CHALMERS

Rollover Prevention by Rear Wheel steering and Rection Wheel for Unmanned Ground Vehicle

Master's Thesis in System Control And Mechatronics

Johan S Lövgren

Department of Signals & Systems
Chalmers University of Technology
Gothenburg, Sweden 2014
Master's Thesis 2014:1

Abstract

This thesis describes the development of a low-cost *Unmanned Ground Vehicle* (UGV), a prototype and base-platform for various sensors which are used for research tests. The project is done at FOI, "Swedish Defence Research Agency" a research institute in the areas of defence and security located in Linköping, Sweden [1].
When external load (in form of sensors) is added the UGV's dynamic changes and the risk of roll-over increases during cornering. The objective is to keep it stable regardless of external load.
A control-system is created that handles the implemented *reaction wheel* together with the *back-wheel steering* which keeps the UGV from roll-over.
This thesis also gives details of mechanical and electronics construction as well as software development. Mathematical modelling of the UGV was done with assumption that it can be seen as a scaled vehicle, the modelling gives the basic knowledge about vehicle dynamics. A MATLAB^{TM}/Simulink^{TM} simulation environment was created to understand the behaviour of a vehicle by using the equations derived from the modelling. A complete 3D- mechanical-model of the UGV was created in Catia in order to get the mechanical properties to the simulation-model. It also gives the drawing of the UGV, so it could be manufactured.
Scheduling methods were used to automatically adjust controller parameters to the system, which ensured the stability of the UGV.
Several field tests with the UGV were done with different external loads. By changing the hight of the *center of gravity* simulation of heavy sensors was done. The UGV remained stable throughout the cornering with the controllers activate.

Acknowledgements

I want to thank my examiner **Balázs Adam Kulcsár** who has managed to keep me motivated during the project, and taken time for my questions even though his schedule has been full. Thanks to my supervisor at Chalmers **Oskar Wigström**, my supervisors at FOI, **Erika Bilock** and **Joakim Rydell**.

Special thanks to **Håkan Thoren**, who I feel might be a reflection of me in the future, he has helped me with CNC milling of the main body.
Thanks to **Viktor Påsse** one of my idols at Chalmers Robotics (CRF) who is a Jedi when it comes to construction, control design and PCB design. There is nothing he won't share and willingly give to others to make their projects proceed.

Johan S Lövgren, Gothenburg 11/9/14

In a galaxy far far away...

Nomenclature

UGV Unmanned Ground Vehicle

DOF Degrees Of Freedom

COG Center Of Gravity

m_t Total vehicle mass $[kg]$

m_s Vehicle sprung mass $[kg]$

m_u Vehicle un-sprung mass $[kg]$

T_w Vehicle Track width $[m]$

h_{COG} Height of COG $[m]$

h_{ms} Height to COG of sprung mass $[m]$

h_{rc} Height to roll axis (roll center) $[m]$

K_ϕ Spring-constant $[N/m]$

C_ϕ Damper-constant $[N - s/m]$

I Moment of inertia $[kg.m^2]$

L Vehicle length $[m]$

L_f Distance of the front axle to COG $[m]$

L_r Distance of the rear axle to COG $[m]$

δ_f Steer angle of front wheels $[rad]$

δ_r Steer angle of rear wheels $[rad]$

F_x Longitudinal force acting on vehicle at COG $[N]$

F_y Lateral force acting on vehicle at COG $[N]$

F_z Vertical force acting on vehicle at COG $[N]$

$F_{yf,l}$ Longitudinal force on front, left wheel $[N]$

$F_{yf,r}$ Longitudinal force on front, right wheel $[N]$

$F_{xf,l}$ Lateral force on front, left wheel $[N]$

$F_{xf,r}$ Lateral force on front, right wheel $[N]$
$F_{yr,l}$ Longitudinal force on rear, left wheel $[N]$

$F_{yr,r}$ Longitudinal force on rear, right wheel $[N]$

$F_{xr,l}$ Lateral force on rear, left wheel $[N]$

$F_{xr,r}$ Lateral force on rear, right wheel $[N]$

V_x Longitudinal velocity in global coordinate system $[m/s]$

V_y Lateral velocity in global coordinate system $[m/s]$

v_x Longitudinal velocity at COG $[m/s]$

v_y Lateral velocity at COG $[m/s]$

a_x Vehicle longitudinal acceleration $[m/s^2]$

a_y Vehicle lateral acceleration $[m/s^2]$

β Vehicle body side slip $[rad]$

ψ Vehicle yaw angle $[rad]$

$\dot{\psi}$ Vehicle yaw rate $[rad/s]$

ω_f Angular velocity of front wheel-pair $[rad/s]$

ω_r Angular velocity of rear wheel-pair $[rad/s]$

r_e Effective wheel radius $[m]$

RW Reaction-wheel

R_a Armature resistance in DC-motor, the wire in the coils $[Ohm]$

L_a Armature inductance of the motor $[H]$

i_a Current flowing through the electric parts in DC-motor $[A]$

V_{in} Voltage input $[V]$

V_{emf} is the induced voltage, due to the kinetic energy stored in the inertia of the rotor $[V]$

V_{R_a} Voltage over the armature resistor $[V]$

V_{L_a} Voltage over the armature inductor $[V]$

b_m Internal friction coefficient in DC-motor [-]

b_L External friction coefficient in the gearbox [-]

J_m Inertia of the DC-motor shaft $[kg \cdot m^2]$

J_L External load (inertia), the gearbox, and two wheels $[kg \cdot m^2]$

K_e DC-motors Back-emf constant (voltage constant) $[V/rad/sec]$

K_t DC-motor Torque-Constant $[N \cdot m/A]$

PCB Printed Circuit Board

MCU Micro Controller Unit

IC Integrated Circuit

IMU Internal Measurement Unit

DMP Digital Motion Processor

I^2C Inter-Integrated Circuit (serial computer bus)

ADC Analogue to Digital Converter

PWM Pulse-Width Modulation

$LiPo$ Lithium polymer battery

RC Radio Controller

Contents

1

Introduction

In this chapter the background, goals, delimitations can be found. Also the structure of the report is presented, if there are some specific information that the reader wants to find pleas read *outline of the report*.

1.1 Background

One important aspect for safety in the car industries is vehicle control to prevent roll-over. The car industries have good knowledge in both mechanical and electronic way to prevent rollover. One of the most famous car brands that had problem with a unbalanced vehicle dynamic, is Mercedes A-Class [2]. During an "älg-test", constructed by *Teknikens Värld* journal, in Sweden 1997, the car showed unstable behaviour in low velocity. The company solved this problem by implement control to prevent roll-over [3].

Rollover occurs in two ways, un-tripped and tripped [4]. The tripped rollover happens due to external inputs to the dynamic system. When a car leaves the road and slides down sideways, hits a curb or rail, this is an example of tripped roll-over.
The un-tripped one happens when high lateral acceleration acting on the body, due to narrow cornering or sharp curve when the speed of the vehicles is greater than the dynamics is constructed for.
As the specification from FOI states, the unmanned ground vehicle (UGV) is to carry several sensors (which acts like an external load) that have a great more value than the UGV itself, the prevention for roll-over is there fore essential to keep the sensor from getting damaged [**Appendix A**].
There are several ways to prevent roll-over, control the steering of the vehicle, adding actuators to the system or using speed-control to minimizing the yaw-rate [5].
An unmanned ground vehicle has an *operator* that "drives" the vehicle. Compared to a

personal vehicle driver/operator has no sense about the movement of the vehicles, such as acceleration or the feeling of the road condition through the steering-wheel. Therefore sensors are implemented that can detect lateral- and longitudinal- acceleration, yaw- and roll- rate a feedback to the system can be done. In fig.1.1 schematic of the parts connected to the UGV is presented.

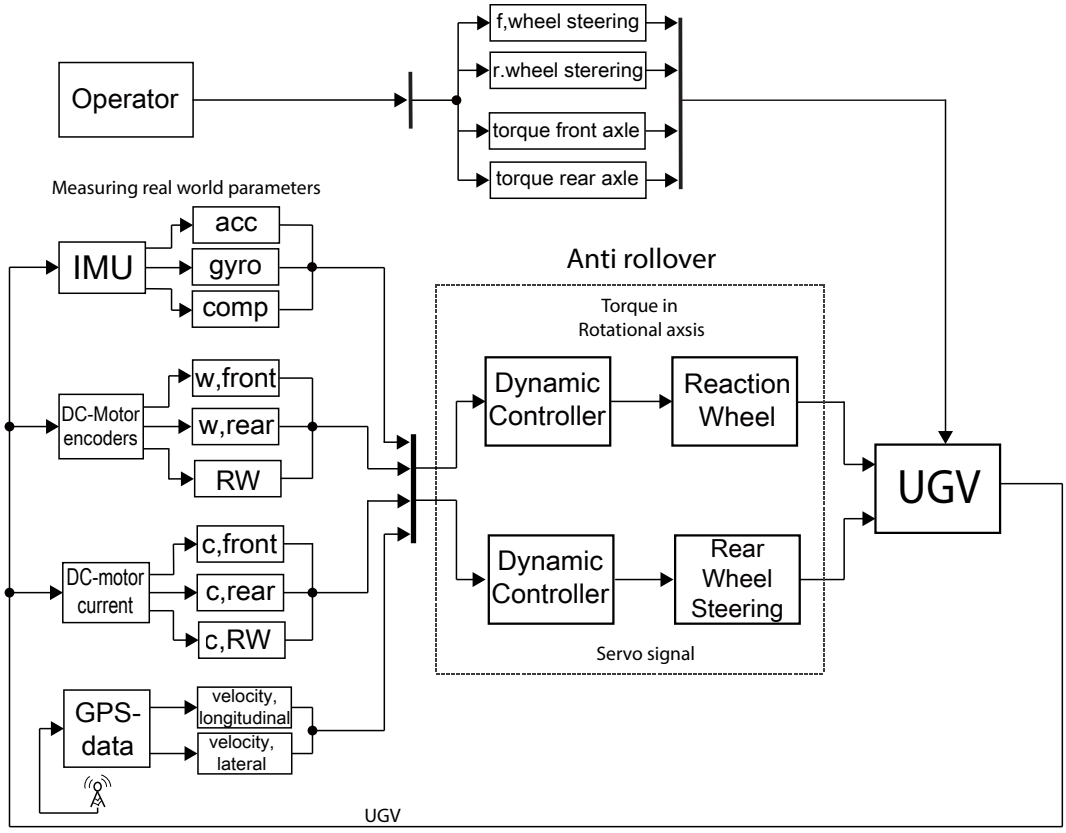


Figure 1.1: flow diagram, the abbreviations stands for: *acc; acceleration*, *comp; compass*, w,(f,r); angular velocity of front and rear wheels. *RW; reaction wheel*

1.2 Project Description and Approach

To understand vehicle dynamic several studies are done in the subject, literature was studied in form of articles, master thesis and books. Mathematical modelling of a simple bicycle-model is the basis for the full *two-track model* that is later built in Matlab/Simulink, and simulated. Analysis of the open-loop response gave an understanding of how the vehicle behaves during cornering and how the back-wheel steering reduces the roll-rate.

Construction of the UGV is done with the specification from FOI as reference. Note that the design is done in this thesis project, there were no drawings or plans on the design from the beginning, only what the UGV should manage [**Appendix A**].

A reaction wheel together with back-wheel steering is the two part that are controlled by the two separate regulators. The steering helps the UGV to maintain balance while driving, the reaction wheel is used as a recipient of impulses, while driving or stationary. The external load may have some windbreak, and it must not tip over even if the UGV is stationary. The flowchart in fig.1.1 illustrates the inputs, outputs and controls to the system.

Filed-test of the constructed UGV is done, with and without the controllers active. The data is collected visually with video recording and by logging the sensor data. Analyse is done to yield a conclusion to see if the designs are valid to use or not. Fig.1.2 illustrated the work-flow in the project.

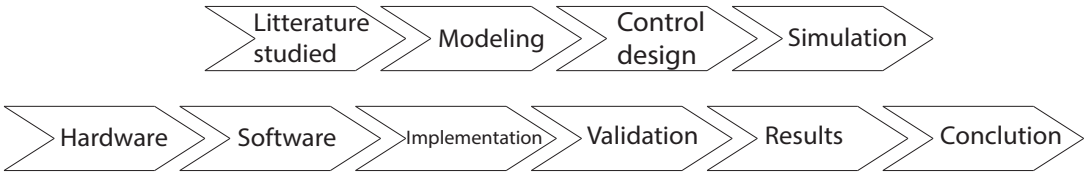


Figure 1.2: flowchart of project approach

1.2.1 Purpose

The purpose of the project is to prevent rollover and minimize impulses in the lateral direction.

1.2.2 Goals

The goals of the project are,

• Design a physical model of the UGV in CAD, and export drawings and files to FOI, for replication.

• Build a prototype UGV, and it should atleast manages an load at 3 kg

• Schematic of the electronics in the project.

• PCB design from the prototype, ready to run.

• Construction of an controller that prevents rollover.

• Construct a way to handle noise in form of small vibrations and impulses in the lateral direction.

• A complete software, ready to run.

• Manual of the project that covers all the designs and facts about the implementations of the UGV.

• A way of analyse the UGV's performance live in the computer.

• Make live tests to validate the performance.

1.2.3 Delimitations

• The complete code for the project will not be presented. Due to the length of the code and for copyright reason.

• FOI wants a control method that is simple to understand so that they can implement it themselves in the future.

• Only parts of the schematics are presented in the report. Due to copyright reasons.

1.3 Outline of Report

The report is a mix of an thesis and a manual. The natural way of reading the report is in the following order it is constructed. But every chapter has all the information needed to understand just that chapter.
Therefore the reader can choose the chapter designated for the information searched for.

Chapter 2 till 4 is theoretical and the basic understanding of the mathematical construction, control theory, and modelling. If information about DC-motor and parameter estimation is needed, then Chapter 4 is the one the reader should focus on. Chapter 5 presents the simulation model used in the project. Chapter 6 and 7 handles the more practical parts in the project, the hardware and software. Chapter 8 presents the results, discussion and conclusion of the project.

More ingoing information about the Chapters.
Chapter 2 starts with an introduction to *tyre-dynamics*, then vehicle modelling of a *single-track model* (bicycle-model) that expands to an 3-DOF single-track model. Then the final version, that is used in the simulations is presented, the *two-track model*.
The bicycle-model is presented because it is easier to follow the modelling steps in a single track model rather then start with the two-track model. And expand bicycle-model until the final model is reached.
Chapter 3 handles the modelling of the *DC-motor*, and parameter estimation. Followed with the modelling of the *reaction wheel*.
Chapter 4 is the main objective of the control strategic, and control design that is to be tested in the project.
Simulation, description of the *Simulink* model are done in chapter 5.
Chapter 6, covers the construction of the UGV, both mechanical and electrical. The chapter is written i a way so that others can benefit from the build, in a way one can see it in somewhat as a guide.
In Chapter 7 the basis of the software is provided, in the report only test parts of the software is included, due to the length of the complete code.
Chapter 8 presents all the results from the project. Simulations, field-tests of the UGV, and the logged data from the different tests are presented and analysed.
In Chapter 9 conclusion about the results are presented and, recommendation for further work is given.

Appendix A handles the drawing and images of the mechanical parts.
Appendix B handles the schematics, PCB-boards and data-sheet of the electronics.
In Appendix C, test codes for the several parts is located. Not that the complete code is not presented in this thesis.
Appendix D, i various images from the assemble, field tests and parts that don't fit in the report but is needed to be done for the project to be completed.

2

Vehicle Modelling

This chapter describes the dynamic modelling of a vehicle. Starts with the basis of tyre-modelling, then a simple *bicycle model*, that expands to bicycle model with roll (3-DOF model). The *two-track model* is the final model that is used in the simulations. In order to understand overturning, roll-dynamic is presented and ways of detecting tilt and rollover.
In the reference coordinate system, the *x-axis* represents the longitudinal direction (forward direction) and *y-axis* the lateral direction (sideways direction). And *z-axis* is positive upwards.

2.1 Assumption and Simplifications

The final vehicle is a 1:5 scale of an personal vehicle, and therefore some simplification is applied to the model.

- Aerodynamic effects on the vehicle have been neglected.
- Body pitch is not considered.
- Same tyre constants on all wheels.
- Steering angle of right and left wheel are considered to be the same.
- Rolling resistant of the wheels, assumed to be small, and therefore neglected.
- Longitudinal and lateral friction coefficients are considered to be the same.
- Angular velocity of the front wheel-pair will consider the same, and the angular velocity

of the rear wheel-pair will be considered the same.

2.2 Tyre Model

One of the elements in the vehicle to be modelled is the tyres. Its parameters are important for control system design and robustness [6]. The model consists of two main parts. The first is to calculate the slip condition of the tyres. The second is to derive the tyre forces due to the slip conditions and steering angle. The tyre model is according to Pacejka [7]. To generate force the tyre must slip. It occurs in different planes of the tyre, longitudinal and lateral [7].

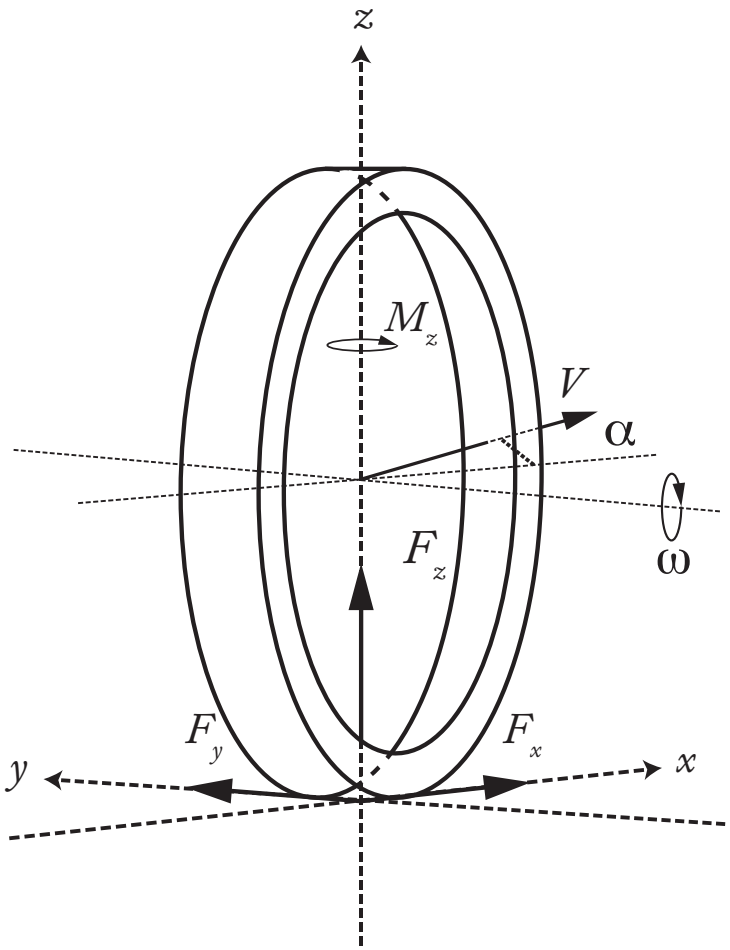


Figure 2.1: Tyre model and the acting forces

2.2.1 Longitudinal slip

From the generated torque, when driving or braking a longitudinal slip developed, it can generally be described in terms of wheel-slip, a measurement between the translational velocity and the rotational speed of the wheel, this gives a value of the slip.
The *slip velocity* is relative velocity of the tyre contact patch with the ground. And is given by,

$$v_x - r_e \cdot \omega \tag{2.1}$$

where ω is the angular velocity of the wheel. r_e is the effective radius of the wheel. v_x is the longitudinal velocity of the wheel, which is a component of the velocity vector v, described in fig.2.1.

Effective radius of the wheel is measured when no external torque is applied up on the spin axis. And therefore the effective radius lies between the un-deformed radius and the static loaded radius [7]. For simplification in the modelling the effective wheel radius will be the same as the wheel radius.
By normalizing equation (2.1) the *slip* is obtained. The normalized equation is presented as,

$$\lambda = -\frac{v_x - r_e\omega}{v_x} \tag{2.2}$$

The equation (2.2) is defined in a way that, if a driving force is applied, then $lambda > 0$, if braking force is applied, then $lambda < 0$

2.2.2 Lateral slip

The tyre slip is the angle between the longitudinal axis and the orientation of the velocity vector (v), measured from the center of the wheel.

$$\alpha = \arctan(\frac{v_y}{v_x}) \tag{2.3}$$

v_x and v_y is the velocity's components of the velocity vector v of the wheel.

2.2.3 Tyre system

The system can be considered to have slip values as input, and F_x, F_y and M_z as output [8].

$$\begin{cases} F_x = F_x(\lambda,\gamma,\alpha,F_z) \\ F_y = F_y(\lambda,\gamma,\alpha,F_z) \\ M_z = M_z(\lambda,\gamma,\alpha,F_z) \end{cases} \tag{2.4}$$

F_x, F_y is the generated forces in the longitudinal and lateral direction, respectively. M_z is the induced moment due to the steering force and patch area between the tyre

and ground. F_z is the wheel load (due to normal force), and it is modelled so that positive is in a upwards direction. The chamber angle (γ) is assumed to be zero in this model.

The accurate way of calculating the lateral force F_y is by using the so called *magic formula* [9],

$$F_y = D \cdot sin[C \cdot \arctan(B \cdot x - E \cdot (B \cdot \alpha - \arctan(B \cdot \alpha)))] \tag{2.5}$$

Where,

$$\begin{cases} B = \frac{C_\alpha}{C \cdot D} \\ D = u \cdot F_z \end{cases} \tag{2.6}$$

E and C are tyre Curvature factor and Shape factor, respectively. For small angles approximation can be done, $sin(\beta) = \beta$ and $atan(\beta) = \beta$. Following linear equation is then derived,

$$F_y = DC(B\alpha - E(B\alpha - B\alpha)) = DCB\alpha = C_{alpha} \tag{2.7}$$

Within the linear region, the approximation becomes,

$$\begin{cases} F_x \approx C_{F_\lambda} \cdot \lambda \\ F_y \approx C_{F_\alpha} \cdot \alpha \\ M_z \approx -C_{M_\alpha} \cdot \alpha \end{cases} \tag{2.8}$$

where,
C_{F_λ} is the longitudinal tyre stiffness coefficient, C_{F_α} is the lateral tyre stiffness coefficient (cornering stiffness) and C_{M_α} is the aligning stiffness.

$$C_{F_\alpha} = c_1 \cdot sin\left(2 \cdot arctan(\frac{F_z}{c_2})\right) \tag{2.9}$$

c_1 is the maximum cornering stiffness. c_2 is the load at maximum cornering stiffness.

2.3 Bicycle Model

By lumping the rear and front wheel-pair into two single wheels (neglecting the track width (T_w)), one gets the simplest way to describe the dynamics of and four wheel-vehicle, the bicycle-model. Even if it has some mayor simplification the bicycle model gives many answers about the behaviour of the system [9].

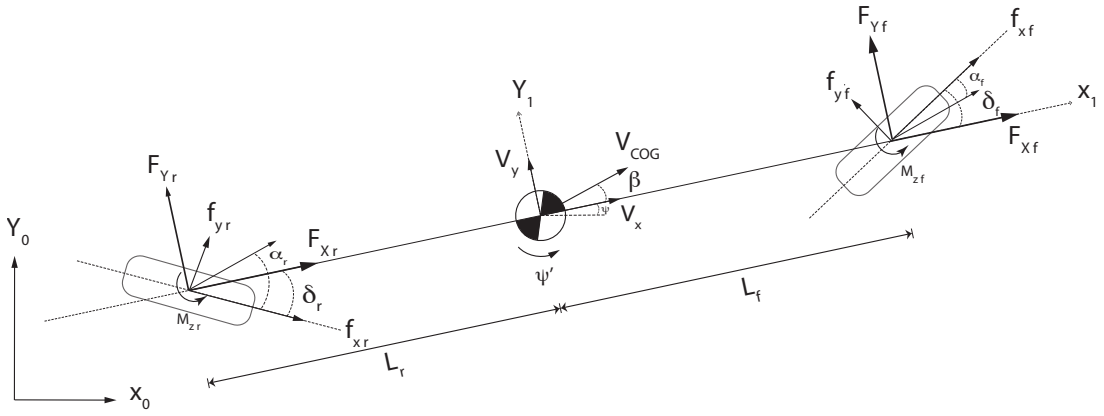


Figure 2.2: Bicycle model top view

Due to applied torque and deformation of the tyres, forces are produced [10]. The forces and moments acting up on the bicycle-model at COG are,

F_X: Longitudinal force along x-axis
F_Y: Lateral force along y-axis
F_Z: Vertical force along z-axis (normal force)
M_Z: Aligning moment about z-axis (yaw)

Transformation between wheel-coordinate system and body-coordinate system has to be done to get the resulting sum of forces and moments. By using fig.2.2, the sum of forces can be described.

$$\sum F_X = F_{xr} + F_{xf} \tag{2.10}$$

$$\sum F_Y = F_{yr} + F_{yf} \tag{2.11}$$

$$\sum M_z = M_{zr} + M_{zf} \tag{2.12}$$

Where F_{xr} and F_{xf} is the rear and front longitudinal tyre-forces , F_{yr} and F_{yf} is the lateral rear and front tyre forces (both in their respective coordinate systems). M_{xr} and M_{fr} is the moment that occurs when turning the rear and front wheel, respectively.

2.3.1 Deviation of tyre forces

The components to get the sums in equation (2.10), (2.10) and (2.12) are derived by following equations, Note that the rear steering angle (δ_r) has a negative direction against the vehicle x-axis in the fig.2.2.

$$F_{xr} = f_{xr} \cdot cos(-\delta_r) + f_{yr} \cdot sin(-\delta_r) \tag{2.13}$$

$$F_{xf} = f_{xf} \cdot cos(\delta_f) - f_{yf} \cdot sin(\delta_f) \tag{2.14}$$

$$F_{yr} = f_{yr} \cdot cos(-\delta_r) - f_{xr} \cdot sin(-\delta_r) \tag{2.15}$$

$$F_{yf} = f_{xf} \cdot sin(\delta_f) + f_{yf} \cdot cos(\delta_f) \tag{2.16}$$

$$M_{zr} = -L_r(f_{yr} \cdot cos(\delta_r) + f_{xr} \cdot sin(\delta_r)) \tag{2.17}$$

$$M_{zf} = L_f(f_{yf} \cdot cos(\delta_f) + f_{xf} \cdot sin(\delta_f)) \tag{2.18}$$

δ_f and δ_r is the front and rear steering angles, L_r and L_f are the distances from the COG of the vehicle to the front and rear axles, respectively.

Combination of equation (2.10), (2.13) and (2.14).
And combination of equation (2.11), (2.15) and (2.16).
And combination of equation (2.12), (2.17) and (2.18).
With the trigonometric rules stated that $cos(-\delta) = cos(\delta)$ and $sin(-\delta) = -sin(\delta)$, following equations are derived respectively.

$$\sum F_X = f_{xr} \cdot cos(\delta_r) - f_{yr} \cdot sin(\delta_r) + f_{xf} \cdot cos(\delta_f) - f_{yf} \cdot sin(\delta_f) \tag{2.19}$$

$$\sum F_Y = f_{yr} \cdot cos(\delta_r) + f_{xr} \cdot sin(\delta_r) + f_{xf} \cdot sin(\delta_f) + f_{yf} \cdot cos(\delta_f) \tag{2.20}$$

$$\sum M_Z = L_f(f_{yf} \cdot cos(\delta_f) + f_{xf} \cdot sin(\delta_f)) - L_r(f_{yr} \cdot cos(\delta_r) + f_{xr} \cdot sin(\delta_r)) \tag{2.21}$$

2.3.2 Lateral and Longitudinal tyre forces

From the Chapter *Tyre model*, *Tyre system*, the tyre forces within the linear region is stated (equation (2.8)). The separate tyre forces in the bicycle model (within the linear region) then becomes,

$$\begin{cases} f_{xr} \approx C_{\lambda_r} \cdot \lambda_r \\ f_{xf} \approx C_{\lambda_f} \cdot \lambda_f \\ f_{yr} \approx C_{\alpha_r} \cdot \alpha_r \\ f_{yf} \approx C_{\alpha_f} \cdot \alpha_f \end{cases} \tag{2.22}$$

Lateral slip

The body-frame side slip angle can be described as the angle between the longitudinal axis and the velocity vector of the body orientation. From fig.2.2, β can be represented in form of δ and α,

$$\beta = \delta - \alpha \tag{2.23}$$

The body frame slip angle is defined as,

$$\beta = arctan(\frac{V_y}{V_x}) \tag{2.24}$$

The tyre slip-angle is the angle between the orientation of the velocity vector at COG, (thus the direction of travel) and the tyre itself. shown in fig.2.1 This will give the following definition to the tyre-slip angle,

$$\alpha = \delta - arctan(\frac{V_y}{V_x}) \tag{2.25}$$

Observe that there is an additional velocity-component to V_y, due to the yaw-rate of the ridged body frame and the distance from the COG to the front and rear wheel. For the front and back tyre following slip angles is then defined,

$$\alpha_f = \delta_f - arctan(\frac{V_y + \dot{\psi} \cdot L_f}{V_x}) \tag{2.26}$$

$$\alpha_r = \delta_r - arctan(\frac{V_y - \dot{\psi} \cdot L_r}{V_x}) \tag{2.27}$$

Longitudinal slip

From equation (2.2), the longitudinal slip is calculated, in terms of v_x, r_e and ω. In the bicycle model, there are two wheels, and therefore two longitudinal slip equations.

$$\lambda_f = -\frac{v_{xf} - r_e\omega_f}{v_{xf}} \tag{2.28}$$

$$\lambda_r = -\frac{v_{xr} - r_e\omega_r}{v_{xr}} \tag{2.29}$$

Where v_{xf} is the longitudinal velocity of the front wheel, v_{xf} is the longitudinal velocity of the rear wheel. ω_f and ω_r is the angular velocity of the front and rear wheel, respectively.

2.3.3 Dynamics of the bicycle-model

Using Newton's second law the equation of motions can be derived. The forces and moments is acting up on the vehicles own reference coordinate system (X_1 and Y_1, in fig.2.1).

With assumption that,
• The longitudinal velocity V_x is known and constant, during the calculation process of the the differential equations.
• The aligning moments caused by the turning if the tyre is neglected.
• Only two wheels, front and rear (bicycle model)

$$\sum F_Y = m \cdot a_y \tag{2.30}$$

$$\sum M_Z = I_z \ddot{\psi} \tag{2.31}$$

Note that this is not the same sum's as before. F_Y in the differential equation only consider the lateral part of the tyre. Before, when deviation of tyre-forces was done, extra components from the *fx-force* was added to the F_Y part, but is not to be consider. The influence of longitudinal force component in lateral direction is neglected. fig.2.3 shows the lateral component of the tyre.

