CHALMERS

Design, construction and verification of a self-balancing vehicle

Mikael Arvidsson

Jonas Karlsson

Department of Signals and Systems

Chalmers University of Technology

Göteborg, Sweden, 2012 EX050/2012

Abstract

The Segway Personal Transporter is a small footprint electrical vehicle designed by Dean Kamen to replace the car as a more environmentally friendly transportation method in metropolitan areas. The dynamics of the vehicle is similar to the classical control problem of an inverted pendulum, which means that it is unstable and prone to tip over. This is prevented by electronics sensing the pitch angle and its time derivative, controlling the motors to keep the vehicle balancing (1).

This kind of vehicle is interesting since it contains a lot of technology relevant to an environmentally friendly and energy efficient transportation industry. This thesis describes the development of a similar vehicle from scratch, incorporating every phase from literature study to planning, design, vehicle construction and verification. The main objective was to build a vehicle capable of transporting a person weighing up to 100 kg for 30 minutes or a distance of 10 km, whichever comes first. The rider controls are supposed to be natural movements; leaning forwards or backwards in combination with tilting the handlebar sideways should be the only rider input required to ride the vehicle.

The vehicle was built using a model-based control design and a top-down construction approach. The controller is a linear quadratic controller implemented in a 100 Hz control loop, designed to provide as fast response to disturbances as possible without saturating the control signal under normal operating conditions.

The need for adapting the control law to rider weight and height was investigated with a controller designed for a person 1,8 m tall weighing 80 kg. Simulations of persons having weights between 60-100 kg and heights between 1,6-1,9 m were performed, showing no need to adapt the controller. The controller could safely return the vehicle to upright positions even after angle disturbances of ± 6 degrees, the highest angle deviation considered to occur during operation.

Acknowledgments

This project was funded and supported by Epsilon, at the department Embedded Systems. Naturally in these kind of projects one comes in contact with a lot of enthusiastic and skilled people, sharing their knowledge and ideas. This has been a great resource and therefore we would like to thank some of those who have been especially helpful and inspiring;

The members of the Chalmers society X-P for their help and consultation when the mechanical parts in the workshop was made, a special thanks to Niklas Tolleson for his workshop lectures.

The members in Chalmers robotics society for their consultation regarding microcontroller programming, especially Benjamin Vedder with his expertise in the XMEGA microcontroller.

Finally we want to thank our supervisors; Torsten Wik at Chalmers, Kalle Lindbeck and Göran Kvick at Epsilon.

Nomenclature

SOC - state of charge

IMU - inertial measurement unit

RTOS - real time operating system

OSMC - open source motor controller

UART - universal asynchronous receiver/transmitter

ADC - analogue to digital converter

IC - integrated circuit

MEMS - microelectromechanical systems

MPU - motion processor unit

DMP - digital motion processor

LED - light emitting diode

IIR - infinite impulse response

LQ - linear quadratic

LQR - linear quadratic regulator

MOSFET - metal–oxide semiconductor field-effect transistor

EEPROM - electrically erasable programmable read-only memory

PWM - pulse width modulation

I^2C - inter-integrated circuit

List of figures

Table of contents

1 Introduction

1.1 Background

Companies working in the field of transportation are required to put more work into building environmentally friendly vehicles than ever. Air pollution, global warming and the need for sustainable energy pushes the demands for efficient, green energy powered vehicles. The European Union has in Regulation 443/2009 (in effect since June 8, 2009), set specific demands for the reduction of CO_2 emissions for manufacturers of new passenger cars. From 2012 Manufacturers failing to achieve their goal are required to pay an additional premium, and by 2020 no new passenger cars can be registered within the European Union that emits more than 95 g CO_2/km.

The manufacturers have a great need for competence in the field of hybrid vehicle technology or even fully electrical vehicle technology as a step towards fulfilling these goals. These new vehicles are complex machines and require engineering competence in many fields; mechanics, vehicle dynamics, automatic control, power electronics, battery technology, software engineering, microcomputer programming, network and communication engineering to name a few.

At the consultancy firm Epsilon, development of the employees' competence in these areas is of high priority. There is also an encouragement for the employees to learn areas outside their field of speciality, to form a better holistic understanding for these kinds of system. As a part of this encouragement Epsilon arranges regular project evenings were the employees meet and construct something together.

This project aimed to be a feasibility study and preparation for building two wheeled balancing vehicles in one of these evening projects at Epsilon. The vehicle is similar to the Segway Personal Transporter, invented and released in 2001 by Dean Kamen with the intention to revolutionize city transportation (1). This kind of vehicle contains many of the technologies required to build a hybrid or electrical car, and is sufficiently small to be a good platform for experimentation and learning for engineers aiming to build more environmental friendly passenger cars.

Another aim of the project was to implement a linear controller to make the vehicle balance and investigate if safe operation can be achieved without adapting the controller for riders of different weight and height.

Yet another contribution of this work concerns the update frequency of the control loop. The Segway Personal Transporter runs a controller which sends new commands to the motors at a rate of 1000 Hz (1). The system dynamics should allow for a slower update frequency and in this project it was investigated and proven the possibility for vehicle operation with a control loop running at 100 Hz.

1.2 Purpose

The main purpose was to design and construct a fully functional two wheeled balancing vehicle which can be used as a means of transportation for a single person. It should be driven by natural movements; forward and backwards motion should be achieved by leaning forwards and backwards. Turning should be achieved by tilting the handlebar sideways. To provide untethered operation the vehicles' energy source was designed to be a battery. One of the goals was to implement easy recharging of this battery. More specifically, the project goals were stated as follows:

Vehicle goals:

- Speed should be controlled by the rider leaning forwards and backwards.
- Turning should be controlled by tilting the handlebar.
- Balance and transport persons weighing up to 100 kg.
- Drive continuously for 30 minutes or 10 km, whichever comes first.
- Be rechargeable from a standard 220V 50Hz wall socket.
- Warn the rider when running low on battery power.
- Contain a user interface which communicates information about speed, current drawn from battery, total travelled distance, voltage to the motors and state of charge in the battery.
- Allow for changes of the control parameters without reprogramming.

General goals:

- The controller should be a linear controller.
- The controller should be implemented in the real time operating system FreeRTOS.
- Investigate theoretically if the controller response is good enough to provide safe operation of the vehicle for riders of different weights and heights.

- Implement a possibility for controller adaptation for riders of different weight and height.

1.3 Delimitations

To design and build a vehicle matching similar commercial options would take a lot more time than the 20 weeks allotted for this project. Therefore some delimitations were set up at the beginning of the project:

- No extra finish like paint and plastic details will be done.
- The motor controllers will be bought, no construction in this area will be made.
- The mathematical model will not be made entirely from scratch. Instead other models will be evaluated and then finally one of them will be used with necessary modifications to fit the project.
- The budget of the project must not exceed 30 000 SEK.

1.4 Method

This section explains the methods used in attempt to reach the purpose and goals setup in section 1.2. Since time was short, a to the authors well-known CAD-software and well proven and asserted methods taught in engineering educations were consistently chosen for each phase of the project. Some examples of the adopted work methodologies are:

- All mechanics, electronics and software were first designed using a top down approach, to promote good structure and division into subsystems. The top down approach is also a good way to make sure no functionality is left unimplemented (2).

- The most critical hardware was compared to alternatives and then a selection process based on the Kesselring matrix (2) was used to decide on the final choice.

- The control system was developed using model-based design. A linearized version of a mathematical model for the vehicle was used in the controller design stage. This controller was verified by simulations with the nonlinear mathematical model in the verification stage, before implementation in the actual system. The use of a nonlinear model in the verification stage is better since the linearized model is only valid in a narrow region around the operating point (3).

1.4.1 Preliminary modelling and investigation of the dynamical system

The first step was to gain knowledge about the dynamical system such that the hardware could be designed to be controllable from the start. Many books on automatic control use the inverted pendulum problem as an example, and the type of vehicle constructed in this project is basically an inverted pendulum mounted on wheels, where the rider and the chassis corresponds to up the pendulum itself.

It was decided to use an existing model found in literature and adapting it rather than deriving it completely from scratch. This was motivated by the possibility to review several models and compare them to eachother in the same time as it would take to derive a new model. Thorough investigations of papers on modelling of two wheeled balancing vehicles were then conducted, rendering a number of dynamical models.

The models could basically be divided into two types; those derived using Newtonian mechanics and those derived using Lagrangian mechanics. The resulting equations of motions should be the same for all models based on the same assumptions but the derivation is different. Finally a model of an inverted pendulum mounted on wheels, but without a rider, was selected. More on this model and the motivation for using it can be found in Chapter 3.

1.4.2 Mechanical and electrical design

When a study of the proposed system dynamics had been conducted and a final model had been selected it was time to design the real system.

All mechanical design was made with the help of the CAD-software Autodesk Inventor, using a top-down approach. Every part of the model was then converted to a drawing (selected drawings can be found in Appendix B) and printed for use in the mechanical workshop where all mechanics were constructed. The project members were able to attend and finish a course granting access to a workshop called ‘Prototyplabbet’. This workshop is owned by Chalmers University of Technology and provided mills, lathes, drills, band saw, a CNC-mill and welding gear that was used to build the mechanical parts.

The electrical system was also designed using a top-down approach, starting with a schematic over the complete system that would enable the vehicle to fulfil the specification. The system was then divided into subsystems which could be designed and specified in detail, breaking the initial large problem into smaller problems that could be solved by each subsystem.

1.4.3 Software design

Since one of the goals of the project was to implement the main control software in a microcontroller running a real time operating system, a lot of constraints that helped deciding on software structure were present from the start. Also the division of the electrical system into subsystems as explained in the previous section enabled the division of the software as well, breaking it into smaller and more easily manageable parts.

The subsystems software models where then described in detail in text and flow charts. Flow chart is a useful tool to design program functionality before actually writing any code. This gave the benefit of separating program functionality problems (handled in flow charts) from specific hardware and communication problems (handled when writing code for a specific microcontroller).

1.4.4 Selection of main components

The component selection required some basic theory of electric motors, drive circuits and batteries acquired from educational engineering books. This theory in combination with the project goals served as reference when comparing components on the basis of performance, cost, availability and robustness.

1.4.5 Testing and verification

Finally the finished system was tested and assessed relative to the initial purpose. The controller designed with the aid of the mathematical model was implemented in software. Extensive test riding was performed and data was logged to MATLAB.

2 Theory

This chapter is intended to bring the reader up to speed on the theory required to understand this report. The reader is expected to have an engineering background and some knowledge of dynamical systems, control theory and real time control systems. The material in this chapter goes through specific aspects of these subjects and readers who are familiar with the material may consider this chapter optional.

There are several ways of deriving the equations of motion for a mechanical system. One commonly used method is to draw a free body diagram with forces and torques. The equations are then derived by applying Newton's laws and solving for accelerations to arrive at a differential equation that describes the motion of the system. This method requires good understanding of the system (having to know the significant forces) and for complex systems the equations can become very long and tedious to manipulate.

Another modelling approach is to analyze the energies in the system to derive the equations of motion. The benefit of this approach is that energies are scalars and not dependent of directions like forces are. One such method is based on the Lagrangian equations and is called Lagrangian mechanics.

2.1 Modelling based on energy – The Lagrangian method

To use the Lagrangian method a number of simple steps needs to be applied (4):

1. Determine a set of independent coordinates that describes the system motion in the most convenient way, these are called generalized coordinates. The number of generalized coordinates will be equal to the degrees of freedom in the system.

2. Derive expressions for the energies in the system and apply the Lagrangian function:

$$L = T - U \tag{1}$$

where T = Kinetic energy and U = Potential energy.

3. Evaluate the Lagrangian equations:

$$\frac{d}{dt}\left(\frac{dL}{\partial \dot{q}_i}\right) - \frac{dL}{\partial q_i} = F_{q_i} \qquad [2]$$

where q_i is the i:th generalized coordinate, and F_{q_i} is the i:th generalized force. The generalized forces are the sum of forces projected in the direction of q_i.

The result of the last step is the equations of motion for the mechanical system. Both the Lagrangian method described here and the method using free body diagrams combined with Newtons' laws will yield the same equations of motion when applied to the same system using the same assumptions.

2.2 Control strategies

This section will briefly introduce some control strategies that are relevant to this work.

2.2.1 PID-controller

The PID-controller is by far the most commonly used controller today (5). It tries to minimize the error between the reference signal and the actual output signal. One benefit of working with this type of controller is that it is rather intuitive to tune. When the proportional (P) part is increased the controller gets faster and more aggressive, the stability margins are generally decreased (for systems of higher order than one), the ability to handle load disturbances is increased but to the price of more control signal activity. The integral (I) part gives higher gain for low frequencies and therefore it compensates for low frequency process disturbances. Generally the integral part also eliminates residual steady state error. The derivative (D) part may add to the stability margins, with the drawback of high frequency gain is increased. This can lead to problems when the measured signal is noisy which leads to unwanted control signal activity. Therefore, the D-part is almost always implemented with a low pass filter. Since the stability margin is increased by the D-part, it can partly compensate for the decreased stability margins introduced by the integral part.

The PID-controller is often implemented with some extra functions to make it more realizable in a practical implementation. One phenomenon that can occur is integrator windup. This occurs when you have integral action combined with an actuator that becomes saturated. The integral part builds up integral action to a very large value. When the error finally decreases the integral part is still large so it will keep the actuator in the saturated state for a considerable amount of time.

Below is a PID controller in continuous time domain:

$$u(t) = K\left(e(t) + \frac{1}{T_i}\int_0^t e(s)ds + T_d\frac{de(t)}{dt}\right) \qquad [3]$$

In frequency domain:

$$U(s) = K\left(E(s) + \frac{1}{sT_i}E(s) + T_d sE(s)\right) \qquad [4]$$

2.2.2 State feedback

Pole placement state feedback is a way of controlling the behaviour of the closed loop system by placing the closed loop poles at positions where the desired behaviour of the system is achieved (5).

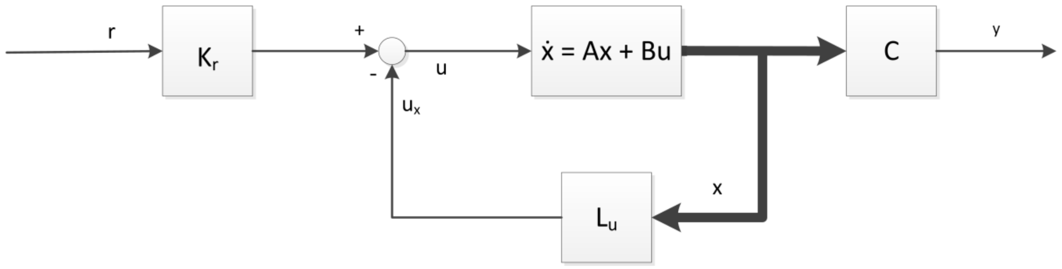


Figure 1 - State feedback controller for a linear system

The equations describing the states and output of a linear time invariant system are given by:

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) \end{cases} \qquad [5]$$

All the states are assumed to be known or observable and this gives the control law:

$$u(t) = -L_u x(t) + K_r r(t) \quad [6]$$

Where L_u is the row vector that is used for the pole placement. The gain parameter K_r is selected so that the low frequency gain is 1 from the reference signal r to the output y. The transfer function from reference value to the output is given by:

$$G_{ry}(s) = C(sI - A)^{-1} B K_r \quad [7]$$

By setting $G_{ry}(0)=1$ one gets the following expression for K_r:

$$K_r = \frac{1}{C(-A)^{-1}B} \quad [8]$$

2.2.3 Optimal control

The drawback with ordinary state feedback controller is that it is not very easy to tune. It is not always easy to get a good grasp of what actually happens with the performance when the poles are placed differently (6). The linear quadratic (LQ) controller is implemented as a state feedback controller with a feedback gain vector L which is determined such that J is minimized.

$$J = \int_0^\infty (x^T(t)Q_1 x(t) + 2x^T(t)Q_{12}u(t) + u^T(t)Q_2 u(t))dt \quad [9]$$

The controller will depend on the matrices Q_1 and Q_2, which tell us how we penalize control signal activity and control error. Since this is very seldom given explicitly, these matrices are considered as design parameters. The matrices should be symmetric and matrix Q_1 must be semi definite ($Q_1 \geq 0$) and Q_2 must be positive definite ($Q_2>0$) (3). The feedback law that minimizes the cost function J is given by (3):

$$u(t) = -Lx(t) \quad [10]$$

where L is given by:

$$L = {Q_2}^{-1} B^T S(t) \quad [11]$$

and *S(t)* is given by the following continuous time Riccati equation:

$$A^T S(t) + S(t)A - S(t)BQ_2{}^{-1}B^T S(t) + Q_1 = 0 \qquad [12]$$

The controller can be combined with a Kalman filter to estimate unknown states and to reduce the impact of the reference signal noise and measurement noise. Then it is called a LQG controller, where the G stands for Gaussian. The R matrix is a covariance matrix and it describes the noise affecting the states (R_1), the noise affecting the measurements (R_2) and how they might be correlated (R_{12})(3).

$$R = \begin{bmatrix} R_1 & R_{12} \\ R_{12}^T & R_2 \end{bmatrix} \qquad [13]$$

The problem of determining the Kalman gain *K* (see Equation [14]) is very similar to calculate the *L* vector in the previous LQ-case and they are therefore sometimes referred to as dual problems. The *K* that minimizes the estimation error variance when the noise is assumed to be white and Gaussian is:

$$K = (PC^T + NR_{12})R_2^{-1} \qquad [14]$$

Here *P* is given by the following Riccati equation:

$$AP + PA^T + NR_1N^T - (PC^T + NR_{12})R_2^{-1}(PC + NR_{12})^T = 0 \qquad [15]$$

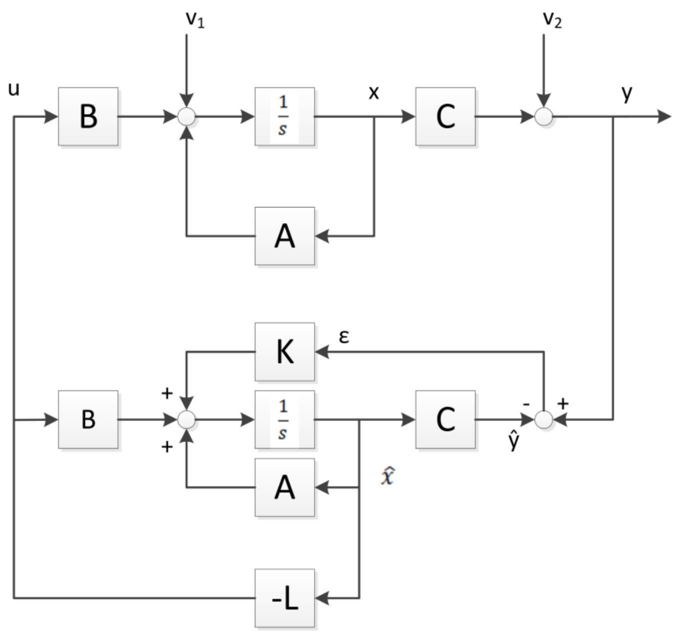


Figure 2 - LGQ controller

2.3 Real world sensor behaviour

When implementing a control system in an actual system the outputs and the states of the system may be far from the ideal signal considered in textbook examples. The signals have to be measured by sensors, and sometimes the relevant information cannot be measured directly but must be calculated from some other signal. The imperfections in the signals may include measurement noise, inaccuracies due to resolution problems, offsets etc. and will therefore have to be processed in some way. Usually the noise is of high frequency character compared to the dynamics of the system under control and requires low pass filtering. This filtering will introduce phase lag in the control loop which decreases the stability margins.

This section aims to explain how the signals relevant to a balancing vehicle can be measured and which distortions these measurements may contain. Usually the pitch angle and the pitch angular rate are needed independent of control strategy. Sometimes it may also be of value to know the wheels' position and speed, either for control purposes or simply rider awareness.

2.3.1 Accelerometer

An accelerometer is a sensor device capable of sensing acceleration. These sensors may be built in numerous different ways. In modern applications where low cost and small size is important, the microelectromechanical system (MEMS) type accelerometers present a feasible option. MEMS devices began appearing on the commercial market in the late 80's and has since evolved into extremely light weight, cheap miniature sensors with the same accuracy as traditional techniques which are larger and more costly (7).

A modern accelerometer often includes a spring loaded structure whose deflection in response to external forces can be capacitively sensed and converted to an electrical signal (8). In balancing vehicles where the pitch angle is of interest the accelerometer can be used to measure the orientation of the vehicle with respect to gravity. The gravitational force will affect the spring loaded structure and the sensor sends a signal representing the acceleration along its sensing axis.

The drawback of accelerometers for pitch angle estimation in vehicles is that they are affected by linear acceleration occurring naturally while driving as well as rotational acceleration if the sensor is placed with a distance to the vehicle's rotational axis.

2.3.2 Gyroscope

A gyroscopic sensor (gyro) is used to measure the angular rate of an object with respect to an inertial system. They are commonly of MEMS type like the accelerometers discussed in the previous section if low cost and small size is desirable. The mechanical system used for sensing may vary, but a common type uses two vibrating masses to measure the Coriolis acceleration. The Coriolis acceleration comes from rotational motion and by capacitively sensing the distance between the vibrating masses, the amount of Coriolis acceleration they are affected by can be deduced. This allows the angular rate to be calculated (9).

The gyro output signal can also be integrated to find the angle of the motion. This estimate will be less affected by linear acceleration than the accelerometer output, but will suffer from drifting when numerical integration is used. Gyro sensors also suffer from bias, meaning that the output when completely still is not equal to zero. This bias can be dependent on temperature, vibrations, sensor supply voltage etc. and is therefore extremely hard to predict and remove, and will also add to the drift of the angle estimation (7).

2.3.3 Sensor fusion algorithms

Knowing the properties of these two sensors, smart filtering can be used to fuse the outputs and attain a better estimation of the angle. Characteristics that are relevant to pitch angle estimation are listed in Table 1.

	Useful properties	Bad properties
Accelerometer	Absolute reference of the pitch angle when unaffected by other accelerations than gravity	Misleading output when affected by accelerations
Gyro	Can estimate the pitch angle through integration of sensor output independent of motion	Only incremental estimation which drifts due to varying bias and numerical integration

Table 1 – Properties of the accelerometer and gyro sensors

A filter commonly seen in hobby applications is the 'Complementary filter'. This is a simple filter which demands very little processing power and is simple to implement. The filter basically uses the gyro for short term estimation and correcting this estimation based on the accelerometer's absolute reference of the gravity. The filter equation can be written in pseudo code as:

$$angle = (1-\beta)\left(angle + gyro_{output} * sampletime\right) + \beta * accelerometer_{output} \qquad [16]$$

It basically applies low pass filtering of the accelerometer output and high pass filtering of the gyro output. By tuning the constant beta the time constant of the filter is changed until a satisfactory result is achieved, and gyro bias drifting is removed. A significant drawback with this filter is that while removing the drift from the angle estimation, it cannot give an estimate of the bias on the gyro output. This means that control systems requiring the angular rate as an input, eg a PD-controller with the pitch angle as reference, cannot use the output of the gyro directly because of the bias.

