3, 1995
- Autonomous Guided Vehicle, Modelling, M. Nørgaard, N.K. Poulsen and O. Ravn, IMM, DTU 1997
- Computer Vision detection of weed plant in row crops, af Ph.D. studerende Hans Jørgen Andersen, Lab. for billedanalyse, AaU 1999
- Control of soil engaging tools and implements in plant production, af Finn Conrad, IKS og Martin Heide Jørgensen, DJF, 1999
- Datamatbaserede Reguleringssystemer, M.L. Levin, Servolaboratoriet 1982
- Development of an autonomous navigation system for an outdoor vehicle, K. Rintanen, H. Mäkelä, K. Koskinen, J. Puputti, M. Sampo and M. Ojala, Control Eng. Practice, Vol. 4, pp. 409-505, 1995
- "Elementær Analogelektronik" Forelæsningsnoter, O.H. Olesen 1992.
- Evaluation of Image-Based Landmark Recognition Techniques, Y. Takeuchi and M. Hebert, The Robotics Institute Carnegie Mellon University Pittsburgh PA 15213, 1998
- Forebyggelse af ukrudtsproblemer, samt mekanisk bekæmpelse af ukrudt og effekter på frøpuljen, M. Tersbøl, G. Mikkelsen, I. Rasmussen og S. Christensen, Danmarks Jordbrugsforskning og Landbrugets Rådgivningstjeneste.
- Fruit harvesting robots in Japan, M. Kondo, M. Monta and T. Fujiura, Adv. Space Res. Vol. 18, No. ½ , pp. 181-184, 1996
- Modelling and control of the motion of a trolley: Effect of motor dynamics on the dynamical model, Y. Yavin, Mathematical and Computer Modelling 30 p. 141-146, 1999
- Navigation and control of an autonomous horticultural robot, N.D. Tillett and T. Hague, Mechatronics Vol. 6, No. pp. 165-180, 1995
- Off-the-road locomotion, M.G. Bekker, The University of Michigan Press 1960
- Reguleringsteknik, O. Jannerup and O.H. Sørensen, Polyteknisk Forlag 1995
- Samplede Reguleringssystemer 1 & 2, Kurt Andersen, Servolaboratoriet 1982
- Teknik til rækkedyrkning af Afd. for Jordbrugsteknik og Produktionssystemer, Forskningscenter Bygholm 1999.
- Teknologi til differentieret plantepleje af Afdeling for jordbrugsteknik, DJF 1999
- The Agrobot Project, F. Buerni, M.Massa, G. Sandini and G. Costi, Adv. Space Res. Vol. 18, No. ½, pp. 185-189
- Automatic tractor guidance using carrier-phase differential GPS, B. Thomas, Computers and Electronics in Agriculture 25, 53-66, 2000
- The Automotive Chassis, Engineering Principles, J.Reimpell and H. Stoll, Arnold 1996
- The succesful development of a vision guidance system for agriculture, J. Billingsley and M. Schoenfisch, Computers and Electronics for Agriculture 16, pp. 147-163, 1996

- Theory of Ground Vehicles, J.Y. Wong, John Wiley & Sons 1978
- Tracking of row structure in three crops using image analysis, J.A. Marchant, Computers and Electronics in Agriculture 15, pp. 161-179, 1996
- Upgrading and repairing PC's, eigths edition, Macmillan computer publishing
- Vision Guided Precision Cultivation. Precision Agriculture, D.C. Slaughter, P. Chen and R.G. Curley, 1, pp. 199-216, 1999

9.3. Patents

- University of California, Davis, Department of Biological & Agricultural Engineering, USA. US patents no. 5,442,552 and 5,544,813 describes a Robotic Cultivator. Key scientist D.K.Giles.
- Rotorrenser PA 1998 01216

9.4. Product manuals

[HONDA] "Honda product manual DDW 2015 – 2020 DDW 4030 – 4060". Honda Motor Europe Ltd. 1997
[OLDHAM]
[ROSTA] "Rosta rubber suspension units" Rosta AG.
[TCM2] "TCM2 Electonic compass module – User's Manual" Revision 1.08, December 9, 1999, Precision Navigation Inc.
[Trimble] "Trimble 4700 Receiver Operation Manual" 1998 Trimble Navigation Limited
[USDIGITAL] Encoders, Inclinometers and Motion Control, USDigital 2000
[VALEO] "SWF Drive Technology". Valeo Scheibenwischsysteme & Elekt. Motoren.

9.5. Product sales material

Below is a list with companies, which we have received sales material from.

Batteries

Exide Danmark
Hawker A/S
M.P.X Electra Aps

Couplings

Manicus
Indutrans
Acton
Centa
Jens-S
Brdr. Klee A/S

Motors and amplifiers

SLO-SYN DC step motors, gearmotors … AVN-Elektronik

BJ-Gear A/S
Bosch
Compower
Faulhaber DC motors, Jenk
Lenze Kleinantriebe, Leo-Motor
Servo motors and controller, JVL
Leroy Somer Danmark
API Portescap, Micmotor
Tima A/S (Importing Honda motors)
Thrige Electric
AGV Products, Inc.

Wheel chair manufacturer

UB-Let (www.ub-let.dk)
Permobil (www.permobil.dk)
Sportster (www.sportster.dk)
Scandinavian mobility (Now dissolved)

Wheels

Brødrene Vestergaard A/S
Soco System A/S
Scan-Wheel
Scangros
Per Ejlersgård
Skandinavisk Dæk Import A/S
Texas (Garden equipment manufacturer)

Electronics etc.

Farnell
RS-Components
Danbit
PMA Cable Protection from Bagger-Nielsen