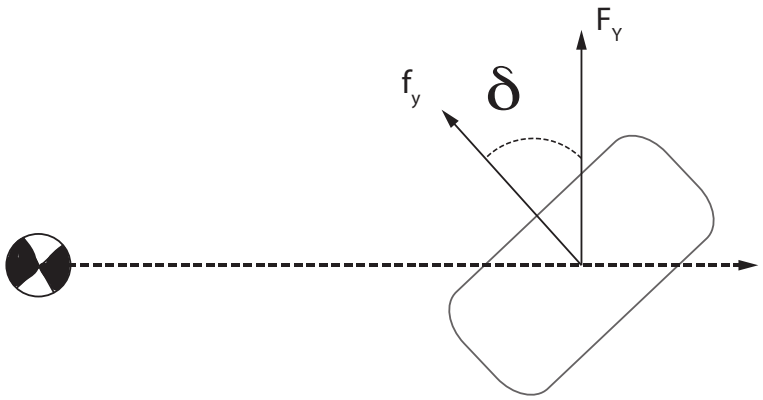


Figure 2.3: one tyre with lateral forces

$$\begin{cases} \sum F_Y = F_{yf} + F_{yr} \\ \sum Mz = L_f \cdot F_{yf} - L_r \cdot F_{yr} \\ F_{yf} = f_{yf} \cdot cos(\delta_f) \\ F_{yf} = f_{yr} \cdot cos(\delta_r) \end{cases} \tag{2.32}$$

$a_x = \dot{V}_x$, $a_y = \dot{V}_y$ The acceleration has an additional component due to the body-frame rotation, the longitudinal and lateral acceleration then becomes,

$$a_x = \dot{V}_x - V_y\dot{\psi} \tag{2.33}$$

$$a_y = \dot{V}_y + V_x\dot{\psi} \tag{2.34}$$

Equation (2.30), (2.32) and (2.34) in combination with (2.22) yields,

$$m(\dot{V}_y + V_x\dot{\psi}) = C_{\alpha_f}\alpha_f \cdot cos(\delta_f) + C_{\alpha_r}\alpha_r \cdot cos(\delta_r) \tag{2.35}$$

$$I_z\ddot{\psi} = L_f \cdot C_{\alpha_f}\alpha_f \cdot cos(\delta_f) - L_r \cdot C_{\alpha_r}\alpha_r \cdot cos(\delta_r) \tag{2.36}$$

Approximation for small angles ($cos(\delta) = 1$) together with equation for lateral slip (2.26) and (2.27) gives following result,

$$m(\dot{V}_y + V_x\dot{\psi}) = C_{\alpha_f}\left(\delta_f - \frac{V_y + L_f\,\dot{\psi}}{V_x}\right) + C_{\alpha_r}\left(\delta_r - \frac{V_y - L_r\,\dot{\psi}}{V_x}\right) \tag{2.37}$$

$$I_z\ddot{\psi} = C_{\alpha_f}\,L_f\left(\delta_f - \frac{v_y + L_f\,\dot{\psi}}{v_x}\right) - C_{\alpha_r}\,L_r\left(\delta_r - \frac{v_y - L_r\,\dot{\psi}}{v_x}\right) \tag{2.38}$$

Expanding the equations,

$$m(\dot{V}_y + V_x\dot{\psi}) = C_{\alpha_f}\,\delta_f + C_{\alpha_r}\,\delta_r - \frac{C_{\alpha_f}\,V_y}{V_x} - \frac{C_{\alpha_r}\,V_y}{V_x} - \frac{C_{\alpha_f}\,L_f\,\dot{\psi}}{V_x} + \frac{C_{\alpha_r}\,L_r\,\dot{\psi}}{V_x} \tag{2.39}$$

$$I_z\ddot{\psi} = C_{\alpha_f}\,L_f\,\delta_f - C_{\alpha_r}\,L_r\,\delta_r - \frac{C_{\alpha_f}\,L_f\,v_y}{v_x} + \frac{C_{\alpha_r}\,L_r\,v_y}{v_x} - \frac{C_{\alpha_f}\,{L_f}^2\,\dot{\psi}}{v_x} - \frac{C_{\alpha_r}\,{L_r}^2\,\dot{\psi}}{v_x} \tag{2.40}$$

The bicycle model written in state space form, with steering angle (δ_f, δ_r) as input. The states are V_y and $\dot{\psi}$. Output is $\dot{V}_y$ and $\ddot{\psi}$.

$$\begin{pmatrix} \dot{v_y} \\ \ddot{\psi} \end{pmatrix} = \begin{pmatrix} -\frac{C_{\alpha_f}+C_{\alpha_r}}{m\,v_x} & -v_x - \frac{C_{\alpha_f}\,L_f - C_{\alpha_r}\,L_r}{m\,v_x} \\ -\frac{C_{\alpha_f}\,L_f - C_{\alpha_r}\,L_r}{I_z\,v_x} & \frac{C_{\alpha_f}\,{L_f}^2}{I_z\,v_x} + \frac{C_{\alpha_r}\,{L_r}^2}{I_z\,v_x} \end{pmatrix} \cdot \begin{pmatrix} v_y \\ \dot{\psi} \end{pmatrix} + \begin{pmatrix} \frac{C_{\alpha_f}}{m} & \frac{C_{\alpha_r}}{m} \\ \frac{C_{\alpha_f}\,L_f}{I_z} & \frac{C_{\alpha_r}\,L_r}{I_z} \end{pmatrix} \cdot \begin{pmatrix} \delta_f \\ \delta_r \end{pmatrix} \tag{2.41}$$

Note that further simplification can be done, due to the fact that the tyre constants are the same on front and rear wheels.

2.4 Bicycle Model with Roll

The outcome of the project is to handle overturning, therefore equations for roll-dynamics is added into the bicycle model.
The roll dynamics equation is based on the torque balance around the vehicle body x-axis. The equation are,

$$\begin{cases} m(\dot{V}_y + V_x\dot{\psi}) = C_{\alpha_f}\,\cos(\delta_f)\,\left(\delta_f - \frac{V_y + L_f\,\dot{\psi}}{V_x}\right) + C_{\alpha_r}\,\cos(\delta_r)\,\left(\delta_r - \frac{V_y - L_r\,\dot{\psi}}{V_x}\right) \\ I_z\ddot{\psi} = C_{\alpha_f}\,L_f\,\cos(\delta_f)\,\left(\delta_f - \frac{v_y + L_f\,\dot{\psi}}{v_x}\right) - C_{\alpha_r}\,L_r\,\cos(\delta_r)\,\left(\delta_r - \frac{v_y - L_r\,\dot{\psi}}{v_x}\right) \\ mh(\dot{V}_y + V_x\dot{\psi}) = C_\phi\,\dot{\phi} + I_x\,\ddot{\phi} + K_\phi\,\phi \end{cases} \tag{2.42}$$

Where, K_ϕ and C_ϕ is the spring and damping coefficients respectively. ϕ, $\dot{\phi}$ and $\ddot{\phi}$ is the roll-angle, roll-rate, and roll-acceleration, respectively.

2.5 Two-Track Model

The final and most complex model in this project is the two-track model. All four wheels are connected to each other in an ridged body frame. From the fig.2.4, the forces from the tyres can be transformed to the body-frame. Note that the rear steering angle is opposite against the front steering angle (sign(δ_f) = -sign(δ_f)·δ_r).

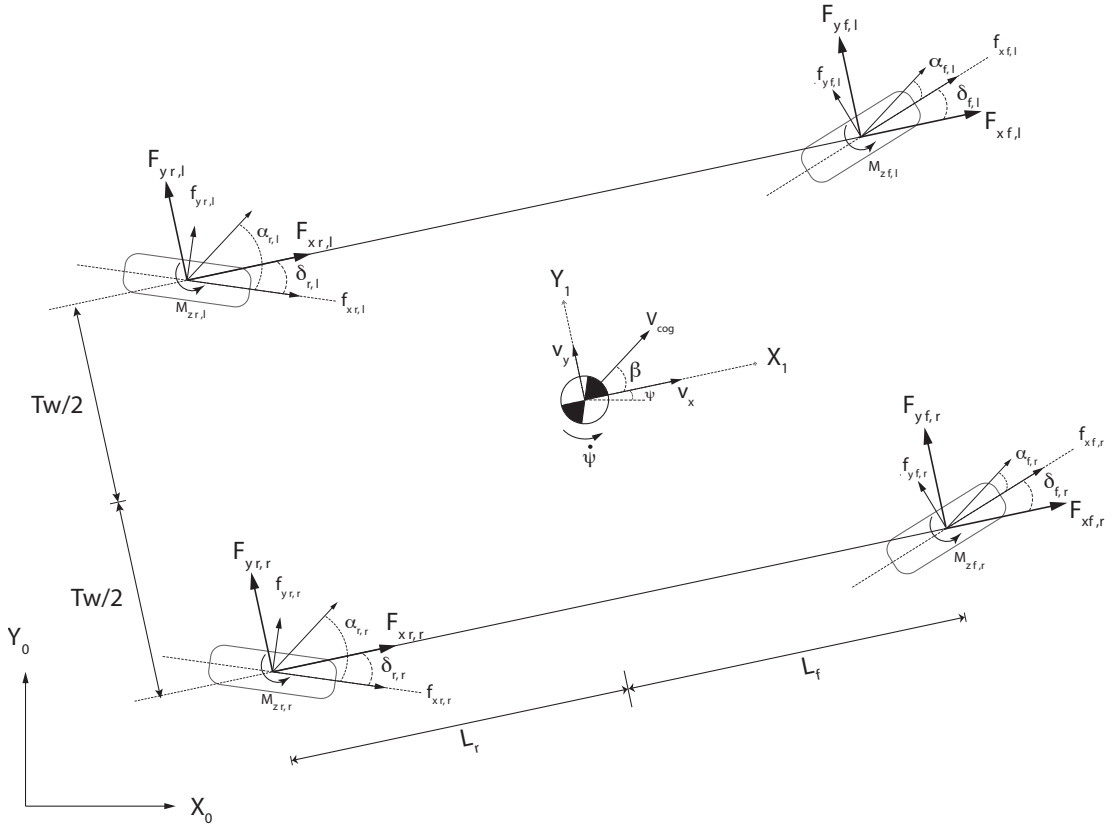


Figure 2.4: Two-track model

2.5.1 Deviation of tyre forces

In the bicycle-model, the forces induced due to the longitudinal and lateral slip has to be mapped to the body frame. Same is done for the two-track model. The mapped forces will then be acting from/on the COG, form which the acting force on the sprung mass can be estimated.

$$\begin{cases} F_xf,l = f_{xf,l}\ \cos(\delta_f) + f_{yf,l}\ \sin(\delta_f) \\ F_yf,l = f_{yf,l}\ \cos(\delta_f) + f_{xf,l}\ \sin(\delta_f) \\ F_xf,r = f_{xf,r}\ \cos(\delta_f) + f_{yf,r}\ \sin(\delta_f) \\ F_yf,r = f_{yf,r}\ \cos(\delta_f) + f_{xf,r}\ \sin(\delta_f) \\ F_xr,l = f_{xr,l}\ \cos(\delta_r) - f_{yr,l}\ \sin(\delta_r) \\ F_yr,l = f_{yr,l}\ \cos(\delta_r) - f_{xr,l}\ \sin(\delta_r) \\ F_xr,r = f_{xr,r}\ \cos(\delta_r) - f_{yr,r}\ \sin(\delta_r) \\ F_yr,r = f_{yr,r}\ \cos(\delta_r) - f_{xr,r}\ \sin(\delta_r) \end{cases} \tag{2.43}$$

Where Fxf,i ($i = l,r$) the l and r after the comma sign, stands for left and right wheel side, respectively. f and r before the comma sign stands for the front and rear wheel, respectively.

2.5.2 Dynamics of the Two-track model

The differential equations for the *Two-Track model* is taken from [11].

$$m_t(\dot{v_x} - \dot{\psi}v_y) = m_u\ (L_f - L_r)\ \dot{\psi}^2 - 2\,h_{rc}\,m_s\,\dot{\phi}\,\dot{\psi} + F_{xfl} + F_{xfr} + F_{xrl} + F_{xrr} \tag{2.44}$$

$$m_t(\dot{v_y} + \dot{\psi}v_x) = F_{yfl} + F_{yfr} + F_{yrl} + F_{yrr} - m_u\,\ddot{\psi}\ (L_f - L_r) + h_{rc}\,m_s\,\ddot{\phi} \tag{2.45}$$

$$I_xz\ddot{\phi}+I_z\ddot{\psi} = L_f(F_{yfl}+F_{yfr})-L_r(F_{yrl}+F_{yrr})+c(-F_{xfl}+F_{xfr}-F_{xrl}+F_{xrr})+(L_r-L_f)m_u(\ddot{y}+\dot{\psi}\dot{x}) \tag{2.46}$$

$$(I_x + m_s \cdot h_{rc}^2)\ddot{\phi} + I_{xz}\ddot{\psi} = h_{rc}\,m_s\,\left(\ddot{y} + \dot{\psi}\,\dot{x}\right) - 2\,K_\phi\,\phi - 2\,C_\phi\,\dot{\phi} + g\,h_{rc}\,m_s\,\phi \tag{2.47}$$

Where c is half the track-width ($T_w/2$) and, m_t, m_s, m_u is the total vehicle mass, vehicle sprung mass and vehicle unsprung mass, respectively. The UGV is symmetric, therefore $L_f = L_r$

Inserting equation (2.43) yields the differential equations that is to be used. The equation can be rewritten in a *descriptor from*, for the non linear system. With a mass matrix M a compact differential equation in the following form is stated,

$$\dot{\xi}(t) = f(\xi(t),u(t)) \tag{2.48}$$

with the states,

$$X = \begin{pmatrix} \dot{y} & \dot{x} & \dot{\phi} & \dot{\psi} & y & x & \phi & \psi \end{pmatrix}^T \tag{2.49}$$

the matrices becomes,

$$M = \begin{pmatrix} 0 & m_t & m_u\,(L_f - L_r) & -h_{rc}\,m_s & 0 & 0 & 0 & 0 \\ m_t & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & m_u\,(L_f - L_r) & I_z & I_{xz} & 0 & 0 & 0 & 0 \\ 0 & -h_{rc}\,m_s & I_{xz} & m_s\,{h_{rc}}^2 + I_x & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.50}$$

$f(\xi(t),u(t)) =$

$$\begin{pmatrix} F_{yfl} + F_{yfr} + F_{yrl} + F_{yrr} - m_t\,\dot{\psi}\,\dot{x} \\ F_{xfl} + F_{xfr} + F_{xrl} + F_{xrr} + m_t\,\dot{\psi}\,\dot{y} - m_u\,\dot{\psi}^2\,(L_f - L_r) - 2\,h_{rc}\,m_s\,\dot{\phi}\,\dot{\psi} \\ L_f\,(F_{yfl} + F_{yfr}) - L_r\,(F_{yrl} + F_{yrr}) - c\,(F_{xfl} - F_{xfr} + F_{xrl} - F_{xrr}) - m_u\,\dot{\psi}\,\dot{x}\,(L_f - L_r) \\ g\,h_{rc}\,m_s\,\phi - 2\,K_\phi\,\phi - 2\,C_\phi\,\dot{\phi} + h_{rc}\,m_s\,\dot{\psi}\,\dot{x} \\ x \\ y \\ \psi \\ \phi \end{pmatrix} \tag{2.51}$$

This will yield following form,

$$M \cdot \dot{\xi}(t) = f(\xi(t),u(t)) \tag{2.52}$$

The input vector to the system is,

$$u(t) = \begin{pmatrix} \delta_f & \delta_r & \omega_f & \omega_r \end{pmatrix} \tag{2.53}$$

Not that the M-matrix first four rows consist of the values, and that they are in the first four columns, thus it is a square, and therefore invertible.
With the M-matrix inverted, it will yield the following non-linear equation,

$$\dot{\xi}(t) = M^{-1} \cdot f(\xi(t),u(t)) \tag{2.54}$$

2.6 Roll dynamics

The body of the *two-Track model* is represented as a sprung-mass, that rolls around an imaginary axis (roll center) in the longitudinal direction, illustrated in fig2.5.

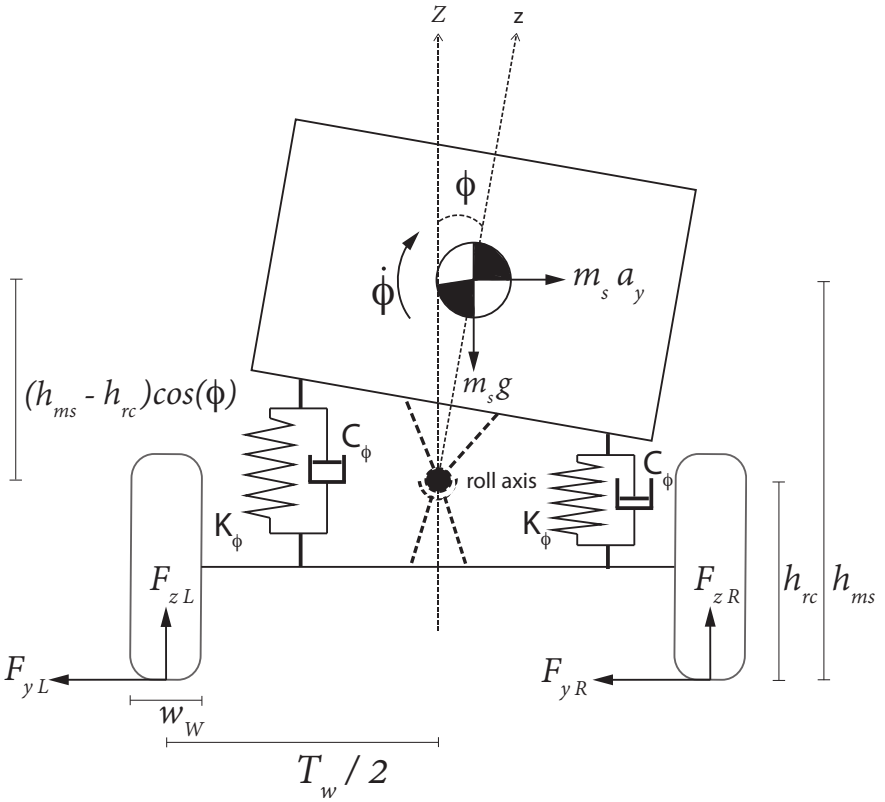


Figure 2.5: Two-track model, roll view

When the vehicle has movement in an circular trajectory a centripetal acceleration a_y occurs. The centrifugal force $K = m \cdot a_y$ can be said to act on the vehicle body at the center of gravity in the opposite direction, which contributes to the total moment acting on the rotational axis. From the vehicle modelling it is shown that the torque due to roll is dependent on the hight of COG.

$$\tau_x = (F_{yfl}+F_{yfr}+F_{yrl}+F_{yrr})\cdot(h_{ms}-h_{rc})\cdot cos(\phi)+m_s g\cdot(h_{ms}-h_{rc})\cdot sin(\phi)-2C_\phi\cdot\dot{\phi}-2K_\phi\cdot\phi \tag{2.55}$$

(This is a part of the differential equation 2.47)

2.7 Tilt/Rollover detection

The main cause for un-tripped roll-over of an vehicle is the motion from the rotation occurring when the vehicle makes a turn. fig.2.7 illustrates the UGV during cornering, with the radius R_{curve} of the curvature. The pseudo-force $m \cdot a_y$ acting from the center of gravity, and inwards to the center of the curvature is shown. External forces are the forces acting on the tyre (the tyre-road contact point).

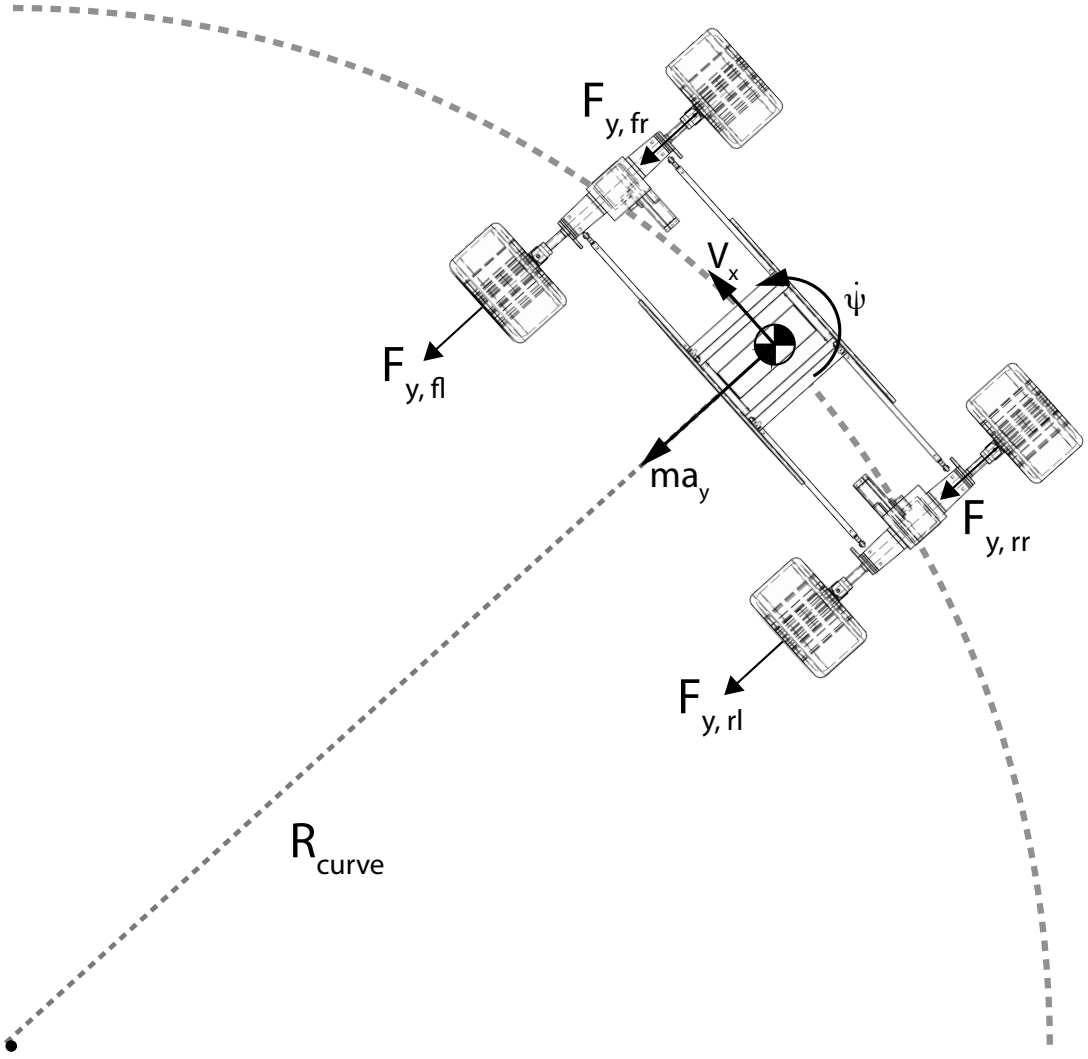


Figure 2.6: UGV during cornering

Depending on the hight of the center of gravity and acting force moment is generated, shown in equation.(2.55). The moment that is induced is counteracted by the normal forces acting on the tyre on the outside of the turn [12] (the track width is dependent

on the magnitude of the acting moment on the tyres). When the moment exceeds the normal-force, the UGV will start to roll.
To illustrate that the height of the COG has great impact on the outcome of stability, assuming that there is no shock-suspension on the UGV. The UGV would then be illustrated as in fig.2.7.
If the resulting force is outside the track width, the UGV begins to roll, and it is dependent on the hight of the COG how much external force the vehicle can handle. For roll-over to occur following statement must be fulfilled,

$$m \cdot a_y \cdot h_{COG} > mg \cdot \frac{T_w}{2} \tag{2.56}$$

$$a_y > g \cdot \frac{T_w}{2h_{COG}} \tag{2.57}$$

Clearly the equation.(2.57) the height of COG (h_{COG}) influences the magnitude of the lateral acceleration that is needed to start an rollover. Due to the fact that the right side of the mathematical inequality has only one variable, the h_{COG}, which is in the denominator.

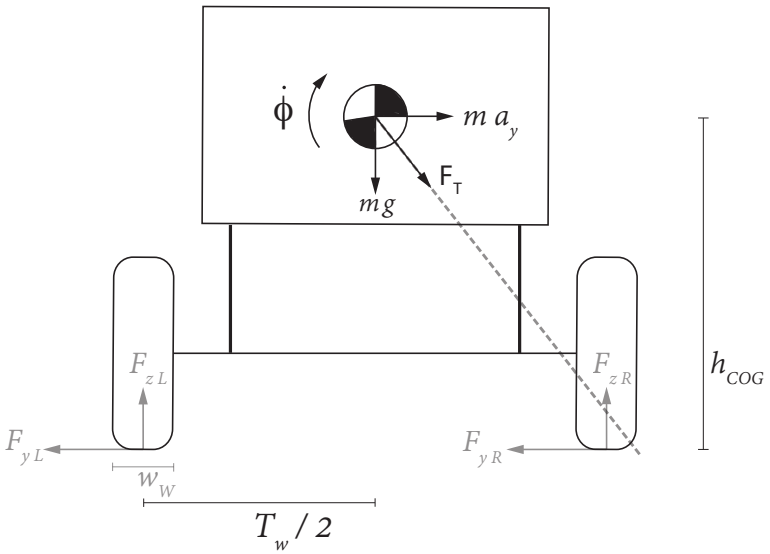


Figure 2.7: No Shock suspension, and with the resultant force

2.7.1 Roll-angle and Roll-rate

The UGV is equipped with a nine-degrees of freedom (9-DOF) IMU sensor, the roll-angle ϕ and the roll-rate $\dot{\phi}$ can easily be estimated. By defining a threshold value for the roll-angle, and the roll rate, (ϕ_{max}, $\dot{\phi}_{max}$), statement when the controller should be active can be done.
active if; $|\phi| > \phi_{max}$ or $|\dot{\phi}| > \dot{\phi}_{max}$

3

DC-motor and RW Modelling

In the project the driving force comes from an electric permanent DC-motor. There are several of them used on the UGV. In order to use the DC-motor in the simulations, mathematical modelling has to be done, in order to yield the relations between the current, voltage and rotational speed [13]. In this chapter parameter estimation of several constants of the DC-motor is also done.

3.1 Differential equations

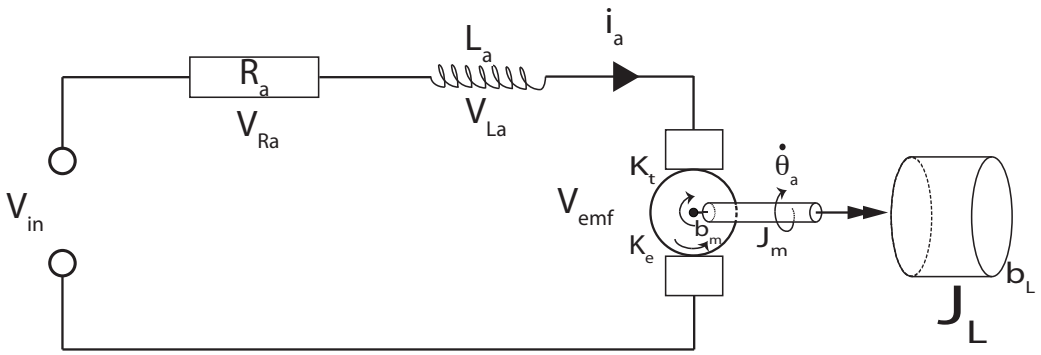


Figure 3.1: DC-motor model

The system shown in fig.3.1 is a linear model of a permanent magnet DC-motor with a static inertial. The model description is in tow parts, the electrical characteristic and the mechanic characteristic.

3.1.1 Electrical characteristics

By using *Kirchoff's voltage law* in the DC-motor model according to fig.3.1 following equation is derived,

$$V_{in} - V_{Ra} - V_{La} - V_{emf} = 0 \tag{3.1}$$

Where, V_{in}, V_{R_a}, V_{L_a} and V_{emf} is the input-voltage, voltage over the armature resistance, voltage drop over the armature inductance and voltage induced by the coil respectively. The equations for the voltage parts in the DC-motor are,

$$V_{R_a} = i_a \cdot R_a \tag{3.2}$$

$$V_{La} = L_a \cdot \frac{d}{dt} i_a \tag{3.3}$$

$$V_{emf} = K_e \cdot \dot{\theta}_a \tag{3.4}$$

Substituting equation (3.1) with (3.2), (3.3) and (3.4) yields following differential equation,

$$V_{in} - i_a \cdot R_a - L_a \cdot \frac{d}{dt} i_a - K_e \cdot \dot{\theta}_a = 0 \tag{3.5}$$

$$\frac{d}{dt} i_a = \frac{1}{L_a} \cdot (i_a \cdot R_a + K_e \cdot \dot{\theta}_a - V_{in}) \tag{3.6}$$

3.1.2 Mechanical Characteristics

By torque balance (energy balance) in the system the mechanical equations can be stated.

$$T_e - T_{\ddot{\theta}_a} - T_b - T_L = 0 \tag{3.7}$$

Where T_e is the electromagnetic torque. $T_{\dot{\theta}_a}$, is torque generated from the rotational acceleration of the rotor. T_b, is the torque due to the friction and angular velocity in the motor. T_L is the torque of the mechanical load (external load). Equations for the first three parts are,

$$T_e = K_t \cdot i_a \tag{3.8}$$

$$T_{\ddot{\theta}_a} = J_m \cdot \frac{d}{dt} \dot{\theta}_a \tag{3.9}$$

$$T_b = b_m \cdot \dot{\theta}_a \tag{3.10}$$

Substituting equation (3.7) with (3.8), (3.9) and (3.10) yields following differential equation,

$$K_t \cdot i_a - J_m \cdot \frac{d}{dt}\dot{\theta}_a - b_m \cdot \dot{\theta}_a - T_L = 0 \tag{3.11}$$

$$\frac{d}{dt}\dot{\theta}_a = \frac{1}{Jm} \cdot (K_t \cdot i_a + -b_m \cdot \dot{\theta}_a + T_L) \tag{3.12}$$

3.2 Identifying parameters

By using the data sheet of the motor (Maibuschi RS550VC, **Appendix B**) unknown parameter can be estimated. Note that the external load is zero during the estimation of the parameters ($T_L = 0$)

3.2.1 Torque-constant (Kt)

Usage of the stated equation the torque-constant (K_t) and the back-emf-constant (K_e). From equation (3.8), the torque-constant can be derived. Using both the data of the stall- , max-efficiency torque and current. The mean-value of this two gives a valid approximation of the torque-constant.

$$K_t = T_e/i_a \tag{3.13}$$

Rewriting the equation first with $T_e = T_{stall}$ and $i_a = i_{stall}$. Where T_{stall} is the stall torque form the data-sheet and i_{stall} is the stall-current from the data-sheet, and then with $T_e = T_{effe}$ and $i_a = i_{effe}$, which is the max-efficiency, -torque and -current respectively. This yields,

$$K_{t_{stall}} = T_{stall}/i_{stall} \tag{3.14}$$

and then with max-efficiency, -current and -torque.

$$K_{t_{effe}} = T_{effe}/i_{effe} \tag{3.15}$$

Mean-value of this two gives the usable torque-constant.

$$K_t = \frac{K_{t_{stall}} + K_{t_{effe}}}{2} \tag{3.16}$$

Table of the parameters that is used, (taken from the data-sheet).

constants	Stall	Max effe	unit
T_e	0.5880	0.0647	$[N \cdot m]$
i_a	85.0	10.5	$[A]$

With the stated equations and given parameters the *torque-constant* is calculated to be,

$K_t = 0.0065$.