Another common filter is the Kalman filter, requiring more processing power and effort to implement. The Kalman filter is derived from optimization theory and is the optimal estimator when the disturbances in the system are normally distributed white noise. The filter is described in more detail in Section 2.2.3, the only thing to remember here is that the model used in the filter is a model of the sensors behaviour. This model includes the gyro bias, which then can be obtained as an output from the filter. This makes it possible to attain a better measurement of the angular rate since the bias can be removed from the gyro output.

Another interesting approach is to try to estimate and remove as much of the gyro bias as possible before using the sensor fusion filters described above. This method can be based on information about the particular gyro sensor used, eg how temperature, supply voltage and vibrations are affecting the bias. However since extensive knowledge of the selected sensor is required it may not be feasible if sufficient data is not available from the sensor manufacturer.

2.3.4 Quadrature encoders

A sensor commonly used to measure wheel and/or motor speed and position is a rotary encoder. This sensor contains a rotating disc on a shaft which is connected to the motor or wheel shaft and spins with the same rate. The sensor then measures the rotation of the disc in some way. For example, optical encoders use tiny slots in the disc and photo detectors to obtain the current movement.

Rotary encoders come in two types, incremental which outputs the motion of the shaft and absolute, which gives the current position of the shaft. The quadrature encoder is an incremental encoder which outputs two digital signals as seen in Figure 3. The signals are displaced by 90 degrees, making it possible to calculate the current position, speed and direction of the revolution (10).

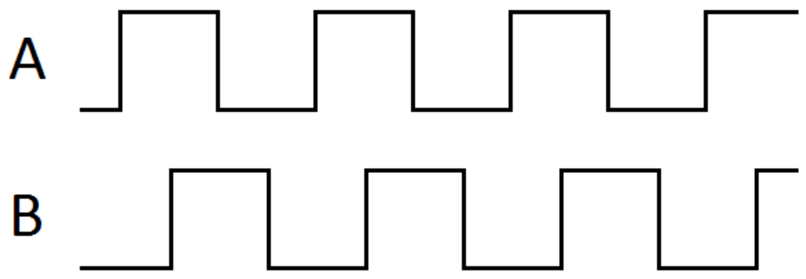


Figure 3 - Quadrature signals

3 Modelling and simulations

This section describes the mathematical model used to design and verify the control system. As mentioned in the method section, the model was not derived from scratch but rather selected after reviewing several existing models. The chosen model was derived by Yorihisa Yamamoto (11) using Lagrangian mechanics. The model also included dynamics for the yaw angle. Those equations were discarded since only dynamics concerning pitch angle were considered in this project. Other modifications to include a rider with varying mass and height were also carried out.

Initial simulations without the real parameters were then carried out to get insight into the dynamics of this kind of system. This simplified constructional design choices, since simulations provided an easy way to verify model response when changing certain parameters.

3.1 Mathematical model

The system can roughly be represented by a pendulum with its end attached to two wheels. The pendulum has two equilibrium points: the pendulum standing straight up or hanging straight down. In this case the upright position is the only point of interest.

The model used in this project is actually two models, one nonlinear model and one linear version of the nonlinear model. The linear model was derived from the nonlinear version at the operating point where the pendulum is standing straight up. This model was mainly used for control design since many controller design methodologies are based on linear theory. Because of the nonlinearities of the real system this model can only be a good approximation in a region close to the operating point.

The nonlinear model is however able to represent the pendulum in all operating points and therefore makes it a candidate to verify by simulation that the designed controller works.

3.1.1 Model parameters

Some of the parameter values used in the model described in the next section need a bit of explanation. For example, the inertia of the body and the friction coefficients are very hard to calculate exactly and can be time consuming and hard to find by experiments. This section motivates some particular parameters:

- The inertia of the chassis is part of the total body inertia $J_{b.}$ This parameter was found after designing the whole chassis in Autodesk Inventor, and then using that program to calculate the inertia.

- The motor parameters were estimated from a dynamometer test run published by the manufacturer.

- The friction coefficients were estimated and considered very unreliable.

3.1.2 Nonlinear model

Below follows a couple of figures that describes the vehicle. The mathematical model is based upon these figures with some small modifications. For example, the yaw angle ϕ is disregarded (always zero).

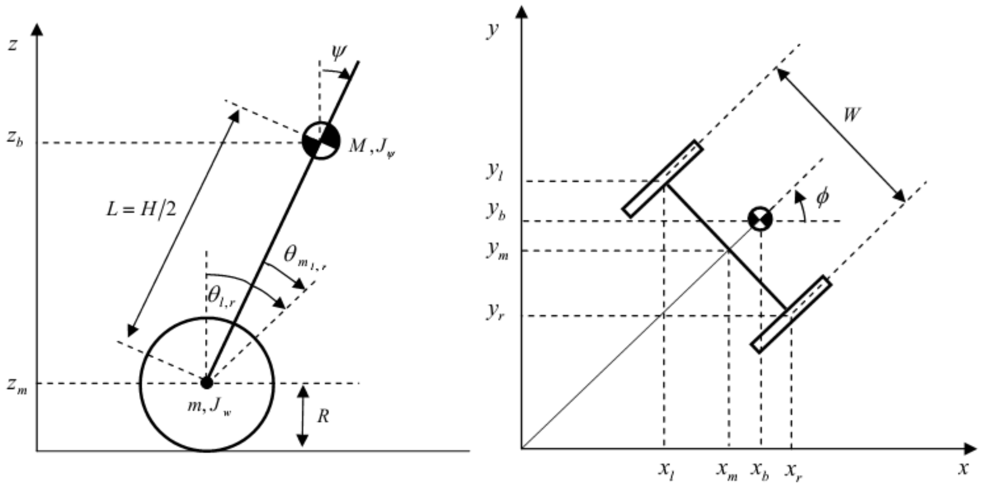


Figure 4 - Description of the coordinate system and angles. The figures come from (11).

The parameters used in the simulation of the two wheeled vehicle:

[Parameter]	[Unit]	[Comments]
$g = 9{,}81$	$[m/s^2]$	Gravitational constant
$m = 4{,}6$	[kg]	Wheel weight
$R = 0{,}2413$	[m]	Wheel radius
$J_w = (mR^2)/2$	$[kgm^2]$	Wheel inertia
$M_v = 30{,}06$	[kg]	Vehicle body weight
$M_{rider} = 80$	[kg]	Rider weight
$M = M_v + M_{rider}$	[m]	Total weight
$H = -0{,}03$	[m]	Body height
$L_{rider} = 1{,}8$	[m]	Rider height

L = (Mv*H/2+Lrider*0.55*Mrider)/M	[m]	Distance to the centre of mass from the wheel axle
$J_{rider} = M_{rider}L^2_{rider}/3$	[kgm^2]	Rider inertia
$J_b = J_{rider} + 1{,}49$	[kgm^2]	Body pitch inertia
$J_m = 0{,}0075$	[kgm^2]	DC motor inertia
$R_m = 0{,}141$	[Ohm]	DC motor resistance
$k_b = 0{,}722$	[Vs/rad]	DC motor back EMF constant
$k_t = 0{,}833$	[Nm/A]	DC motor torque constant
n = 14	-	Gear ratio
$f_m = 0{,}3$	-	Friction coefficient between body and DC motor
$f_w = 0$	-	Friction coefficient between wheels and ground

Table 2 - Model parameters

The derivation of the equations of motion can then be performed by the three steps from Section 2.1:

1. Selection of general coordinates:

   The number of general coordinates is equal to the degrees of freedom in the system. For balancing control purposes these coordinates were selected as:

   ψ : The pitch angle of the pendulum
   θ : The average of the right and left wheel angle

2. Applying the Lagrangian function to the expression for energy:

   The energies can be divided into translational kinetic energy (T_1), rotational kinetic energy (T_2) and potential energy (U):

$$\begin{cases} T_1 = \frac{1}{2}m\left(\dot{x}_l^2 + \dot{y}_l^2 + \dot{z}_l^2\right) + \frac{1}{2}m(\dot{x}_r^2 + \dot{y}_r^2 + \dot{z}_r^2) + \frac{1}{2}M\left(\dot{x}_b^2 + \dot{y}_b^2 + \dot{z}_b^2\right) \\ T_2 = \frac{1}{2}J_w\dot{\theta}_l^2 + \frac{1}{2}J_w\dot{\theta}_r^2 + \frac{1}{2}J_w\dot{\psi}^2 + \frac{1}{2}n^2J_m\left(\dot{\theta}_l - \dot{\psi}\right)^2 + \frac{1}{2}n^2J_m\left(\dot{\theta}_r - \dot{\psi}\right)^2 \\ U = mgz_l + mgz_r + mgz_b \end{cases} \quad [17]$$

   where the coordinates can be expressed as:

$$(x_m, y_m, z_m) = \left(\int R\dot{\theta}\,dt, 0, R\right) \quad [18]$$

$$(x_l,\, y_l,\, z_l) = \left(x_m, \frac{W}{2},\, z_m\right) \qquad [19]$$

$$(x_r,\, y_r,\, z_r) = \left(x_m, -\frac{W}{2},\, z_m\right) \qquad [20]$$

$$(x_b,\, y_b,\, z_b) = (x_m + L\sin\psi\,, y_m, z_m + L\cos\psi) \qquad [21]$$

3. Evaluate the Lagrangian equations:

   The next step is putting *L* from the Lagrangian function into the Lagrangian equations:

$$\begin{cases} \dfrac{d}{dt}\left(\dfrac{\partial L}{\partial\dot{\theta}}\right) - \dfrac{\partial L}{\partial\theta} = F_\theta \\ \dfrac{d}{dt}\left(\dfrac{\partial L}{\partial\dot{\psi}}\right) - \dfrac{\partial L}{\partial\psi} = F_\psi \end{cases} \qquad [22]$$

   The result is the equations of motion for this system:

$$\begin{cases} \big((2m+M)R^2 + 2J_w + 2n^2J_m\big)\ddot{\theta} + (MLRcos(\psi) - 2n^2J_m)\ddot{\psi} \\ \qquad -MLR\dot{\psi}\sin(\psi) = \; F_\theta \\ \\ (MLRcos(\psi) - 2n^2J_m)\ddot{\theta} + (ML^2 + \; J_b + 2n^2J_m)\ddot{\psi} \\ \qquad - MgL\sin(\psi) = \; F_\psi \end{cases} \qquad [23]$$

The last step needed before the model is complete is to find the generalized forces F_θ and F_ψ. Before that, however, the equations were rewritten for easier Simulink model building by solving for the accelerations. This can be made by rewriting the system on a matrix form and invert the left side matrix:

$$\begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \begin{bmatrix} \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} f_1 + \; F_\theta \\ f_2 + \; F_\psi \end{bmatrix} \qquad [24]$$

$$M_{11} = \; (2m + M)R^2 + 2J_w + 2n^2J_m \qquad [25]$$

$$M_{12} = \; M_{21} = MLRcos(\psi) - 2n^2J_m \qquad [26]$$

$$M_{22} = ML^2 + \; J_b + 2n^2J_m \qquad [27]$$

$$f_1 = MLR\dot{\psi}\, sin(\psi) \qquad [28]$$

$$f_2 = MgL\, sin(\psi) \qquad [29]$$

Inversion of the left side matrix solves the system for the accelerations:

$$\begin{bmatrix} \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}^{-1} \begin{bmatrix} f_1 + F_\theta \\ f_2 + F_\psi \end{bmatrix} \qquad [30]$$

The system is powered by two DC-motors, and the general forces can thus be derived by considering the DC motor torque and viscous friction (i_l and i_r denotes the currents in the left and the right motor.):

$$\begin{cases} F_\theta = nK_t(i_l + i_r) + f_m(\dot{\psi} - \dot{\theta}_l) + f_m(\dot{\psi} - \dot{\theta}_r) - f_w(\dot{\theta}_l + \dot{\theta}_r) \\ F_\psi = -nK_t(i_l + i_r) - f_m(\dot{\psi} - \dot{\theta}_l) - f_m(\dot{\psi} - \dot{\theta}_r) \end{cases} \qquad [31]$$

Applying Kirschoff's voltage law to the electrical part of the DC-motor, ignoring the inductance (since the electrical time constant of the motor is much smaller than the mechanical time constant) and solving for the current gives:

$$i_j = \frac{v_j + k_b(\dot{\psi} - \dot{\theta}_j)}{R_m} \quad , j = l, r \qquad [32]$$

where v_j denotes the input voltage to the right (*j*=*r*) and left (*j*=*l*) motor. Putting this result in Equation [31] renders the final expression for the generalized forces:

$$F_\theta = \alpha(v_l + v_r) - 2(\beta + f_w)\dot{\theta} + 2\beta\dot{\psi} \qquad [33]$$

$$F_\psi = -\alpha(v_l + v_r) + 2\beta\dot{\theta} - 2\beta\dot{\psi} \qquad [34]$$

$$\propto = \frac{nK_t}{R_m} \qquad [35]$$

$$\beta = \frac{nK_tK_b}{R_m} + f_m \qquad [36]$$

The final equation used when building the simulation model in Simulink is Equation [30] where the general forces are given by Equation [33]-[36].

3.1.3 Linear model

The linear model was used for control design and was derived by linearizing the nonlinear model in the operating point where the vehicle is standing straight up. This means that $\psi \rightarrow 0$, $\sin(\psi) \rightarrow \psi$ and $\cos(\psi) \rightarrow 1$ was considered. This gives the following system:

$$\begin{cases} \left((2m+M)R^2+2J_w+2n^2J_m\right)\ddot{\theta}+(MLR-2n^2J_m)\ddot{\psi}=F_\theta \\ (MLR-2n^2J_m)\ddot{\theta}+(ML^2+J_b+2n^2J_m)\ddot{\psi}-MgL\psi=F_\psi \end{cases} \qquad [37]$$

To achieve a controllable model on state space form the vector x = $\{\theta,\psi,\dot{\theta},\dot{\psi}\}$ was chosen. To shorten the equations for the full state space, a matrix E is defined:

$$E=\begin{bmatrix} (2m+M)R^2+2J_w+2n^2J_m & MLR-2n^2J_m \\ MLR-2n^2J_m & ML^2+J_b+2n^2J_m \end{bmatrix} \qquad [38]$$

By letting E(i,j) denote the i:th row and the j:th column of E, and det(E) be the determinant of E, the linear model can be described on state space form as:

$$A=\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \dfrac{-gMLE(1,2)}{\det(E)} & \dfrac{-2\left((\beta+f_w)E(2,2)+\beta E(1,2)\right)}{\det(E)} & \dfrac{2\beta(E(2,2)+E(1,2))}{\det(E)} \\ 0 & \dfrac{gMLE(1,1)}{\det(E)} & \dfrac{2\left((\beta+f_w)E(1,2)+\beta E(1,1)\right)}{\det(E)} & \dfrac{-2\beta(E(1,1)+E(1,2))}{\det(E)} \end{bmatrix} \qquad [39]$$

$$B=\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \dfrac{\alpha(E(2,2)+E(1,2))}{\det(E)} & \dfrac{\alpha(E(2,2)+E(1,2))}{\det(E)} \\ \dfrac{-\alpha(E(1,1)+E(1,2))}{\det(E)} & \dfrac{-\alpha(E(1,1)+E(1,2))}{\det(E)} \end{bmatrix} \qquad [40]$$

$$C=\begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} \qquad [41]$$

$$D=0 \qquad [42]$$

$$\begin{cases} \dot{\boldsymbol{x}}=A\boldsymbol{x}+Bu \\ \boldsymbol{y}=C\boldsymbol{x}+Du \end{cases} \qquad [43]$$

3.1.4 Open-loop test of nonlinear and linear model

To test if the output of the models seemed reasonable a simple open loop test was performed. The pendulum started at -6 degrees, which means slightly displaced from the upper equilibrium point, and was then simulated for a duration of 30 seconds (see Figure 5).

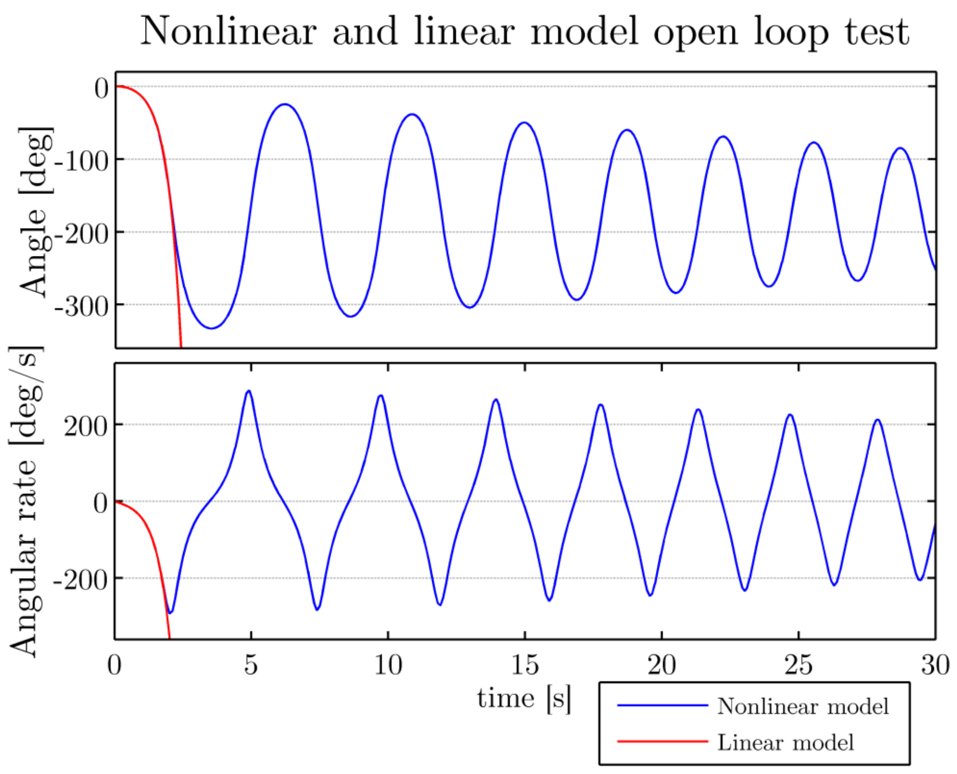


Figure 5 - Open loop response with initial angle of -6 degrees

The nonlinear model shows that the pendulum falls down from the starting point and starts to oscillate round the bottom equilibrium point. It is clear that the oscillations are decreasing over time which can be explained by the energy loss in the resistors as a voltage is generated by the rotation of the motors. In reality the oscillations would probably diminish faster due to dampening not accounted for in the model.

It is also clear that the linear model exhibits similar dynamics for the region close to the upper equilibrium point, but as expected it cannot give an accurate representation of the system when moving too far from this point, although it does follow the nonlinear system surprisingly long (more than 200 degrees from the equilibrium point).

4 Controller design

As stated in the purpose of this project, the vehicle was designed to use a linear controller for balancing and driving. The basis for the controller design was the linear model. This chapter deals with the strategy used to control the vehicle, the design of the controller and the simulated system response.

4.1 Control strategy – using the error to get speed

As seen in Chapter 3 the equilibrium point when the vehicle is standing is unstable, meaning that it will tip over if not controlled. Naturally the first objective of the controller must be to prevent this from happening. The second objective of the controller is to make the rider able to control the speed of the vehicle.

Most control strategies solve similar requirements by changing a reference signal, using the controller to make the output of the system track this reference. If the rider is supposed to control the speed by leaning however, it becomes a tricky behaviour to implement. When the rider leans forward or backward the centre of gravity of the vehicle changes, and the system begins to tip over. It would now be impossible for the controller to know if this change has occurred because the rider wants to move or if the system has been exposed to a disturbance, say for example a bump in the road or a heavy gust of wind. Therefore it is hard to design a control strategy which uses for example pitch angle or rotational speed of the wheels as reference to make the vehicle move.

To fulfil the goal of controlling speed by leaning backwards or forwards, a different control strategy was used, where the goal is always to keep the pitch angle at zero degrees. By designing this controller sufficiently 'bad' (not being able to remove steady state errors caused by disturbances), the rider's leaning motions can be viewed as a disturbance causing a steady state error. The controller will continue to try to attain a zero pitch angle, giving power to the motors and the steady state error will result in a speed of the vehicle.

4.2 Discrete time LQ-controller design

A controller very well suited for the task described in the previous section is the LQR-controller. This type of controller uses negative feedback of the states to push the states to zero.

4.2.1 Controller specification

To design this controller the designer specifies the cost for the states deviating from zero, as well as the cost for control signal activity. A simple method is to tune the costs until a specification of the controller performance is fulfilled. For this system the following specification was chosen:

- The controller should return the vehicle pitch angle to zero from an initial starting point of ±6 degrees, with the control signal just barely saturating.

This specification was chosen since it was easier to find the limitations of the control signal than it was to find out how fast the controller needs to be for pleasant operation of the vehicle. This way the response will basically be as fast as it can be under the limitations placed on the control signal. The limitations on the control signal comes from the following assumptions; normal riding conditions constitutes a pitch angle displacement of ±3 degrees, and the control signal needs some headroom to handle bumps in the road without saturating. These bumps were assumed to cause a temporary deviation of totally ±6 degrees and for safe operation the control signal should not saturate excessively to maintain rider safety.

4.2.2 Model discretization and LQ cost matrices

Before the controller was designed according to the specification, the model of the system was converted to discrete time with a sampling frequency of 100 Hz, i.e. the same as the proposed control loop. This gave the eigenvalues of the system:

$$\begin{array}{c} 1.0000 \\ 0.8291 \\ 1.0210 \\ 0.9807 \end{array} \qquad [44]$$

By linear control theory this is an unstable system which agrees with the discussion in Chapter 3. Now, the cost matrices were selected and at this time it was known that the battery to be used would deliver up to 24 V to the motors as input signal. By trial and error, the matrices from Equation [9] were tuned until the specification was fulfilled. With the state vector x = $\{\theta, \psi, \dot{\theta}, \dot{\psi}\}$, and disregarding the matrix Q_{12}, the matrices Q_1 and Q_2 became:

$$Q_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2*10^8 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad [45]$$

$$Q_2 = \begin{bmatrix} 1000 & 0 \\ 0 & 1000 \end{bmatrix} \quad [46]$$

The gain vector L for a rider of weight 80 kg and height 1,8 m is then:

$$\boldsymbol{L} = [-0.0190 \;\; -272.4750 \;\;\; -1.5801 \;\;\; -18.2560] \quad [47]$$

The eigenvalues of the closed loop system shows that by discrete time linear control theory, this system is now stable:

$$\begin{gathered} 0.7567 + 0.1675i \\ 0.7567 - 0.1675i \\ 0.9994 \\ 0.9996 \end{gathered} \quad [48]$$

4.3 Closed loop system response simulations

This section presents some graphs from the simulations of the controller designed in the previous section, so that the reader can verify that the specification is indeed fulfilled. All simulations to verify the controller was performed on the nonlinear model.