3.2.2 Back-emf constant (Ke)

Using equation (3.4) the *Back-emf-constant* can be calculated,

$$K_e = \frac{V_{emf}}{\dot{\theta}_a} \tag{3.17}$$

In order to use (3.17) the V_{emf} must be identified. In the data sheet, the non-load -speed and -current and the nominal voltage is used.

$$V_{emf} = V_{nominal} - (R_m \cdot i_{non-Load}) \tag{3.18}$$

The internal resistance is measured to be, $R_m = 0.873$.

constants	Non-Load	unit
$\dot{\theta}_a$	2073	$[rad/sec]$
i_a	1.3	$[A]$
$V_{nominal}$	14.4	$[V]$

The calculated *back-emf constant* is,

$K_e = 0.0064$

Calculated and measured parameters. Note that the friction is yield by iteration in the simulation model (gray-box analysis), due to the fact that the maximum angular velocity at non-load is in the data-sheet.

constants	Value	unit
K_t	0.0065	$[N \cdot m/A]$
K_e	0.0064	$[V/(rad/sec)]$
R_m	0.837	$[Ohm]$
L_m	0.0008	$[H]$
b_m	$4.1210 \cdot 10^{-6}$	$[-]$
J_m	$3.87 \cdot 10^{-7}$	$[kg \cdot m^2]$

3.3 Reaction Wheel

In addition to the control of steering and torque a Reaction Wheel (RW) is mounted up on the sprung mass (mid section/body) of the UGV. The RW has the ability to handle noise in form of impulses, that acts up on the body. It also contributes to handle stabilisation of the sprung mass, which contributes to keep the mounted sensor in level.
The objective is to minimize the torque acting on the virtual roll-axis which is parallel with the x-axis. The induced torque is due to lateral acceleration acting on the sprung mass and the height from the roll-axis.

When accelerating a rotational mass with a certain inertia, a moment is induced. Due to the fact that the RW is hanging in the air, (therefore, also the name flywheel) the moment that holds back the accelerating mass, is equal to the induced moment.
The moment that is holding back is acting up on the DC motors mounting points on the body, and therefore on the sprung mass.
Depending on what direction the RW is accelerated the induced torque will be in the opposite direction, illustrated in fig.3.2. In the section calculating inertia of the reaction wheel shows that in order to yield high inertia with low mass, the mass should be at the radios of the reaction wheel. therefore holes are milled out in the center of the mounting disc.

3.4 Modelling of the RW

The mathematical modelling of the RW is done so that the induced torque can be derived. The torque of the reaction wheel is according to *Newton's law*, an accelerating mass with certain inertia gives a certain torque.

$$\tau_{wr} = J_{rw}\,\ddot{\theta} \tag{3.19}$$

Where J_s is the inertia of the reaction wheel and $\ddot{\theta}$ is the flywheels angular acceleration.

Note that the equation is strongly related to *Newton's laws of motion*, $F = m \cdot a$. The angular acceleration is yield by using equation.(3.11). Observe that the reaction wheel inertia (J_{rw}) must be added as an external input to the DC-motor system, shown in equation.(3.20). Note that the DC motor used for the reaction wheel has a mounted axial gear-reduction box. The ratio is 1:29, therefore the added load must be divided by the ratio.

$$\dot{\theta_a} = \frac{1}{J_m + \frac{J_{rw}}{29}} \cdot (K_t \cdot i_a + -(b_m + b_{rw}) \cdot \dot{\theta}_a) \tag{3.20}$$

The reaction wheel is hanging in the air, therefore the external friction from it is zero ($b_{rw} = 0$).

The resulting torque at the roll-axis will become:

$$\tau_{tot} = \tau_x - \tau_{rw} \tag{3.21}$$

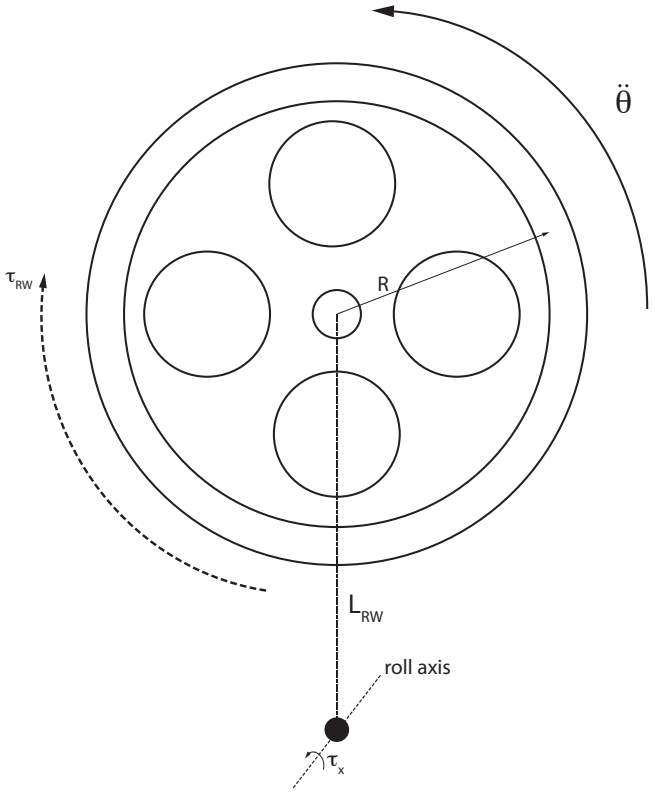


Figure 3.2: Reaction wheel

3.5 Calculating Inertia

Calculation of the inertia of a rotating mass around its own center-axis, as shown in the illustrated fig.3.3.

The mounting disc is i plastic and has a low mass compared to the outer ring, therefore it is neglected, and the calculations only considers the steel-rings. The equations are stated as,

$$J_{rw} = \frac{1}{2} \cdot m_{rw}(R_{in}^2 + R_{out}^2) \tag{3.22}$$

Calculating the mass of the reaction wheel (m_{rw}),

$$A_{rw} = \frac{\pi \cdot (R_{out} - R_{in})}{4} \tag{3.23}$$

$$V_{rw} = A_{rw} \cdot T \tag{3.24}$$

$$m_{rw} = v_{rw} \cdot \rho \tag{3.25}$$

Where ρ is the density of the material. As seen in equation.(3.22) the inertia is strongly connected to the radius of the reaction wheel. In order to yield a high inertia the radius should be large. In that, large inertia can be yielded with low mass.

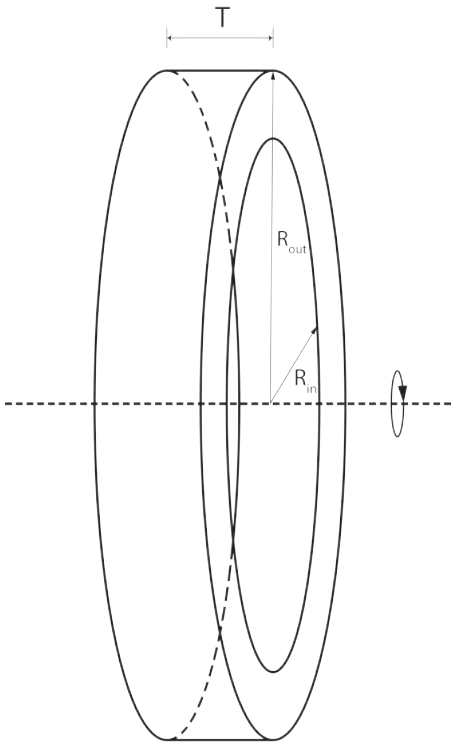


Figure 3.3: Illustrated image of a rotating mass around its center axis

4

Control Design

This chapter is about the control design that is to be implemented in the UGV. For several reason PID controllers are used in this project, mainly because one can implement such controllers in to a small MCU. Using other methods such as an *model predictive controller* demands the capability to do large calculations within a short amount of time, which is not possible with the micro processors used in this project.

The mechanics that is to be controlled to prevent rollover is the reaction-wheel and the back-wheel steering.
The two parts have separate regulators, which gives the operator the ability to run them separately. The four separate modes the operator can switch between is presented in the table below.

Regulator	Mode 1	Mode 2	Mode 3	Mode 4
$ReactionWheel$	activated	activated	deactivated	deactivated
$Back-wheelsteering$	activated	deactivated	activated	deactivated

In cases of no external load, battery can be saved by turning of the controllers. Note that the operator can choose to use the back-wheel steering, but if the controller for the back-wheel steering is activated, in a event of "*risk of rollover*" the regulator will take over the steering.

4.1 Switching PID-regulators

By using switching method between separate PID-regulators the control-system can enter different modes, which in turn can have specified tuning parameters within its range [14]. The switching is dependent on the state of the input to the switching algorithm.

From the calculations in Chapter 2, the critical angle, and roll-rate can be derived. These two values are used as threshold's, where one side of the thresholds is handled with one regulator, and the other side of the threshold with another regulator, illustrated in fig:4.1.

The two regulators is divided in such way that, the *agressivePID* has a high P-gain and is in practical more aggressive, the *constitutivePID* is softer and has the capability to minimize smaller disturbances. Together they form a controller that has the capability as a two step dynamic regulator. It changes the controllers performance depending on the input.

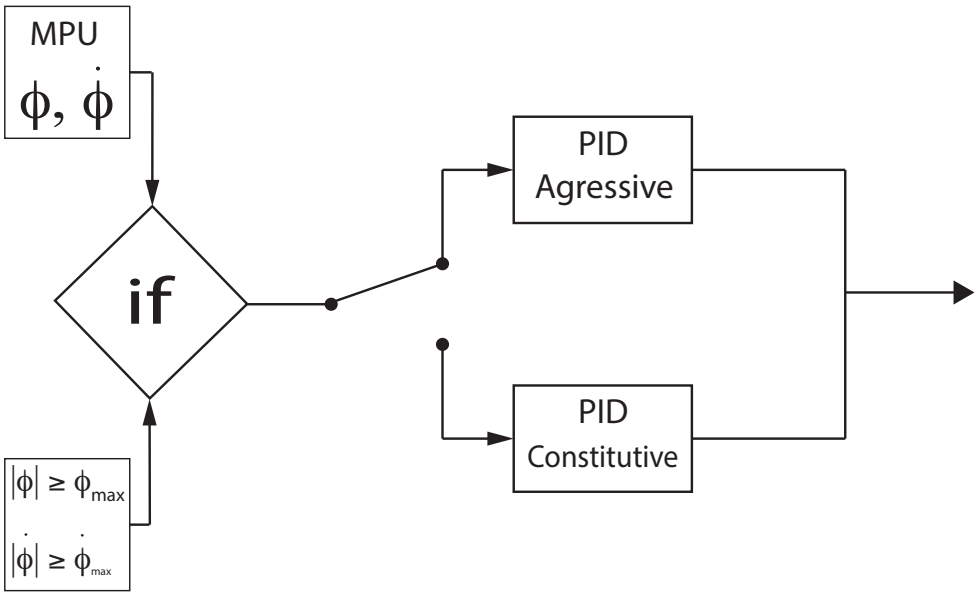


Figure 4.1: Illustrating image of switching control with PID-regulators

4.2 Parameter Scheduled PID

The switching PID-regulators can in some way be seen as dynamic PID, but only with two steps dynamics, *Constitutive* and *Aggressive*. To get a full dynamic range between the *constitutivePID* and *aggressivePID* a mathematical proposal of *parameter scheduling* can be implemented [15]. With this method the controller will not get any transient due to switching, and over all it gets smoother. In the project the parameter scheduling depends on the *roll-rate* ($\dot{\phi}(t)$), which is the only variable input to the controllers. Illustrating fig.4.2 of the *Parameter Scheduled PID*.

The PID-tuning parameters will then become time dependent, defined as,

$$\begin{cases} p \implies p(t) \implies p(\dot{\phi}(t)) \\ i \implies i(t) \implies i(\dot{\phi}(t)) \\ d \implies d(t) \implies d(\dot{\phi}(t)) \end{cases} \tag{4.1}$$

The constrains of the p,i and d in the parameter scheduled PID-regulator, is set by using the same values from the "Constitutive" (con) and "Aggressive" (agg) PID-parameters, following constrains are then derived,

$$\begin{cases} p = [p_{con}, p_{agg}] \\ i = [i_{con}, i_{agg}] \\ d = [d_{con}, d_{agg}] \end{cases} \tag{4.2}$$

By setting,
$p_{con} = p_{min}$, $p_{agg} = p_{max}$
$i_{con} = i_{max}$, $i_{agg} = i_{max}$
$d_{con} = d_{min}$, $d_{agg} = d_{max}$

$$\begin{cases} p = p_{min} + \frac{p_\phi \left|\dot{\phi}\right|}{\dot{\phi}_{max}} \\ i = i_{min} + \frac{i_\phi \left|\dot{\phi}\right|}{\dot{\phi}_{max}} \\ d = d_{min} + \frac{d_\phi \left|\dot{\phi}\right|}{\dot{\phi}_{max}} \end{cases} \tag{4.3}$$

Where $\dot{\phi}_{max}$ is the threshold to yield the "aggressive" (max) PID-parameters. Calculating p_ϕ, i_ϕ and d_ϕ,

$$\begin{cases} p_\phi = p_{max} - p_{min} \\ i_\phi = i_{max} - i_{min} \\ d_\phi = d_{max} - d_{min} \end{cases} \tag{4.4}$$

This will then give the maximum values when $\dot{\phi} = \dot{\phi_{max}}$. shown in equation.(4.5).

$$\lim_{\dot{\phi} \to \dot{\phi}_{max}} \frac{p,i,d_\phi}{\dot{\phi}_{max}} \cdot \dot{\phi} \to p,i,d_\phi \cdot 1 \tag{4.5}$$

With equation.(4.3) this will become,

$$\begin{cases} p = p_{min} + p_\phi = p_{max} \\ i = i_{min} + i_\phi = i_{max} \\ d = d_{min} + d_\phi = d_{max} \end{cases} \tag{4.6}$$

which are the parameters for the aggressive PID-regulator. Note that between the *min* and *max* value, the *p*,*i* and *d* parameters have a infinite number of values that can be chosen, compared against the switching that only has tow parameters values, *con* and *agg*.

Special case is when $\dot{\phi} > \dot{\phi}_{max}$, the *p*,*i* and *d* in equation.(4.6) will exceed the stated maximum value, therefore saturation to the input (roll-rate, $\dot{\phi}$) must be done to be certain that the *p*,*i* and *d* values are within the specified region. Saturation is stated by following function,

$if(|\dot{\phi}| > \dot{\phi}_{max})$ {
$|\dot{\phi}| = \dot{\phi}_{max}$
}

Note that even if Parameter Scheduling seams more complex that standard PID-controller, the implementations is likewise (see the chapter about software).

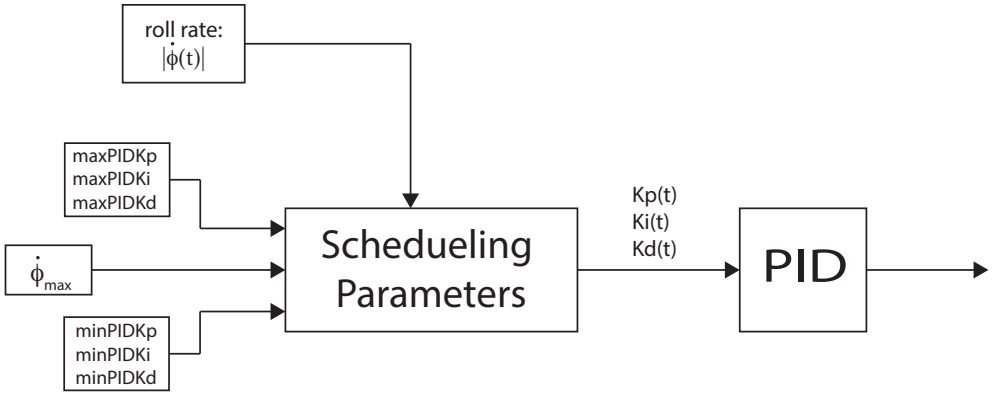


Figure 4.2: Illustrating image of Scheduled PID

5

Simulation Model

In the project *Matlab* and *Smulink* is used for computation and simulation of the UGV. It gives a good aspect of how the different states in the model depend on etch other.

5.1 DC-motor

The DC-motor model is built from the mathematical model in Chapter 3. The output from the model is the *armature-resistance* (i_a), *angular velocity* ($\dot{\theta}$), *angular acceleration* ($\ddot{\theta}$), *torque* (τ) and $torque_2$ (τ_2).

τ_2 is the output from the summation from the electric-torque and the mechanic *torque*. Where the mechanical torque is generated from the equation, $\theta \cdot b_m$ and b_m is the friction in the DC-motor.
In fig.5.2 the feedback of the parameters in the DC-motor is shown. The saturation of the voltage and current is based on the max-Voltage input to the DC-motor [**Appendix B**] and the stated max-current of the motor-driver [16].

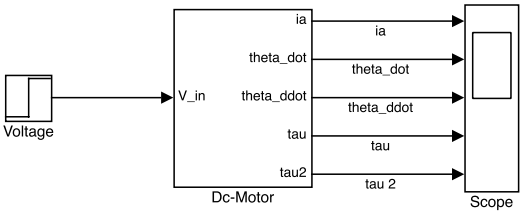


Figure 5.1: DC-motor block

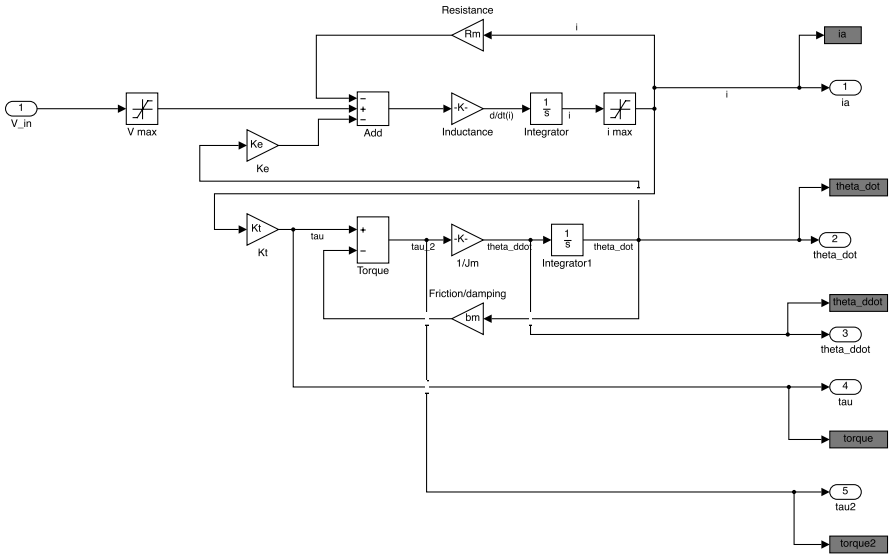


Figure 5.2: DC-motor model

5.1.1 DC-Motor with external Load

In order to use the DC-motor with the UGV-model, the external-load and external-friction/damping from the UGV's gears, axis and wheels must be considered. These parameters become inputs to the DC-motor system (fig.5.1) and together they build up a new model, shown in fig.5.3

In the new subsystem (fig.5.4) $function - block$ is used to calculate the summation of the internal and external -inertias $(J_m + J_L)$ same as internal and external -friction $(b_m + b_L)$.

A new differential equation (equation.(5.1)) is now derived using equation.(3.11), and the additional external parameters.

$$\frac{d}{dt}\dot{\theta}_a = \frac{1}{(J_m + J_L)} \cdot (K_t \cdot i_a - (b_m + b_L) \cdot \dot{\theta}_a) \tag{5.1}$$

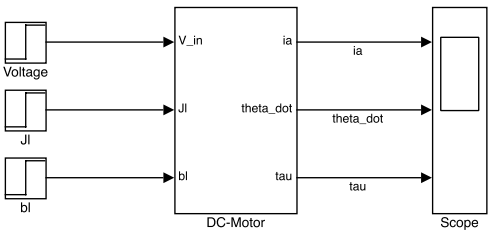


Figure 5.3: DC-motor block, with external Load and friction as input

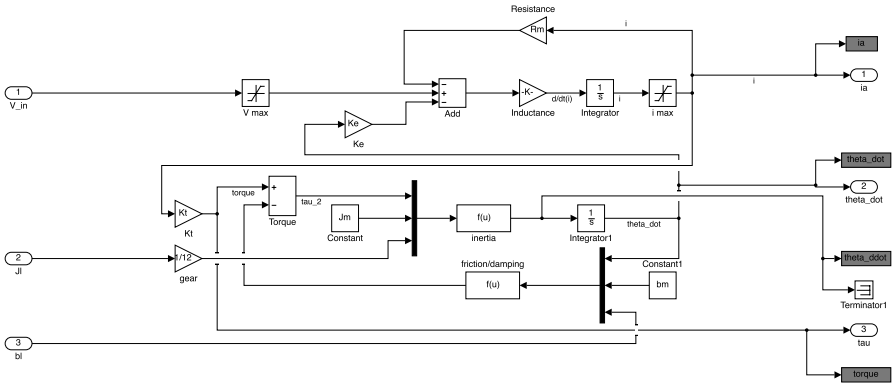


Figure 5.4: Subsystem, of the DC motor with external inputs

5.2 UGV-Model

The UGV/vehicle dynamic model in *Simulink* is built up by several blocks.

The full view, shown in fig.5.8, consist of, DC-motor-, Tyre-, chassi- and loadtransfer-blocks.

The inputs to the system are the steering angles (δ_f and δ_r) and torques (τ_f and τ_r) to the drive axis.

5.2.1 Tyre

The tyre model has two blocks (fig.5.5), one for the calculation of the forces due to the tyres slip-values, (lateral slip) and the inputs to the system. The second block is to map the forces of the tyres to the body-frame.

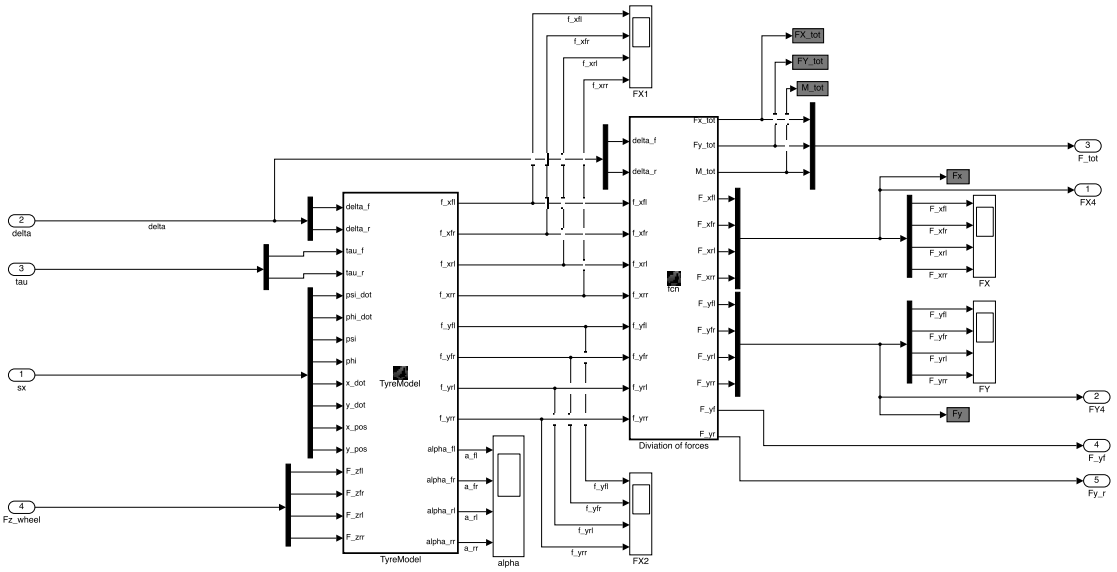


Figure 5.5: Simulink part of the Tyre

5.2.2 Chassis

In the chassis-block the dynamic model of the UGV is stated. The output from the differential equations becomes the states and feedback to the system.

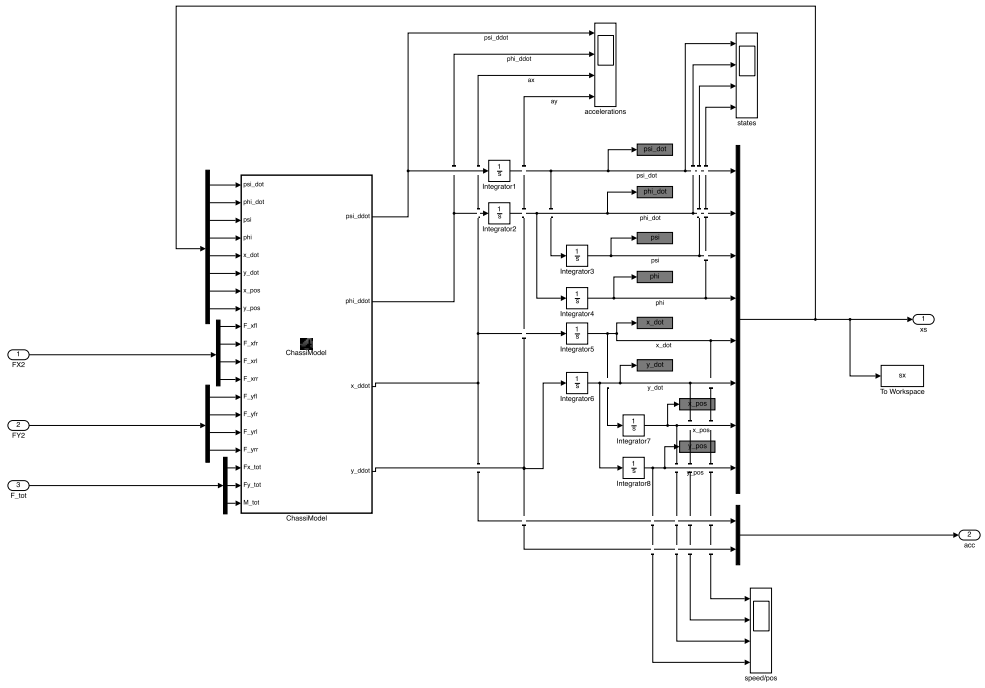


Figure 5.6: Simulink part of the Chassi

5.2.3 Load transfer

The load-transfer-block calculates the normal load on each wheel. The normal load is feedback to the tyre model, and is used to calculate the tyre forces, for each wheel.

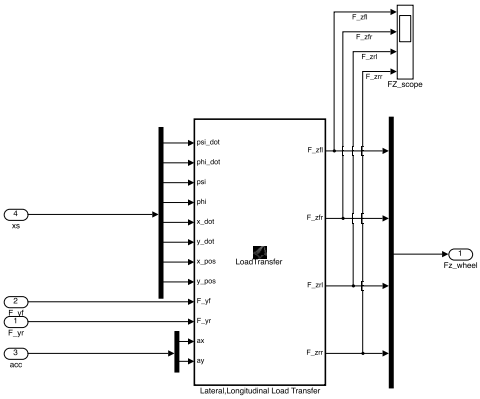


Figure 5.7: Simulink part of the Loadtransfer

5.2.4 Complete model

The separate parts are connected together and simulations with the DC-motor as actuator can be done.

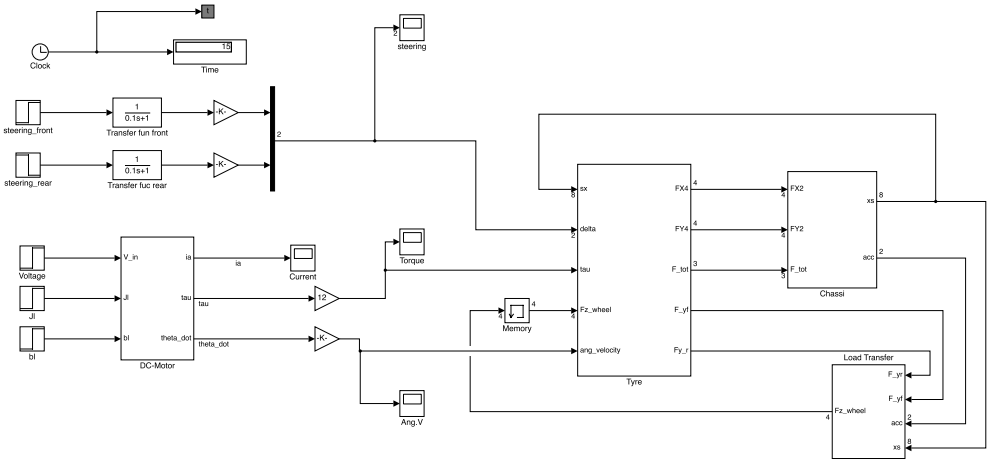


Figure 5.8: Simulink part of the Loadtransfer

6

Hardware

This Chapter presented the hardware which is used in this project, the steps which were taken to develop a suitable Unmanned ground vehicle (UGV) for the project, both mechanic and electronic are presented.

6.1 Mechanics

Drawings for all the manufactured parts are in **Appendix A**].

6.1.1 1/8 Scaled RC-car

The project was from the project-plan built on a radio controlled car (RC-car). The type of RC-car is known as a Rock Crawler. It has the benefits of being robust and manage some terrain.
Except the motors and batteries, the UGV must be able to carry some weight, therefore a reconstruction of the vehicle is made because the original platform was smaller and weaker than expected.
The redevelopment includes extension of the drive shafts, larger shock-suspensions, a new body section including new servo holder, so that the entire width of the vehicle and length were increased.

Figure 6.1: car before

6.1.2 CATIA Parts

The various new parts are designed and assembled in CATIA, this is to ensure that everything fits together before the real parts are manufactured.
The several parts are exported from Catia to *.dxf* files, which is the input file for CNC milling machines, 3D-printers and Laser-cutters. CATIA also provides all the mechanical data about the UGV, dimensions, inertia and weight, which are used in the Matlab/Simulink models.

6.1.3 Axis-Extension

The spacers that make up the extension is made of steel, the rest of the parts of the vehicle are made of aluminium. It makes it become very light. Therefore the selection of material for the spacers is based on the fact that steel is durable and heavy, so adding the steel-parts helps giving some weight to the UGV. More weight in a vehicle's base, results in increased stability [7]. The extension is in two parts, one inner and one outer. The inner part is the extension for drive-axis, the outer part is the extension that holds everything together.
The extensions are cone-shaped, shown in the fig.6.2, so that it can be fixed hard to the transmission box. The total width of the car, measured from the center of the tyres becomes 43cm, it has then increased 15cm from the original width.

Having a wide base is a good basis to maintain stability during cornering [9], the risk of roll-over is also reduced. It also allows the robot to carry larger objects, which is one of the specifications from FOI. [**Appendix A**]. The assembled part is illustrated in fig.6.7.