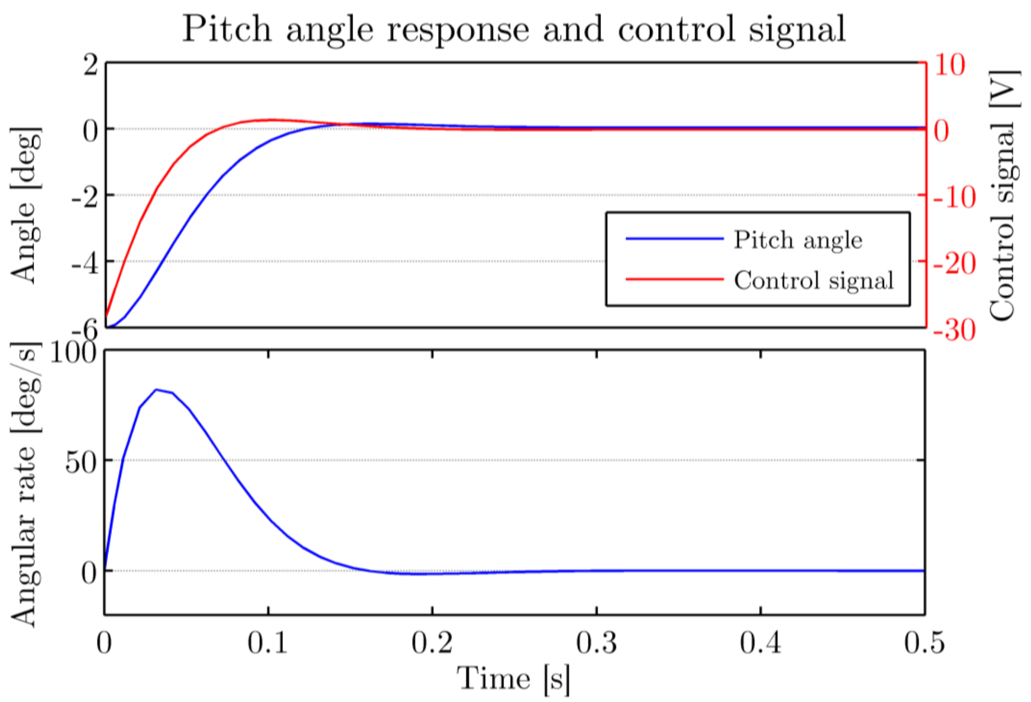


Figure 6 - Closed loop system response with the system at rest, with an initial pitch angle displacement of 6 degrees. The control signal is slightly saturated in the beginning of the simulation.

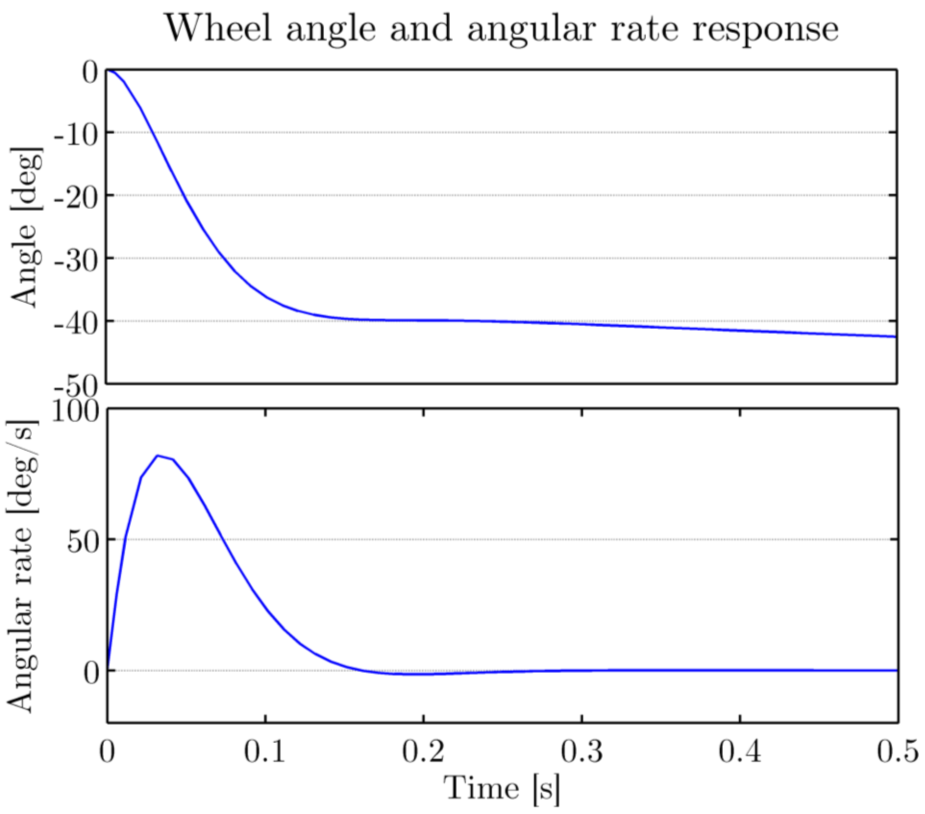


Figure 7 - The average wheel angle and angular velocity, same simulation as Figure 6.

It seems like the wheels moved backwards when trying to push the pitch angle to zero. This was a good hint that the proposed control strategy may be feasible in practice.

5 Overview of the vehicle

This chapter gives an overview of the final vehicle. It precedes Chapter 6, which describes the design and implementation of the control system on a subsystem level. The aim of this overview is to provide the reader with some orientation to better understand the vehicle before going into the details of the subsystems.

5.1 Mechanical overview

The mechanical system is mainly composed of custom made aluminium parts that have been processed in the workshop. The wheels are attached to the motor gearbox with a steel hub, and the motor is bolted to the chassis. The chassis itself consists of bent aluminium sheet metal formed into a box which protects the vital components while serving as a standing platform for the rider and mounting place for the handlebar. To reduce the risk of slipping, the platform has been covered with grip tape.

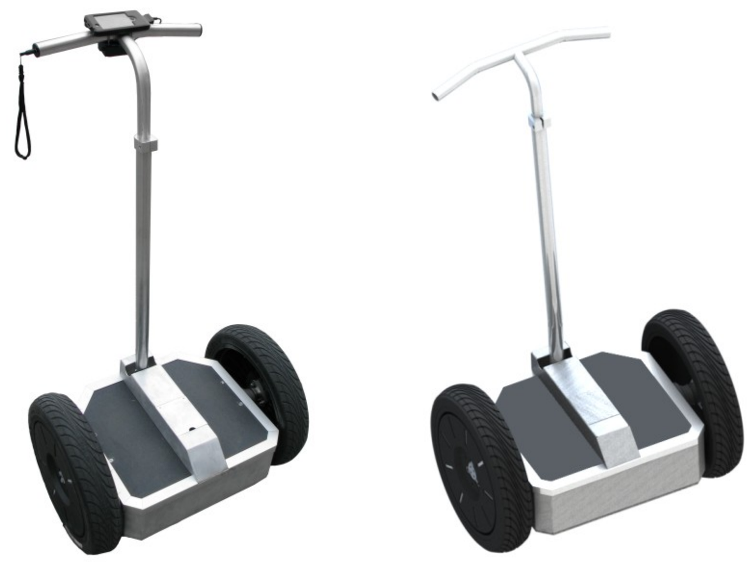


Figure 8 - The vehicle in reality to the left and the CAD-model to the right.

Some of the features include a detachable handlebar, steer-by-leaning mechanics, a dead mans' switch and a centre console with a lid granting access to the main circuit breaker and charging plug. On top of the handlebar is the mounting bracket for the graphical user interface.

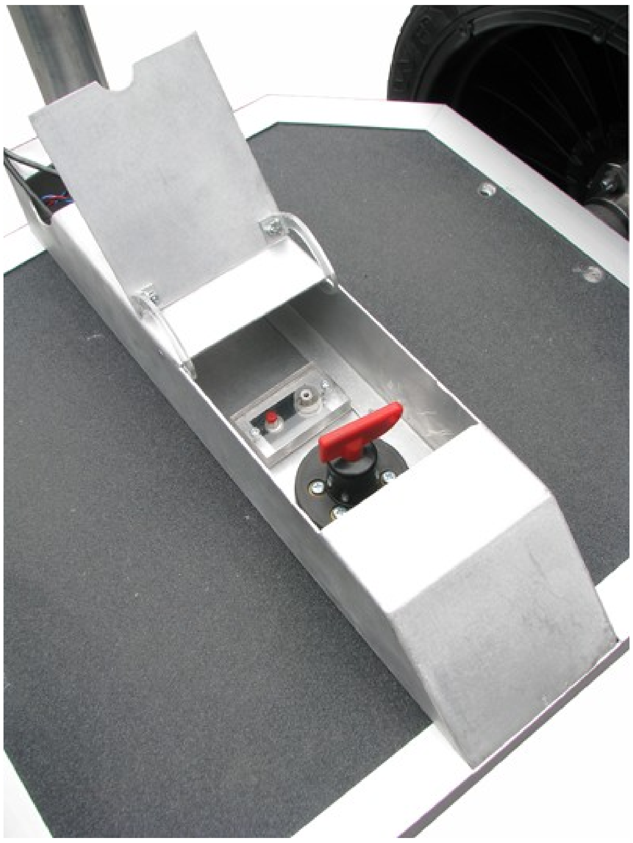


Figure 9 - Lid open to show the main circuit breaker and the charger plug

5.2 Electrical overview

The electrical system is centred on the battery and the distribution of the battery voltage and current to the other subsystems. Since the battery for this kind of vehicle contains a lot of energy it is necessary to monitor the operational status of the main electrical system and have safety features to turn it off in case of an emergency or malfunction. The safety measures in this system consist of fuses, a dead man's switch and a power management system. The next figure shows a conceptual diagram of the electrical system:

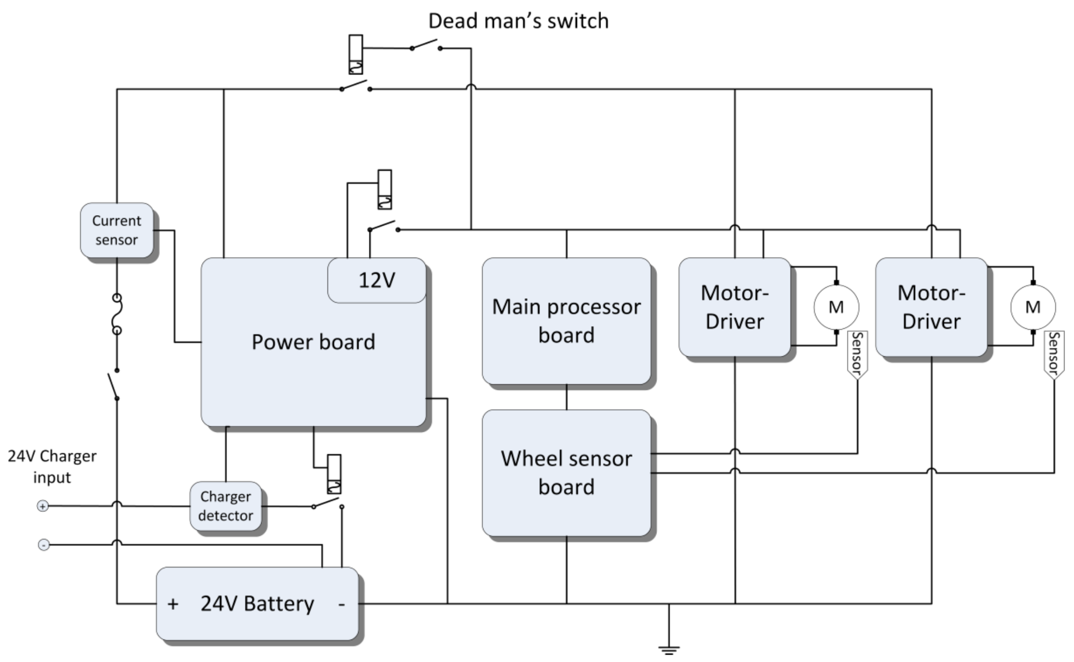


Figure 10 - Overview of the electrical system.

Power source, a 480 Wh $LiFePO_4$ battery

The battery was selected to fulfil the goal of 10 kilometres range on one charge, as well as provide sufficient current for the torque needed to drive the vehicle. A 24 V $LiFePO_4$ rated at 20 Ah and 60 A continuous discharge current was chosen. This battery also came with an internal battery management system, which monitors and balances each cell of the battery during charging. This made the integration of the battery into the electrical system easier and less time consuming.

Safety features

The most important design parameter for the main electrical system was the current rating and the safety measures. All components and cabling has been selected to withstand 50 A continuously and 100 A peak. The main cabling is a 6,5mm^2 copper wire.

The power board in Figure 10 is responsible for controlling relays to distribute power to the other subsystems and shut down the motors when a malfunction is detected. It can also control relays related to charging logics, to turn off sensitive systems and make sure the vehicle cannot drive while being charged. This system also has other functionality further described in Section 6.4.

The dead man's switch is a wrist band that at all times should be worn by the rider. If the rider falls off the band will disconnect a 3,5 mm plug which turns off a relay that cuts the power to the motors. In this way the vehicle cannot continue to drive, which otherwise could result in collisions with other people or objects. Fuses are mounted between the battery and the main switch to limit the current in case of a severe short circuit failure in the vehicle.

5.3 Software overview

The software of this vehicle is running on four different processors. This makes the system modular which reduces complexity during software design and facilitates easy troubleshooting and system replacement. These benefits come at a price: the need for communication between the processors. This communications is based on a UART serial interface.

The most critical software runs on the main processor system which incorporates a NXP LPC1769 ARM Cortex-M3 processor. A real time operating system called FreeRTOS was selected since it provides all the necessary software as well as being free to use. This software is designed to run the control loop, collect and compile information from the other software systems and send this information to the graphical user interface, and optionally to a PC. The PC connectivity is mainly used for debugging and logging with MATLAB.

The other software modules are incorporated and described in the appropriate subsystem section below. Figure 11 shows a diagram of the entire software system:

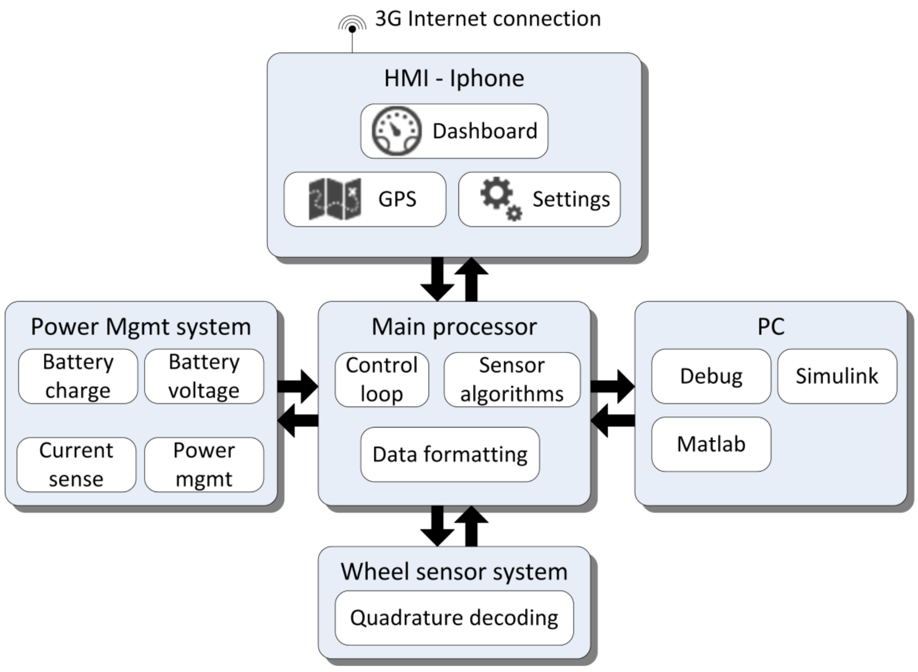


Figure 11 - Overview of the software system, all modules communicate through serial interfaces

6 Subsystem design and implementation

This chapter lists all the subsystems and provides a more in depth coverage of the vehicle. Each subsystem section start with a design section which tells the original intent of the subsystem and a list of design goals. They also contain the actual implementation which describes how it was finally constructed.

6.1 Handlebar

The handlebar is attached to the chassis and serves as support for the rider and placeholder for the HMI.

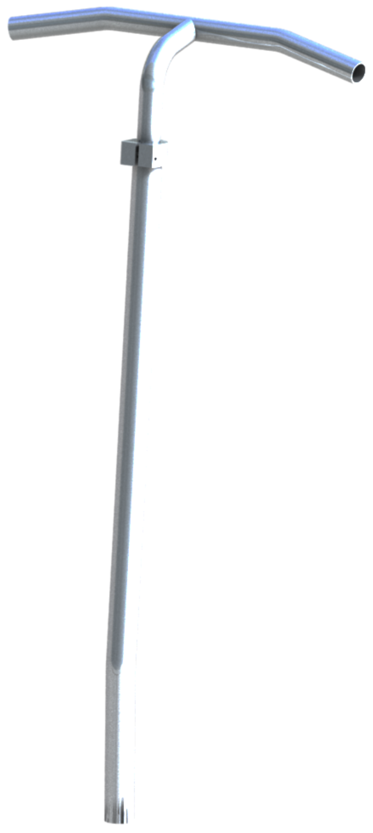


Figure 12 - CAD image of the handlebar

6.1.1 Handlebar design

The handlebar provides support for steer-by-leaning. The rider should feel some resistance when tilting the handlebar. A dead-mans grip is also implemented in the handlebar so that the power to the vehicle is cut when the rider falls off.

Design goals:

- Adjustable height to provide a stable support for riders of heights between 1,6-1,9 m.
- Detachable to minimize space requirements of the vehicle when not in use.
- Provide steer-by-leaning of the vehicle.
- Sturdy enough to handle light collisions.

6.1.2 Handlebar implementation

The construction was made using 1,5mm thick aluminium tube which was bent, cut and welded together. To provide height adjustment the handlebar pole was divided into a lower and an upper section, where the upper section slides into the lower to achieve different heights. A sleeve made of Polyoximethylene plastic was manufactured to provide a smooth, low friction sliding surface and a milled aluminium clamp that can be tightened is used to lock the height (see Figure 13).

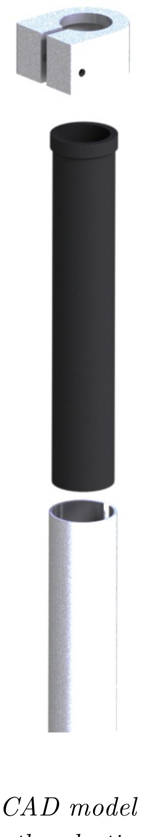


Figure 13 - Exploded view from CAD model of the handlebar height adjustment components; the clamp, the plastic sleeve and the lower pole.

The handlebar is mounted between two pillow blocks and has a spring loaded bottom end. The springs give resistance as the rider tilts the handlebar by leaning, giving the rider a natural way of turning the vehicle. When the handlebar is tilted the angle is sensed by a magnetic rotational sensor which outputs the tilt angle as a voltage to an ADC-converter on the main processor. There is also a 3,5mm connector in the left side of the handlebar. A plug with a wrist band is attached to the connector and needs to be worn by the rider at all times. As soon as the plug is disconnected the power is cut to the motors.

6.2 Chassis

The chassis is the heart of the mechanical system which all other mechanical parts in some way are connected to.

6.2.1 Chassis design

The chassis was designed to provide the rider with a safe and robust standing platform and to protect the electrical system and components.

Design goals:

- Protect components from impacts and crashes.
- Provide easy access to vital components during service and repairing.

6.2.2 Chassis implementation

Bent alumina sheet metal was used to design a chassis in the shape of a box. The top plate of the box is easy to detach and the space inside is big enough to allow easy servicing of components.

The top surface of the chassis has been covered with grip tape to reduce the risk of slipping accidents, and a thin anodized aluminium profile has been added to the edge for esthetical reasons. The top also has a centre console with a lid, which provides access to the main circuit breaker of the system as well as the charging plug. The hinges for this lid was designed and cut by laser from a piece of plastic to allow for opening angles larger than 90 degrees while not having a visible hinge on the top of the lid.

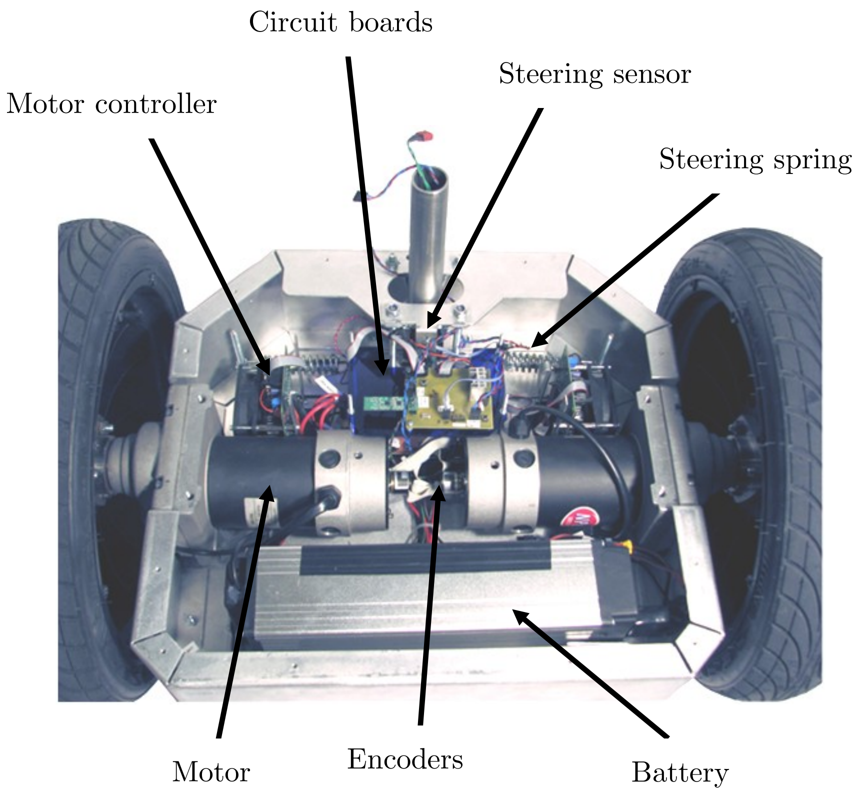


Figure 14 - Inside the chassis. The circuit boards are screwed to plastic shelves mounted on top of each other. The power board and the wheel sensor board are covered by the top shelf which has the main processing board.

6.3 Powertrain

This system enables the propelling and control of the vehicle, and was considered as one of the most tolerance sensitive systems during the design phase. It is paramount to have a strong powertrain that can handle bumps and curbs, with low backlash in the driving mechanism. Too much backlash can introduce oscillations in a control system and make it harder or impossible to control.

6.3.1 Powertrain design

The powertrain makes it possible for the electric motors to propel the vehicle. It is defined as the base plate with the motors, gearbox and wheels. During design and evaluation of the powertrain, it was soon concluded that a solution which requires a minimum of mechanical calculations and simulations was desired. The strategy was to minimize the number of custom components and bearings which requires a lot of workshop hours while still getting a highly robust powertrain.

This narrowed the selection of motors to those with attached gearboxes capable of handling high loads on the output shaft.

Design goals:

- Support a single rider weighing less than 100 kg.
- Be strong enough to handle minor bumps and going up and down curbs at low speed.
- Propel the vehicle safely at the maximum speed of 20 km/h.
- Provide enough torque to balance a 100 kg rider in inclines of 20 degrees.

6.3.2 Powertrain implementation

The powertrain basically consists of the motors, gearboxes, connector hubs and the wheels (see Figure 15).

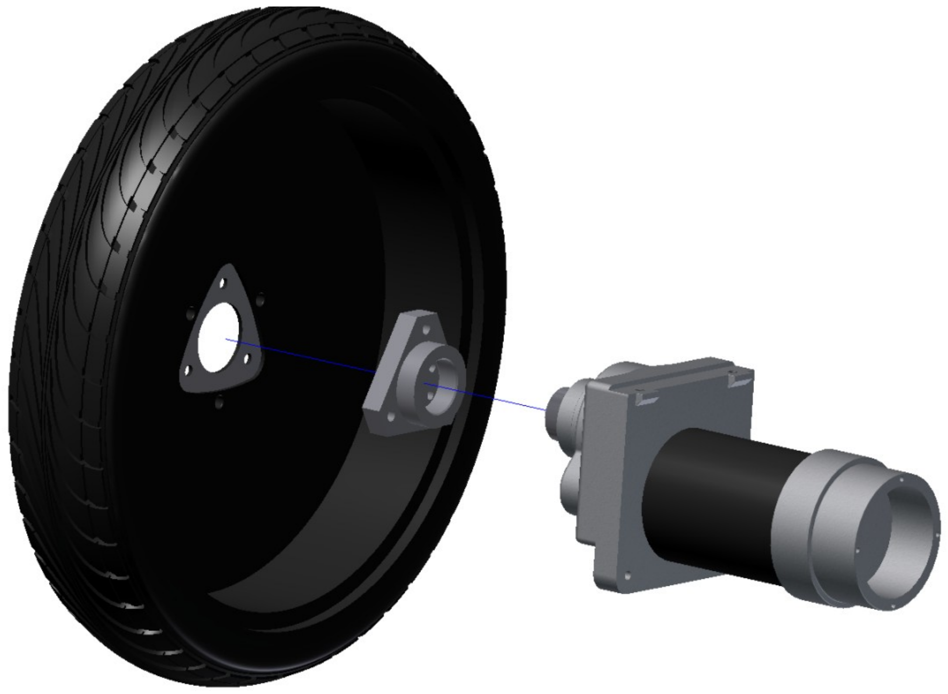


Figure 15 - Exploded CAD view of the powertrain, showing the motors with the attached gearbox, connector hub and the wheel.

Motor and gearbox

Since the regular DC machine is very easy to model and control and can be bought with an attached gearbox it was the primary choice for this vehicle. A suitable model called NPC T-74 was chosen. It was mainly chosen on account of the price, the robust gearbox and the connection possibilities to the chassis and wheel.

Parameters w/respect to output shaft of gearbox	Unit	Value
Maximum speed	[rpm]	248
Maximum torque	[Nm]	167
Gearbox ratio	-	14:1
Maximum output power	[kW]	1,24
Maximum radial load on gearbox axis	[kg]	135

Table 3 - Motor specifications

The gearbox consists of metal gears with a ratio of 14:1, and it was bolted straight onto the base plate. An adaptor hub used to connect the gearbox and the wheel was designed and then machined in a lathe and a CNC-mill. The hub is bolted to the gearbox shaft with four bolts in the centre of the hub. The hub is attached with three bolts to the wheel.

Motor controller

The motor controller is called 'open source motor controller' (OSMC) and is capable of handling 160 A of continuous current. It is mainly composed of a power stage with a total of sixteen MOSFETs and a control part which consists of a driver chip and surrounding support components. It also has a switched buck converter to support a stable 12 V source to the driver chip. A fan is mounted above the power stage to cool the MOSFETs. There is also a disable pin available to turn off the motor controller completely without cutting the power to the power stage. This pin is also connected to the main processor.

The board is controlled with a PWM signal from the main processor. The PWM is fed through an inverter. The non-inverted together with the inverted PWM signal is the only control signal required to control the motor controller. The non-inverted signal is connected to the B-gate seen in Figure 17 and the inverted PWM is connected to the D-gate. The gate driver of the motor controller makes sure that gates A and B or C and D does not conduct at the same time. This prevents short circuit through the motor controller, a phenomenon called 'shoot-through'.

The control strategy of the motors is called "lock anti-phase". Gates A and D will be active during the blue area of the PWM signal seen in Figure 16. Gates B and C will be active during the red area. The current path of the motor controller can be seen in Figure 17.

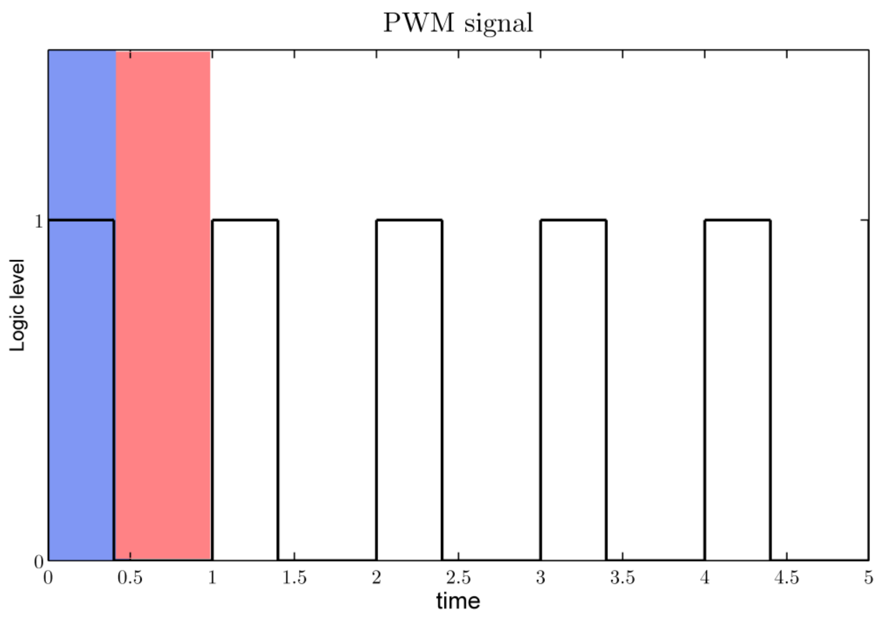


Figure 16 - PWM signal

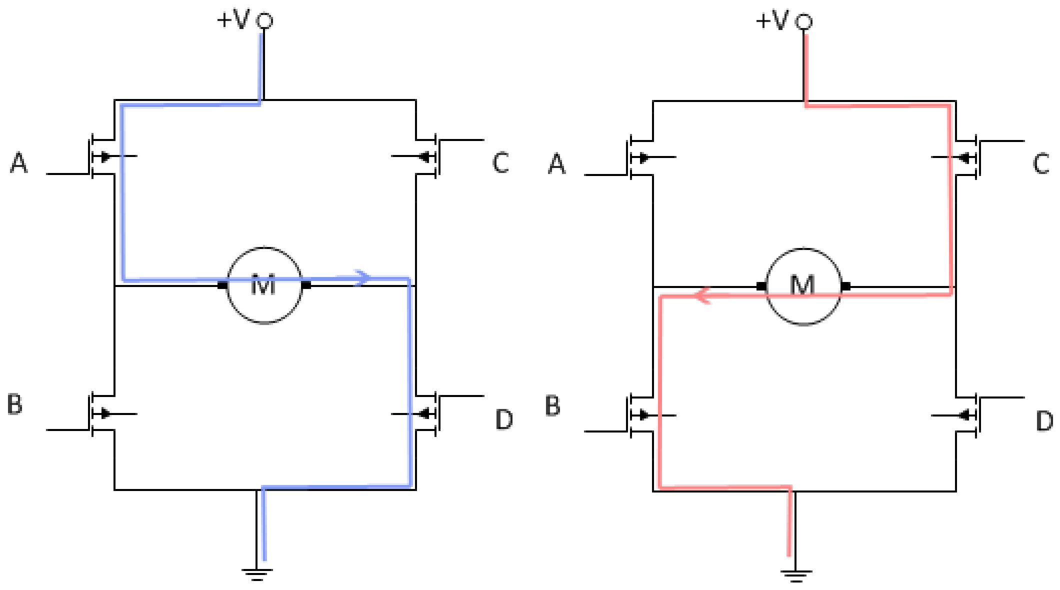


Figure 17 - Motor controller.

Wheels

The wheels were bought as spare parts from Segway. The reason for buying these wheels was that they had good size, appropriate weight and good mounting interfaces.

Figure 18 - The selected wheel, a 19" wheel weighing 4,6 kg.

6.4 Power management system

The power management system is the supply hub for all low power electrical subsystems.

6.4.1 Power management design

The purpose of the power board microcontroller is to monitor the battery and distribute the power to the different subsystems. The information acquired by the power management system is transmitted to the main processor to enable presentation on the user interface.

Design goals:

- Warn the user of low battery voltage.
- Estimate state of charge in the battery.
- Measure current flowing from the battery.
- Shut down the other subsystems when the charging plug is inserted.
- Only allow charging when the correct voltage is applied to the charge input.
- Send information about current, state of charging, battery voltage and warnings to the main processor through a serial UART channel.
- Convert the main 24 V to 12 V that is utilized by the different subsystems.

6.4.2 Power management implementation

The heart of the power management board is an Atmel ATMEGA168P microcontroller. It has been chosen due to previous knowledge of this certain processor, sufficient performance and because it is cheap. The microcontroller runs at five volts and this voltage is supported from a linear regulator. The linear regulator is chosen because of simplicity and the low current draw makes it a feasible choice.

The circuit board has an input socket for the charger plug that is internally connected to a relay which is activated when the voltage is at the appropriate level. The activation of this relay lets the current flow into the battery and the battery is charged. The charger relay is turned off when the charging plug is disconnected and the relay that enables current to the other subsystems is turned on.

The board has a switched regulator to support 12 V to the relays and the other circuit boards. The regulator is of switched type due to the fact that they enable higher efficiency than simple linear voltage regulators.

The system is designed for full charging at each charging event. When the charging is completed the voltage will reach a threshold and the SOC value is set to 100%. The battery voltage reading is also used to activate a warning system to indicate when the battery level is too low. The warning system consists of a buzzer that makes a sound and a warning-flag is set in the serial package that is sent on the UART communication channel to the main processor.

The 12 V supply provided by the switched regulator also activates the main relay that allows current to the motor controllers. A switch is inserted in series with the cable that activates the coil in the main relay. This switch is used as a dead-mans grip located at the handlebar.

In series with the cables going from the battery is a main switch to turn off everything on the vehicle and also two fuses in parallel to limit to high power surge.

The current sensor that is in series with the battery is read by the ADC and used to calculate the charge going out of the battery. The SOC value is stored to a EEPROM memory in the microcontroller so the SOC value is kept even if the power is turned off.

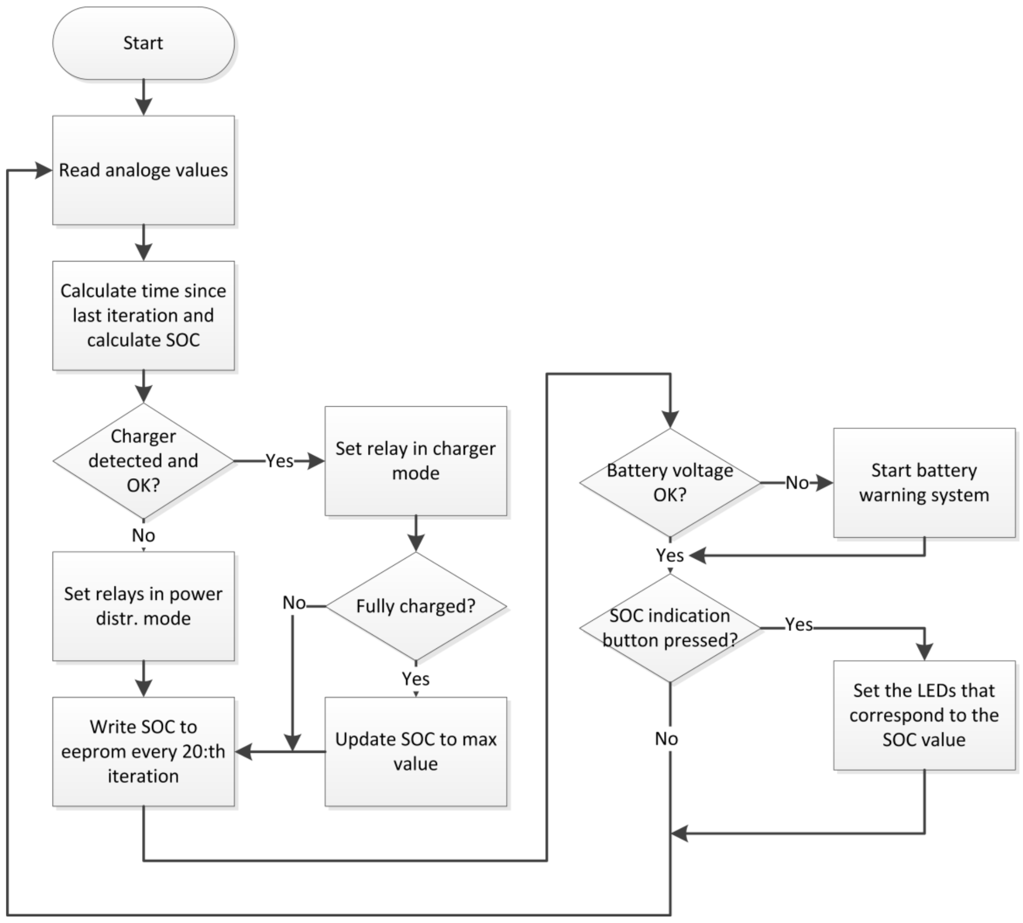

Start
Read analoge values
Calculate time since last iteration and calculate SOC
Charger detected and OK?
Yes
No
Set relay in charger mode
Set relays in power distr. mode
Fully charged?
No
Yes
Write SOC to eeprom every 20:th iteration
Update SOC to max value
Battery voltage OK?
No
Start battery warning system
Yes
SOC indication button pressed?
Yes
Set the LEDs that correspond to the SOC value
No

Figure 19 - Power board program flowchart

6.5 Main processing system

The main processing system is the most complex software module in the vehicle. It works as the communication hub between all the other software systems (see Figure 11), and has the responsibility to answer to all communication events within a finite time. It is also responsible for gathering and converting all sensor information to a format usable by the control system, and to run the control loop.

6.5.1 Main processing system design

This system was designed to be a single circuit board with a microprocessor. All communications was designed as serial communications using UART with baud rates ranging from 9600 baud to 115200 baud.

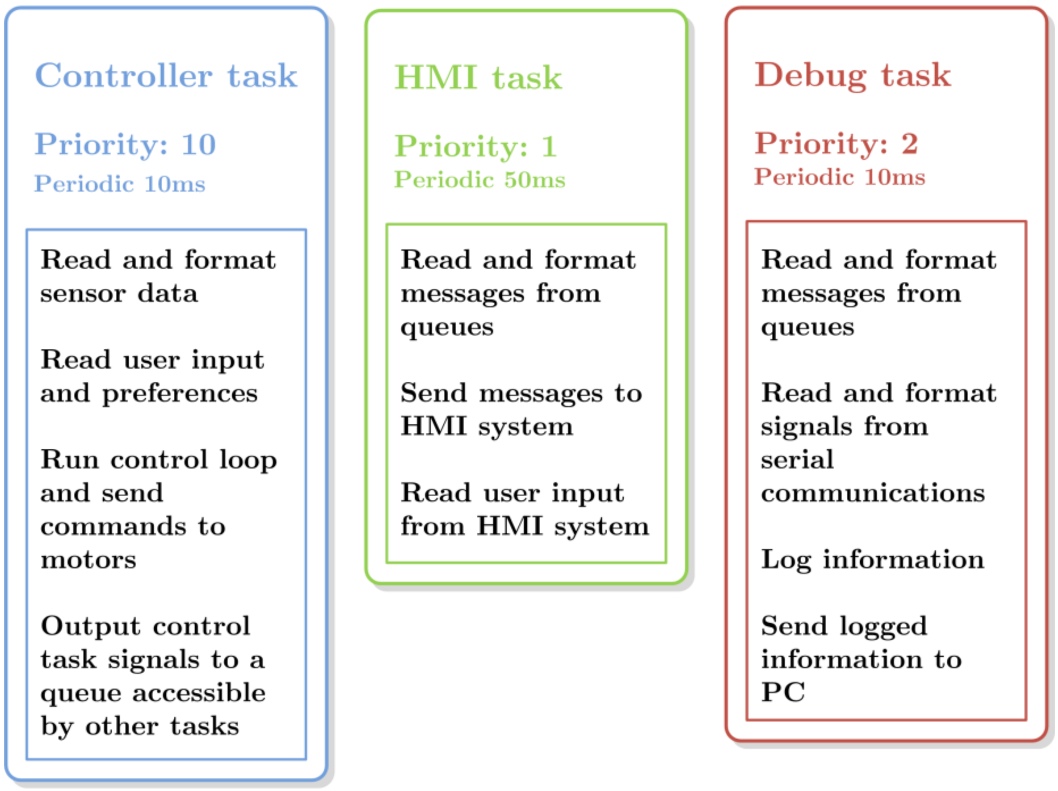
Design goals:

- Answer to all communication requests, interpret and perform actions based on the received data.
- Run the real-time operating system FreeRTOS.
- Collect and format all sensor data to be usable in the control loop.
- Run the control loop at 100 Hz.
- Send collected signals from the system during riding to a PC running MATLAB.
- Format and send messages containing information to the graphical user interface.

6.5.2 Main processing system implementation

The FreeRTOS real time operating system was already from the beginning selected as the basis for the software in this system. This operating system is open source and free to use. At its heart is a pre-emptive scheduler utilizing round robin style scheduling for tasks of equal priorities. The microcontroller selected to run the software is NXP LPC1769 ARM Cortex-M3 based on a 32-bit architecture. The microcontroller CPU clock runs at 100 MHz and the clock for the peripherals was selected to run at 25 MHz.

To fulfil the design goals the first implementation step was to divide the work of the system into tasks. These tasks and their responsibility can be seen in the next figure:



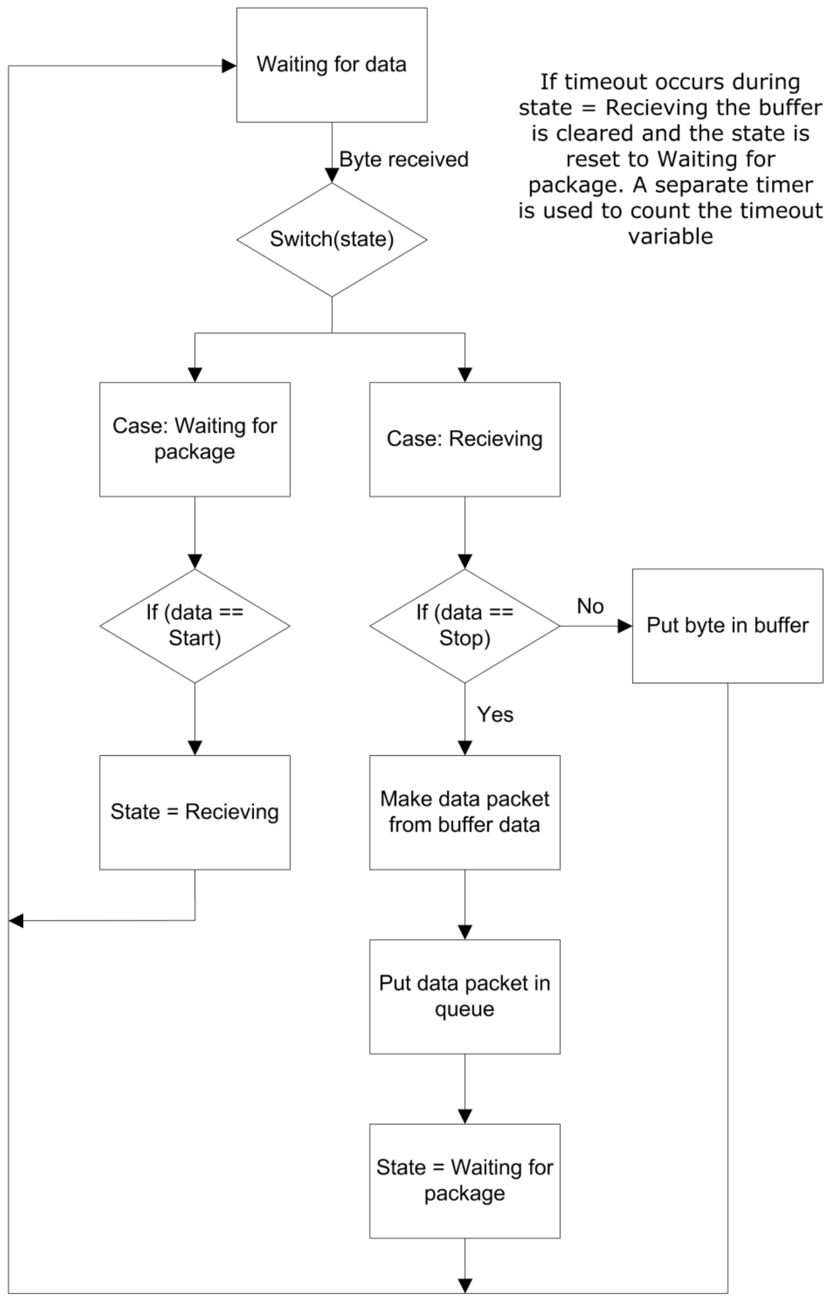
Controller task
Priority: 10
Periodic 10ms
Read and format sensor data
Read user input and preferences
Run control loop and send commands to motors
Output control task signals to a queue accessible by other tasks
HMI task
Priority: 1
Periodic 50ms
Read and format messages from queues
Send messages to HMI system
Read user input from HMI system
Debug task
Priority: 2
Periodic 10ms
Read and format messages from queues
Read and format signals from serial communications
Log information
Send logged information to PC

Figure 20 - Implemented tasks in the main processing system. Higher numeric priority means higher priority

These tasks perform the main workload of the processor in a periodic manner. To avoid data corruption when several tasks try to read and/or write to the same memory locations, all inter-task communication is handled by queues. These queues are thread safe and provides a safe way to let each task access the data they need.