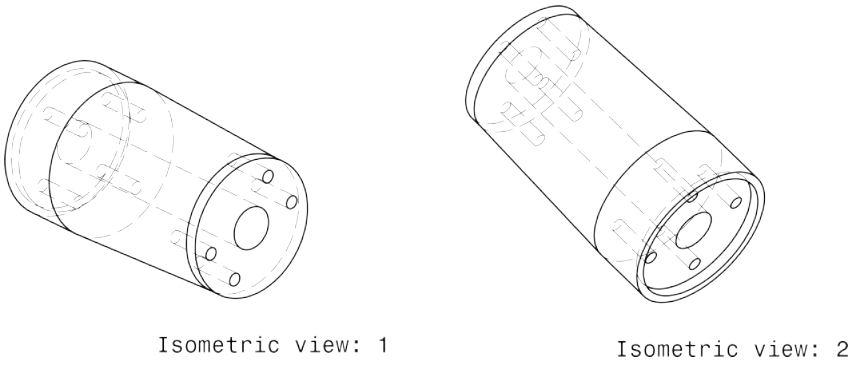


Figure 6.2: Axsis extention outer view

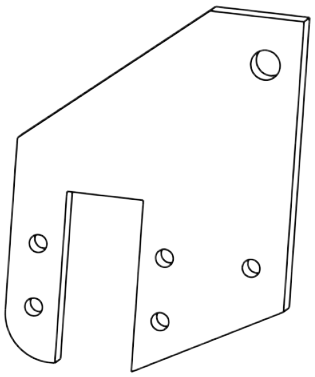
6.1.4 Suspension

FOI's specifications tells that the UGV is to be loaded with equipment that has some weight. The suspensions which follow with the RC-car where very small, soft and not strong enough to hold the small original body of the car. New, larger, stronger suspension from HSP is used instead [17].

These are for 1:5 scales RC-cars and will manage more weight than the original suspensions. Image.6.3 shows the size difference between the old and the new ones.

To make the new suspensions fit the wheel-axis, a chock-suspension holder is made (fig.6.4). Where the 10mm ball-link from the suspension can be mounted. The strut form the midsection is also mounted in the shock-suspension holder, therefore several mounting holes are made in the holder.

Figure 6.3: shock-suspension Old and New one



Isometric view

Figure 6.4: shock-suspension and strut holder

6.1.5 Servo Holder

The servos that are used in the project are much larger than the mountings are on the original RC-car. Construction of new servo holder is done. Due to the fact that the new servos pulls 35kg on 1cm [**Appendix B**], the servo holder must be robust and durable. In fig.6.5 and 6.6, the tow parts that builds up the servo holder is presented.

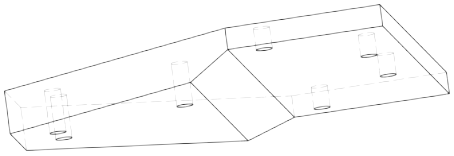


Figure 6.5: Servo mounting plate

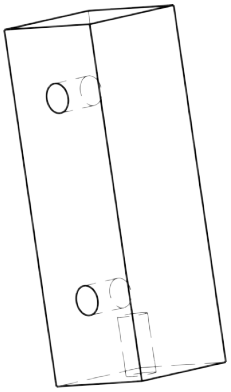


Figure 6.6: Servo stag

6.1.6 Assembled drive axis

The several parts (shock-suspention holder, axis extensions and servo holder) which are to be mounted on to the driving axis are presented in an assembled model from CATIA.

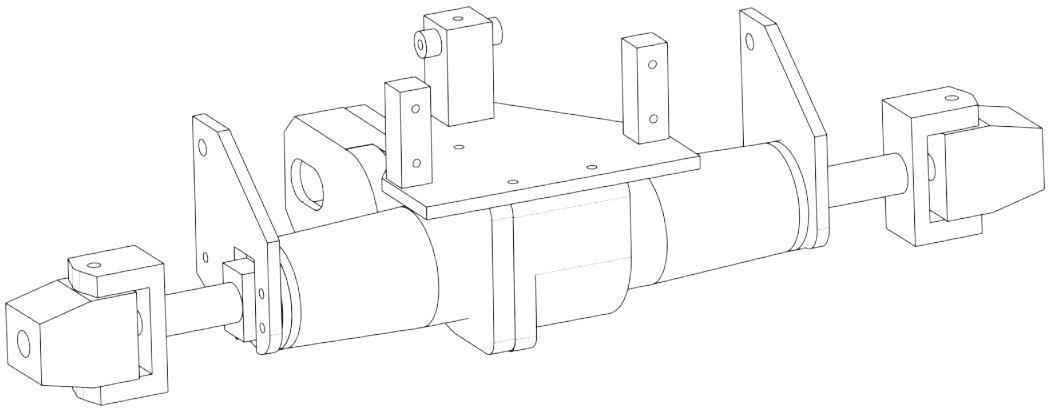


Figure 6.7: Assembled parts on the drive axis

Catia data provides the inertia of the internal drive-axis and the two wheels mounted on it. This inertia is the external load that the DC motor has as input.

$J_L = 16.1266 \cdot 10^{-5}$ $[kg \cdot m^2]$ (Inertia for tow wheel + axle)

6.1.7 Midsection (Body)

Mid section of the car is replaced with a new design that is longer, and higher, so that the electronics and batteries can be placed within the mid-section, and FOI's sensors on top. The aluminium is cut-out in CNC-miller to get the accuracy that is needed to fit the parts together. 5mm thick aluminium insures that it will be robust, the bottom-plate and the strut holds the midsection together.

Midsection/Sides

The sides are mirrored in a symmetrical-line in the middle of the part, fig:6.8. In that way it is easy to mount and easy to replace. There are several mounting holes for the struts, that is to be connected down to the drive-axis. Depending on what holes the strut are mounted in, the elevation of the UGV, midsection (body) is ether increased or decreased.

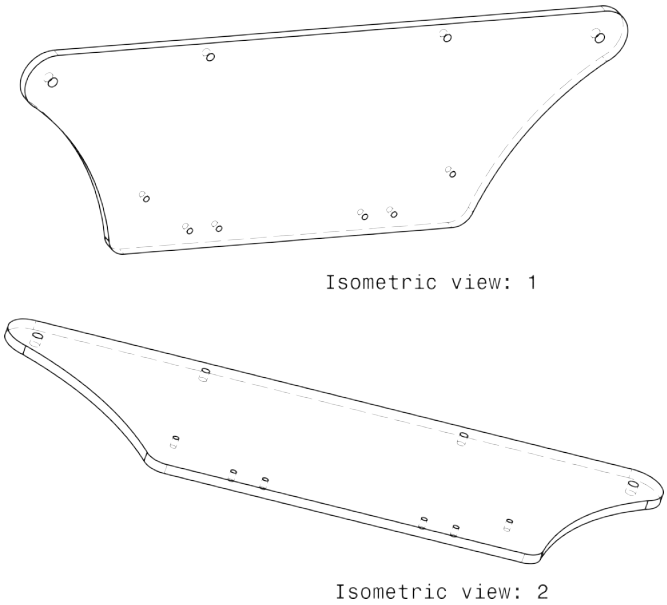


Figure 6.8: One of tow sides that builds up the mid-section of the UGV

Midsection/Bottom

The bottom plate of the midsection is cut-out so that a standard hard-case 3-cell LiPo battery at 5500mAh [18] can be placed without and risk of sliding sideways, see fig.6.9 for the placement of the batteries. The bottom plate also helps holding together the midsection .

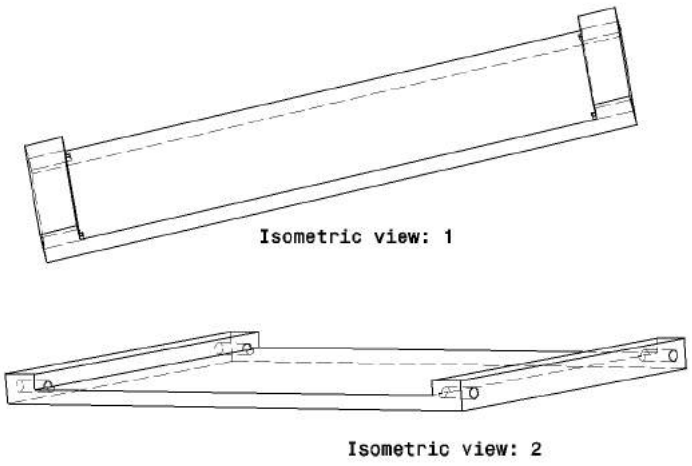


Figure 6.9: Bottom plate of the midsection (body)

Midsection/Strut

To increase the stability of the midsection, and to protect the electronic that is mounted within the midsection, two robust 16mm struts are designed (fig.6.10).

Figure 6.10: Strut of the midsection (body)

6.1.8 Assembled design

The assembled design of the CATIA-model. Note that the shock-suspensions are not in the design. This due to the complex form of the shock-suspensions. Within the midsection a block is placed, this is to illustrate the LiPo-battery.

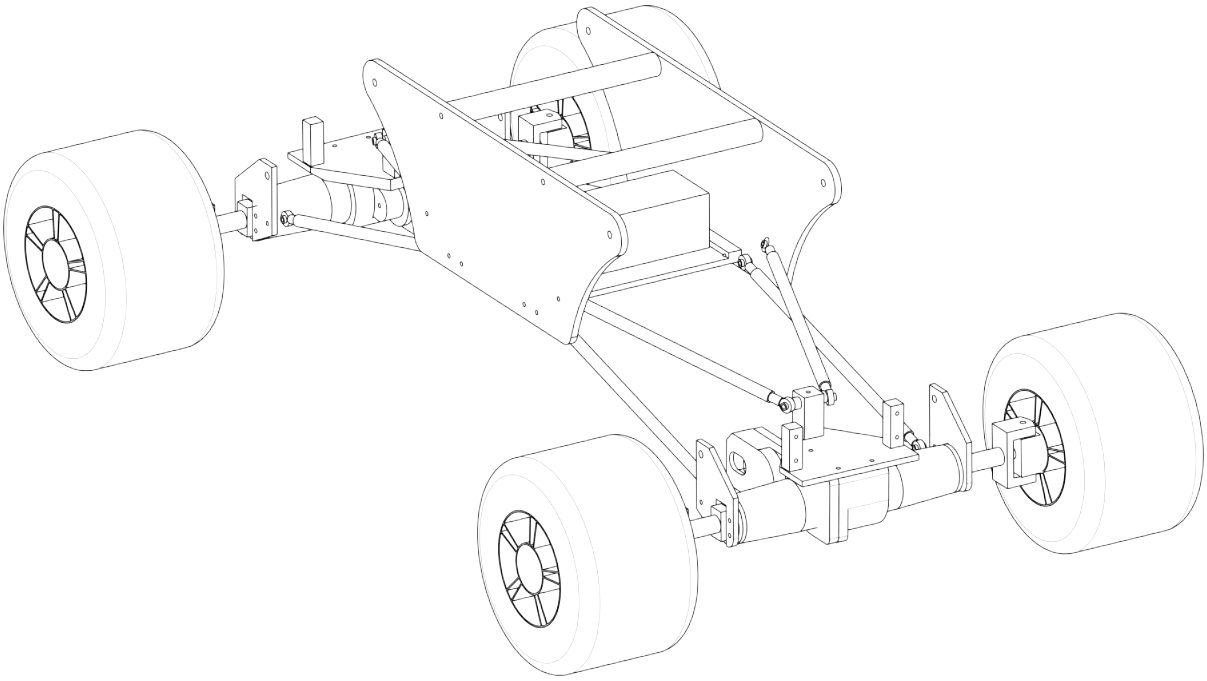


Figure 6.11: UGV assembled, without shock-suspension

6.1.9 Reaction Wheel

For the project an reaction wheel is cut out. The maximum diameter the reaction wheel is allowed to have is $\approx$ 16 cm, which is the spacing between the body-side parts where it is to be mounted.

In order to yield a high inertia with low mass, the total amount of mass should be placed at the radius of the reaction wheel (see section 3.5). Therefore the material of the reaction wheel is chosen to be steel due to the fact that is has a high density and it is durable.

A steel-ring is cut out and mounted on a plastic disc. As stated, as low mass as possible in the center of the disc, therefore several holed are cut out in the disc. To get the accuracy, laser cutter was used.

Figure 6.12: Plastic Disc, the base for the steel-rings

In the project one steel-rings was cut out in the CNC milling machine.

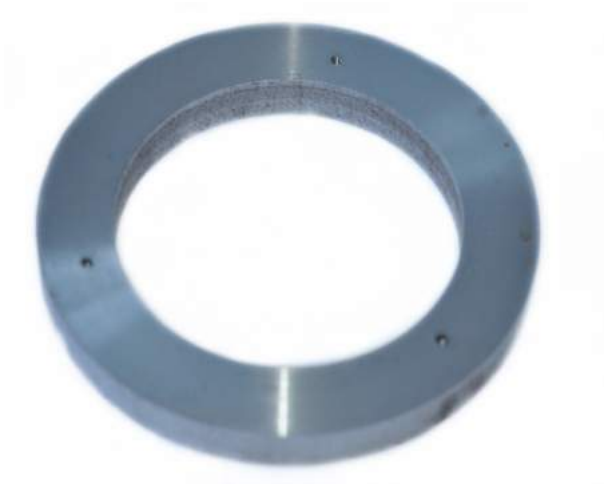


Figure 6.13: Steel-ring uses as an Reaction Wheel

By using the equation from chapter 4, the calculated inertia is: $9.1553 \cdot 10^{-}3\ [kg \cdot m^2]$

6.1.10 Final Build

The parts that is constructed are mounted together, image.6.14 and 6.15 presents the results. With this type of mechanical design large flexibility is derived. The UGV will manage to climb obstacles easier.

Figure 6.14: UGV rear view

Figure 6.15: UGV front view

6.2 Electronics

In this section the electronic that is used in the project is presented. There are several more parts, but non of them are necessary for the projects outcome, therefore they will not be presented in this report.

6.2.1 MCU (Micro Controller Unit)

The projects software and hardware are based on *Arduino* (open source) [19].

The advantages with Arduino is its extended library that is available for download. Arduino's hardware is based on the Atmel, Atmega-chipset, where several different microprocessors are available. In the project Atmega328p is used as the slave processors.

Specification for the main processor is that it has to be fast, have a big memory and several I/O ports. Atmegas ATSAM3X8EA [20] in figure.6.17), has a clockspeed of 84MHz (compared to the usual 16MHz on other Arduino boards). It also features 32bit M3-cortex, several hardware interrupt pins (all 103 I/O ports), an internal *real time clock* (RTC), which can be powered by a coincell-battery (RC2032), and can therefore manage the MCU to go in to a low-power mode when needed, which makes it suitable for the project.

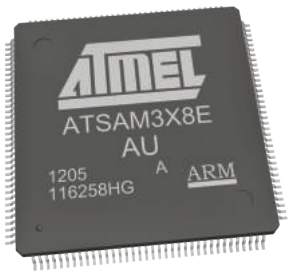


Figure 6.16: Atmel's ATSAM3X8EA-AU MCU

Atmega328p slave processor. This one is used for the rotational encoding.

Figure 6.17: Atmel's atmega328p-AU MCU

6.2.2 PCB

In this project several processors are needed, several H-bridges (DC-motor controllers) to the DC-motors, connections to encoders and sensors. It is then easier to make a special designed PCB, rather than to have several Arduino-boards with wires connected between them and the sensors.
Using PCB-design program from Cadsoft, Eagle PCB [21]. Their free-ware is a non-profit licence and only manages tow layers. With the CAD-program special designs can be done and be sent to various companies for printing.
The PCB is designed, so it gets the ability to have module feature. In this way the future expansion of the UGV can be done and parts can easier be replaced. In the PCB all the pins that are not mainly used is "pinedout" to several pin-arrays, so they might be used for other external features.

In addition there are pin-arrays with +5V, +3.3V and GND on the board, for external sensors. The ATSAM3X8EA has three I^2C-buses. The on-board slave-MCU (*atmega 328p*, schematic in **Appendix B**) is connected to one of them 8SDA1 and SCL1 on ATSAM3X8EA with a I^2C bi-directional level-shifter (*NXP-PCA9306D*) in between, due to the fact that the slave-MCU runs on +5V and the main MCU runs on +3.3V. The reset-button on the PCB resets all MCU's and the bluetooth.

The on-board slave-MCU is connected to the motor-controller's temperature sensor, the motor-controller cooling-unit and the li-Po battery cell reader (used for measure the battery-level). Each cell on a standard Li-Po battery has the maximum value of +4.2V, therefore the slave-MCU is used instead of the main-MCU, due to the fact that the main-MCU only handles voltage levels up to +3.3V, feeding an I/O pin on the main-MCU with higher voltage level may damage the MCU. The on board slave-MCU works as a "*observer*" and is meant to have one standard program, note that the pin's not used is pin-out, so that the user can use them. In that case the user must re-program the slave-MCU.
An CR2032 con-cell battery-holder is added, and is designated for the internal RTC

(real time clock), the MCU can then be set to low-power mode and execute function on a preprogrammed schedule.

In the report parts of the schematics will be presented, but not the full schematic [**Appendix B**].

6.2.3 DC-Motors

In this project brushed DC motors are used. Two Maibuchi RS550VS, with high torque for driving [**Appendix B**]. And one Pololu geared DC-motor as actuator to the reaction wheel [**Appendic B**]
The beneficial is that a simple H-bridge can be used to control the motors through a micro controllers *PWM signals*.

Figure 6.18: Mabuchi RS550 Motor

Figure 6.19: Pololu geard DC-motor

6.2.4 Motor Driver

As stated before, brushed DC motors are used in the project. To control the DC-motors, a motor-controller is needed. $ST - Microelectronics$ double H-bridge is used as the driver circuit [16], schematics in [**Appendix B**].

The selection is made from the stated maximum input voltage of the DC-motors and the simulation results of the DC-motor, the start-up and steady state current with external load.
The MCU uses PWM signal to the motor driver, with this method the motor is able to have high torque at low rotational speed.

Figure 6.20: VNH2SP30 chipset

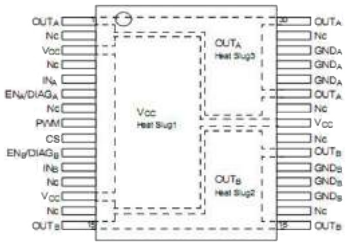


Figure 6.21: VNH2SP30 pinout

6.2.5 Active cooling

By using a LM335, which is a temperature sensor from *Texsas Instruments* [22], the temperature can be measured on the PCB. Due to high current in the VNH2SP30 IC, there is a risk of overheat. It makes the soldering to loose and there is a risk that the IC "floats" away from its position. Note that for extreme currents there is also a risk of burning the IC.

To prevent over-heat, a CPU-fan is placed over the tow VNH2SP30 IC. The fan has a controller that takes the temperature from the LM335, and has the fan speed as output, it will cool off the VNH2SP30 IC. The temperature sensor is placed just between the VNH2SP30 IC, shown in fig.6.23

Figure 6.22: Fan with heatsink 40x40 mm

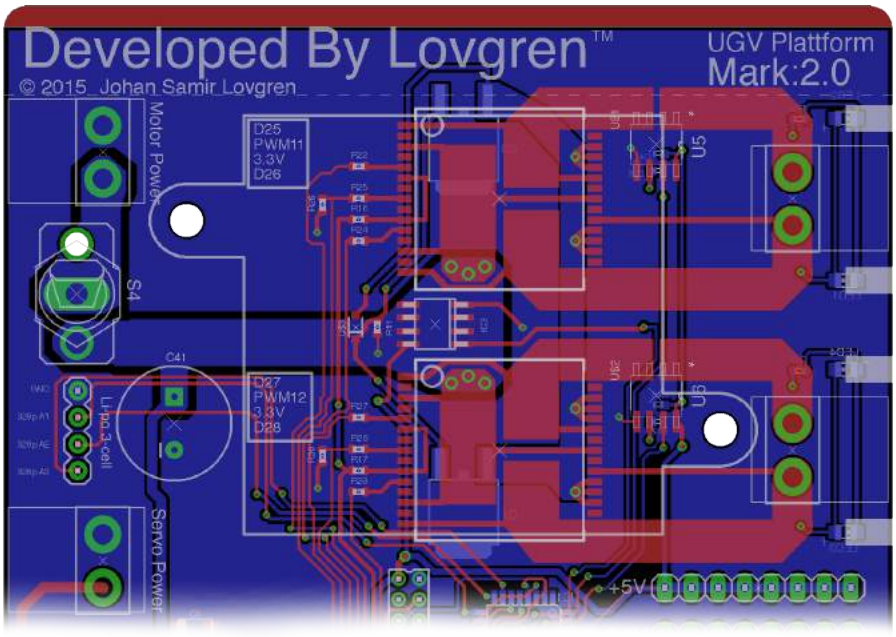


Figure 6.23: Motor controller part of the main pcb

The largest white square over the two VNH2SP30 IC, is where the CPU-fan is placed. The fan has tow spring-clippers that is pressed through the tow mounting holes which keeps it pressed to the VNH2SP30 IC.

6.2.6 Rotational encoder

In order to verify the Matlab-Model the angular velocity of the wheels are needed to be logged. In many other projects odometers and rotational encoders are used, the rotary encoders that are most commonly used are the optical encoders [23]. These are cheap and parts can usually be taken from old computer mice to build one. However, these are fragile and is the combination of many more parts compared to a magnetic encoders. Some of the features of an magnetic encoder.

Some of the advantage of using magnetic encoders,

- Galvanic separated towards the DC motor.
- Ideal in harsh environments due to contact-less position sensing.
- The magnet is durable.

Magnetic encoder

Austria Micro Systems IC chipset, AS5040 is a 10bit 360 deg programmable magnetic rotary encoder that has everything needed to read the magnetic field from a rotating permanent magnet [24].

The AS5040 has a quadratic output. It gives the ability to detect not only speed, but also in which direction the rotation is. It is essential to know in what direction the wheels are spinning.

Simplified explanation. The IC circuit has an hall effect sensor inside. This is the part that detects a magnetic field, in combination with operational amplifiers and flip-flops, it creates pulses corresponding to the rotation of the detected magnet.

In fig.6.24 there is a round magnet that is dimetric magnetised, witch is the way the magnet has to be against the IC circuit for it to be able to scene the magnetic field difference.

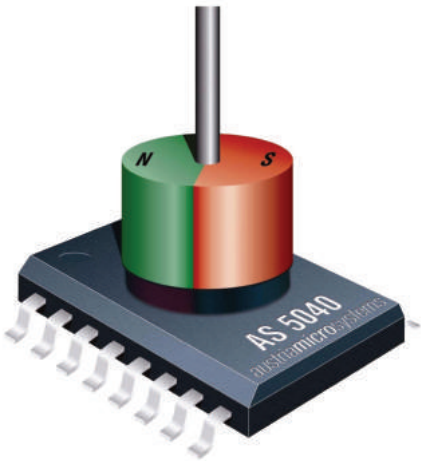


Figure 6.24: The IC circuit and an rotating permanent magnet (image borrowed from [24])

The stronger the magnet , the larger the difference between the positive and negative field is illustrated in fig.6.25 , in with the IC circuit is able to estimate the position easier. Earth Magnets are preferred according to specifications [24].

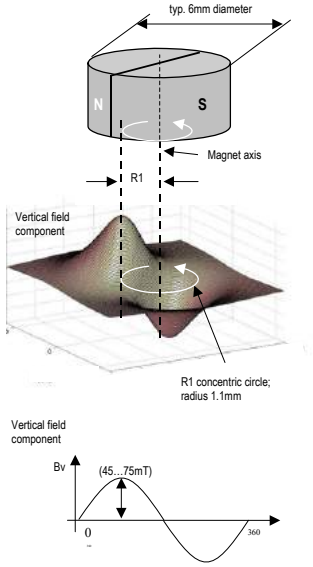


Figure 6.25: Magnet and magnetic field distribution (image borrowed from [24])

Assembly

For assembling, a PCB-CAD that fits the mechanics was done. A two layer PCB is designed with all the pin-out that is needed for reading the pulses from the sensor. Three Chip-LED is added and is used as an indication when the IC is centred above the magnet. It also indicates when the magnet is too far or too close to the IC.

In fig.6.31 the PBC is presented. The round shape and the mounting holes fit the *Maibuchi RS550VS* DC-motor. As shown in fig.6.27, the magnet is glued on to the front of the DC-motors shaft.

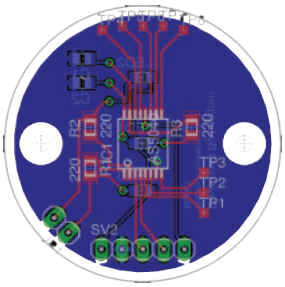


Figure 6.26: Designed PCB for the rotational encoders

Figure 6.27: Magnet fixed on the motor shaft

Figure 6.28: Implementation of encoder

In fig.6.29 the maximum speed at 11.1V is plotted in radians per seconds, note that the rotary encoder is mounted such that it is the DC-motors rotational speed that is measured, not the wheel.
The fig.6.30 shows that the DC-motor has higher angular velocity in one of the tow directions. The brushed DC motor has a rotational direction that is preferred [**Apendix B**], this is due to the carbon brushes in the motor that drag against the motor coil. If the rotation is in the non preferred direction the carbon brushes will give higher friction against the shaft, and the result will be a lower angular velocity.

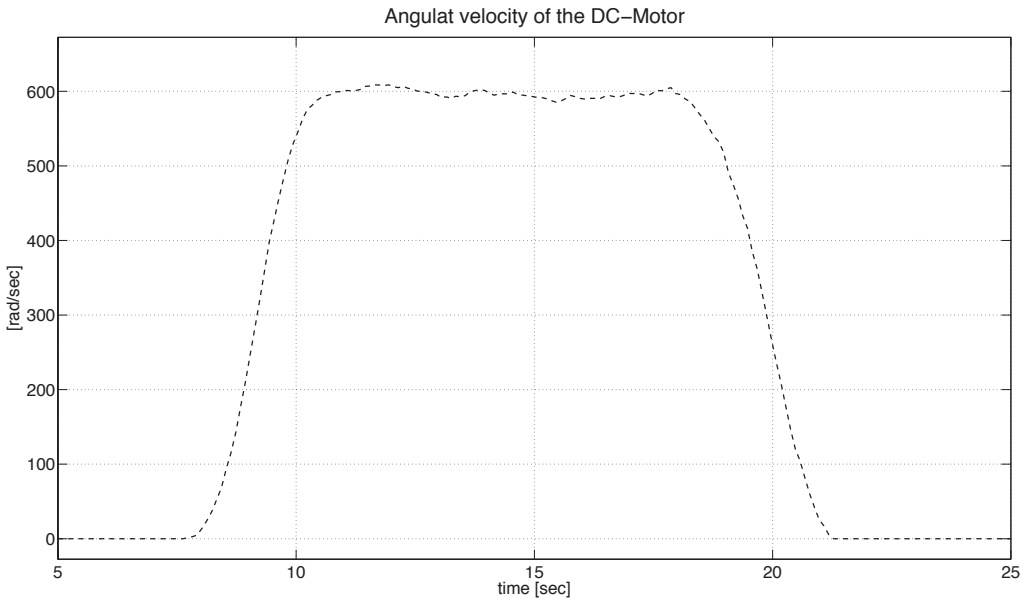


Figure 6.29: Angular velocity of the DC-motor (forward direction)

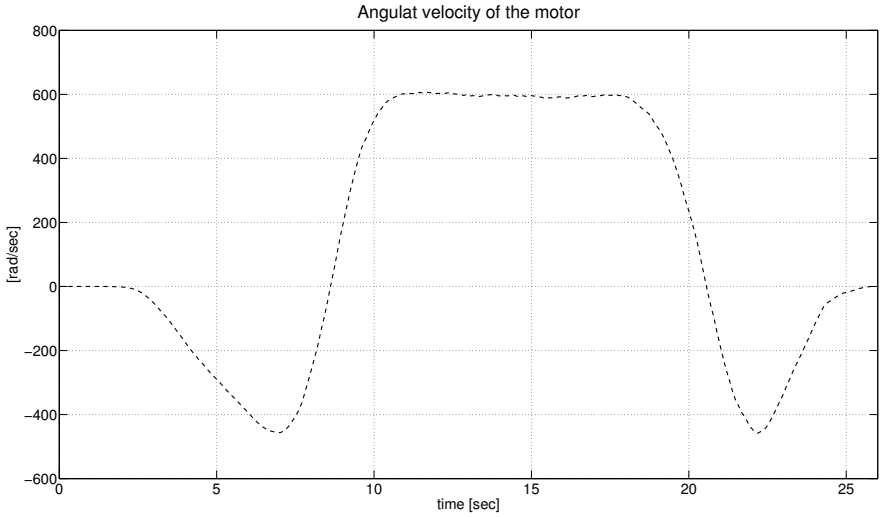


Figure 6.30: Angular velocity of the DC-motor (reverse and forward direction)

6.2.7 Servo

The steering actuators are servos from the manufacture *Savöx* [25]. Due to the size of the UGV, the size of the wheels, it is necessary to have a powerful servo. *Savöx 0235* manages to pull 35kg at 1cm, and has the speed of 60 degree/100ms [25].

Figure 6.31: Servo used for steering (image borrowed from [25])

The servo control signal has a minimum voltage level at 6V, the microprocessors TTL-levels are 5V. Therefore a servo-driver had to be constructed to get the MCU's signal to the right level in order to operate the servo [**Appendix B**].
The servo has maximum voltage level at 7.4 volt. By using a separate power source with the same level to drive the tow servos no extra power-regulator is needed. An two-cell LiPo battery at 7.4 V is used as source for the servos.

6.2.8 Integrated Measurement Unit (IMU)

IMU is a chip-set with sensors that can be used as input to the system-plant. It consists of 9-axis, three each from, *accelerometer*, *gyro* and *magnetometer*, which in turn corresponds to the three axis, X, Y and Z (fig.6.32). By using sensor fusing between this three parts, estimation about angle, direction, and angular rate can be made.

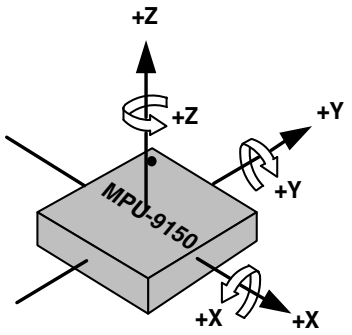


Figure 6.32: X,Y,Z -axis of the IMU sensor (image borrowed from [26])

Placement of the IMU

The IMU is placed in a plastic hard case cover. And is mounted according to the CATIA models COG 6.33, which is just underneath the bottom plate (the belly of the UGV). It is essential that the IMU is mounted close to the COG, so that calibration and correction are avoided due to misplacement. Note that the case cover is in plastic, so that the magnetometer dose not get shielded.

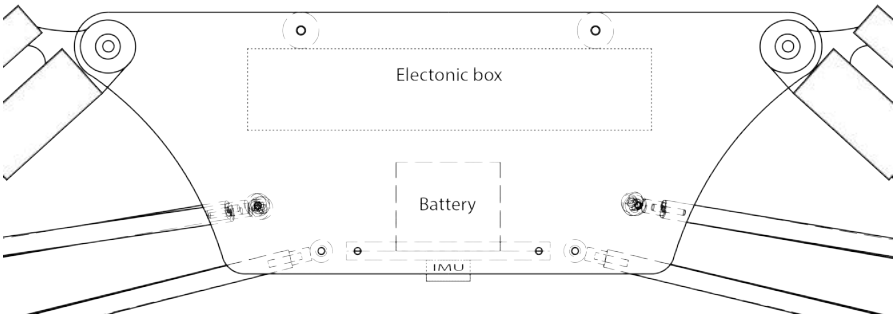


Figure 6.33: Side view; Placement of IMU on the UGV

The MPU 9150

In the project a MPU-9150 (fig.6.34, borrowed from [26]), from the manufacturer *InvenSense* is used [26] . The MPU-9150 has a InvenSense's MotionFusion (fig.6.35), which delivers optimal raw data to the user. The MPU 9150 has a 1024-byte FIFO register, this register is accessible via serial interface. The user can use this to get interrupt when new data from the three sensors are available. The DMP (Digital Motion Processor) gives a Motion-Fusion for all the nine axis, it offloads the computation of the motion processing algorithm from the host MCU.

Figure 6.34: MPU 9150 from InvenSense

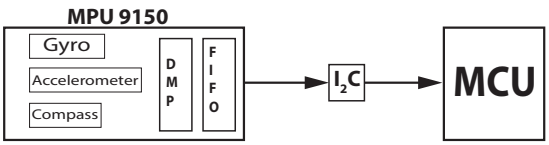


Figure 6.35: MPU 9150's flowchart

Understanding RAW data from the MPU

Measure Yaw-,Roll-,pitch- rate is the same as taking the gyroscopic data from the IMU.

The raw data that is processed and sent is scaled, due to the default measurement range and the 16bit ADC. Therefore the raw data has to be transformed to get the right

form (SI-standard).
In order to transform the the data, the *range* and the *resolution* of the measured unit has to be known. According to data sheet [26] the MPU 9150 gives ± 250 deg/sec as default. And that the gyroscope has 16-bit Analogue to Digital Converter (ADC).
In order to get the gyroscopic data in [rad/sec] the range of the MPU's gyroscope-unit is divided with the numerical range of the 16bit ADC (signed) shown in equation.(6.1).

$$[deg/sec] = raw_{data} \cdot \frac{250}{32768.0} \tag{6.1}$$

$$[rad/sec] = [deg/sec] \cdot \frac{\pi}{180} \tag{6.2}$$

Validation of the Sensor

To verify that the parameters are correct, and properly scaled. An external IMU is compared with the MPU. There are several sensor on the market that can be rented for a high amount of money. One way of use a non expansive way to do this is to use a smart-phone. Almost all smart-phones has there own IMU implemented, there are several applications that plots the measurement units data. The application used in this project is named *gyroRecorder* and is developed by OS-Craft [27].

Placing the phone at the same spot as the MPU, and record the data from the two sensors, comparison can be made.
Not that the used smart-phones maximum sapling rate is 2Hz. This will give some variation to the signals in the time line. But the objective is to see if the stated scaling is correct or not.

In fig.6.36, the blue dotted line is the Yaw-rate recorded from the IMU-sensor, and the black line is the recorded data from the Iphone4. One can clearly see that the two data lines corresponds well. And assumption that the scaling is correct is done.

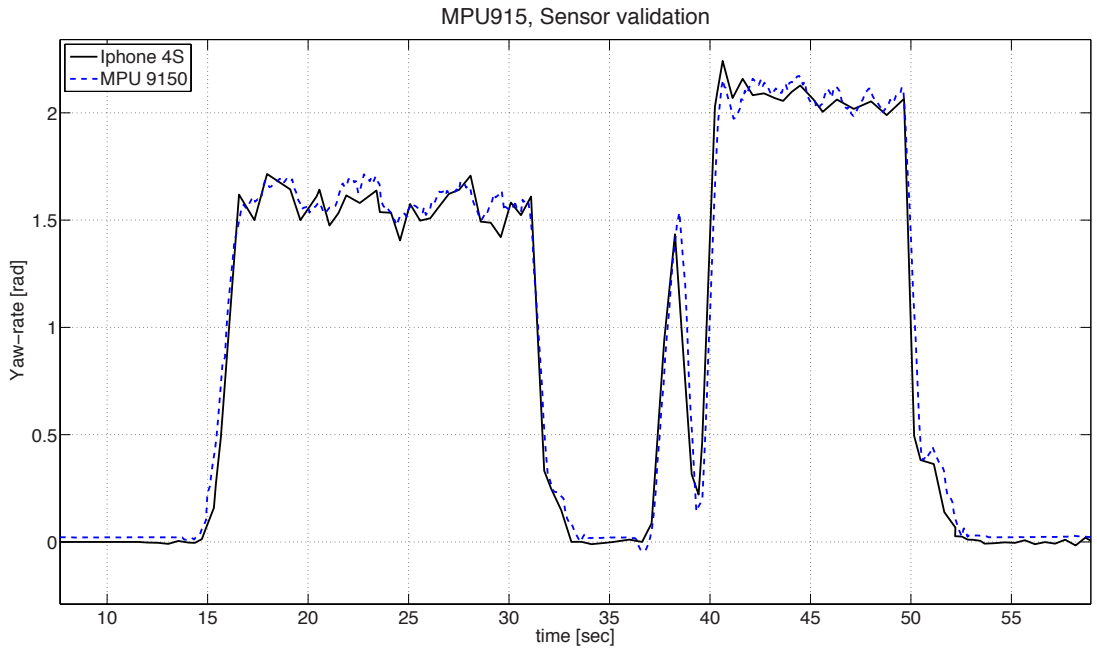


Figure 6.36: MPU vs Iphone.4 results

6.2.9 Bluetooth (BT)

Bluetooth is used to send live data from the sensors to MATLAB, the data is sent via serial-communication. In the MCU a code sends the data, and a Matlab script handles the received data.

Roving Networks RN42 Bluetooth-module is used, the module i programmable, and can be adjusted to fit the project's need. Bluetooth V.2.0 and have a range up to 25 meters [28]. Due to the fact that the project is a prototype, telemetry dose not need to be long range. In future expansion of the project the bluetooth can be replaced with long range telemetry. The images (6.37 and 6.38) below shows the module and its pinout (images are borrowed from [28]).

The blue tooth module is added to the PCB and is connected to the main-MCU's serial-port no:0 (*TX0 and RX0*). In addition the factory-reset on the bluetooth is connected to the main-MCU, factory reset is may needed when reprogramming the bluetooth. An smd slide switch is used as ON-OFF of the bluetooth, it saves power to switch-off the bluetooth when not needed.

Figure 6.37: Roving Network's RN-42 Bluetooth module (picture borrowed from [28])

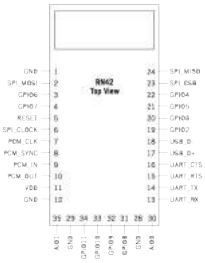


Figure 6.38: Roving Network's RN-42 Pinout (picture borrowed from [28])

7

Software

7.1 Costume Bootloader

In able to use the Arduino's version of C++, Arduino's own bootloader has to be "burned" in to the chip-set. Every version of Atmels chip that is available for Arduino has its own bootloader. Due to the fact that the chips have several different features, such as, number of I/O ports, buses and clock-speed, each one of them must have a bootloader of its own.
The bootloader is available in Arduino's software [19]. The advantages of this is that it is easy to rewrite the existing bootloader in order to get the features which are best for the project. In some cases it is necessary to save power when the microprocessor is not executing any tasks. It is then good to write a special bootloader that features this sort of functions. The bootloader for this project will not be presented in this report.

7.2 Main Loop

The main loop is constructed only of calls to the different functions in the same order as the presented flowchart in fig.7.1. The functions dose the following,

- *readReceiever*, read the operators command.
- *moveMotors*, send the operators command for throttle to the motor controller.
- *moveSteering*, check if the operator wants to use the back-wheel steering. Send the operators command for steering to the steering-servos.
- *updateMPU*, read the IMU sensors values.
- *updateRWPID*, update the reaction wheel PID-regulator parameters.
- *calculateRWPID*, calculates this cycles output to the reaction wheels DC-motor.
- *moveRW*, send the calculated output for the reaction wheel to the motor controller for the reaction wheels DC-motor.
- *updateWheelPID*,update the back-wheel steering PID-regulator parameters. And send

the output to the back-wheel steering servo.
• *updateRWStatuses*, call slave processor no.1, and get the rotational speed and position of the reaction wheel.
• *updateMotorStatues*, call slave processor no.2, and get the rotational speed and position of the front and rear wheels.
• *currentRead*, read the current of the three DC-motors.
• *sendData*, send all the data from the UGV to Matlab.

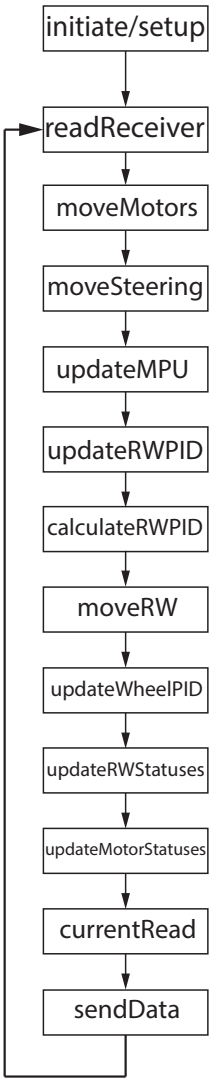


Figure 7.1: Flowchart of the main loop

7.3 MCU to Matlab

In the project it is necessary to have the ability to send live data from the UGV. To yield this a communication protocol is to be created.

The hardware part of the communication is a blue-tooth module, blue-tooth version 2.0. It has a maximum range of ≈ 25 meters, it works well during the project, but for future development, a telemetry with greater range is more profitable. The blue-tooth is configured so that it becomes a serial port when paired to a computer.
Matalab-scipt is created that reads the flow of the serial port. Note that the MCU must send the data in the right structure to Matlab, else the data will be stored wrong. [29]

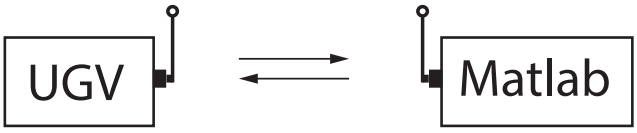


Figure 7.2: communication between UGV and Matlab

7.4 MPU

The MPU, has to be configured via the connected I_2C bus from the MCU at start-up. The complete test code is in Appendix A.

7.5 RC-receiver

The operator controls the UGV with a 7-channel RC-controller. The ports from the receiver are connected to the MCU, and the signals are interpreted by software code. The standard RC-controller gives out pulses that has the time spacing of 20 milliseconds (ms) and the amplitude of the maximum logic-level, which on the receiver is 5[V].

The pulse have the range of $1ms \leq pulse \leq 2ms$ as shown in fig.7.3. Where 1 ms is full reverse or brake, 1.5 ms is the neutral pulse and 2 ms is full throttle.

For one channel the spacing is 20 ms between the pulses. If the same thing were done on all seven channels it would would take $7 \cdot 20 = 140$ ms to read them, but that is not the case it only takes 20 ms.

The reason for the 20 ms gap between the pulses is for the ability to use *Pulse Position Modulation* (PPM). It is basically several PWM signals lined up, each one for every channel of the receiver. By using a oscilloscope the signals were viewed and an illustrating fig.7.3 were made.

In fig.7.3 a single- and seven- channels are viewed, channel three changes pulse-width, to illustrate that all of them are independent.

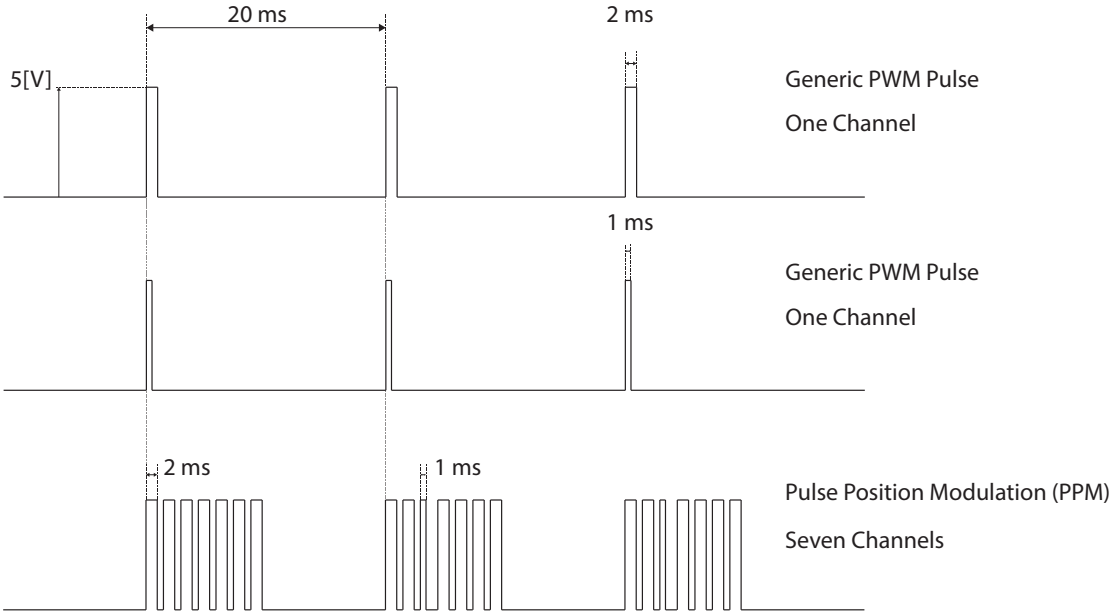


Figure 7.3: On channel and seven channel. With the PPM

The code is structured so that the data is between 1000 - 2000 pulses. Note that the data from the receiver cant be used to drive the motors. In order to do so the data is mapped to PWM signal as in equation.7.1.

$$\begin{cases} move = map(inChannel, 1000, 2000, -500, 500); \\ move = constrain(move, -255, 255); \end{cases} \tag{7.1}$$

The tow lines do the following,
· center over zero (1500)
· only pass values whose absolutes are valid PWM values, which is between 0-255, the sign just tells us which direction.

7.6 Hardware interrupt for Encoders

The rotational encoders give quadratic pulses with an correspondent amplitude of the logic input voltage of the AS5040 IC.
The most common way of encoding is to sample the pulses within a specified time, but with that method the encoding function holds up the main program while reading the pulses.

Usage of hardware interrupt in the MCU makes the registrations of pulses easier. The main program can ”run” as usual, and jump to the *Interrupt Service Routine* (ISR) when encoding is needed.

A separate MCU, Atmega328p is used to handle the rotational encoders, calculations of the position and speed, the data is sent via I_2C-buss to the main processor. The Atmega328p has two hardware interrupts and can therefore handle two quadratic encoders. (Only one of the tow signals in the quadratic encoders is needed to make an interrupt)
In fig.7.4 the flow of the code is shown. The initiation is to setup the hardware interrupt pins (IRP), INTO and INT1. When the motors are moving a pulse is given to one of the IRP, depending on which motor it is.
The main program is then ”paused” and an ISR is executed.

Within the ISR the position of the motor is updated, due to the fact that each motor has a quadratic encoder, depending on witch pulse (A or B) that comes first, estimation of rotational direction can be done. In the code for encoders the calculation of angular-velocity is done. By using *TimedAction*, the program can jump to a stated function with a specified time-step. Velocity can be calculated when the time between the calculations and the difference between the *new position* and the *old-position* are known.

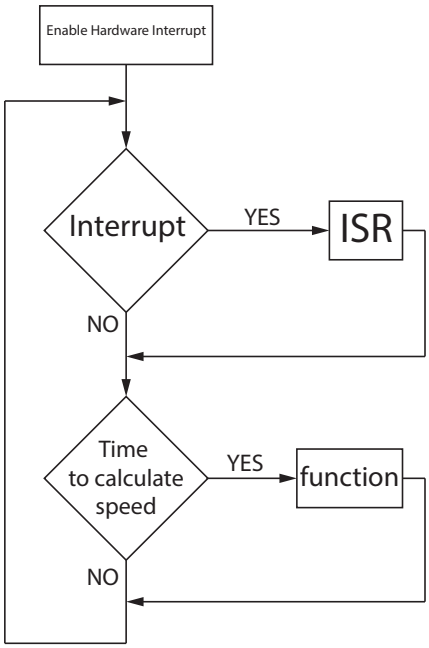


Figure 7.4: Illustrating image of the code structure

If pulse A,i (i = 1,2) comes first, the positions are increased with one step, and if pulse B,i (i = 1,2) comes first, the position is decreased with one step. Example cod for the front encoder,

```
void ISRfrontEncoder(void) {
// found a low-to-high on channel A
if (digitalRead(frontEncoderA) == HIGH) { // read pin 4

// check channel B to see which way if (digitalRead(frontEncoderB) == LOW)
frontMotorPosition++;
else
frontMotorPosition–;
}//end if

else{
if(digitalRead(frontEncoderB)==LOW)
frontMotorPosition ++;
else
frontMotorPosition –;
}//end else

}//end frontEncoder
```

The *TimedAction* is set to 20 $[ms]$, every 20 $[ms]$, the main program jumps to a function that calculates the angular velocity. Example code of the function,

```
 void updateMotorSpeeds(void)
//(present - last)/256* (1/0.02) = revolution/second
//times 2*pi => Rad/sec

frontMotorSpeed =((frontMotorPosition -lastFrontMotorPosition)/(256*0.02))*2*pi;

rearMotorSpeed =((rearMotorPosition -lastRearMotorPosition)/(256*0.02))*2*pi;

//update the motor posistion to next time!
lastFrontMotorPosition = frontMotorPosition;
lastRearMotorPosition = rearMotorPosition;

//end update motorspeed
```

Note; The hardware interrupt bridge between AVR C-code, and Arduino C++ code, is not reliable. Using C-code according to AVR-Programming is preferred.

7.7 I2C BUS

In the project I^2C bus is mentioned several times. Usage of several MCU's and MPU on one bus simplifies the communication between hardware parts. Instead of using Serial -RX -TX -CTS -DTR -RTS for each device, only two signal-wires is needed for all devices.
A simple way of explain the I^2C, it is a protocol that defines the concept of master- and slave- devices on two-wire, shown in fig: 7.5. Where the data is transmitted through *Serial DAta line* (SDA) and timed on *Serial CLock line* (SCL).

The master device is the device that is the "owner" of the bus and handles the clock and generates the "start" and "stop" signals. The slave devices listen to the master and respond to specified commands that are predetermined.
Each slave device has its own address, the master can decide which slave device it wants to communicate with.
Note that the devices must share the same GND, so that the zero-level of the signal is at the same level on all connected devices. It is also preferred for them to share the same VDD, but not necessary.

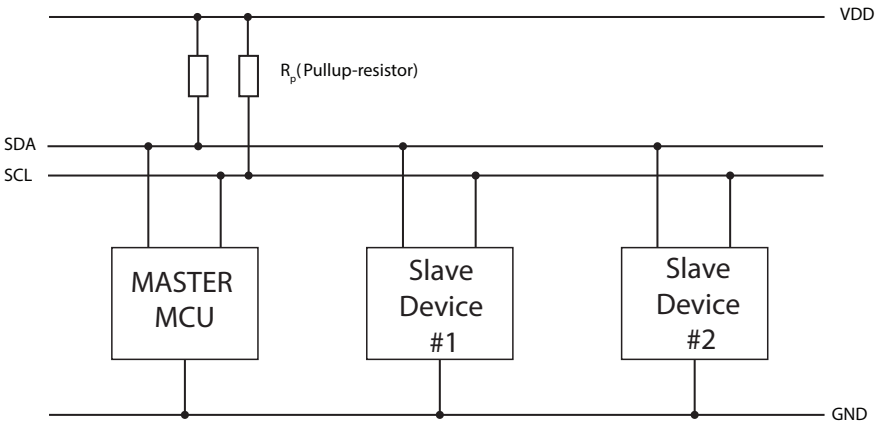


Figure 7.5: Illustrating image of Master and Slave on ISC Bus

It is preferred to have pull-Up resistors to the SDA, and SCL wires to get more stable signals, shown in fig.7.5.

7.8 Switching PID

Parallel PID regulators is used to handle the reaction wheel (RW) and the back-wheel steering. The parallel PID is illustrated in fig.7.6.

When the input value is close to the set-point value an constitutive PID-regulator is used, when the input value is far from set-point value an aggressive PID-regulator is used.
The PID-parameters are stated so that the aggressive PID-regulator has a high P-gain. And the constitutive PID has a low P-gain.
The switch that handles the ”jump” between the two PID-regulators has a pre stated threshold, and jumps between the *agressive-* and the i*constitutive-* PID-regulator.

makes the switch to the aggressive PID-regulator if it is fulfilled. The input to the switch is the roll-angle and roll-rate.

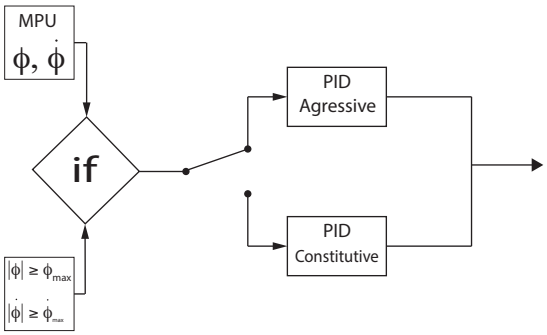


Figure 7.6: Illustrating image of switching PID

Example code of the reaction wheel with the parallel regulator,

```
void updateWheelPID(void){

WheelPIDInput = gyro[1];

if( abs(WheelPIDInput)>WheelRateLowerLimit1){
//we are close to setpoint, use soft tuning parameters
WheelPID.SetTunings(consWheelPIDKp, consWheelPIDKi, consWheelPIDKd);

calculateWheelPID(); //Calculates the output PW to the Reaction wheel
}

else if( abs(WheelPIDInput)>WheelRateLowerLimit2){
WheelPID.SetTunings(aggWheelPIDKp, aggWheelPIDKi, aggWheelPIDKd);
calculateWheelPID();
}
} //end updateAnglePID
```

7.9 Parameter Scheduled PID

As stated in chapter 3, the parameter scheduled PID, the input has to be saturated just to be sure that the max-(p,i,d) will not be exceeded. Illustrating fig of the Scheduled PID-parameters: 7.7

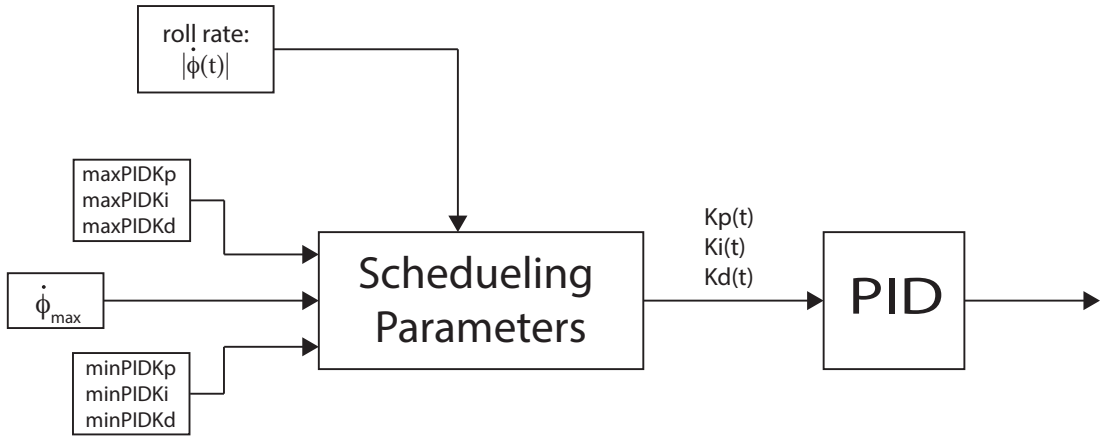


Figure 7.7: Illustrating image of Scheduled PID

Example code for the reaction wheel of the parameters scheduled PID-regulator,

```
//this parameters should be stated outside the function

double rollRateLimit = 1.5; //[rad/s]

void updatePID(void)
//gyro[1] is the roll angle of the UGV

PIDinput = gyro[1];

//saturate the roll-rate input to the mapped PID parameters
if(abs(PIDInput)>rollRateUpperLimit)
WheelPIDInput = rollRateUpperLimit;

//Parameter Scheduled PID
Kp = minPIDKp + ((maxPIDKp-minPIDKp)/rollRateLimit)*abs(PIDInput);
Ki = minPIDKi + ((maxPIDKi-minPIDKi)/rollRateLimit)*abs(PIDInput);
Kd = minPIDKd + ((maxPIDKd-minPIDKd)/rollRateLimit)*abs(PIDInput);

//write the new calculated values to the PID-regulator
WheelPID.SetTunings(Kp, Ki, Kd);
```

8

Results

In this chapter the results from the simulation are presented. Validation that the model is correct is based on the comparison of the simulated and logged data.
Therefore the un-known parameters, such as *spring-coefficient* and *damper-coefficient* is yield by iteration.
Also in this chapter the field tests data are presented the results give an image if this type of control and mechanics is valid to use.

8.1 Simulation Results

In this section the simulation results of the DC motor is presented. With and without external load. Then the complete model is simulated and compared with the logged data.

8.1.1 DC-Motor model

With the calculated and measured parameters, the simulations of the DC-motor is done. Input level of the voltage to the simulation is the same as the nominal voltage of the DC-motor (14.4[V]).
During the start-up of the DC-motor the current ”jumps”, this is the *start-up current*, shown in plot.8.1. When the rotation of the shaft starts the current will decrease until it reaches the steady state value.

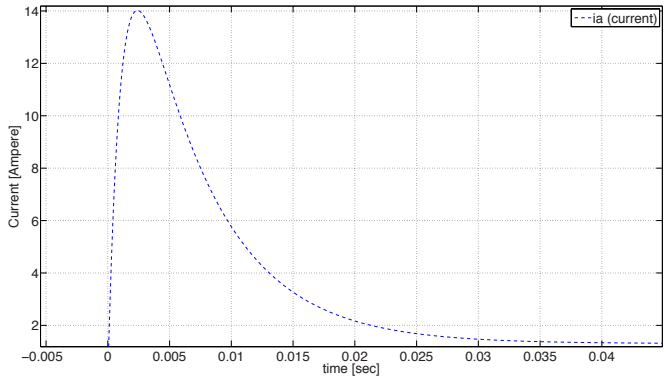


Figure 8.1: Current during start-up

When the angular velocity has increased to the level where the $\theta \cdot b_m = \tau$ the torque will become zero, and there will be a saturation to the angular velocity. This saturation is stated as the maximum angular velocity of the DC-motor (fig.8.2).

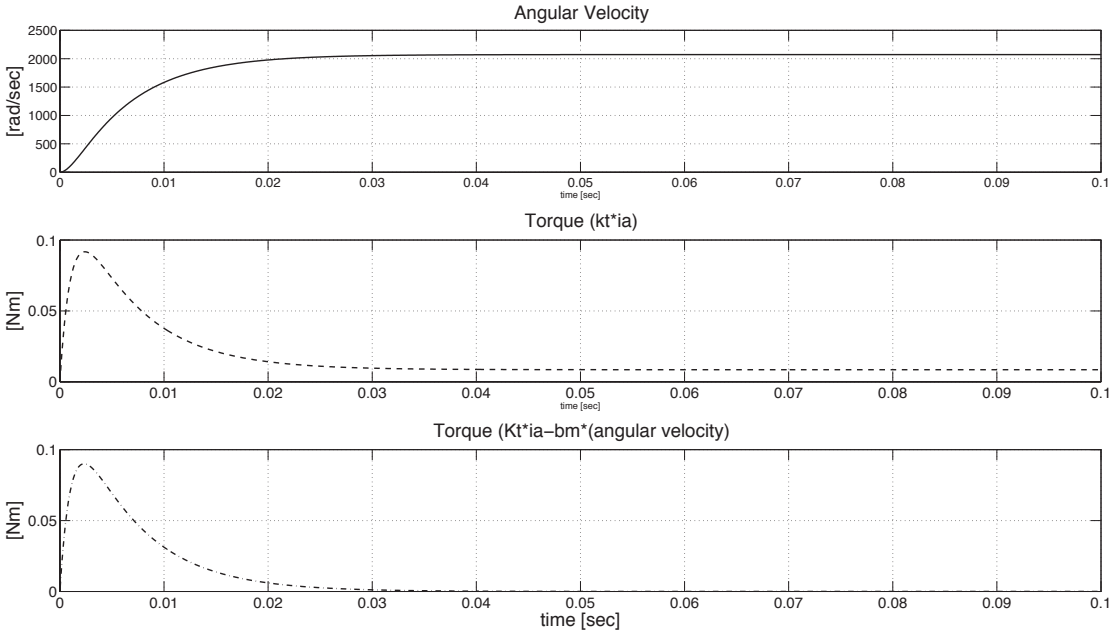


Figure 8.2: Angular velocity, Electric torque, Torque due to friction and angular velocity

From the data-sheet of the Maibuchi RS550VC [**Appendix B**] the maximum velocity and non-load current, corresponds to the simulated ones.

8.1.2 DC-Motor model,with Load

Connecting the external parameters the behaviour of the DC-motor can be plotted. Note that in this simulation the Lipo, battery value (11.1V) is used, or else comparison against the logged data would not be valid.

fig.8.3 shows that the angular velocity saturates close to the logged value of 658 $[rad/sec]$, due to the estimation of the friction in the gear and drive-axis.
Start-up current is plotted in fig.8.4, the choice of motor controller is partly based on the level of this current. The steady state current is close to 13 $[A]$, and the driving torque to 54 $[mN \cdot m]$. According to the data sheet, maximum efficiency of the DC-motor is yielded at this torque and angular velocity.

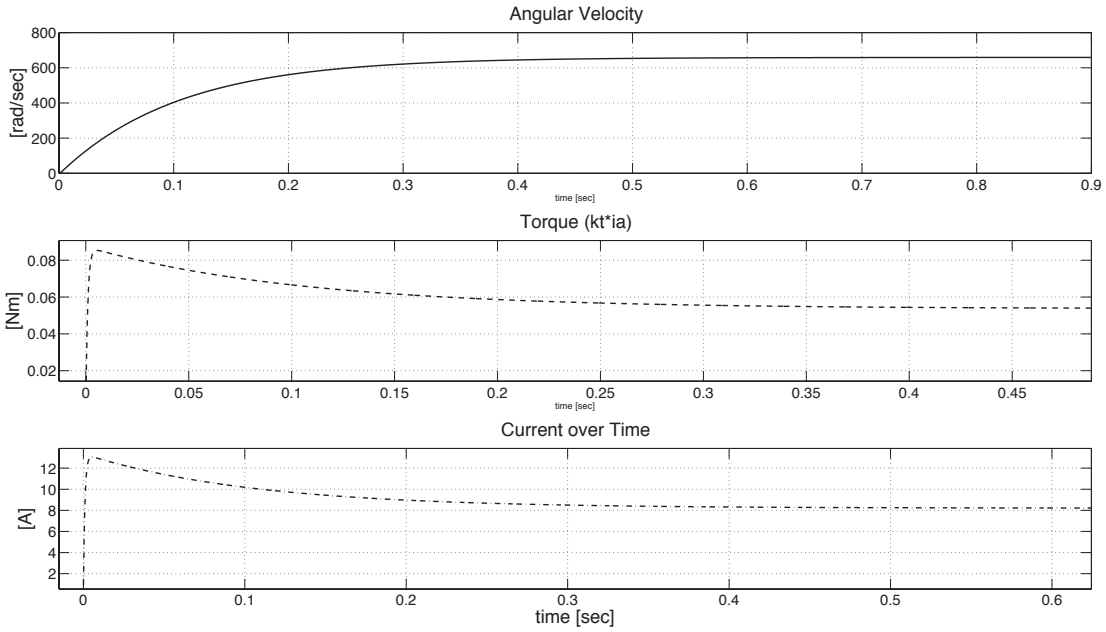


Figure 8.3: Current, Angular velocity and Torque

Note that the start-up current is lower in this simulation even if the external load is applied. This is due to the voltage level (V_{in}), which is 3.3V less than before.