The UART modules that handles the serial communications from the processor was selected so that each task uses its own module. In this way there is no need for mutual exclusion guards on the modules to prevent deadlocks or data corruption.

Since the sending and receiving of data over the UART interface is byte oriented, some kind of overlying protocol is needed if data longer than one byte shall be sent. In this case a simple protocol was formed where each node expects packages of a predetermined size, with one byte signalling the start of a package and another signalling the end of a package. Each receiver uses an interrupt based receiving method, with a simple state machine implemented in software inside the interrupt service routine. This state machine is shown in Figure 21.



Waiting for data
If timeout occurs during state = Recieving the buffer is cleared and the state is reset to Waiting for package. A separate timer is used to count the timeout variable
Byte received
Switch(state)
Case: Waiting for package
Case: Recieving
If (data == Start)
If (data == Stop)
No
Put byte in buffer
Yes
State = Recieving
Make data packet from buffer data
Put data packet in queue
State = Waiting for package

Figure 21 - State machine handling UART packet reception.

6.6 Graphical user interface

One of the main purposes of this project was to build a vehicle that would be controlled only with natural movements. That may be possible during the riding itself, but before and after some extra control are required. Also in many situations, the rider may need information about the current status of the vehicle. Like how much battery power is left, how close the motors control signals are to saturation, how fast the vehicle is travelling etc.

6.6.1 Graphical user interface design

The most important thing of this system during design phase was to allow for communication in both directions between rider and vehicle. To enable a good visual interface and to enable input to the hardware of the vehicle, a touch screen was chosen as the solution.

Design goals:

- Allow for communications in both direction between rider and vehicle.
- Easily viewable during riding.
- Show all signals from the vehicle that may be vital for rider safety.
- Allow the rider to change control parameters without reprogramming the main processor.

6.6.2 Graphical user interface implementation

To implement this system and achieve the goals in a quick and easy way a smartphone, namely an Iphone 4, was selected as the basis of this system. Using a special accessory called Redpark Serial Cable the Iphone is provided with a RS-232 serial communications bus. The RS-232 bus was then converted to work with the UART of the main processor system with the help of a level converter attached to the cable.

The smartphone provides a display with touch screen capability as well as playback of sound, 3G internet connection, GPS sensor and much more. The software was written as an App with three tabs. The first tab is the dashboard tab where the rider can see information sent from the vehicle in real time. Signals such as battery status, current draw, pitch angle, pitch angular velocity, speed, total driven distance, motor controls signals and more (see Figure 22) are updated every 50 ms.

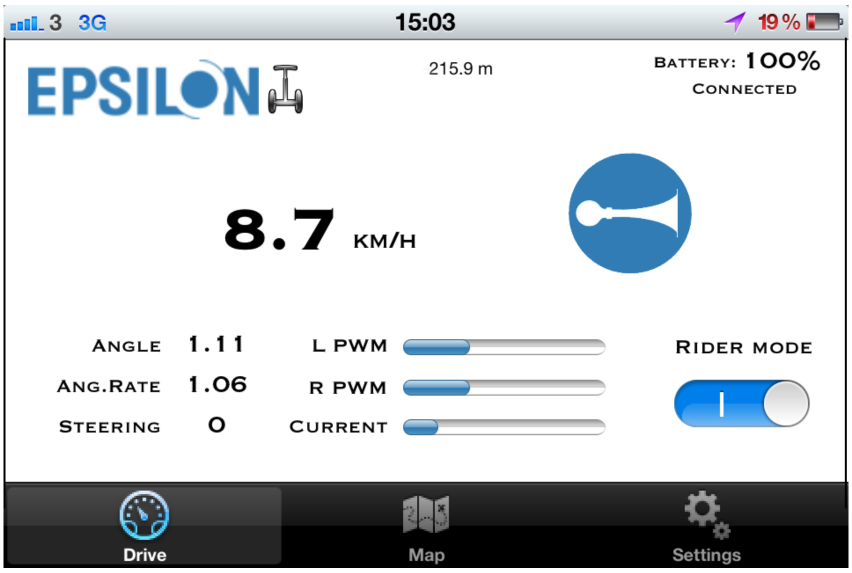


Figure 22 – The drivers tab, which updates every 50 ms with new information from the vehicle. 'Rider mode' is a switch that activates the controller.

Another tab is the settings tab (see Figure 23), where the user can enter information like weight and height to automatically retune the controller for different riders (more on this in Section 7.6). From this tab it is also possible to load the controller settings currently used in the control loop, fine tune and easily update them. It is also possible to set a friction compensation value which is added to the steering signal to overcome the friction of the motors. Finally there are options for turning on and off the motors and the debug mode, where all measured signals are sent to a PC running MATLAB.

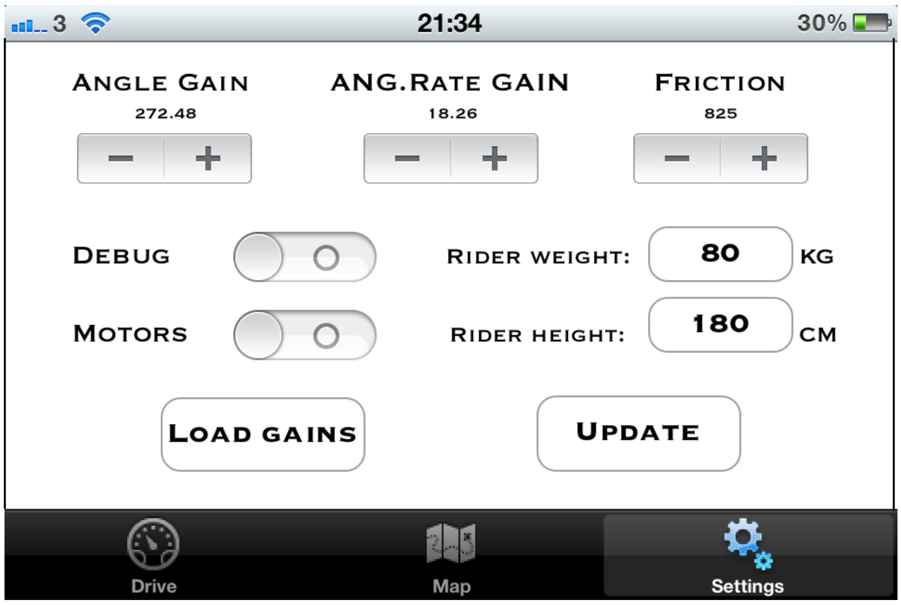


Figure 23 - The settings tab which may be used to read and update control parameters as well as turning on and off the debug mode and motors.

The final tab is a navigation tab that displays the vehicle on a map. The rider can zoom and pan around the map, and there are choices for the map to dynamically track the vehicle, with or without automatic aligning with magnetic north. Finally, this navigational system also includes a driving direction mode where the rider can enter an address in any city and get driving directions to this address. When the rider enters an address the app uses the 3G connection to make a request to Google Directions, which responds with a JSON object containing directions and polylines for the current directions. This object is then parsed with a JSON parser and the polylines are decoded and drawn as a blue line on the map (see Figure 24):

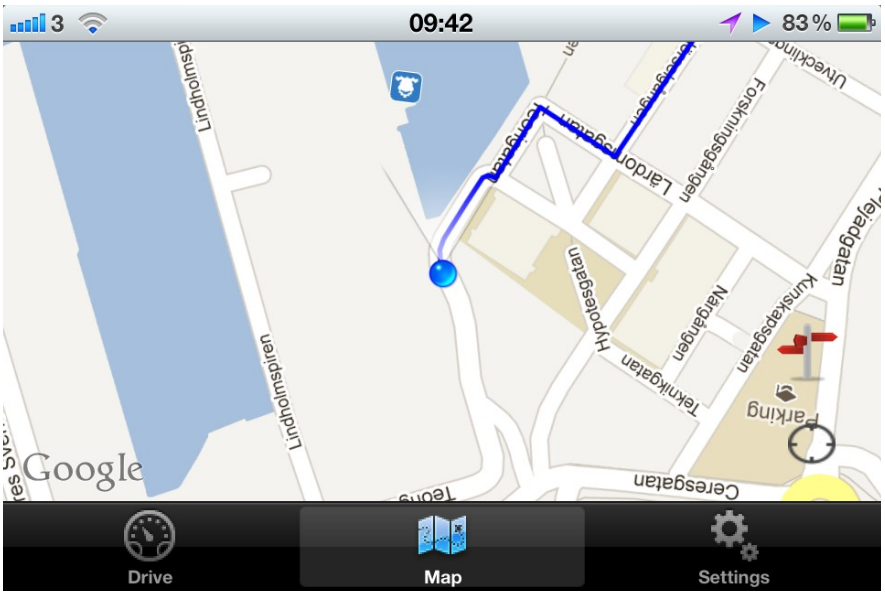


Figure 24 - The Map tab showing the vehicle as a blue dot and showing a blue line for directions towards the destination.

6.7 Pitch angle and angular rate measuring system

As described in Section 2.3, an accelerometer can be combined with a gyro and sensor fusion algorithms to obtain the pitch angle and the angular rate of an object. Both these signals are of high importance when designing this vehicle, since they are required as inputs to the control system.

6.7.1 Pitch angle and angular rate measuring system design

The design goals of this subsystem was to; attain as good measurement as possible of the pitch angle and angular rate with a frequency at least as high as the control loop sampling frequency. Formally:

- Deliver measurements of pitch angle and pitch angular rate taken at 100 Hz update interval.

6.7.2 Sensor selection and implementation

The chosen sensor system is based on InvenSense MPU-6050 which is a 6-axis MEMS device with a 3-axis accelerometer and a 3-axis gyroscope integrated into one IC. This IC has a 400kHz I^2C interface, measures only 4x4 mm and also incorporates a temperature sensor and a digital motion processor (DMP^{TM}) for a cost of 15$ at the time of writing.

The motivates for selecting this sensor were the low cost, the simple communication interface and the ability to access sensor fusion data directly from the built-in DMP^{TM}. This saves valuable processor time on the main processor of the vehicle and removes the need to implement the filtering algorithms (see Figure 25).

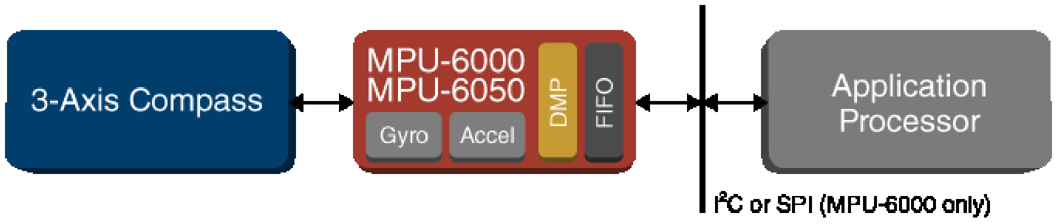


Figure 25 - Diagram over MPU-6050 integration with application processor. The compass is optional (image from www.invensense.com, retrieved 9th May 2012).

However to implement and use this sensor and the DMP^{TM} a large and complex software structure called "InvenSenseMotionApps^{TM} Platform" (shortend MP) needs to be ported and compiled for every distinct implementation. The MP is delivered as an AVR Studio 5 project targeting the AVR A3-UC3 Xplained development board, featuring the AT32UC3A3256 32-bit microprocessor. This means that the problem of filtering and filter implementation commonly associated with combining accelerometer and gyro sensors is replaced by a software engineering problem, consisting of porting the MP project to the desired microprocessor.

The reason for this software platform is according to InvenSense protection of their algorithms and intellectual property. The datasheet (12) of this product contains no information on the DMP^{TM} registers, making it impossible to access the sensor fusion output in a traditional way. Instead the MP is used as a way of encapsulating and hiding the inner workings of the sensor. What this practically means is that the DMP^{TM} code is only available as embedded HEX-data and the MP is used to upload the code to the DMP^{TM} over the I^{2}C bus each power cycle.

6.8 Wheel sensor system

This system utilizes two encoders, one mounted on the motor axis of each motor and a circuit board with a processor dedicated to decoding the encoder signals. It sends the decoded information to the main processor over serial communication.

6.8.1 Wheel sensor design description

The wheel sensor system counts the pulses from the quadrature encoders and sends the information about the number of pulses on a UART communication channel to the main processor.

Design goals:

- Handle 12000 pulses per motor revolution.
- Keep track of the rotational direction of the wheel.
- Output the data through a UART communication channel.

6.8.2 Wheel sensor implementation

The wheel sensor system consists mainly of an Atmel XMEGA32A4 processor. This processor has been chosen because of its built in event system. The event system enables inter-peripheral communication which is used to count the pulses from the quadrature encoders. The event system does not use any CPU, which makes it very efficient and powerful. A counter is connected to a data register with the event system. The register value increases or decreases when the encoder is turning clockwise or counter clockwise.

When the XMEGA microcontroller gets a UART package from the main processor, it responds by sending the number of pulses that has been counted since last call. The board is fed by a 5 Volt source from the main processor board. The 5 Volt source is then converted to a 3.3 Volt source by a linear regulator on the wheel sensor board. Some LEDs has also been added to the board to simplify debugging.

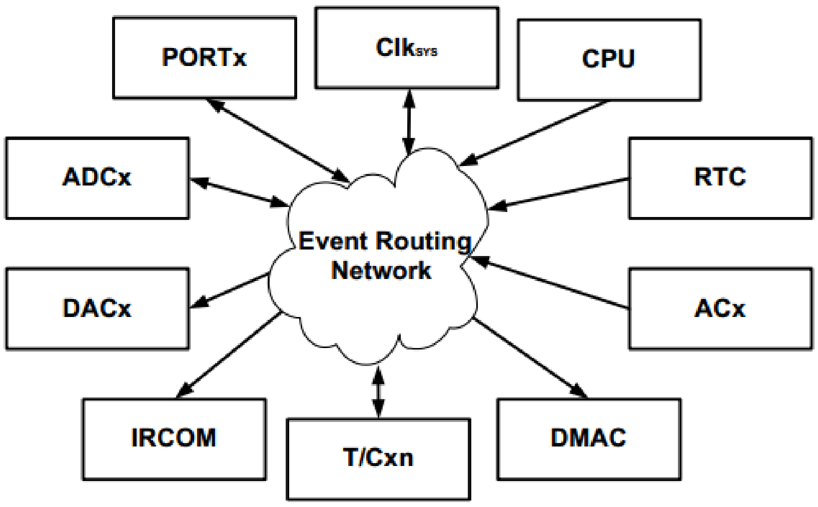


Figure 26 - Event system on the XMEGA microcontroller (image from www.atmel.com, retrieved 10th June 2012).

The quadrature encoders connected to the wheel sensor system are mounted directly on the motor axle, one for each motor. These sensors outputs 12000 pulses per motor revolution and 168000 pulses per wheel revolution, since the motor has a 14:1 gearbox.

7 Results

This chapter presents the results of the project. First comes a short discussion on the implementation of the controller and the steering algorithm. This is followed by some initial driving results and why the vehicle did not behave in a satisfactory manner and what was done to improve the performance. The chapter ends with the final driving results and an investigation of how controller safety is affected by riders of different weight and height.

7.1 Implementation adjustments to the controller

Before performing the first test runs the controllers' L-vector was slightly modified to:

$$\boldsymbol{L} = [0 \ \ -272.4750 \ \ \ 0 \ \ -18.2560] \qquad [49]$$

The gains on the wheel angle and wheel angular rate were set to zero such that the controller would not directly have any influence on wheel angle or speed.

It was also realized that the models static friction coefficient was probably insufficient, since a DC-motor often requires a much larger voltage to start rotating than it needs to keep this rotation. To handle this, a constant friction compensation was always added to the control signal PWM, see Figure 27.

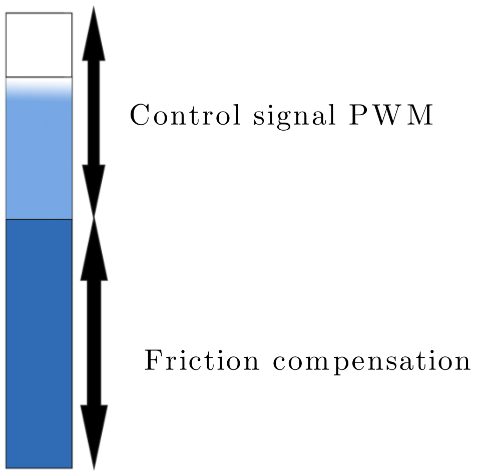


Figure 27 - Friction compensation strategy, the total control signal is made up of two parts; the constant friction compensation and the actual control signal.

To be able to steer the vehicle an open loop steering law was implemented. The idea behind it was to increase the power to one of the motors and decrease the power to the other motor. This way, the vehicle can easily rotate around its own axis, giving it high manoeuvrability in tight areas. To lower the risk of tipping when riding at high speed, the effect of the steering signal was made inversely proportional to the average vehicle speed. The final expression for the turning law became:

$$V_{turning} = \frac{1}{4}\frac{Angle_{steering}}{(1 + 0{,}7 * CurrentSpeed)} \qquad [50]$$

where $V_{turning}$ is the voltage being added and subtracted respectively from the motors, $Angle_{steering}$ is the tilt angle of the handlebar and $CurrentSpeed$ is the average speed as measured by the wheel sensor system.

7.2 Initial driving results

When testing the vehicle for the first time the controller gains were significantly lowered as a precaution. The starting values were an angle gain of -130 and no angular rate gain at all. This test was mainly performed to see that the system functioned as intended, spinning the wheels in the right direction etc. After this was confirmed the first riding test was conducted, and already with this controller the vehicle was drivable. The control strategy with the remaining error serving as a source for speed turned out to be a success and speed control by forward or backwards leaning worked well.

However, the vehicle required a lot of balancing skills of the rider, and resistance when trying to lean forward or backward was weak. To remedy this problem the controller gains were slowly raised towards the design values. Unfortunately it turned out that increasing the magnitude of the angular rate gain to values higher than 3 resulted in heavy oscillations of the system, especially when riding over small bumps or uneven surfaces. The same phenomenon occurred when increasing the angle gain magnitude above 170.

The steer-by-leaning mechanism was then implemented and tested. Although it worked, the steering mechanism was heavily underperforming. When trying to ride straight there were some disturbances causing the vehicle to wiggle from left to right.

Both of these problems lead to an investigation of the system signals during operation. The results of this investigation are presented in Section 7.2 and 7.3 and the final driving results are presented in Section 7.5.

7.3 Angle and angular rate sensor signals

The sensor signals for the pitch angle and the angular rate are paramount to the success of controlling this vehicle. This section presents the results and signal quality of the implemented MPU-6050 inertial measurement unit (IMU).

7.3.1 Angle signal quality

The selected sensor has a lot of tuning possibilities. Through the Motion Processor built into the chip the user can select a wide range of operation characteristics, for example which filters and filter bandwidths to use, as well as the algorithms to estimate the gyro bias. Since the inner workings of these algorithms are hidden, there is no right or wrong answer to which algorithms to use for a particular case.

To get guidance in this process, a test rig for the IMU was constructed. This rig enabled rotational movement to be measured by both the IMU and a potentiometer simultaneously, sending the measured values in real time to MATLAB where the data could be plotted and analyzed. The potentiometer was used to give a reference of the actual movement, and different filters and algorithms were then tried on the IMU in attempt to make the IMU signal track the potentiometer as closely as possible.

Finally the following setup was chosen:

- Sampling rate of gyro and accelerometer: 1000 Hz
  Even though the control loop was planned to run in 100 Hz, an increased performance in terms of noise was seen when using oversampling of the raw sensor values. This is supported by (13), since the oversampling basically gives the same result as using a higher resolution ADC. The DMP^{TM} algorithms may also gain benefits from the oversampling and contribute to the performance difference.

- Cut off frequency for the digital low pass filter applied on the raw measurements: 42 Hz

- Bias tracking algorithms for the gyro:

  Since the implementation of these algorithms is hidden in the DMP^{TM}, the following information is based on (14):

  - Compensation based on factory temperature measurements
    This algorithm is based on a no motion heating test performed on the factory assembly line. The bias is tracked during motionless state while the IC is heated. The bias change with temperature is stored in the memory of the chip.

  - Compensation based on continuous temperature readings
    This algorithm is based on the same principle as the one above, but temperature measurements are taken continuously during operation when a no motion state is detected. When enough measurements are available the bias dependency on temperature is updated and stored in memory.

  - Compensation based on no motion
    Whenever a no motion state is detected this algorithm uses gyro and accelerometer data to track the bias.

  - Compensation based on low pass filtering
    This algorithm basically low pass filters the gyro output with an extremely low cut off frequency. The remaining signal is considered to be the drifting of the bias value.

  - Compensation based on gravity
    This algorithm uses information from all three accelerometer axes to detect states with no linear acceleration, and then considers the orientation of the device according to the accelerometer as the reference, tracking any bias change from the gyroscope that may occur during the same period.

These filters and algorithms seemed to provide the best tracking performance of the IMU compared to the potentiometer reference signal. Figure 28 shows data from a test run in the rig.