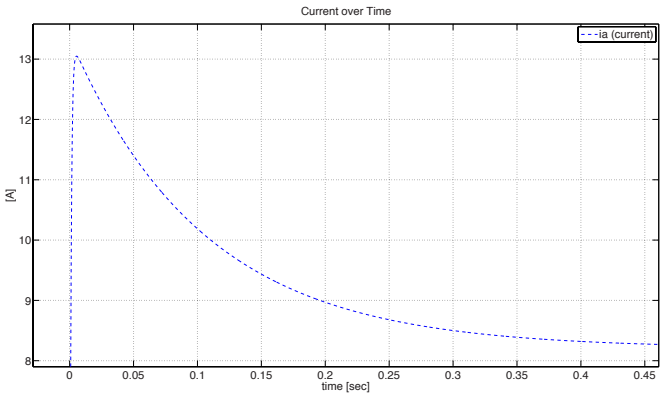


Figure 8.4: The start-up Current with external load to the DC-motor

8.1.3 UGV-Model

Verification of the model is done by using similar input parameters as the ones that are logged during tests-run of the UGV (open loop). The output of the model should then yield the same response as the logged outputs.

Longitudinal velocity

The longitudinal velocity of the simulated and the logged data is presented in plot.8.5. During the collection of data, the steering angles of the UGV was zero ($\delta_f = \delta_r = 0$). Same conditions are set in to the simulation to yield a valid result. As shown the longitudinal velocity corresponds well to the logged data.

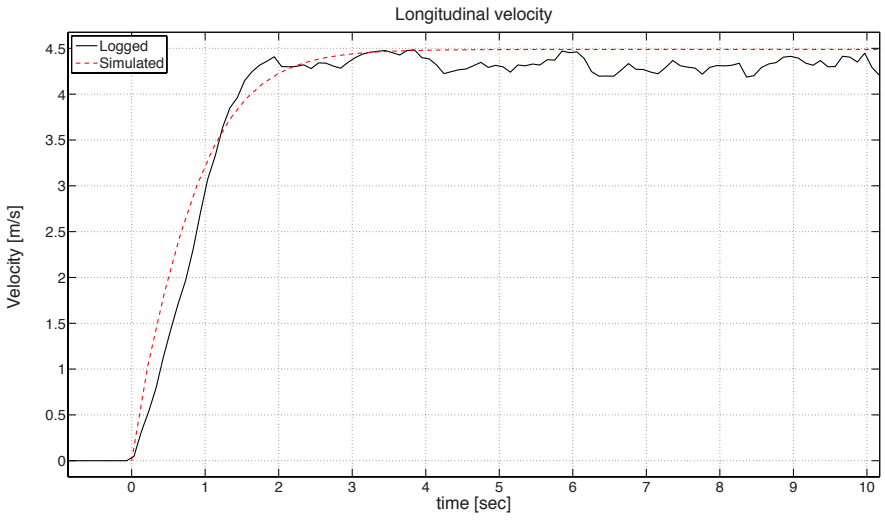


Figure 8.5: Simulation of the simulated vs the logged data

Yaw-rate

In fig.8.6 the Yaw-rate of the simulation and the logged data is plotted. Note that in the simulated environment the signals are ideals. The steering angels are set to the max angle of 20deg ($\delta_f = 20$ $\delta_r = -20$), the same as the UGV's maximum steering angle. With this results, the estimation/tuning of the spring and damper in the shock-suspension can be done.

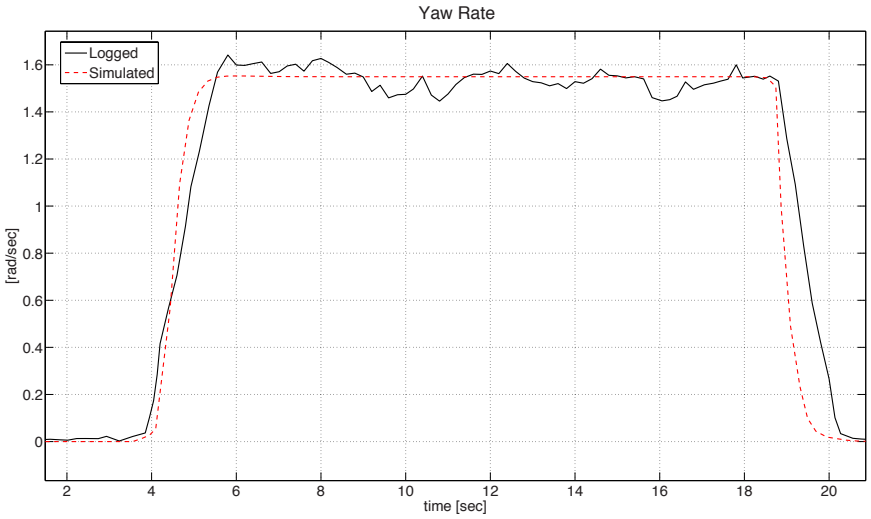


Figure 8.6: Simulation of the Yaw-rate vs the logged Yaw-rate data

8.2 Mechanics

Due to the reconstruction of the original RC-car, the UGV manage to handle weights up to 5kg (when mounted on top). Fig.8.7 and fig.8.8 shows that with the mechanical design in this project the UGV increases in length when loaded. This is due to the shock-suspensions placement. When external load is placed on top it will yield a lower COG, and therefore it will become more stable.

Image 8.7 shows the UGV without any load, and image 8.8 with maximum load, the shock-suspensions is saturated.

Figure 8.7: UGV without any load

Figure 8.8: UGV with heavy load

8.3 Software

Through measurements of the execution-time of the code, shown in the table below. It was noted that the reading of the RC-receiver took the longest time. The library that is used to read the pulses has not the ability to use pulling or threading, therefore the function holds up the time during the sampling of the pulses. This makes the control-loop slower. For a critical system such as inverted pendulum or unicycle, the controller would not be able to update the control signal in the frequency necessary to keep the system at steady state.
One way to improve this for the future is to use hardware interrupts for all seven channels, or let a slave processor handle the reading of the receiver.

Part	Time [μs]
$readReceiver$	210264
$moveMotors$	116
$moveSteering$	12
$updateMPU$	8056
$updateRWPID$	144
$calculateRWPID$	84
$moveRW$	48
$updateWheelPID$	32
$updateRWStatuses$	376
$updateMotorStatuses$	484
$currentRead$	344
$sendData$	2380

8.4 Field Tests

8.4.1 Reaction Wheel

This test is focused on the Reaction Wheel. By hanging the UGV in a line from the ceiling in such way that only one wheel side is in contact with the ground, one can see this as a static image of the UGV during rollover.

The roll-rate is measured during the impulse in the lateral direction, with and without the reaction wheel (RW) activated.

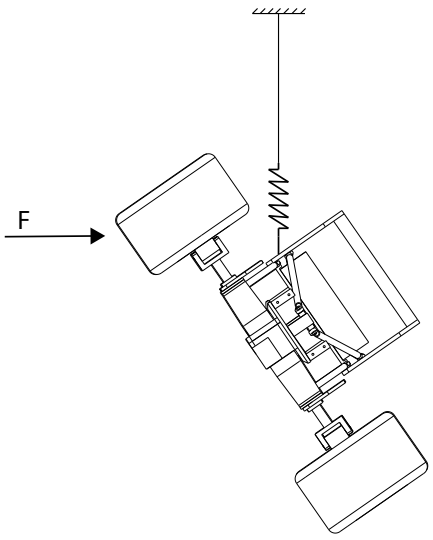


Figure 8.9: Pendulum test, illustrating fig

Manoeuvres; Pendulum test

A spring is linked between the UGV and the line. With this setup the UGV will swing in a harmonic pattern, when a force is applied in the lateral direction.

Test response

Clearly the reaction wheel influences the roll-rate.

Analysis

In fig.8.10 The black line is the roll-rate of the UGV with out the RW, the blue line is with the RW activated. One notation is that the RW handles smaller impulses better than large Ones. This is due to the fact that the RW has a finite force it can generate.

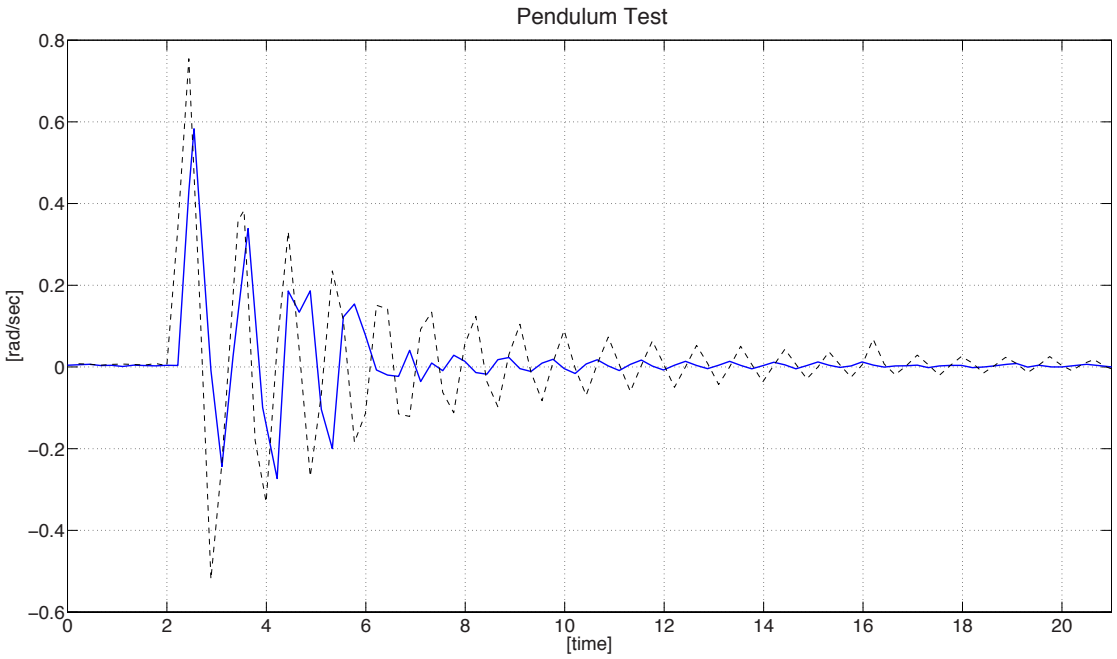


Figure 8.10: RW pendulumTest results

8.4.2 Driving with high COG

The objective in this test is to see how the UGV handles cornering with heavy load with and without the controllers. Using both front and rear wheel steering gives the smallest turn radius [ref acerman], and therefore it will generate the strongest lateral acceleration.

Note that $\delta_f = -\delta_r$. In fig.8.11 the roll-rate and front-wheel steering is plotted. It shows the relation between the steering and roll-rate. This will be some of the parameters that is to be plotted an analysed.

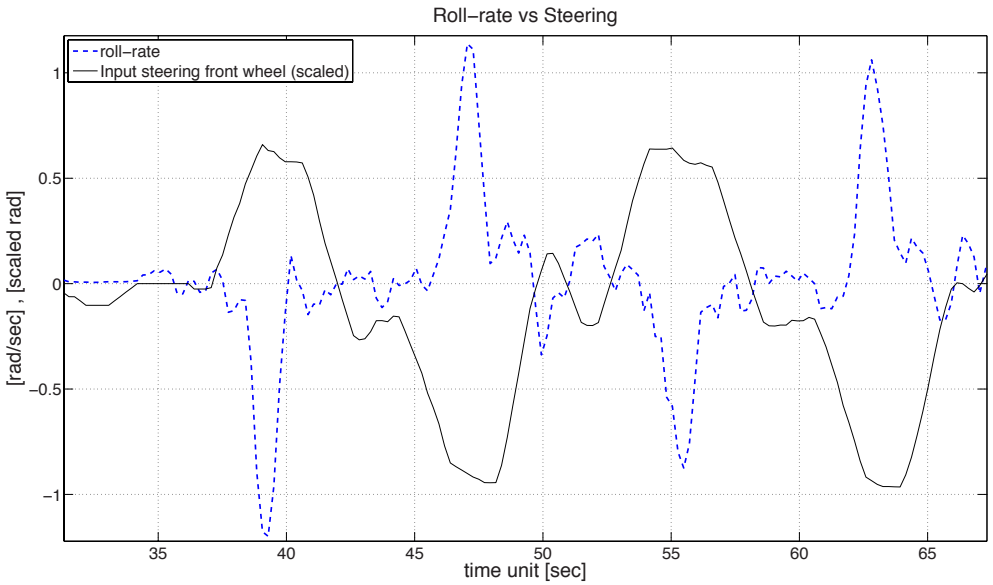


Figure 8.11: plott of the roll-rate and steering

Outline of the tests

accelerate positive longitudinal direction until max-speed is reached, then make several different manoeuvres.
In order to change the hight of the COG the UGV is equipped with a hard paper cylinder where a 2.5kg weight is mounted on top, illustrated in fig.8.12. The height of the cylinder is 31 cm, and this gives a distance of 40 cm above the UGV's COG to the external load. The added weight acts like a replica for the upcoming sensors that are to be mounted on top of the UGV. With this heavy load that is about 31% of the UGV's total weight gives the test a high risk of rollover [30], this test is an extreme case compared to the sensor that is to be used. With the help from the logged data and recording of the test with a GoPro HD3 camera, analysis of the test can be done.

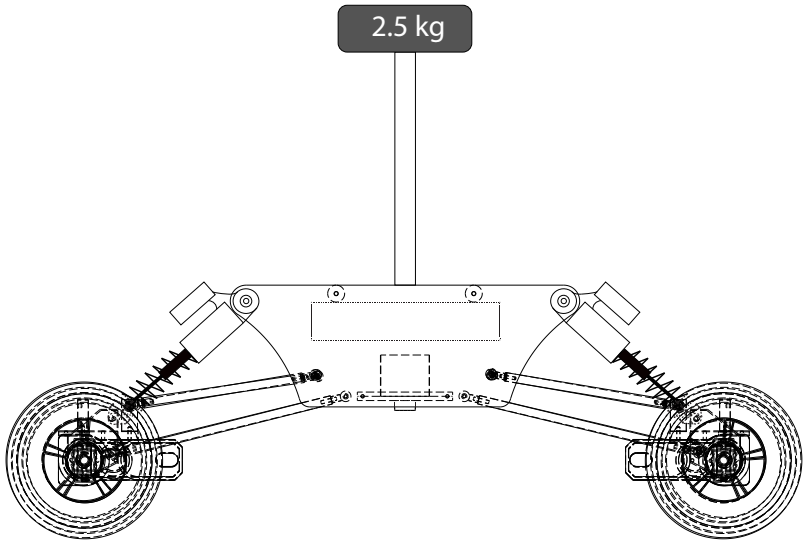


Figure 8.12: UGV with high COG

8.4.3 Without controller

Manoeuvres; J-turn

Accelerate until maximum speed is reached, then make single curvature (narrow cornering), using front and back wheel-steering.

Test response

Due to the added mass, its placement and the high lateral force, the UGV is overturned and roll-over occurs.

Analysis

Analysis of the collected data gives insight to how the behaviour of the UGV is with this specific load, speed and steering angles.
During roll-over, the UGV is balancing at a equilibrium point (on one wheel-side) just before it tips over. Fig.8.13 shows this point in the roll-rate curve, this point can be used as the threshold for roll-over. Clarification; this holds only for this test. By setting this point as the *upperRollrateLimit* in the scheduled PID-parameter algorithm, the saturation of the maxPID-parameters can be stated.

OK

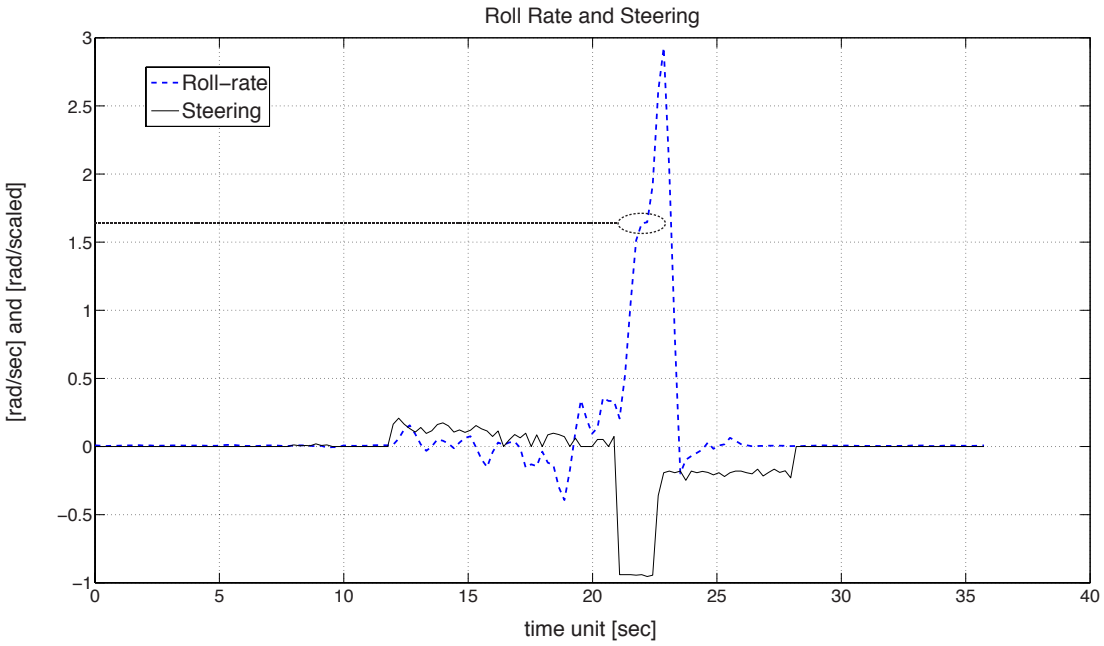


Figure 8.13: Roll-over occurs

The lateral acceleration tells about the lateral force that is acting up on the sprung mass. In fig.8.14 the acceleration just before the UGV is hitting the ground is $2 \cdot g$

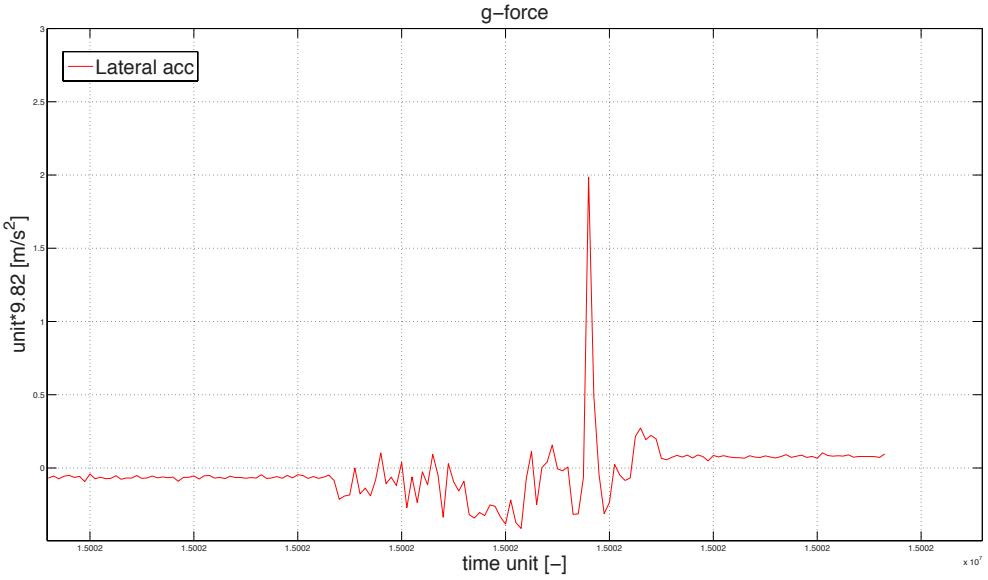


Figure 8.14: Lateral Acceleration $a_y[value \cdot g]$

8.4.4 With Controller

Both controllers are activated in this tests.

Manoeuvres; J-turn

Accelerate until maximum speed is reached, then make single curvature (narrow cornering), using front and back wheel-steering.

Test response

With the controller active and the $upperRollrateLimit$ set, the UGV stays stable through out the cornering in both directions.

Analysis

fig.8.16 is a subset of the collected data. The tree plotted lines, *blue-dotted*, *black-continues*, and *black-dotted* is the roll-rate ($\dot{\phi}$), front wheel-steering (δ_f) and rear wheel-steering (δ_r) respectively.

as stated before $\delta_f = -\delta_r$. In the plot one can clearly see that $-\delta_r$ differs from δ_f in some cases. This is at the time the controller takes over the back-wheel steering δ_r. Whenever the controller takes over the back-wheel steering $\delta_f \neq -\delta_r$. In the plot, when the controller takes over, the roll rate starts to decrees, which means that the impact of the controller is high.

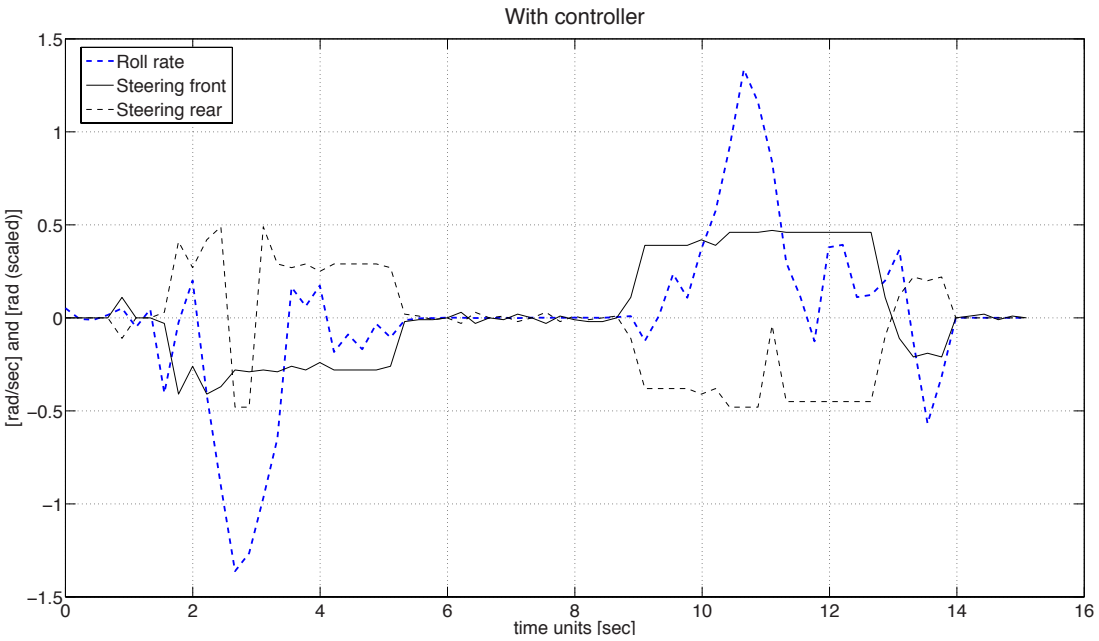


Figure 8.15: front- and back-wheel steering vs roll-rate

When the controllers are active the lateral g-force acting on the UGV's un-sprung mass is reduced, smaller g-force in the lateral direction contributes to less risk of roll-over.

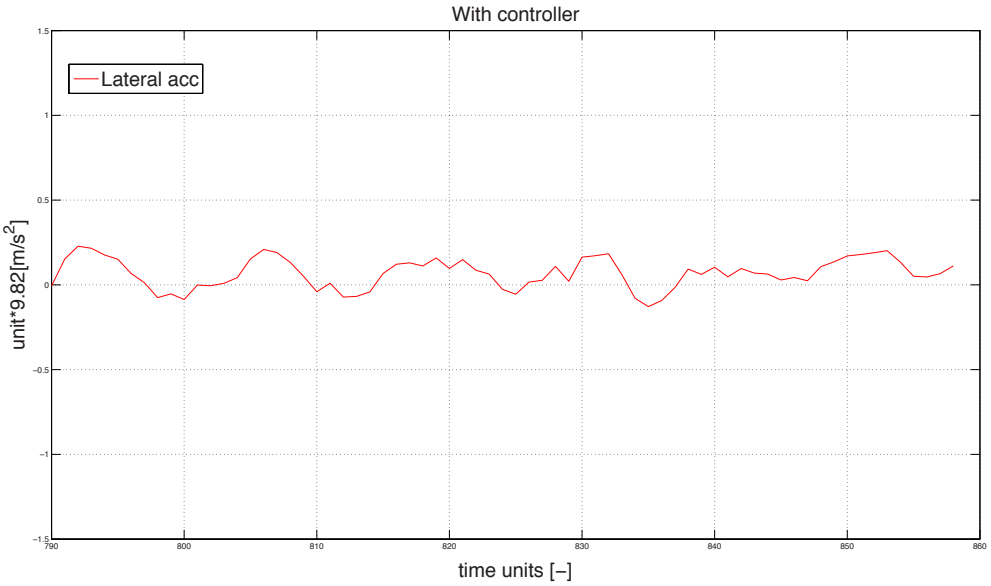


Figure 8.16: g-force with the controller active

Maneuvers; Impulse

Accelerate until maximum speed is reached, nudge the weight by hand as the UGV drives by. An staged external impact.

Test response

The controller handles this staged ”impulse” and corrects the UGV back to steady state.

Analysis

In fig.8.17 the front wheel steering is not used (the black line is close to zero). When the nudge is acting on the UGV the back-wheel steering starts to act. The added weight swings from one side to the other side due to over swing. The back-wheel changes steering angle to compensate this. Finally the roll-rate is minimized and the UGV is stable.

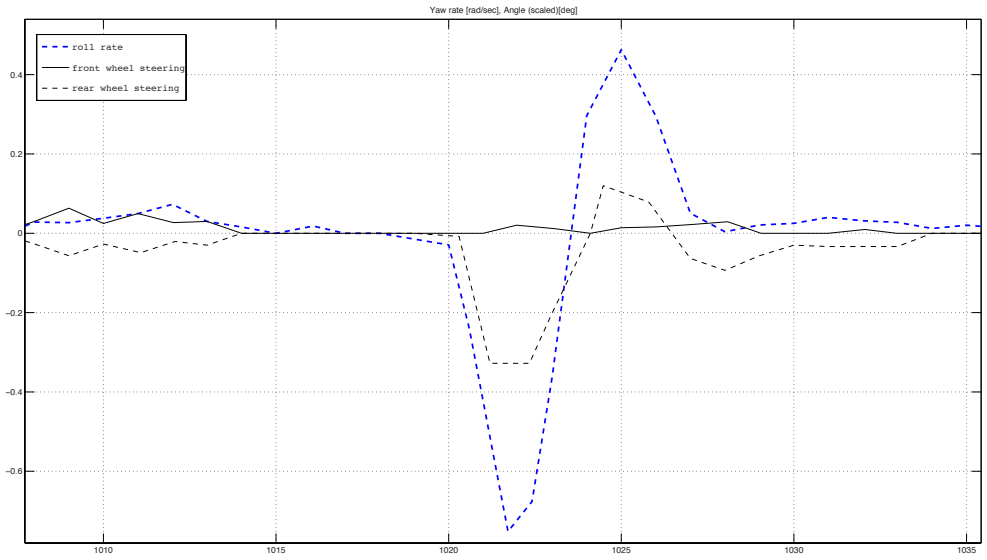


Figure 8.17: front- and back-wheel steering vs roll-rate

9

Conclusion

9.1 Simulations

The simulation of the Longitudinal velocity the logged data and the simulated model correspond well to each other. The estimation of friction and calculation of the inertia must then be very close to the real values.

9.2 Electronics

The fact that an PCB was done for the project simplified the connection of the separate electronic parts in the project.

9.2.1 Motor driver

During the project notation about the DC-motors angular velocity in the different direction had an quit big separation. The carbon-brushes gives some friction in the non-preferred direction but it seamed to be unrealistic amount. Switching the polarity (A-B output) from the motor-driver to the DC-motor, and test-run the motor in the two direction again showed almost the same result.

It seams that the motor-driver has a preferred direction of the max-current it can manage. Notation of this can not be found in the data sheet. It can also be the MOSFET used, they might have to low back-current protection.
An vehicle often has a front and rear, and a preferred direction of travel, therefore it has no big impact on the project.

9.3 Mechanics

The UGV manage all the load that was stated. And even manages to ceep stable with the extreme case

9.4 Controllers

The control strategy in the project is quit simple but works well. The PID-regulators are easy to implement and preforms well, the computation time for the regulator is quite low. The implemented version of the control is an combination of the *switching PID* and the *scheduled PID*. The scheduled PID, takes the roll rate input and changes the PID parameters depending on the roll rate. The interval of value that is used from the roll-rate is,

$0 < \dot{\phi} < \dot{\phi_{max}}$

where $\dot{\phi_{max}}$ is a threshold stated by the operator. But usage of this method prevents the operator to use the back-wheel steering, due to the fact that the control-always handles the back-wheel steering.
In order to solve this a lower threshold had to be stated, $\dot{\phi_{min}}$. Which will, correspond to a switching performance of the control. Note that within the region of the stated thresholds the regulator still has the scheduling-PID performance.

9.5 Future Work

9.5.1 Driving actuators

To expand the control of the UGV, I propose that expantion and recostruction of the mechanics is done in such way that etch wheel has a separate DC-motor. Controls, such as *anti-spin*, *ABS* and *software differential-axle* can be implemented. The DC motor form *Pololu*, with mounted gear-box wold be suitable of this type of expansion. The wheel can be mounted directly on the motor shaft.
Example cad of motor housing that can be used for the Pololu geard-DC motor in fig.9.1

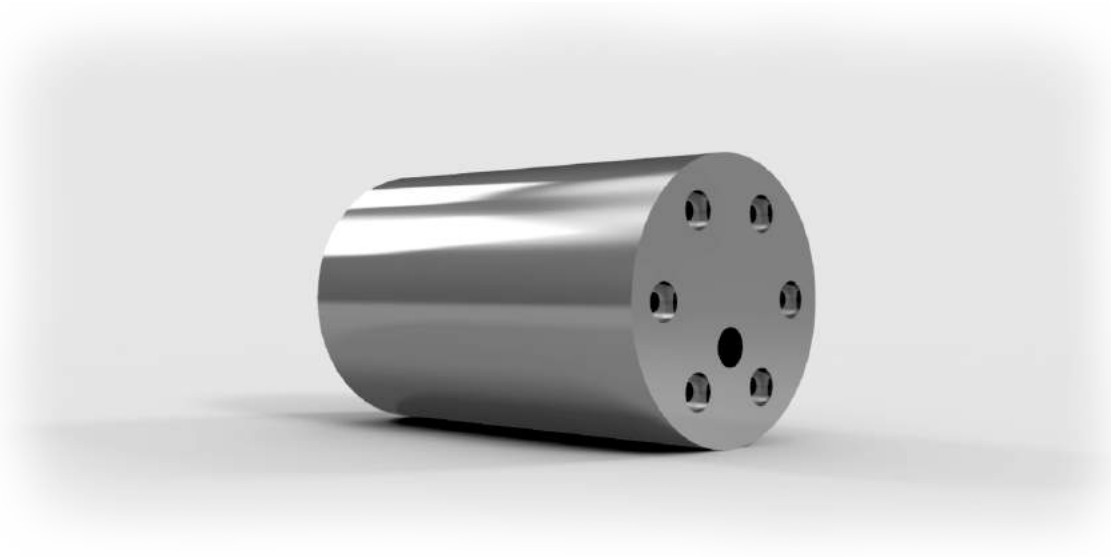


Figure 9.1: Motor housing for Pololu gear-motor

9.5.2 Rotational encoder

The slave processor for the rotational encoders is mounted on the main PCB. The new version of the encoder, I have chosen to have a processor on each encoder. All the calculations of the rotational speed is done on the processors as before. The front and rear encoder have there own address.