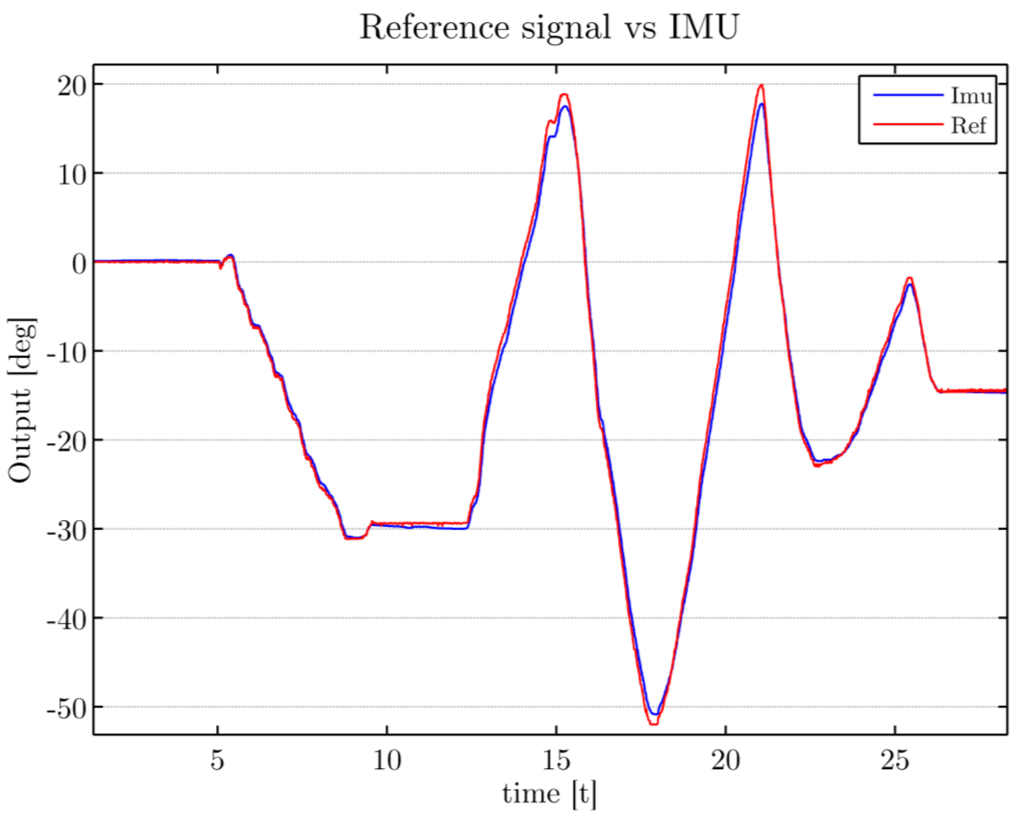


Figure 28 – Comparison of reference angle value from potentiometer and the angle measured by the IMU

It is clear that the filtering introduces very little phase shift, as the IMU responds very fast to rapid changes in tilt. However in some fast turning scenarios the IMU is not able to match the exact value of the potentiometer.

7.3.2 Angular rate estimation

The angle was estimated with the help of the built-in motion processor in the sensor but there is no support for noise reduction of the angular rate. The angular rate data is read directly from the gyro sensor after the motion processor has removed the estimated bias.

Initial driving results proved it difficult to use a feedback constant L_2 for the angular rate below values of -3 (the designed controller should have L_2 = -18,256). Any attempts to step towards the designed controller gain resulted in strong oscillations. After logging the system signals it was concluded that the sensor signal for the pitch angular rate was very noisy.

To reduce the effect of the noise a first order Butterworth IIR low pass filter was designed and implemented. This type of filter was selected since it is easy to implement in code, uses a low amount of computing power and has a steep transition region even for lower order filters (15). The filter cut off frequency was set by comparing the raw signal with different filtered versions until a satisfactory result was found. The aim was to capture the dynamics of the system but remove the higher frequency noise not related to the actual motion. This gave a cut-off frequency as low as 1 Hz. The transfer function of the selected filter was:

$$G_{filter}(z) = \frac{0.03047 + 0.03047z^{-1}}{1 - 0.9391z^{-1}} \qquad [51]$$

and Figure 29 shows the Bode plot for this transfer function.

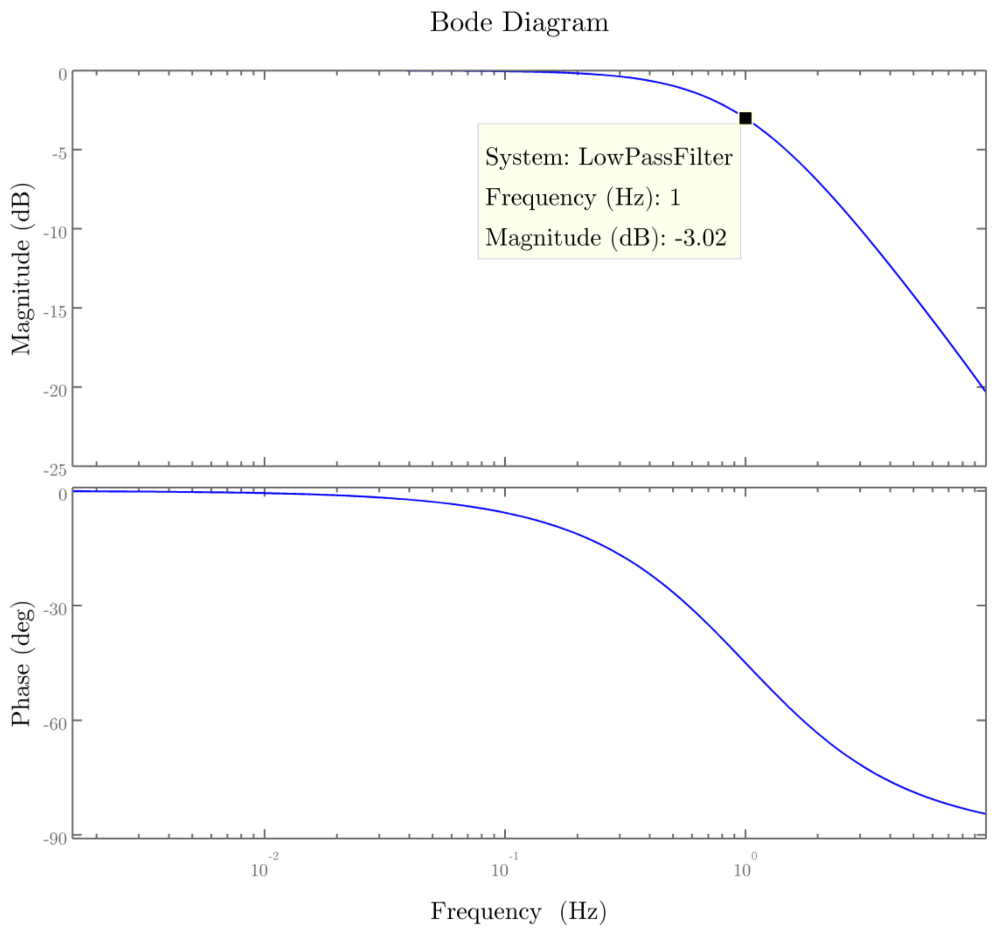


Figure 29 - Bode diagram of the 1st order Butterworth filter

In Figure 30 the raw gyro signal is compared to the filtered signal. One can see that there is some phase lag in the filtered signal compared to the unfiltered signal. The phase lag is small enough not to be noticeable when riding the vehicle.

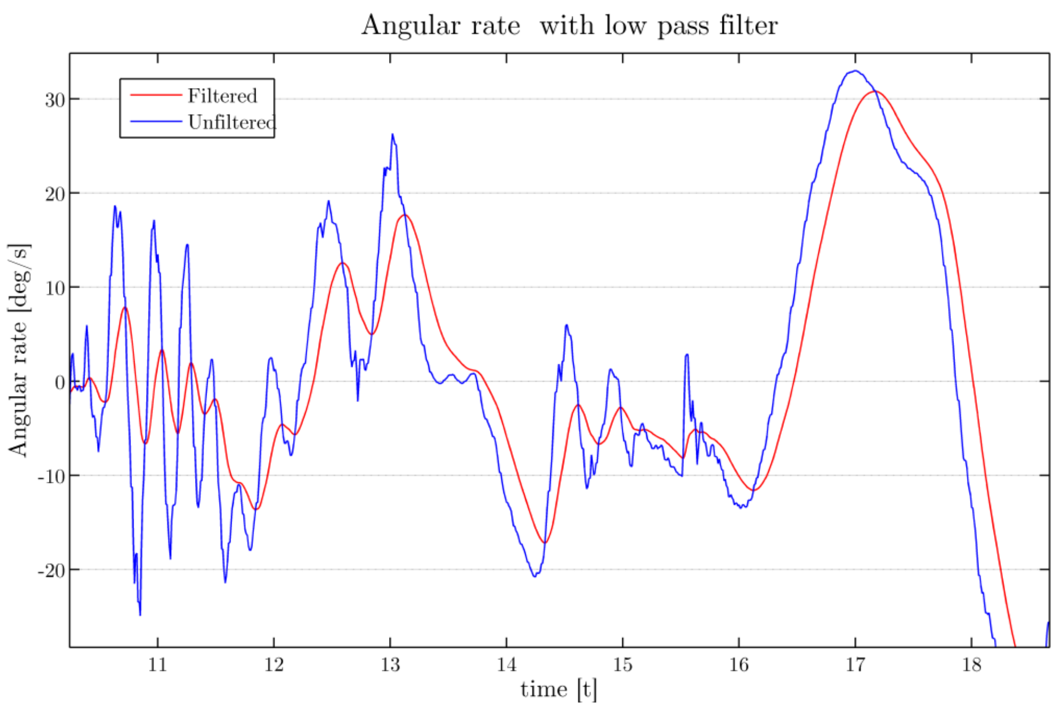


Figure 30 - Filtered and unfiltered angular rate

After implementing this filter and verifying its behaviour in the real system it was possible to set the angle and the angular rate gain to the designed values - 272,475 and -18,256 respectively. This made a huge impact on system performance and a much more stable riding experience was attained. The previous problems with oscillations when riding over small bumps was also significantly reduced.

7.4 Steering angle sensor signal

The second problem concerned the steering which made the vehicle wiggle. After investigation of some data logs from a test run the steering signal turned out to be very noisy (see Figure 31). The small curb of the graph at approximately eight seconds is an actual steering performed by the driver.

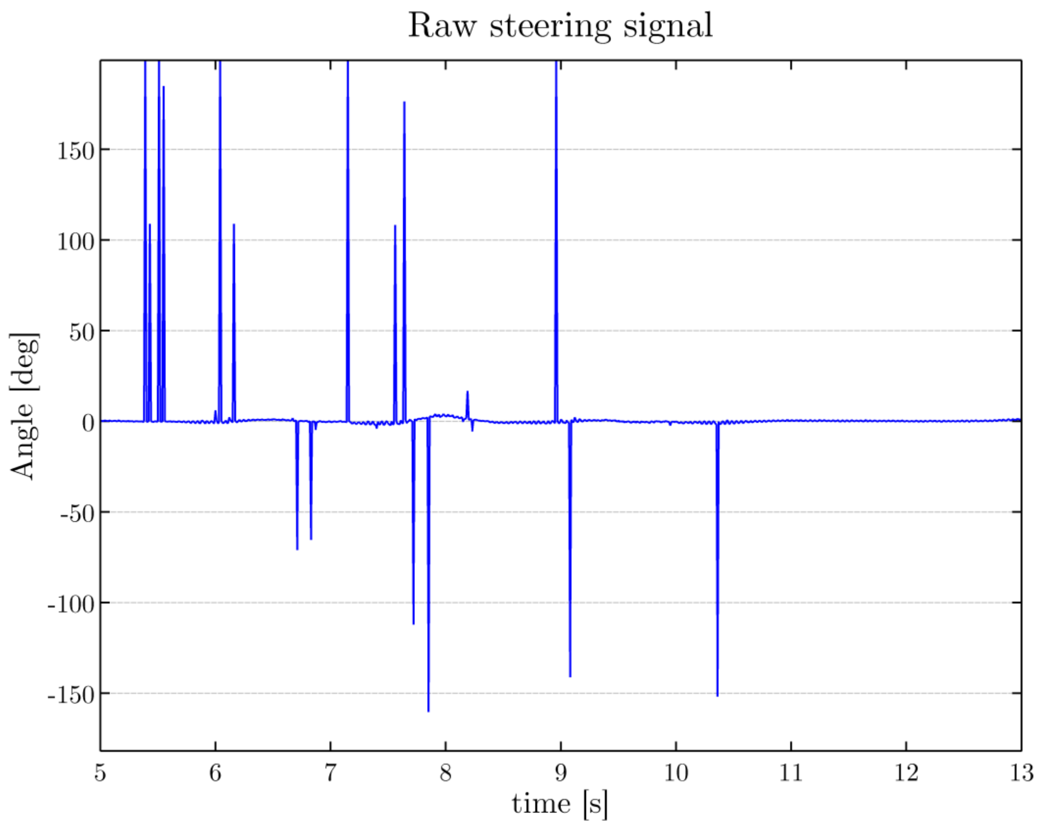


Figure 31 - Raw steering signal during a driving experiment

The outliers were considered to be pure errors and to reduce the effect of these errors the steering signal magnitudes which was greater than a certain threshold was set to the previous steering signal value. This reduced the outliers significantly but the signal was still very noisy. To solve this problem a low pass IIR filter was introduced. Since the expected frequency of the driver signal was judged to be similar to the angular rate dynamics, the same filter was used as for the angular rate filtering. This filter was described in the previous section. Figure 31 shows the steering signal after removing the outliers and the steering signal after removing outliers and adding low pass filtering. This removed the wiggling problems and made the steering feel responsive and correct.

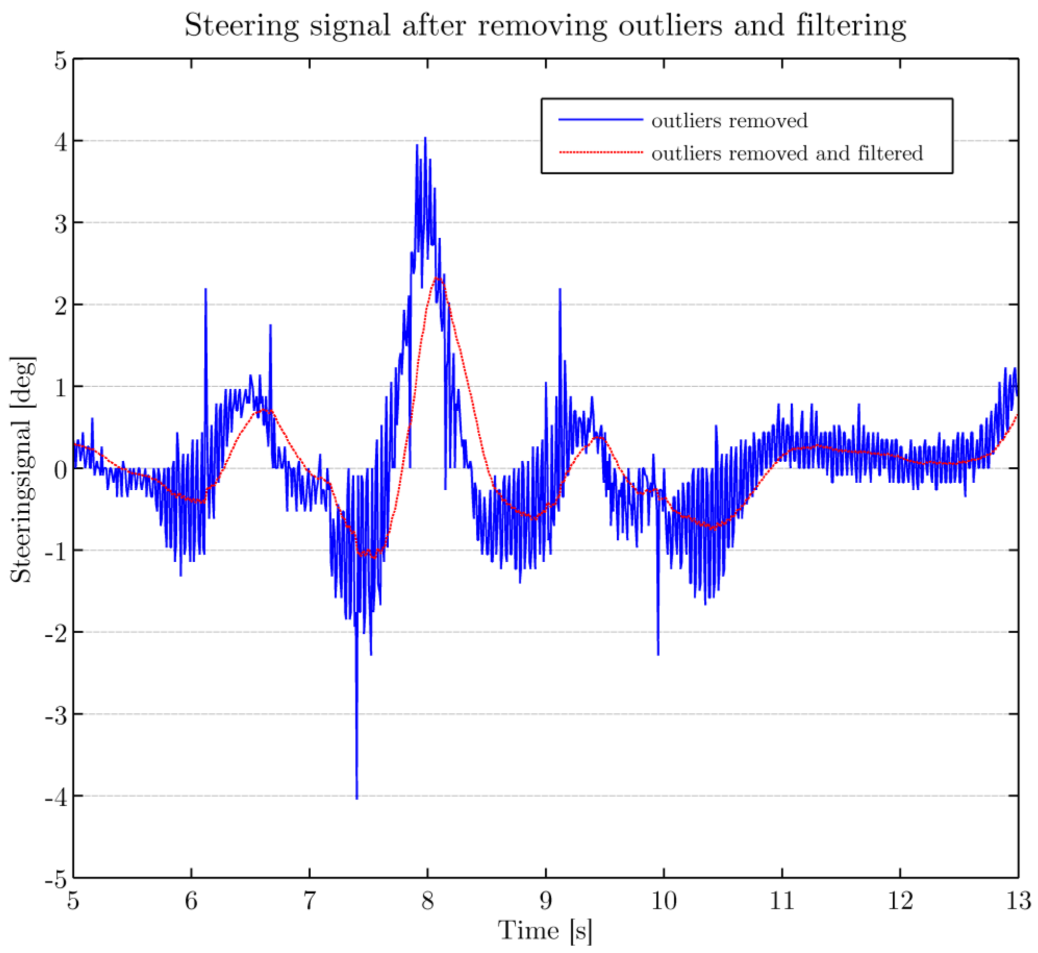


Figure 32 - Filtered and unfiltered steering signal with removed outliers

7.5 Final driving results

With the new filters implemented the vehicle performed significantly better. The designed controller was implemented which made the vehicle much more stable and easy to ride. The resistance when leaning forwards or backwards became much greater and less balancing from the rider was needed. Now it was also possible to ride over small bumps without inducing heavy oscillations.

7.6 Investigation of current drawn from battery

The current drawn from the battery was logged to monitor the size of the currents conducted in the cabling. First, a stress test was made, where the vehicle was driven forward and backward in an oscillating manner to maximize the currents. The current drawn from the battery can be seen in Figure 33 with the highest peak of 123 A.

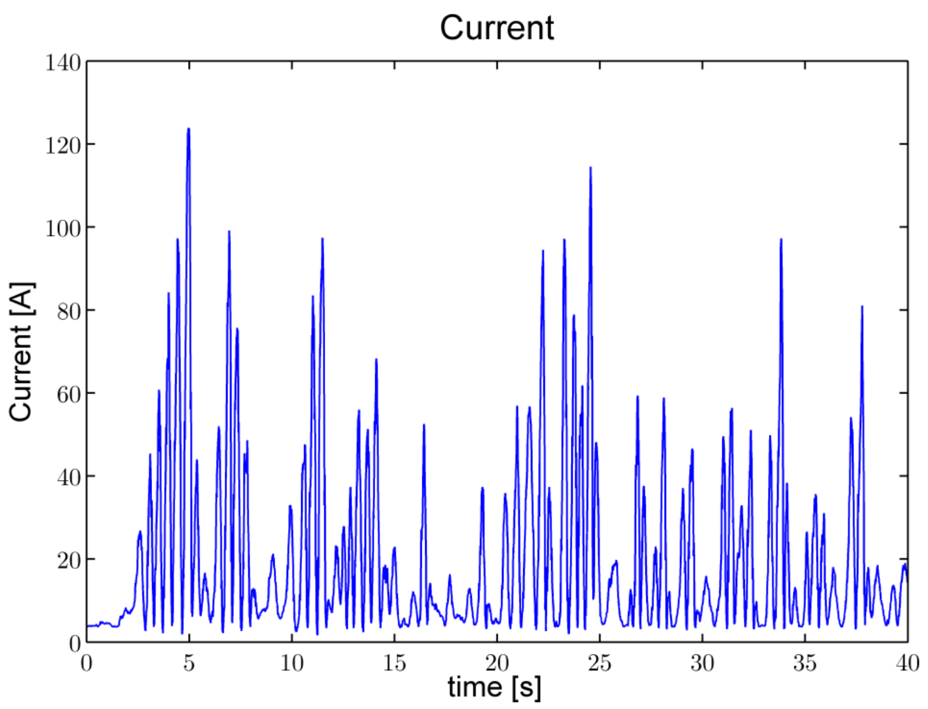


Figure 33 - Current drawn from battery during a stress test.

To estimate the driving range of the vehicle, a simple test was performed where the vehicle was driven at approximately 10 km/h in 30 seconds. The test run also includes a start and a stop, which can be seen in Figure 34. The average current drawn from the battery was 10.1583 A and the mean speed was 9.2347 km/h. Since the battery has a capacity of 20 Ah, this would mean that the vehicle should be possible to ride for approximately two hours. The driving range will then be almost 20 kilometres.

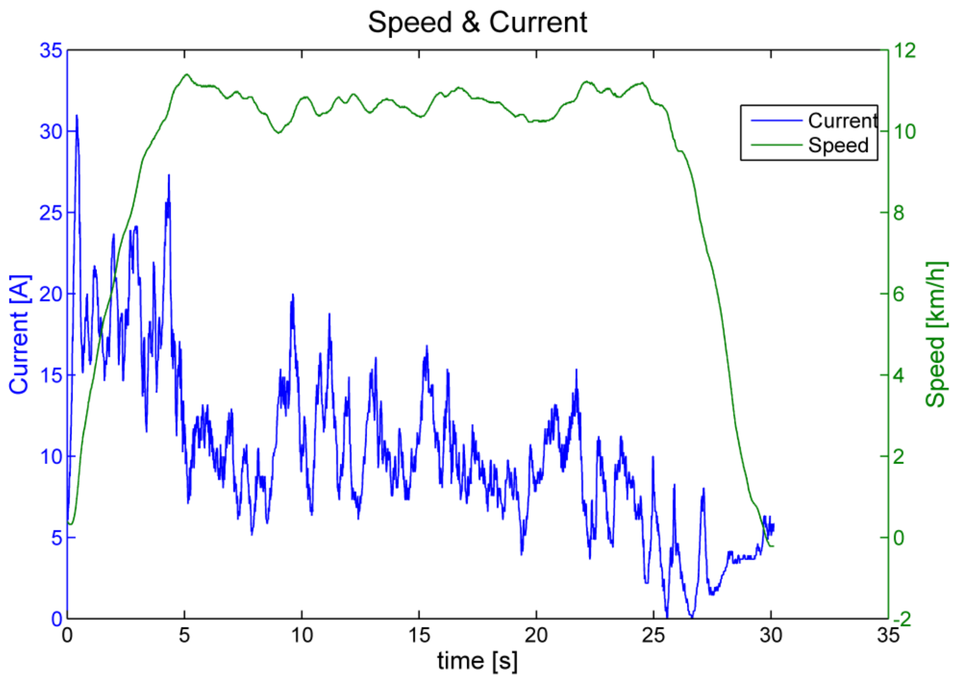


Figure 34 - Driving range test of the vehicle.

7.7 Investigation of controller performance for riders of different weight and height

One of the project goals was to investigate the system response for riders of different weight and height than the controller had been designed for. Simulations were therefore carried out for various combinations of weights and heights using a controller tuned for a rider weighing 80 kg and being 1.8 meters tall. The weights accounted for was selected between 60 and 100 kg and the heights between 1.6 and 1.9 meters.