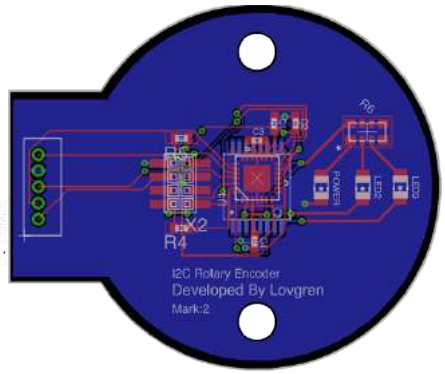


Figure 9.2: Rotational Encoder Version 2

9.5.3 Hub + telemetry

The model can be tuned even more to yield a very accurate Simulation-model of the UGV. Predictive controller as stated in the chapter about *controls*, needs to have a hardware strong enough to make heavy calculations. This can be solved by using one of the small computer that are available in home-electronic stores. With the protocol that I

have written for communication between Matlab and the UGV, *control-parameters* and *control-signals* can be sent live to the UGV. In somewhat that is how future of the car-industries looks. Instead of having all computation and all controls within the car, hubs with long range telemetry will communicate with the car and send the optimal control due to the input data, illustrated in fig.9.3.

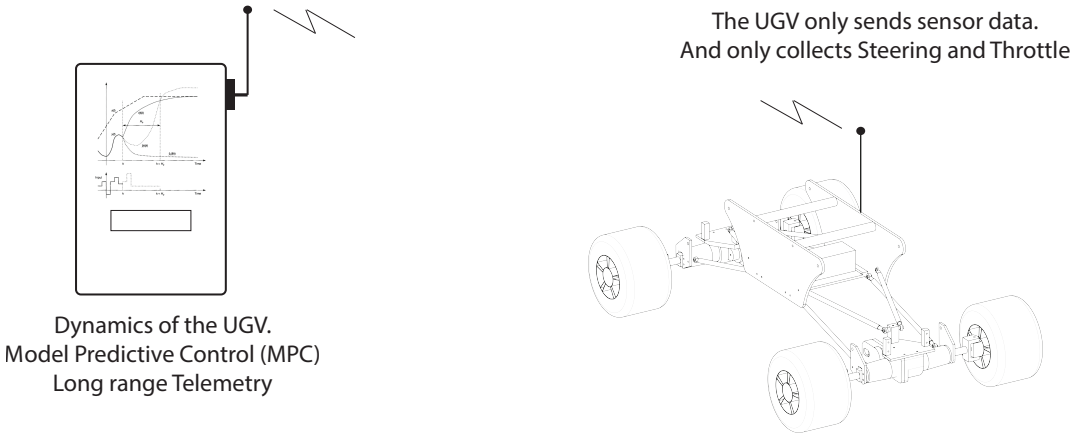


Figure 9.3: HUB and telemetry

9.5.4 Software

If one can use HUB and telemetry the software in the UGV can be reduced to just the operators control, reading the sensor data and communication protocol.
To solve the RC-controllers large execution time, hardware interrupt is proposed. The same way that the encoding of the magnetic encoders are done. Due to the fact that the PPM train is in a specific order, and by making a hardware interrupt on the rising edge and the falling edge of etch pulse. same results can be yield as the way it is done now. In between the interrupts the program can make several executions and thus the main-loop will reduce in computation time.

Bibliography

[1] FOI, Totalförsvarets forsknings institut.
URL http://foi.se/en/foi/About-FOI/

[2] H. Puchan, The mercedes-benz a-class crisis, Corporate Communications: An International Journal 6 (1) (2001) 42–46.

[3] Ø. Ihlen, Defending the mercedes a-class: Combining and changing crisis-response strategies, Journal of Public Relations Research 14 (3) (2002) 185–206.

[4] N. H. T. S. A. (NHTSA), Types of rollovers.
URL http://www.safercar.gov

[5] K. G. Jianyong Cao, Lixin Jing, F. Yu, Study on integrated control of vehicle yaw and rollover stability using nonlinear prediction model, Mathematical Problems in Engineering 2013 (2013) 15, article ID 643548.

[6] W. Cho, J. Yoon, S. Yim, B. Koo, K. Yi, Estimation of tire forces for application to vehicle stability control, Vehicular Technology, IEEE Transactions on 59 (2) (2010) 638–649.

[7] H. B. Pacejka, Tyre and Vehicle Dynamics, Butterworth-Heinemann (an imprint of Elsevier).

[8] G. Rill, First order tire dynamics, 2006.

[9] B. Jacobson, Vehicle Dynamics Compendium for Course MMF062, Chalmers University of Technology, 2013.

[10] G. Rill, Wheel dynamics, in: P.S.Varoto, M.A.Trindade (Eds.), proceedings of the XII International Symposium on Dynamic Problems of Mechanics, 2007.

[11] H. E. T. G. Palmieri, P. Falcone, L. Glielmo (Eds.), A Preliminary Study on the Effects of Roll Dynamics in Predictive Vehicle Stability Control, 2008.

[12] B. Schofield, Model-Based Vehicle Dynamics Control for Active Safety, Media-Tryck, 2008.

[13] K. V. C. R. V. Rajasekhar, Design and analysis of dc motor with pid controller - a state space approach (2013).

[14] S. Armaghan, A. Moridi, A. K. Sedigh, Design of a switching pid controller for a magnetically actuated mass spring damper, in: Proceedings of the World Congress on Engineering, Vol. 3, 2011.

[15] M. Mattei, Robust multivariable pid control for linear parameter varying systems, Automatica 37 (12) (2001) 1997–2003.

[16] C. USA, Stmicroelectronics (2014).
URL `http://www.st.com/web/en/resource/technical/document/datasheet/CD00043711.pdf`

[17] H. Co, Hspracing (2014).
URL `http://www.hspracing.com`

[18] hobbex AB, Lipo battery (2014).
URL `http://www.hobbex.se/sv/artiklar/ack-li-po-111v-5000mah-25c-h-case.html`

[19] Arduino.cc, Arduino.cc (2014).
URL `http://www.arduino.cc`

[20] Atmel, ATSAM3X8EA, Atmel Corporation.
URL `http://www.atmel.com/images/doc11057s.pdf,(2014-11-16)`

[21] C. USA, Eagle pcb design (2014).
URL `http://www.cadsoftusa.com`

[22] T. Instrument, LM335 Precision Temperature Sensor, Texas Instruments Incorporated, (2014-11-16) (03 2013).
URL `http://www.ti.com/lit/ds/symlink/lm335.pdf`

[23] C. S. Mitchell, Optical rotational encoder, uS Patent 4,152,589 (May 1 1979).

[24] ams, Rotary sensor (2014).
URL `https://www.ams.com/eng/Products/Position-Sensors/Magnetic-Rotary-Position-Sensors/AS5040`

[25] savoxusa, Savöx 0235 (2014).
URL `http://www.savoxusa.com/Savox_SV0235MG_Digital_Servo_p/savsv0235mg.htm`

[26] I. Inc, Mpu9150 (2014).
URL `http://www.invensense.com/mems/gyro/mpu9150.html`

[27] OS-Craft, Gyro recorder (2014).
URL `https://itunes.apple.com/us/app/gyrorecorder/id390416376?mt=8`

[28] Microchip, Rn42 bluetooth (2014).
URL `http://www.microchip.com/wwwproducts/Devices.aspx?product=RN42`

[29] J. S. Lovgren, Johan s lovgren (2014).
URL `www.samirlovgren.se`

[30] H. Schouten, Research on the vehicle dynamics of a loaded vehicle (2005).

A

Appendix A

A.1 Specification from FOI

Specification for Master thesis project

At the Swedish Defence Research Agency (FOI) ongoing research addresses the topic of developing more intelligent sensor platforms. In order to evaluate the possibilities of detection of humans and objects in different types of terrain and environments, it is desirable to develop a platform (unmanned ground vehicle, UGV), which can carry sensors (of variable size and weight).

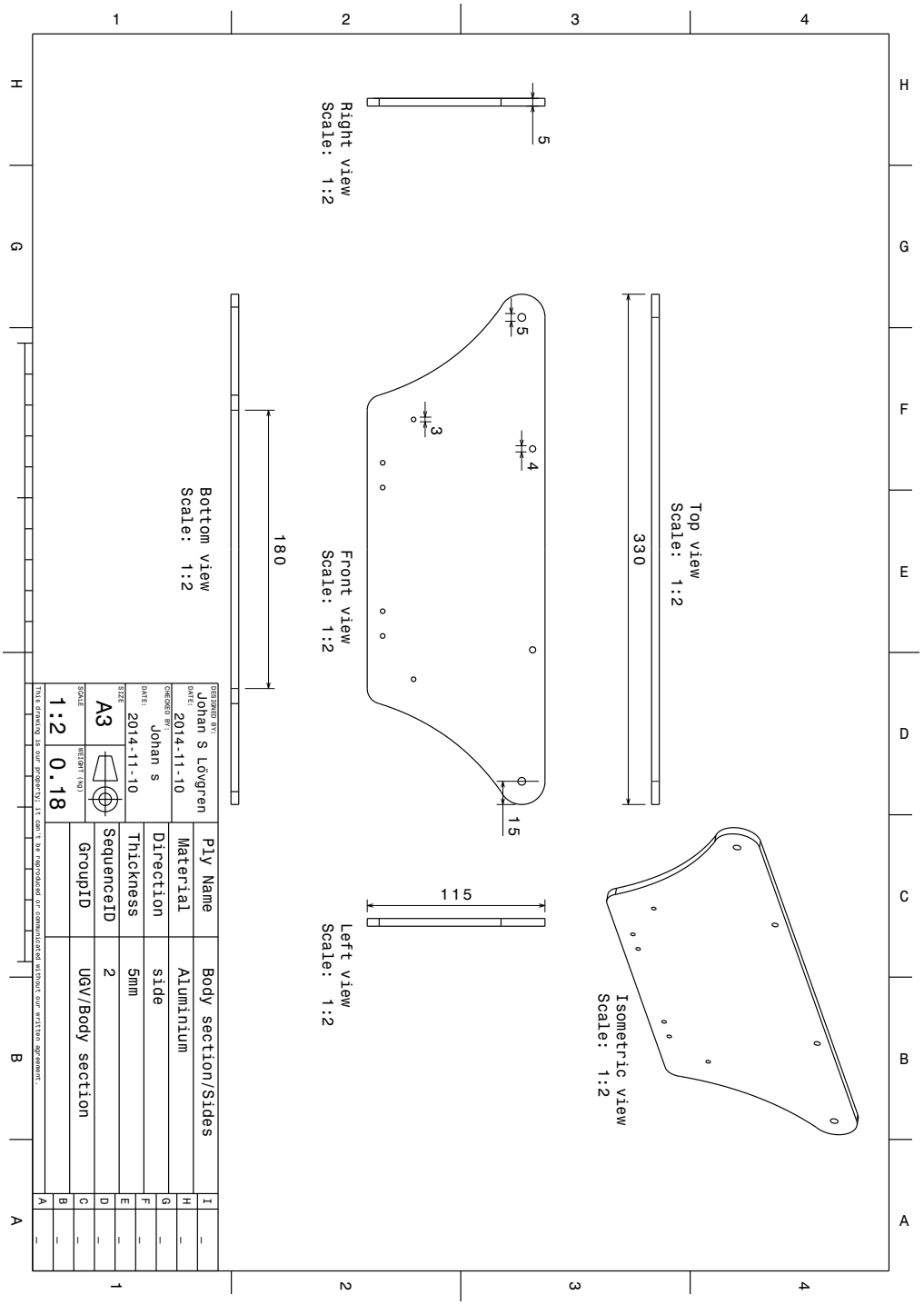
The platform could be constructed based on a larger RC car, and it is desired that the platform is (more or less) autonomous. Since the platform should be able to carry objects that will change the height of center of gravity, the risk of rollover is present. A implementable controller to prevent rollover is to be constructed, implemented, and validated before any sensor is to be mounted.
The UGV is also to be constructed in such a way that it can easily be copied, so that a series of the same type of UGV can be build.