7.7.1 Simulations and comparison

The simulations clearly show that the vehicle is able to balance and regain a zero degrees pitch angle for all tested combinations of weight and height. Figures 35 to 37 show the response of the controller with respect to pitch angle and angular rate for riders at the limits of the investigated weights and heights. The initial condition was a system in rest with a starting pitch angle of -6 degrees.

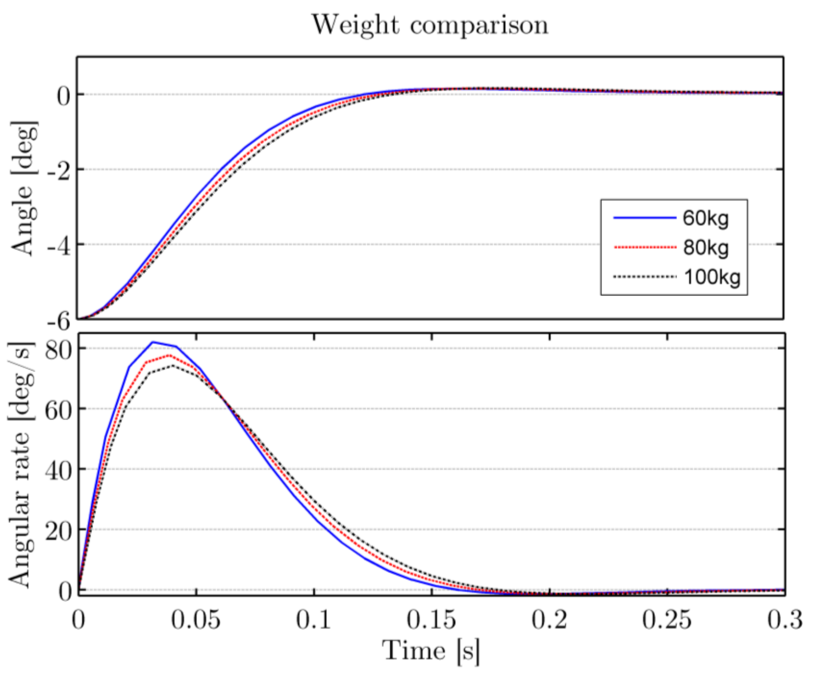


Figure 35 – Controller response for riders of different weights.

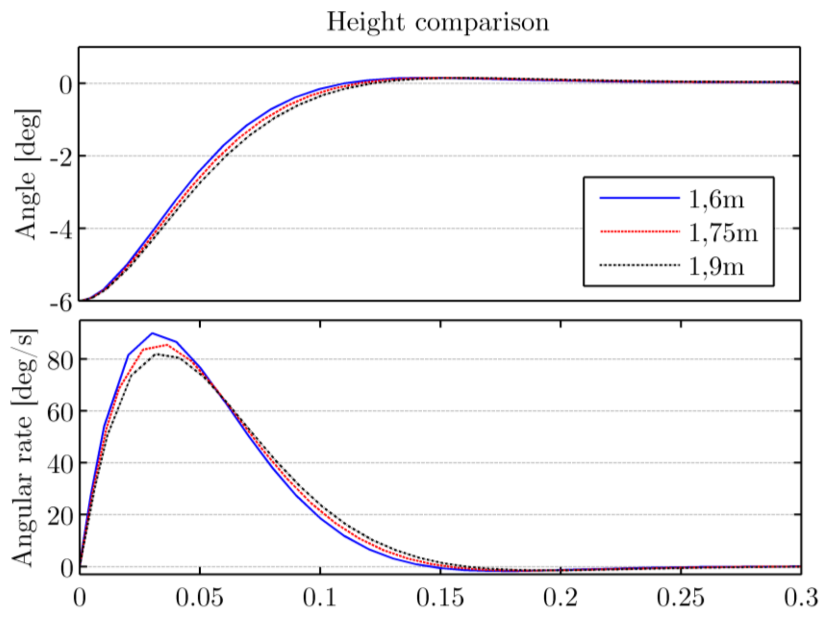


Figure 36 - Controller response for riders of different heights.

The time constant of the system increases with both weight and height, but the difference is small. The next figure shows the system response for two riders at the end points of the investigated interval:

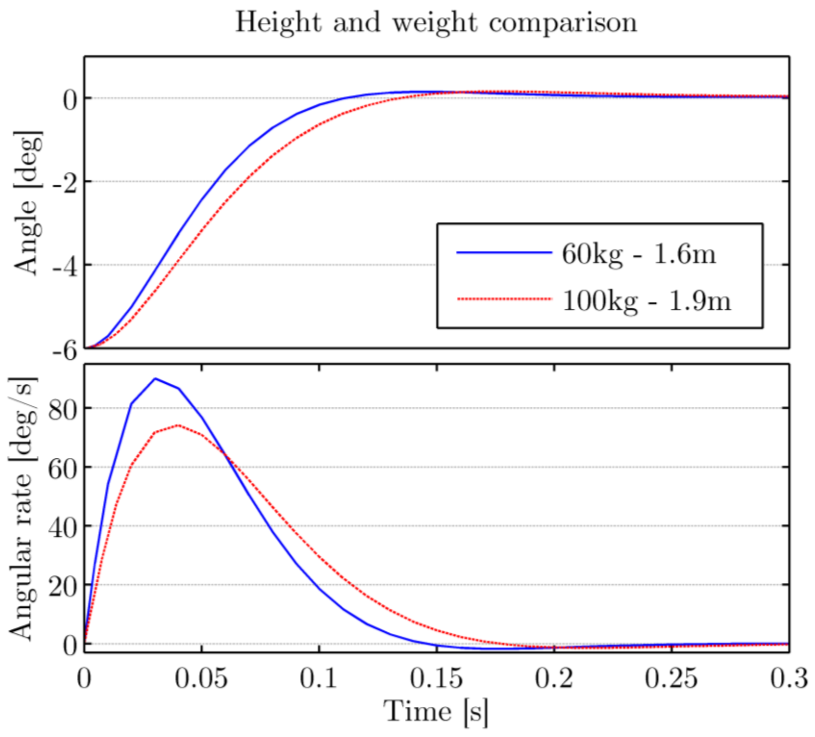


Figure 37 - Controller response for riders at the endpoints of the investigated intervals.

It is possible to think that an experienced rider demanding high performance may feel this difference. To eliminate the different response for riders of varying weight and height, the LQ-controller gains could be recomputed for each individual rider. This makes the model exhibit a similar system response for every rider.

7.7.2 Implementation on real system

The same project goal also stated that the real system should allow for the rider to select a controller tuned for a specific weight and height. This goal was not fulfilled for all weights and heights investigated in the previous section. Because of timing constraints, solving the Riccati equations necessary for designing an LQR-controller in the actual system was never implemented. Instead a few controllers for different weights and heights in the investigated interval were selected and computed offline with MATLAB. These controllers where then implemented in the user interface processor and made selectable from the touch screen. This way the controller can be retuned approximately for each rider.

The result of this controller gain computation can be seen in the following table:

	60kg	65kg	70kg	75kg	80kg	85kg	90kg	95kg	100kg
1.60m	-265.33 -15.44	-266.22 -15.74	-267.03 -16.01	-267.79 -16.27	-268.48 -16.52	-269.14 -16.76	-269.75 -16.98	-270.33 -17.19	-270.88 -17.40
1.65m	-266.43 -15.85	-267.31 -16.15	-268.12 -16.44	-268.86 -16.71	-269.54 -16.96	-270.19 -17.20	-270.79 -17.43	-271.36 -17.64	-271.90 -17.85
1.70m	-267.49 -16.26	-268.36 -16.57	-269.15 -16.86	-269.88 -17.13	-270.56 -17.39	-271.20 -17.64	-271.79 -17.87	-272.35 -18.09	-272.89 -18.31
1.75m	-268.51 -16.67	-269.37 -16.99	-270.15 -17.28	-270.87 -17.56	-271.54 -17.82	-272.16 -18.07	-272.75 -18.31	-273.30 -18.54	-273.83 -18.76
1.80m	-269.49 -17.08	-270.34 -17.40	-271.11 -17.70	-271.82 -17.99	-272.48 -18.26	-273.09 -18.51	-273.67 -18.75	-274.22 -18.99	-274.73 -19.21
1.85m	-270.44 -17.48	-271.27 -17.81	-272.03 -18.12	-272.73 -18.41	-273.38 -18.69	-273.98 -18.95	-274.56 -19.19	-275.09 -19.43	-275.61 -19.66
1.90m	-271.34 -17.89	-272.16 -18.23	-272.91 -18.54	-273.60 -18.84	-274.25 -19.11	-274.84 -19.38	-275.41 -19.63	-275.94 -19.87	-276.44 -20.10

Table 4 - Computed gains for riders of different weight and height

8 Discussion

This chapter presents the authors personal opinions on the projects' accomplishments. Also things that could have worked better and things that might be interesting to develop further are considered.

8.1 Vehicle performance

The results in Chapter 7 show that the vehicle performs in a sufficiently good manner. The vehicle can handle different types of drivers with different types of length and height. After a large number of test drives it as concluded that almost everyone can ride the vehicle with confidence after practicing for 5 minutes. Reducing the noise of the gyro signal had a large impact in the driving experience. It was very sensitive to an increase of the control action depending on the gyro signal when the filter was not implemented. The fact that the IMU is mounted directly to the chassis makes it very exposed to vibrations which may corrupt the signals. This may be improved if mounted on vibration repressive stands.

Some problems were noted with the employed turning strategy. Since power is added to one motor and removed from the other, the stability of the vehicle is affected. Practically this leads to the vehicle slightly tipping forward when turning, which feels uncomfortable and often causes the rider to compensate by shifting the weight further back.

A problem when riding outdoors is the stiffness of the standing platform. The powertrain is bolted on the chassis which is made up of relatively thick and strong aluminium sheet metal. The only real damping comes from the air in the tyres, which feels insufficient when riding on anything but smooth tarmac. Even though the vehicle is capable of handling the bumps without the control signals going crazy, it does not feel good and leads to tired feet and legs after a while.

Finally, there exists a problem in the electrical system when the vehicle is exposed to highly varying manoeuvres. If the rider is shifting the weight forwards and backwards on purpose to stress the control system, very large currents flows through the system. At a number of occasions this has led to the main processing microcontroller to reboot. As a safety precaution the motor drivers are always turned off as a part of the microcontroller start up sequence, which of course makes the vehicle tip over. A probable cause is that the high currents lead to an under-voltage condition on the microcontroller supply pins, which possibly could be solved by adding more capacitors to support the power supply.

8.2 System model and simulations

Further development of the mathematical model would increase the reliability and the practical use of simulations. It is possible that a lot of insight could be obtained if the rider actions could be modelled as well. Other improvements could be made in the area of frictions and viscous damping in the model. The friction is now modelled as a pure friction constant in the motor and the gearbox. It would probably be better to use a coulomb model of the friction, taking into account that the friction is smaller if the motors are turning than if they are standing still.

The effect of wind resistance has not been modelled and may add some errors to the model at higher speeds.

8.3 Future work

To enhance the functionality and performance of the vehicle there are some areas that could be developed further:

- Shielding of the steering sensor signal and/or replace the sensor with a less disturbance sensitive sensor.
- Cushioning of the IMU to reduce the impact of disturbances.
- Add communication to a web server for logging of data.
- Bluetooth communication for remote control and remote logging of data.
- Employing a new remote control mode for the vehicle without rider.

9 Conclusion

The main goal of this project was to build a functional two wheeled transporter and this goal has been fulfilled. The overall functionality and performance of the vehicle has been evaluated thoroughly by a number of test drives. The vehicle has been tested by a number of different people, with and without previous experience of riding this kind of vehicle. All were able to ride the vehicle.

9.1 Goal fulfilment

The vehicle has been able to drive riders weighing more than 100 kg, and the longest distance travelled so far has been 12 km. Calculations from measured current consumption suggests that the range on a single charge is approximately 20 km. The highest measured speed is 20.7 km/h, but the theoretical limit for the powertrain is around 34 km/h.

The electrical system allows for simple recharging and alerts the rider of low battery power by sounding an alarm. The re-charging is accomplished by connecting a DC-adapter between the charging plug and a wall socket. When a LED on the adapter switches from red to green, the battery is fully charged. However, the goal of allowing re-charging straight from the wall socket is not entirely fulfilled since a DC-adapter is needed between the vehicle and the socket.

The graphical user interface shows all required signals in real time and allows for on-line control parameter adjustments. The rider can easily enter weight and height and the control parameters will immediately adapt. However, each weight and height in the theoretically investigated interval described in Section 7.6 is not represented. Due to time constraints, solving of the Riccatti equations were never implemented in the system and therefore only the heights and weights in Table 4 are selectable.

Finally an LQ-controller has been successfully implemented on a microcontroller running FreeRTOS and the project budget was kept. For bill of materials including cost for every component, see Appendix D.

9.2 Additional conclusions

In reality the measurements of the system states can be unreliable or affected by noise. Thanks to a carefully tuned low pass filter it was in this case possible to use the gyro and steering signal even though they were very noisy. The low pass filtering of these signals had a great impact on the overall performance of the vehicle.

The aim when designing and building the vehicle has been a very robust vehicle. In every step of this project, robustness was considered a very vital part of the project. The chassis is sturdy and can withstand large external forces acting on it. The motor controller can withstand much larger currents than they are exposed to and so on. This turned out to be a resourceful strategy since we have had no failures of any subsystem in the vehicle due to under-dimensioning.

The Segway personal transporter has an update frequency of 1000 Hz. This frequency would require a much faster microcontroller and an update frequency of 100 Hz was used instead. Since the vehicle has slow dynamics, the lower frequency still seemed to be reasonable and after testing, it turned out to be sufficient.

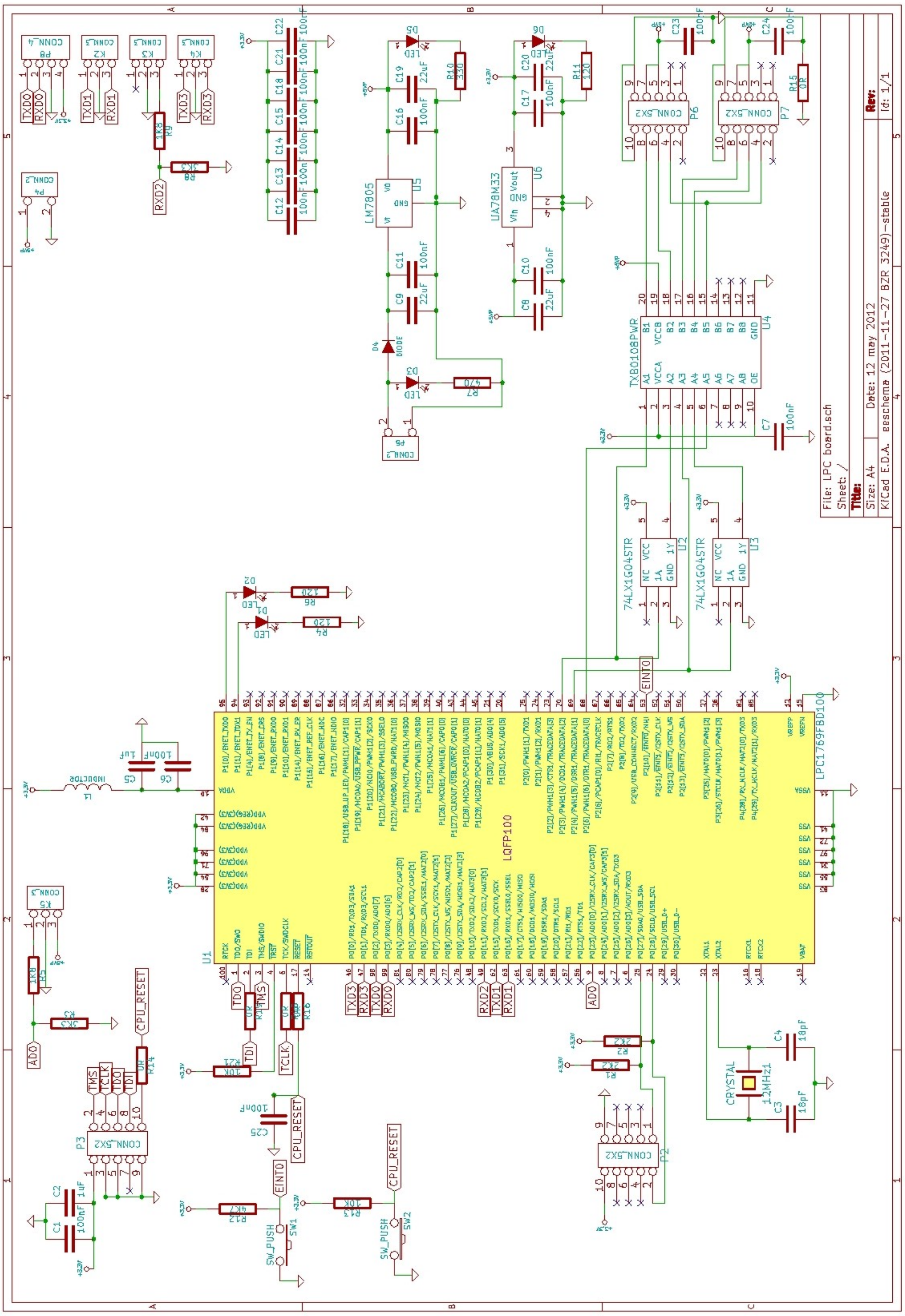
References

1. Kemper, Steve. *Reinventing the Wheel : A story of Genius, Innovation and Grand Ambition.* s.l. : HarperCollins, 2005. ISBN 0060761385.
2. Johannesson, Hans & Persson, Jan-Gunnar & Pettersson, Dennis. *Produktutveckling.* Stockholm : Liber, 2004.
3. Glad, Torkel & Ljung, Lennart. *Reglerteori.* Lund : Studentlitteratur, 2003.
4. Torby, Bruce J. *Advanced dynamics for engineers.* New York : Holt, Rinehart and Winson, 1984. ISBN 0-03-063366-4.
5. Lennartson, Bengt. *Reglerteknikens grunder.* Lund : Studentlitteratur, 2002.
6. Åström, Karl Johan & Wittenmark, Björn. *Computer Controlled Systems.* United States of America : Prentice Hall Inc., 1997. ISBN 0-13-314899-8.
7. Kempe, Volker. *Ineartial MEMS - Principles and Practice.* : Cambridge University Press, 2011.
8. Looney, Joseph Bergeron & Mark. Making MEMS accelerometers work in motion control. [Online] 2007. [Cited: 10 04 2012.] http://www.eetimes.com/design/analog-design/4170121/Making-MEMS-accelerometers-work-in-motion-control-item-1.
9. Randy Torrance, Chipworks. MEMS-based inertial sensor is not your grandfather's gyroscope. [Online] 2008. [Cited: 10 04 2012.] http://invensense.com/mems/gyro/documents/articles/chipworks.html.
10. Alciatore, G. David & Histand, B. Michael. *Inroduction to Mechatronics and Measurement systems.* : McGraw-Hill, 3rd edition, 2007. ISBN 0072963050.
11. Yamamoto, Yorihisa. NXTway-GS Model-Based Design - Control of self-balancing two-wheeled robot build with Lego Mindstorms NXT, Rev1.4. [Online] 2009. [Cited: 15 04 2012.] http://www.mathworks.com/matlabcentral/fileexchange/19147.
12. InvenSense Inc. MPU-6000 and MPU-6050 Product Specification Revision 3.2. [Online] 2011. [Cited: 16 04 2012.] http://invensense.com/mems/gyro/documents/PS-MPU-6000A.pdf.
13. Madapura, Jayanth Murthy. *Achieving Higher ADC Resolution Using Oversampling.* [Online] [Cited: 05 05 2012.] http://ww1.microchip.com/downloads/en/AppNotes/Achieving%20Higher%20ADC%20Resolution%20Using%20Oversampling%2001152A.pdf.
14. InvenSense Inc. Application Note: 9-axis MotionFusion, Calibration algorithms. [Online] 2011. [Cited: 15 04 2012.] http://www.invensense.com/developers/index.php?_r=downloads&ajax=dlfile&fi

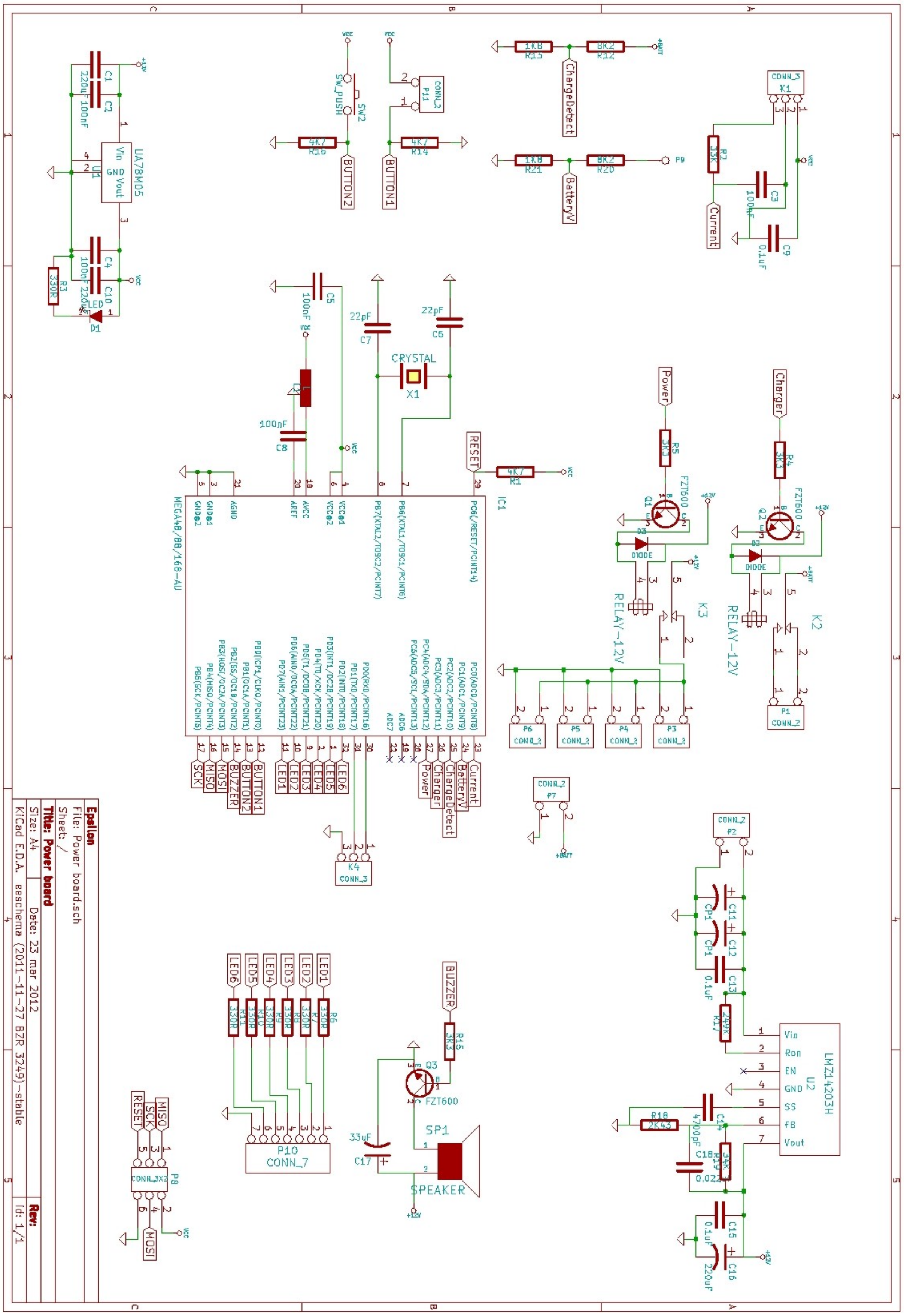
le=AppNote%20-%209-
Axis%20MotionFusion%20and%20Calibration%20Algorithms.pdf.
15. Mulgrew B., Grant P., Thompson J. *Digital Signal Processing - Concepts and applications.* : Palgrave McMillan 2nd ed., 2003.

Appendix A – Electrical schematics

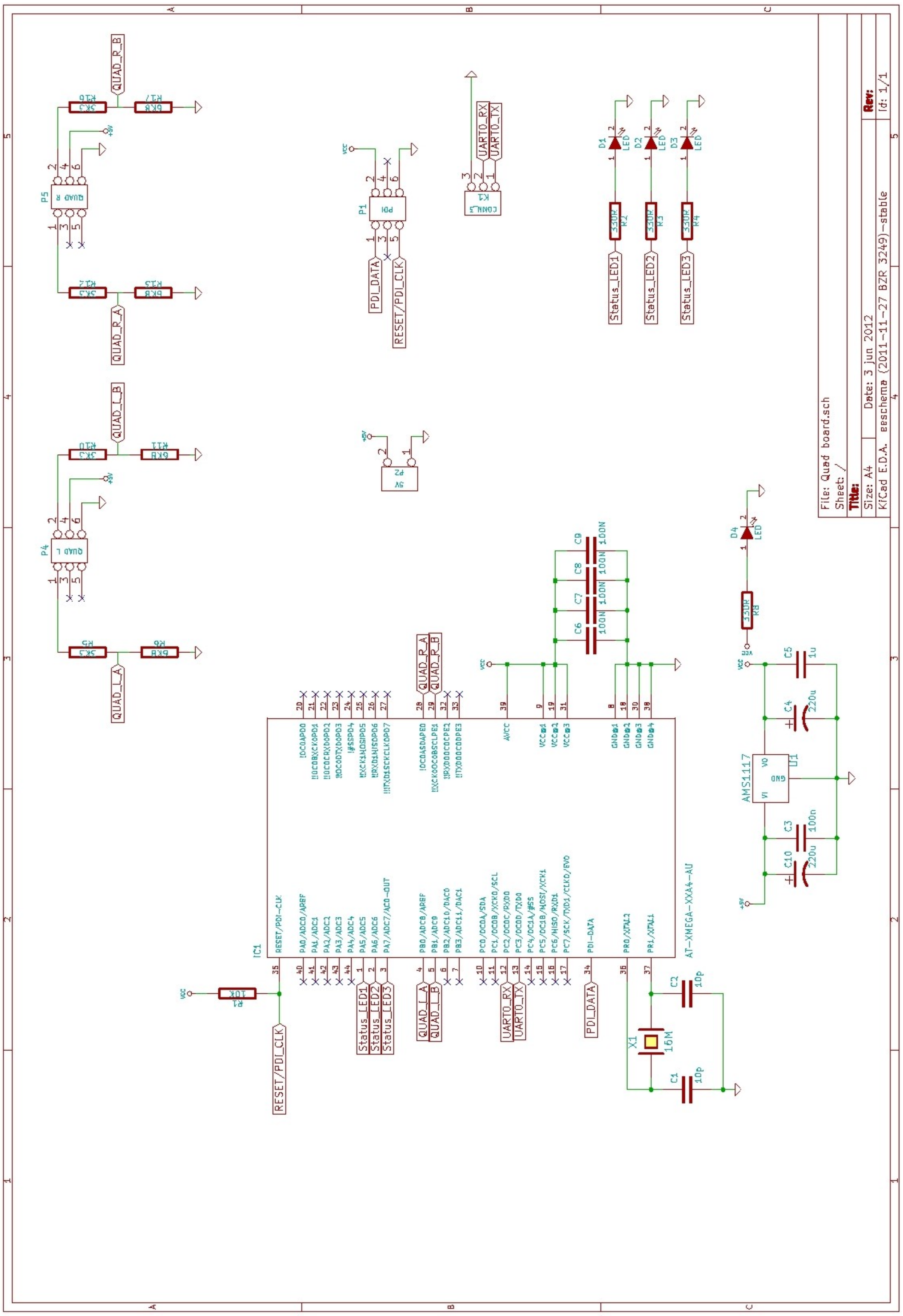
Main processor board:



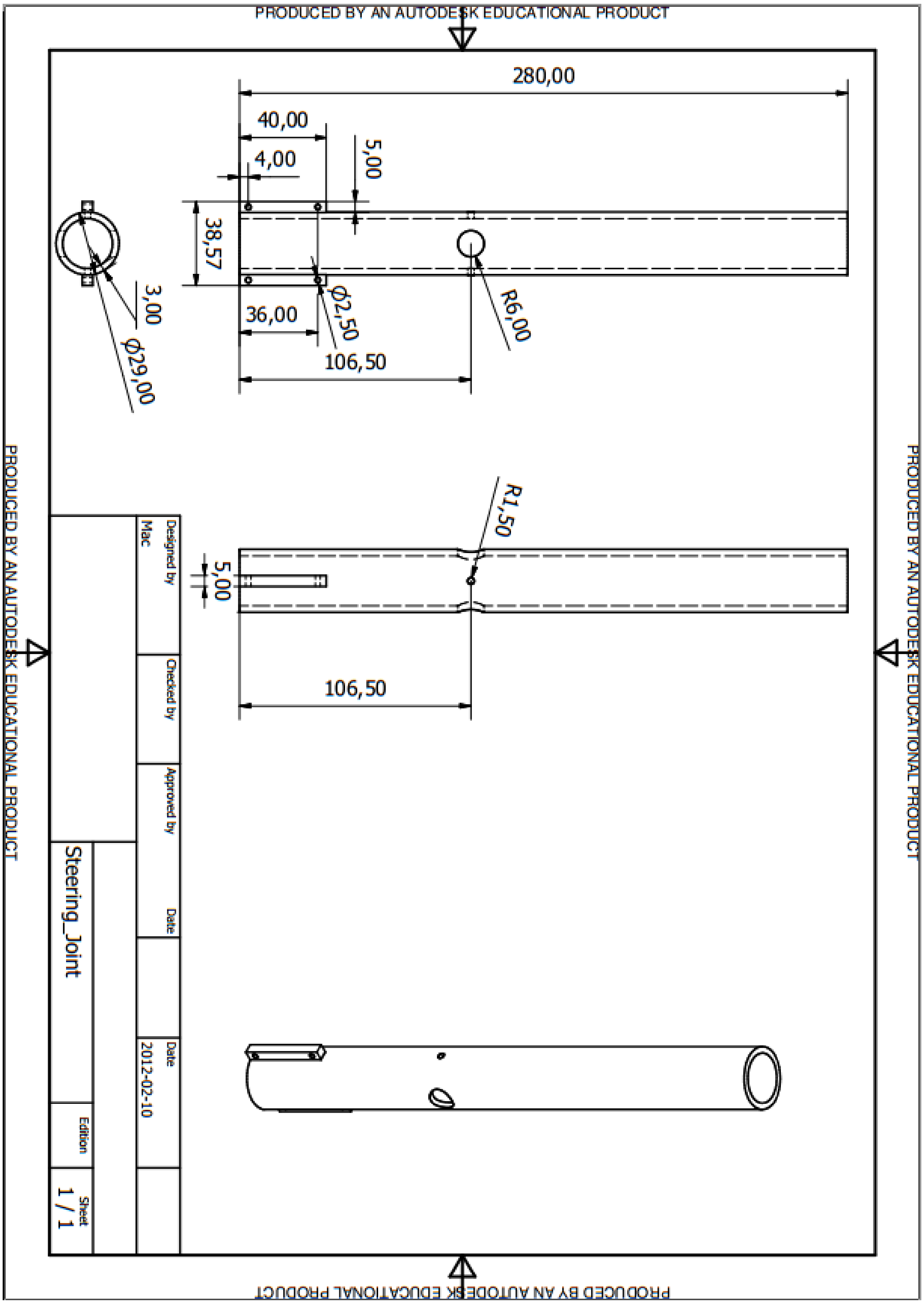
Power board:

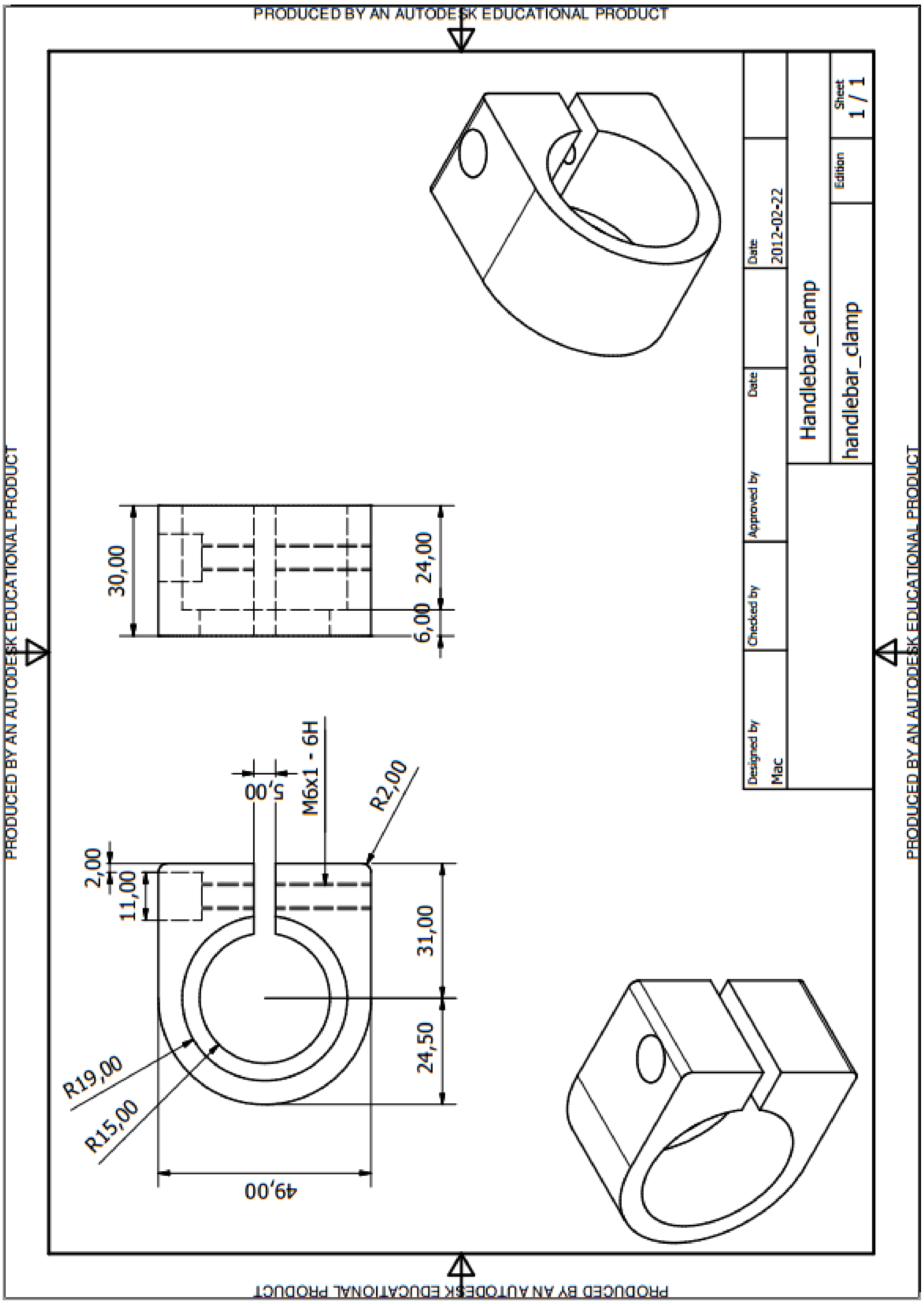

Epsilon
File: Power board.sch
Sheet: /
Title: Power board
Size: A4
Date: 23 mar 2012
KiCad E.D.A. eeschema (2011-11-27 BZR 3249)-stable
Rev:
Id: 1/1
MEGA48/88/168-AU
IC1
UA78M05
U1
LMZ14203H
U2
CRYSTAL
X1
RELAY-12V
K2
K3
SPEAKER
SP1
FZT600
Q1
Q2
Q3
CONN_7
P10
CONN_3X2
P8
BUTTON1
BUTTON2
ChargeDetect
BatteryV
Current
Power
Charger
BUZZER
RESET
LED1
LED2
LED3
LED4
LED5
LED6
MISO
MOSI
SCK

Wheel sensor board:



Appendix B – Selected mechanical drawings




PRODUCED BY AN AUTODESK EDUCATIONAL PRODUCT
49,00
R15,00
R19,00
24,50
31,00
11,00
2,00
5,00
M6x1 - 6H
R2,00
30,00
24,00
6,00
Designed by
Mac
Checked by
Approved by
Date
Date
2012-02-22
Handlebar_clamp
handlebar_clamp
Edition
Sheet
1 / 1

Appendix C - MATLAB simulation script

```
%-------------------------------------------------------------------------
%
%
% Project: Master thesis, Epsilon
% Mikael Arvidsson, Jonas Karlsson
%
% Requires simulink model 'complete_system'
%
%-------------------------------------------------------------------------

clear all
close all
clc

%% Program control

% Use this to control program execution:
SAVE_PLOTS = 0;
NONLINEAR_SELECT = 1;
ENABLE_CONTROLLER = 1;

%% Nonlinear model derivation for simulink

syms theta              % [rad]             pitch angle of the vehicle
syms thetadot
syms thetaddot
syms psi                % [rad]             angle of the wheel
syms psidot
syms psiddot

syms g;                 % [m/sec^2]         Gravity acceleration
syms m                  % [kg]              Wheel weight
syms R                  % [m]               Wheel radius
syms Jw                 % [kgm^2]           Wheel inertia moment
syms M
syms W                  % [m]               Body widht
syms D                  % [m]               Body depht
syms H                  % [m]               Body height
syms L                  % [m]               Distance of the center of mass from the wheel axle
syms Jb                 % [kgm^2]           Body pitch inertia moment
syms Jm                 % [kgm^2]           DC motor inertia moment
syms Rm                 % [ohm]             DC motor resistance
syms Kb                 % [V sec/rad]       DC motor back EMF constant
syms Kt                 % [Nm/A]            DC motor torque constant
syms n                  %                   Gear ratio
syms fm                 %                   Friction coefficient between body and DC motor
syms fw                 %                   Friction coefficient between wheel and floor
syms V                  % [V]               Short for both left and right voltage

syms a
a = (n*Kt)/Rm;
syms b
b = (n*Kt*Kb)/Rm + fm;

%Left side of the equation system
M11 = ((2*m +M)*R^2 + 2*Jw + 2*n^2*Jm);
```

```
M12 = (M*L*R*cos(psi) - 2*n^2*Jm);
M21 = M12;
M22 = (M*L^2 + Jb + 2*n^2*Jm);

Mtot = [M11 M12; M21 M22];

%Right side of the equation system
Rf1 = a*V - 2*(b + fw)*thetadot + 2*b*psidot +       M*L*R*psidot^2*sin(psi);
Rf2 = -a*V + 2*b*thetadot-2*b*psidot +        M*g*L*sin(psi);

%   Matrix structure
%   [M11 M12][thetaddot ]  =    [Rf1]
%   [M21 M22][psiddot   ]       [Rf2]
Rf = [Rf1;Rf2];
stateddot = (Mtot)\Rf;

Mrider = 80;                          % [kg]         Rider weight
Lrider = 1.8;                         % [m]          Rider height
g = 9.81;                             % [m/sec^2]    Gravity acceleration
m = 4.6;                              % [kg]         Wheel weight
R  = 0.2413;                          % [m]          Wheel radius
Jw = (m*R^2)/2;                          % [kgm^2]      Wheel inertia moment
Mv = 30.06;                           % [kg]         Vehicle body weight
M = Mv + Mrider;                         % [m]          Total body weight
H = -0.03;                            % [m]          Body height
L = (Mv*H/2+Lrider*0.55*Mrider)/M;  % [m]          Distance of the center of mass from the wheel axle
Jrider = Mrider/3*Lrider^2;           % [kgm^2]      Rider pitch inertia moment
Jb =  Jrider + 1.49;                  % [kgm^2]      Body pitch inertia moment
Jm = 1/4*4*0.05^2+1/12*4*0.12^2;    % [kgm^2]      DC motor inertia moment
Rm = 0.141;                           % [ohm]        DC motor resistance
Kb = 0.722;                           % [V sec/rad]  DC motor back EMF constant
Kt = 0.833;                           % [Nm/A]       DC motor torque constant
n = 14;                               %              Gear ratio
fm = 0.3;                             %              Friction coefficient between body and DC motor
fw = 0;                               %              Friction coefficient between wheels and ground

%% State vector and initial values
Xvect_str = {'Theta','Psi','dotTheta','dotPsi'};
color_str = {'b', 'r--', 'k-.'};
Theta_init = 0;
Psi_init = -6*pi/180;
dot_Theta_init = 0;
dot_Psi_init = 0;
X0 = [Theta_init Psi_init dot_Theta_init dot_Psi_init]';

%% Linearized state space model
alpha = (n*Kt)/Rm;
beta = (n*Kt*Kb)/Rm +fm;

E = [(2*m+M)*R^2+2*Jw+2*n^2*Jm , M*L*R-2*n^2*Jm;...
    M*L*R-2*n^2*Jm , M*L^2+Jb+2*n^2*Jm];

A1_32 = (-g*M*L*E(1,2))/det(E);
A1_42 = (g*M*L*E(1,1))/det(E);
A1_33 = (-2*((beta + fw)*E(2,2) + beta*E(1,2)))/det(E);
A1_43 = (2*((beta + fw)*E(1,2) + beta*E(1,1)))/det(E);
A1_34 = (2*beta*(E(2,2) + E(1,2)))/det(E);
A1_44 = (-2*beta*(E(1,1) + E(1,2)))/det(E);

B1_3  = (alpha*(E(2,2) + E(1,2)))/det(E);
B1_4  = (-alpha*(E(1,1) + E(1,2)))/det(E);
```

```
A1 =   [0        0        1        0;...
    0        0        0        1;...
    0        A1_32    A1_33    A1_34;...
    0        A1_42    A1_43    A1_44];

B1 =   [0, 0;...
    0, 0;...
    B1_3, B1_3;...
    B1_4, B1_4];

C1 = eye(4);
D1 = [0 0 0 0; 0 0 0 0]';

sys = ss(A1,B1,C1,D1);
dsys = c2d(sys,1e-2);

eig(dsys)

%% Find controller parameters
QQ = eye(4);
QQ(2,2) = 2e8;

RR = 1e3 * eye(2);

[KK,SS,E] = lqr(dsys, QQ, RR);
l_f = KK(1, 1:4)     % feedback gain

if ~ENABLE_CONTROLLER
    l_f = [0 0 0 0];
    l_i = 0;
end


%% Simulate
options = simset('FixedStep', 0.01);

% simulating the model
[time, states, output] = sim('complete_system', [0 0.5], options);

Xvect = {Theta, Psi, dotTheta, dotPsi};

%% Figures
% plotting options
width = 1.2;
size = 15;
title_size = 18;
radToDeg = 180/pi;

% Plot theta vs time

for i = 1:length(Xvect)
    figure(i)
    hold on
    if( ~ENABLE_CONTROLLER )
        handle = plot(time, Xvect{i}.signals.values(:,1)*radToDeg, time, Xvect{i}.signals.values(:,2)*radToDeg);
        legend('Nonlinear model', 'Linear model')
        axis([0 30 -360 360])
    elseif ( NONLINEAR_SELECT )
        handle = plot(time, Xvect{i}.signals.values(:,1)*radToDeg);
        legend('Nonlinear model')
    else
```

```
        handle = plot(time, Xvect{i}.signals.values(:,2)*radToDeg);
        legend('Linear model')
    end
    set(handle,'LineWidth',width);
    title(Xvect_str{i}, 'fontsize', title_size)
    xlabel('time [t]', 'fontsize', size)
    ylabel([Xvect_str{i} ' [deg]'], 'fontsize', size)
    grid on
end

%% Save plots to files
if ( SAVE_PLOTS )

    datestring = datestr(now,'dd-mmm-yyyy HH-MM-SS');
    mkdir('Plots',datestring);
    path = strcat('Plots/',datestring,'/');
    outputFileName = 'ploti';
    for i=1:length(Xvect_str)
        outputFileName= regexprep(outputFileName, 'i', Xvect_str{i});
        saveas(i, strcat(path, outputFileName), 'png');
        saveas(i, strcat(path, outputFileName),'fig');
        outputFileName=char('ploti');
    end

end
```

Appendix D – Bill of materials

Mechanics	Price in SEK excl. VAT
Wheels	3 342,50
Pillow blocks	175,00
Springs	140,00
Sand (used for pipe bending)	71,92
Main circuit breaker	55,92
Cable clips	47,84
Steel for wheel hubs	200,00
Aluminium	1 000,00
Polyoximethylene	105,00
Anodized aluminium profile	210,40
UNC-screws	246,40
Mixed screws	-
Electronics	
Motors	7 507,28
OSCM motor drivers	3 399,00
24V fans	140,00
Battery	2 909,94
Fuse holder	11,92
Red copper cable 5m 6mm2	39,92
Black copper cable 5m 6mm2	39,92
Redpark serial cable	555,00
Relay	100,00
Mixed electronic parts	3 000,00
Encoders	-
Iphone 4	-
Sum:	23 297,96