Erika Bilock

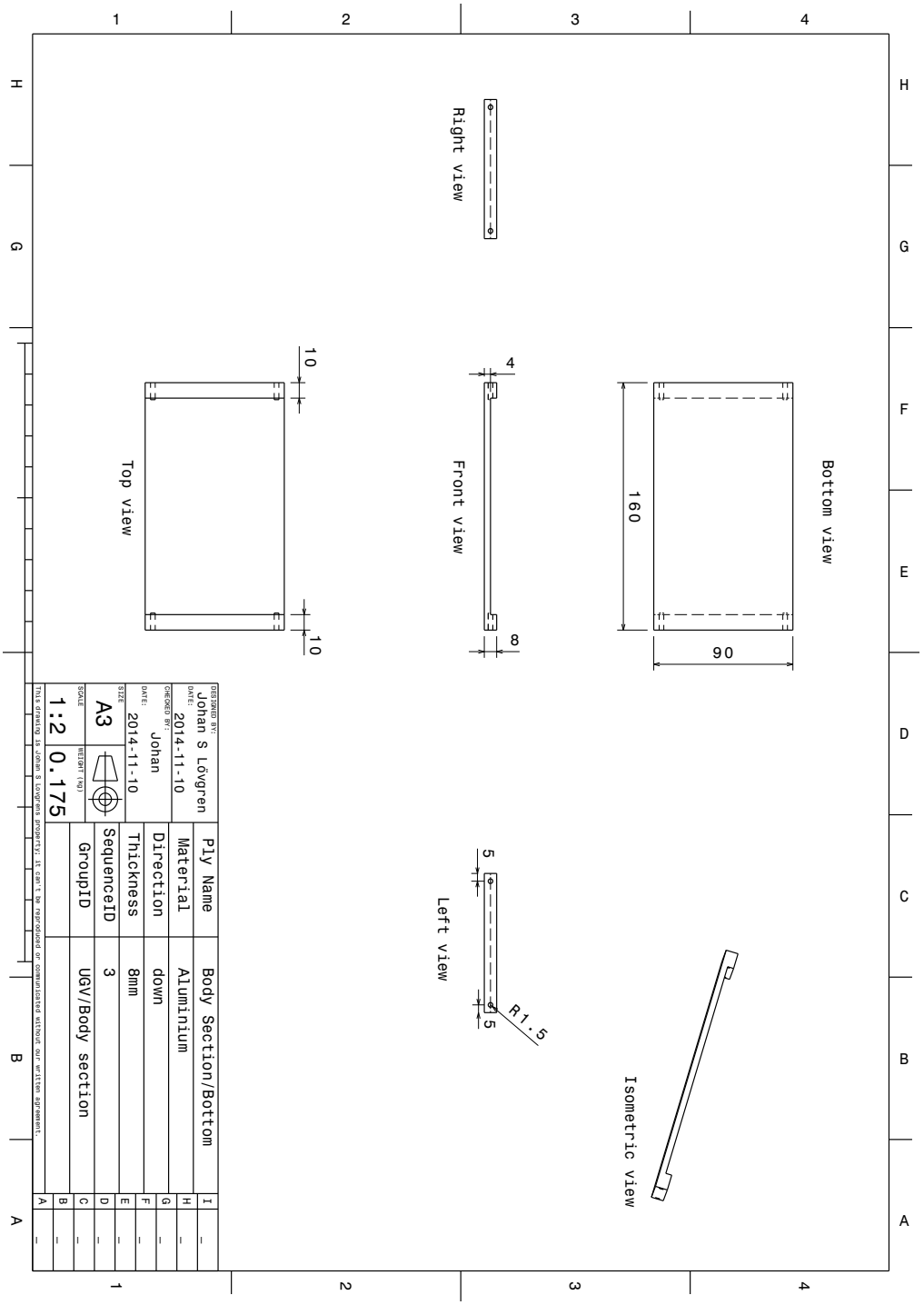
Joakim Rydell

Linköping, Spring 2014

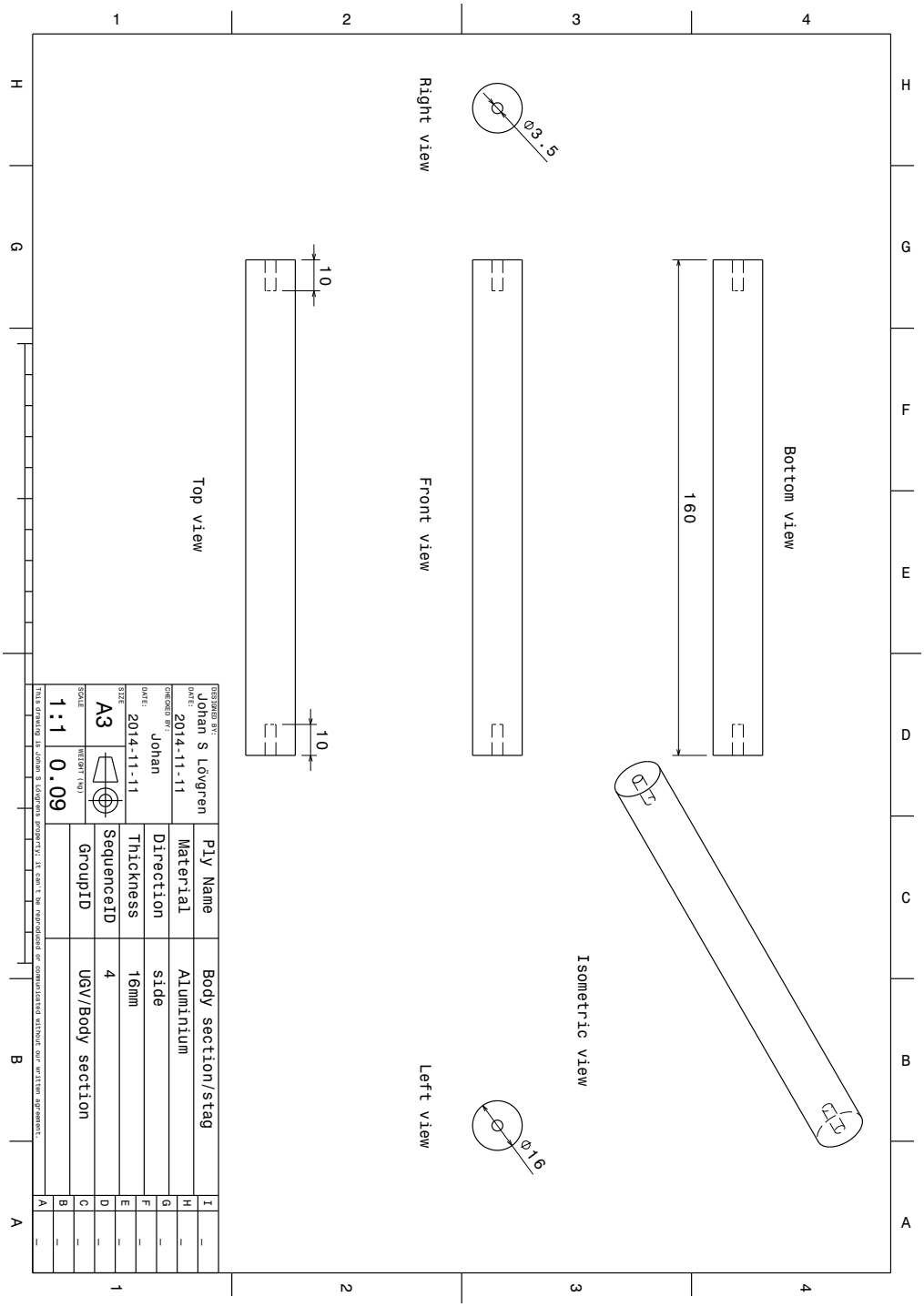
A.2 Midsection/Sides



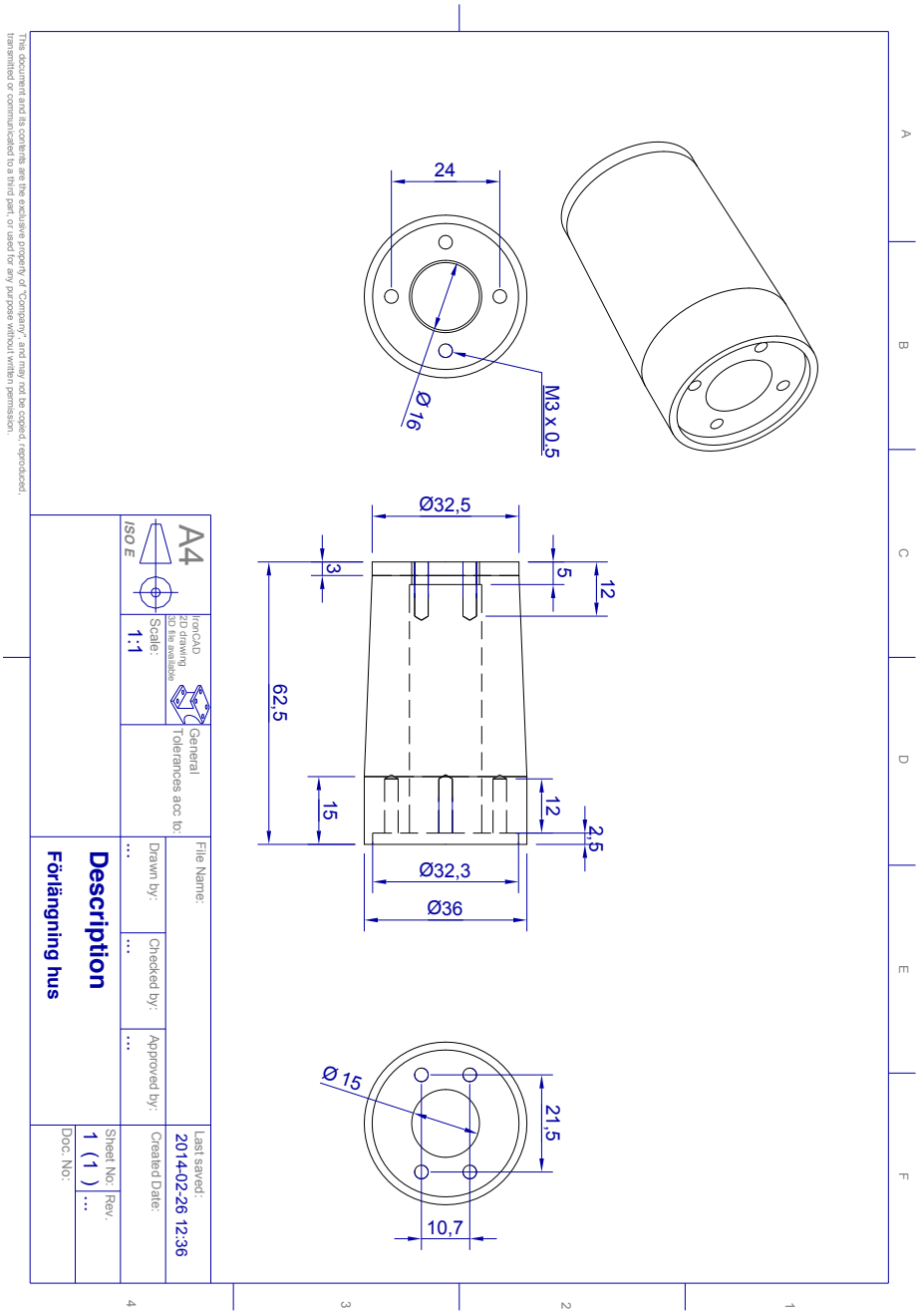
A.3 Midsection/Bottom



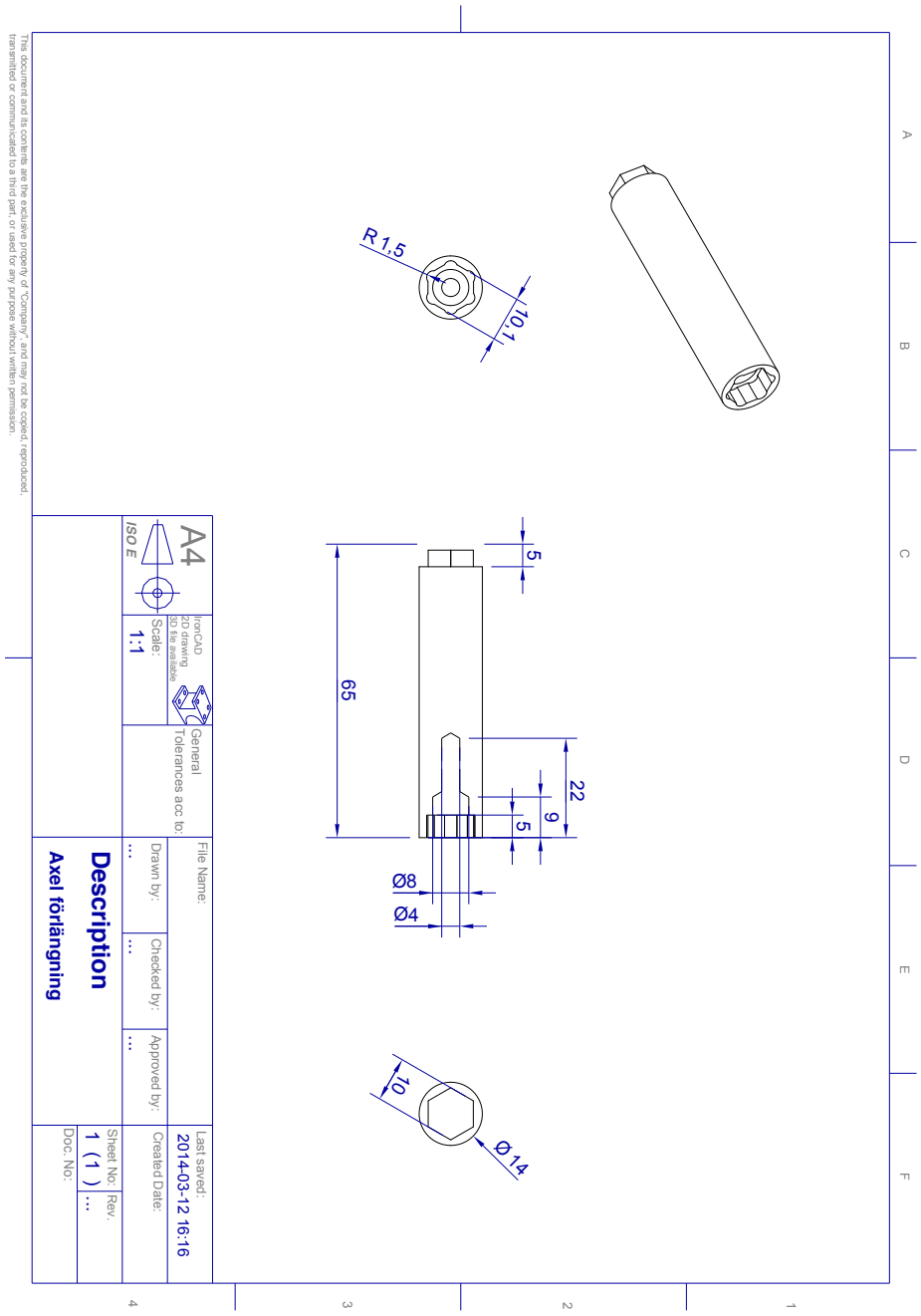
A.4 Midsection/Stag



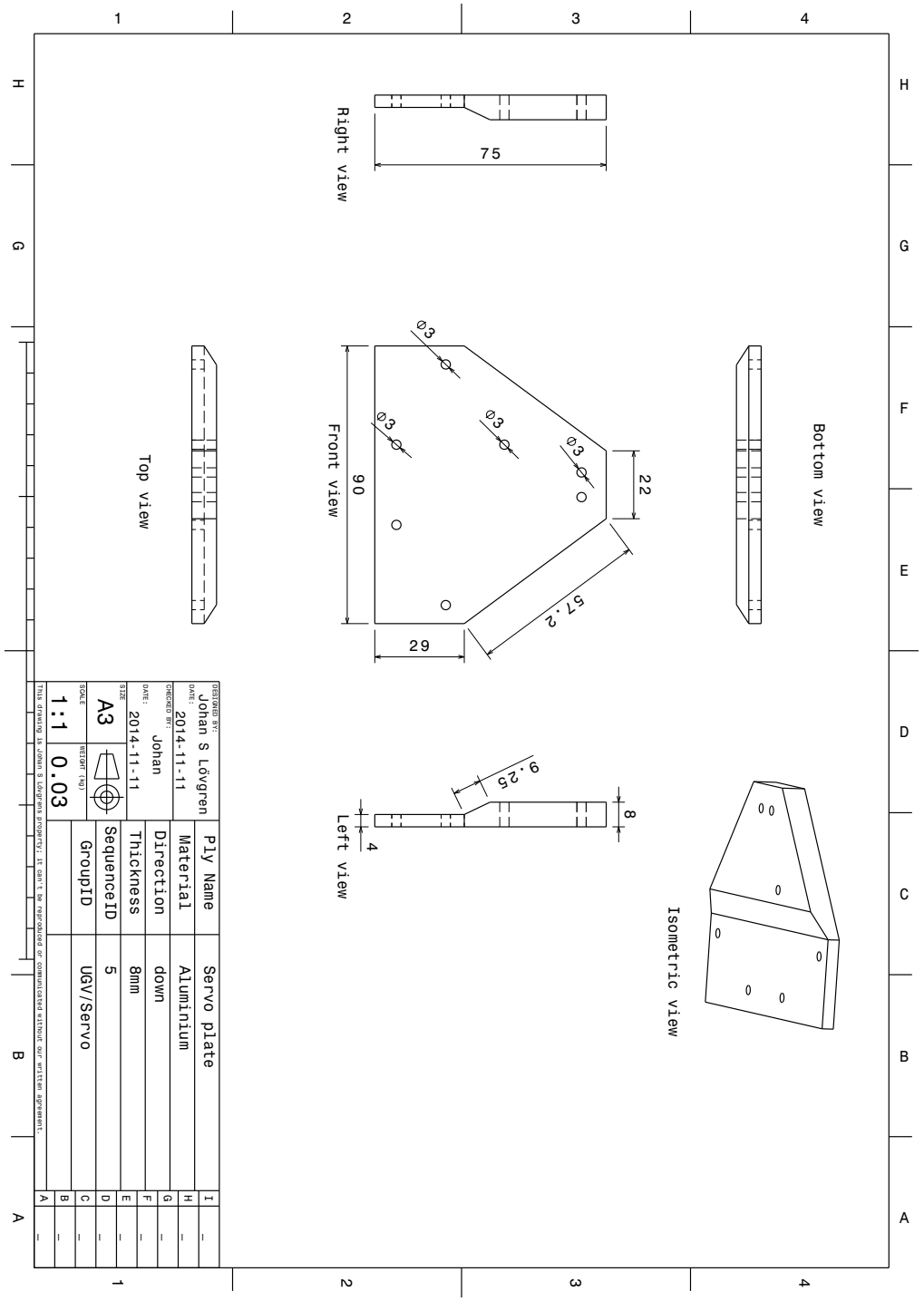
A.5 Extension of wheel axis Outer



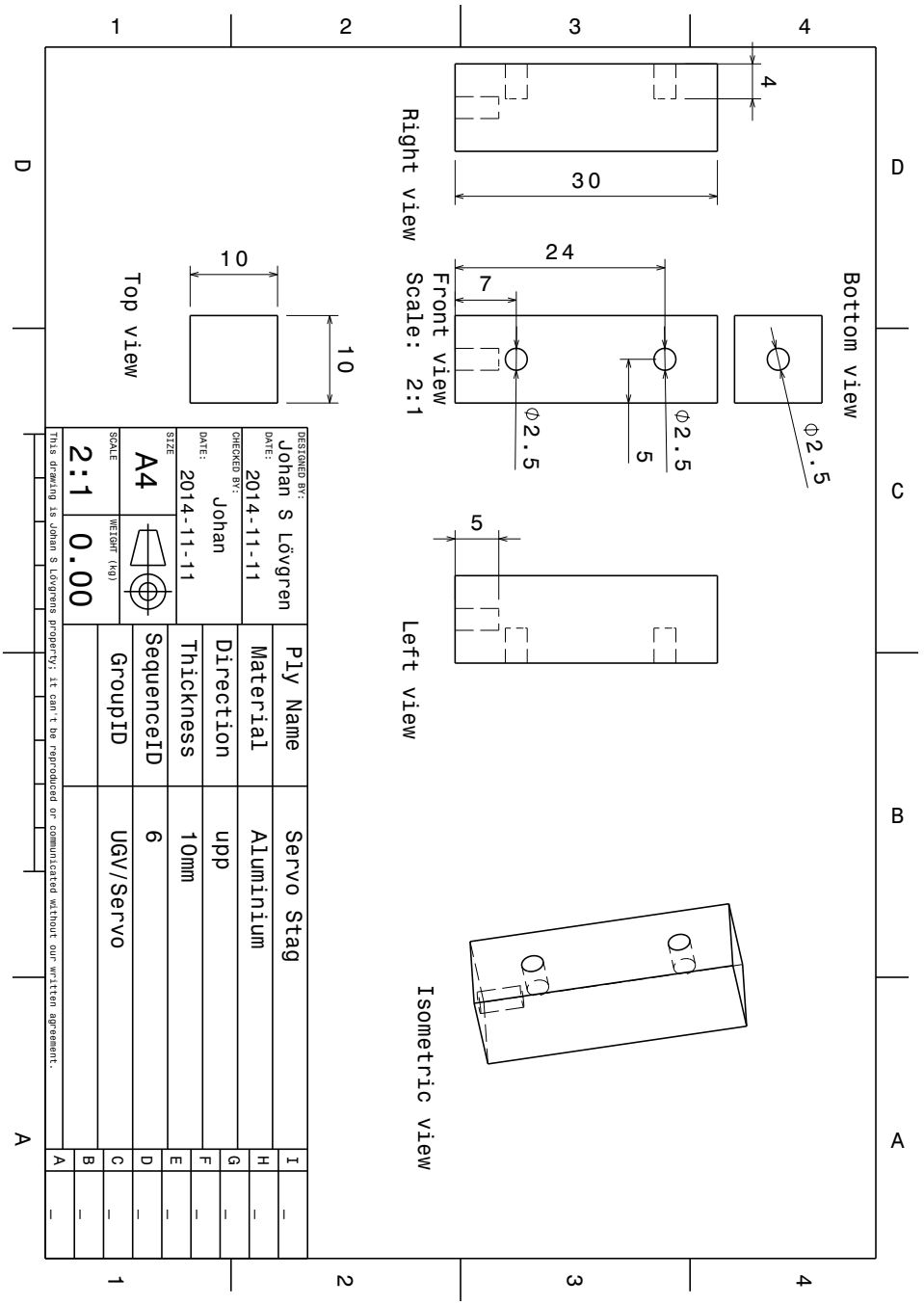
A.6 Extension of wheel axis inner



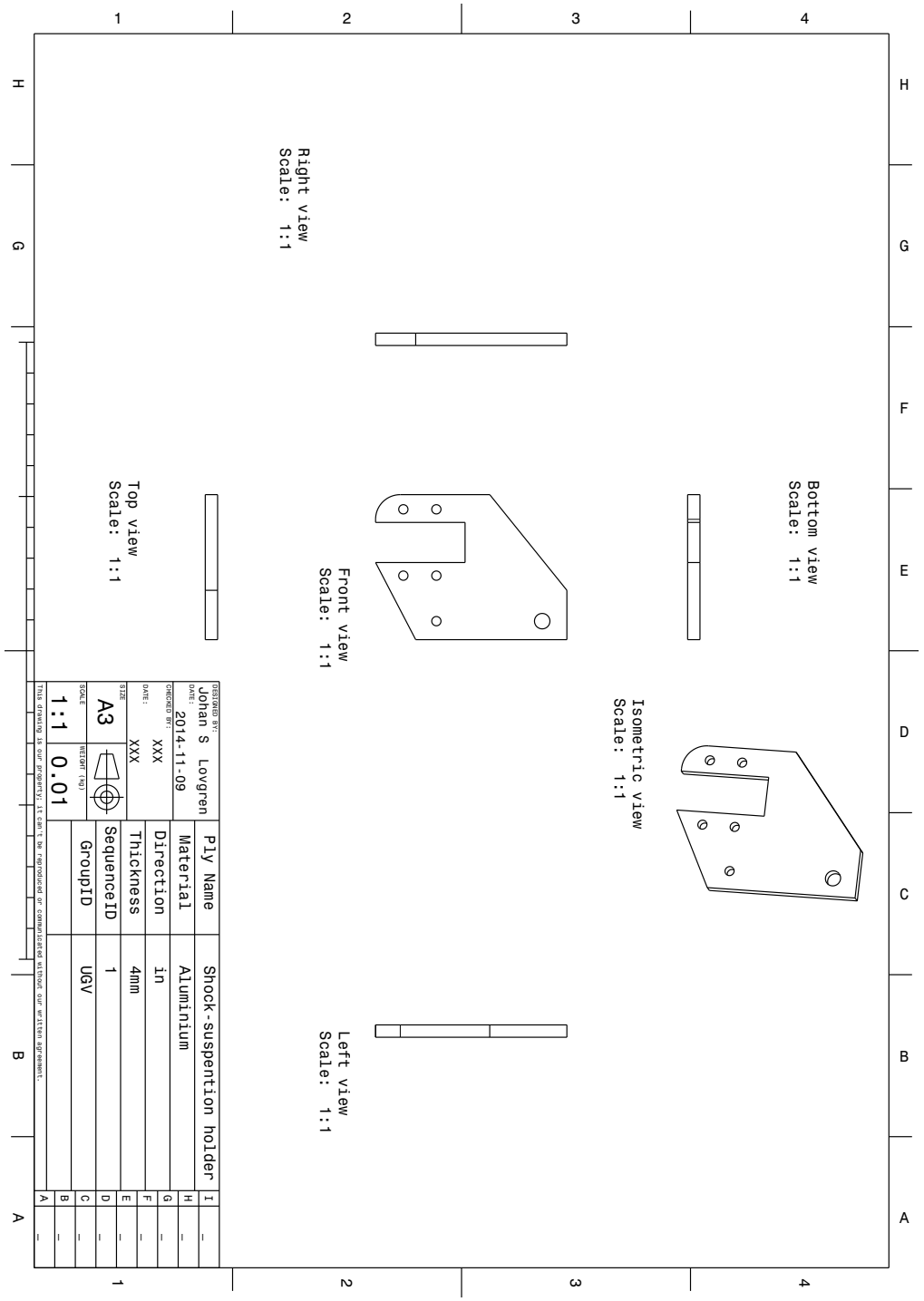
A.7 Servo plate



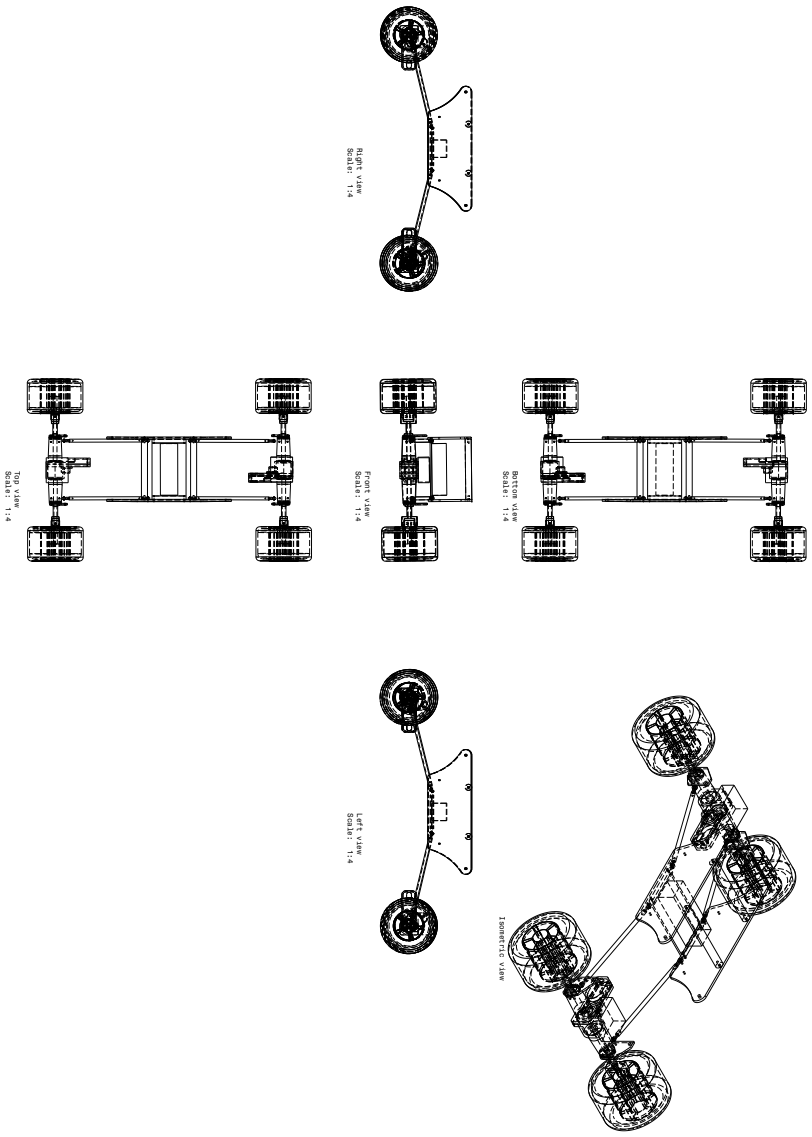
A.8 Servo Stag



A.9 Shock-suspension holder



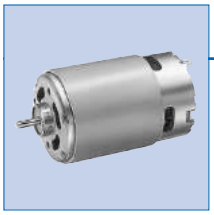
A.10 UGV Assembled



B

Appendix B

B.1 Mabuchi DC motor RS550VC



RS-550PC/VC

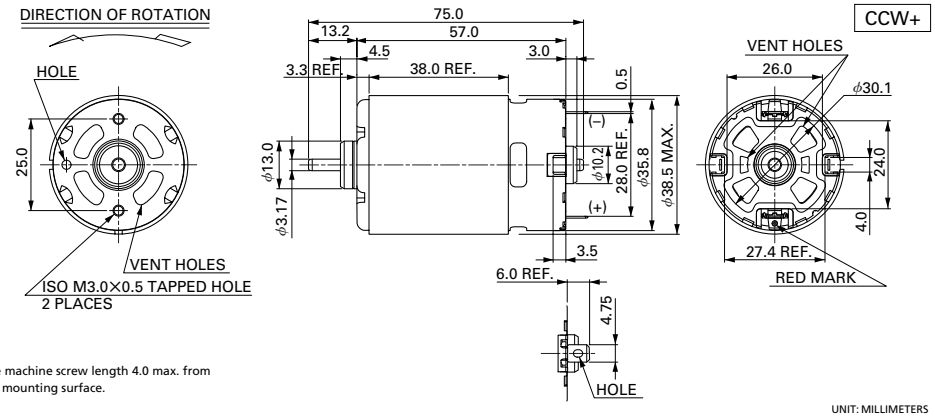
MABUCHI MOTOR
Carbon-brush motors

OUTPUT : 5.0W~100W (APPROX) WEIGHT : 255g (APPROX)

Typical Applications **Cordless Power Tools :** Drill / Cordless Garden Tool / Air Compressor
Toys and Models : Ride-on Toy

MODEL		VOLTAGE		NO LOAD		AT MAXIMUM EFFICIENCY				STALL		
		OPERATING RANGE	NOMINAL	SPEED	CURRENT	SPEED	CURRENT	TORQUE	OUTPUT	TORQUE		CURRENT
				r/min	A	r/min	A	mN·m	g·cm	W	mN·m	g·cm	A
RS-550PC-7527	(*1)	6.0~14.4	12V CONSTANT	18200	1.15	16130	8.97	47.8	488	80.7	421	4292	70.0
RS-550VC-7527	(*1)	6.0~14.4	14.4V CONSTANT	19800	1.30	17620	10.5	64.7	660	119	588	5994	85.0

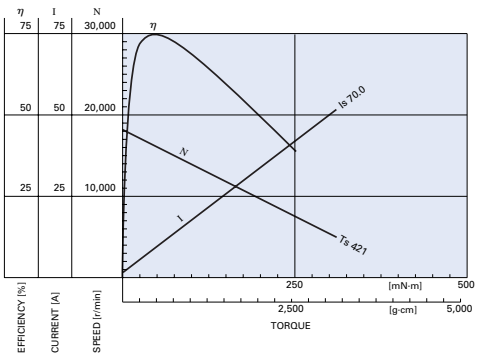
(*1) CCW shifted commutation (CCW+)



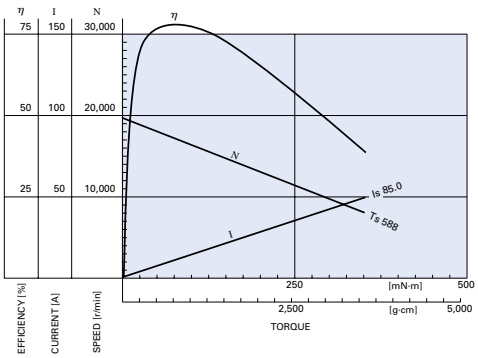
Usable machine screw length 4.0 max. from motor mounting surface.

UNIT: MILLIMETERS

RS-550PC-7527 12.0V



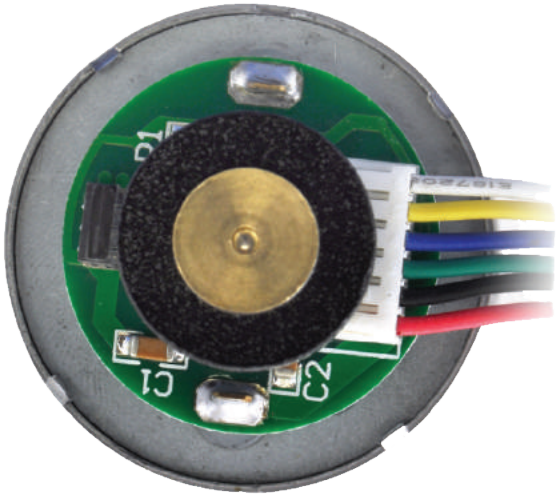
RS-550VC-7527 14.4V



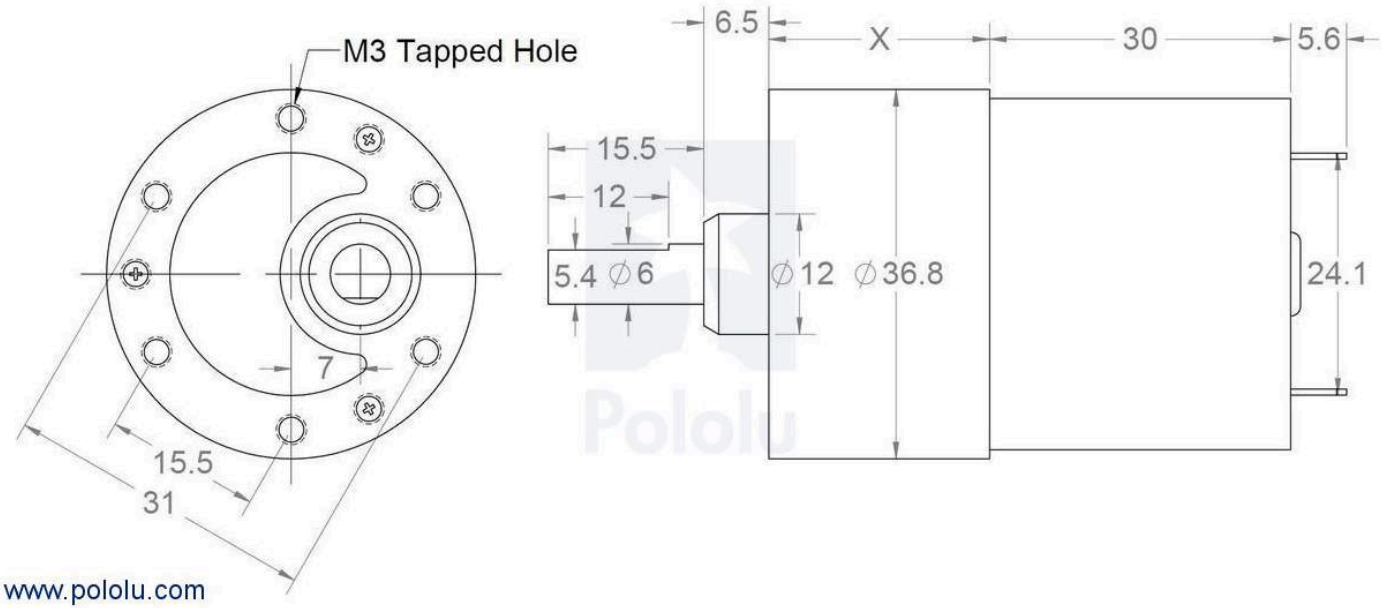
MABUCHI MOTOR CO.,LTD. **Headquarters** 430 Matsuhidai, Matsudo-shi, Chiba-ken, 270-2280, Japan Tel:81-47-384-9523 Fax:81-47-385-2026 (Sales Dept.)

B.2 Pololu Gear DC motor

Encoder for Pololu Gear Motor



Red:	Motor +
Black :	Motor -
Green	Encoder GND
Blue:	Encoder VCC
Yellow:	Endcoder channel A
White:	Endcoder channel B


M3 Tapped Hole
6.5
X
30
5.6
15.5
12
5.4
∅6
∅12
∅36.8
24.1
7
15.5
31

B.3 Servo, Savöx

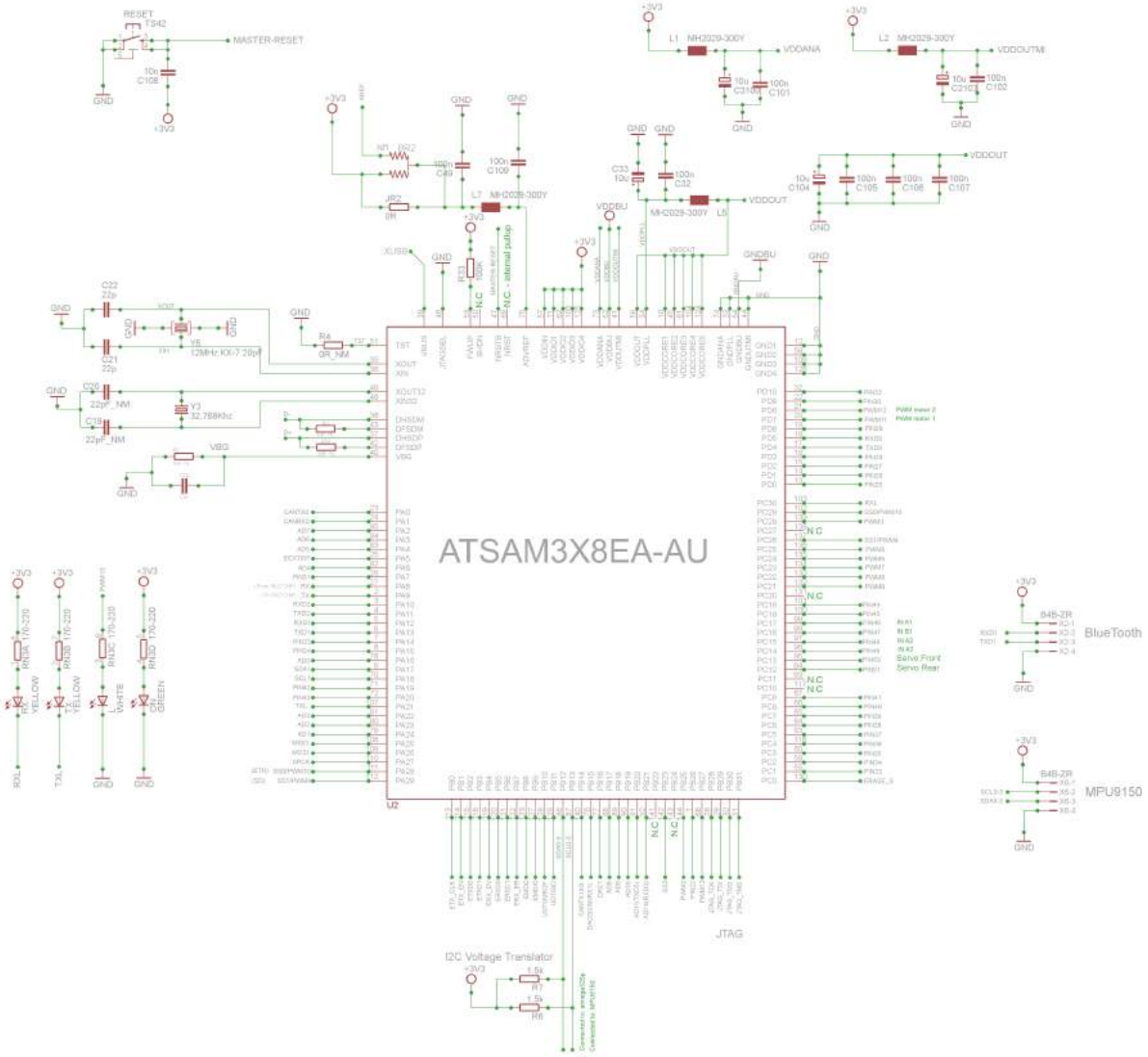
This is the Savox SV-0235MG Super Torque Steel Gear Digital Servo.

Features:

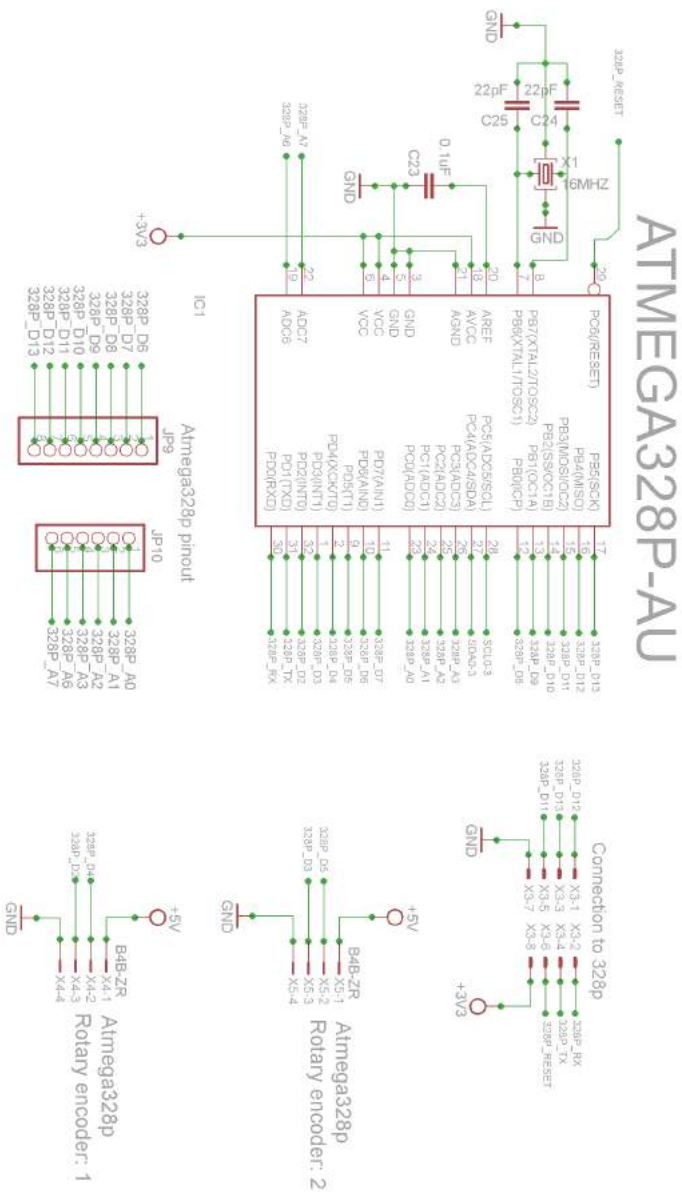
- Combines leading edge technology with super high 12 bit (4096) resolution and unique steel gears.
- Super light-weight.
- High speed, incredible efficiency, and low power consumption.
- Extremely strong metal steel gears ensure long-life and durability.
- The full aluminum case design not only looks good but also allows for cooler and smoother operating temps.
- Our servos are totally green ñ from materials to production, these servos are environmentally friendly.
- Ideal for 5th scale vehicles

Dimensions(mm): 65.8x30x57.4
Weight(g): 200
Speed(@6.0V sec/60): .18
Torque(@6.0V oz-in): 388.8
Speed(@7.4V sec/60): .15
Torque(@7.4V oz-in): 486.1
Gear: Metal
Bearing: 2BB
Case: Full Aluminum
15 Tooth Spline

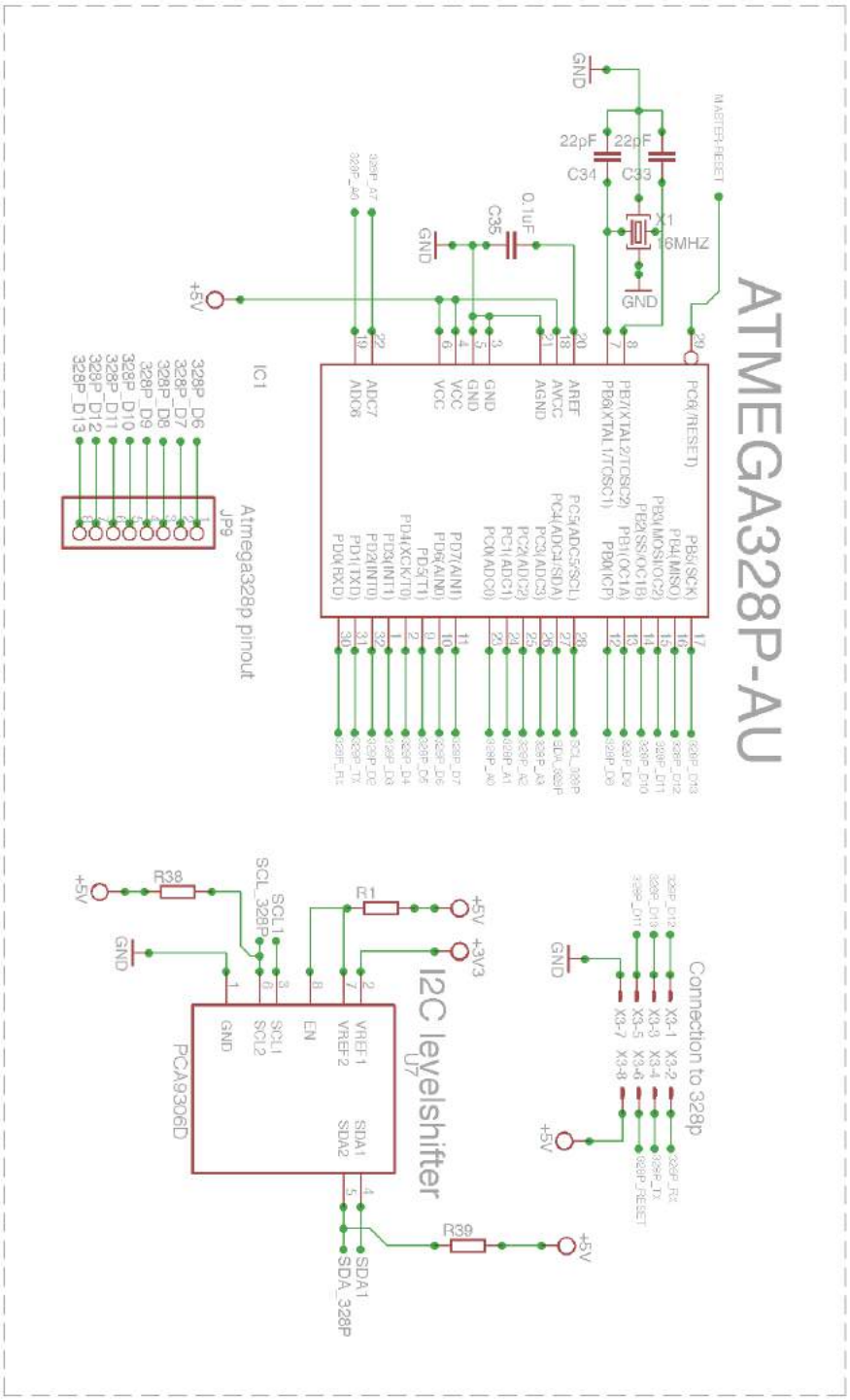
B.4 Main MCU, schematics



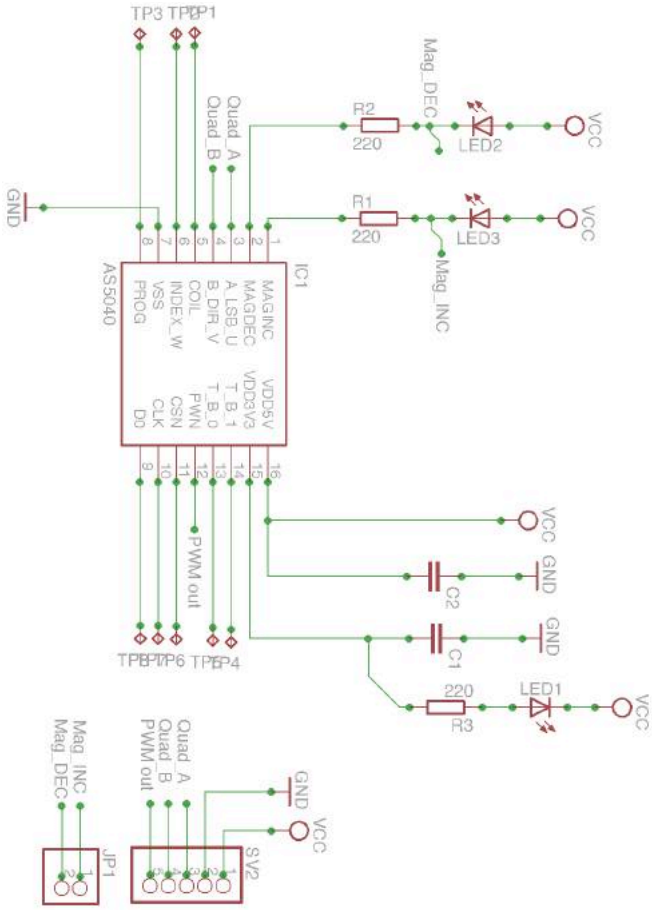
B.5 Slave MCU for Rotational Encoders, schematics



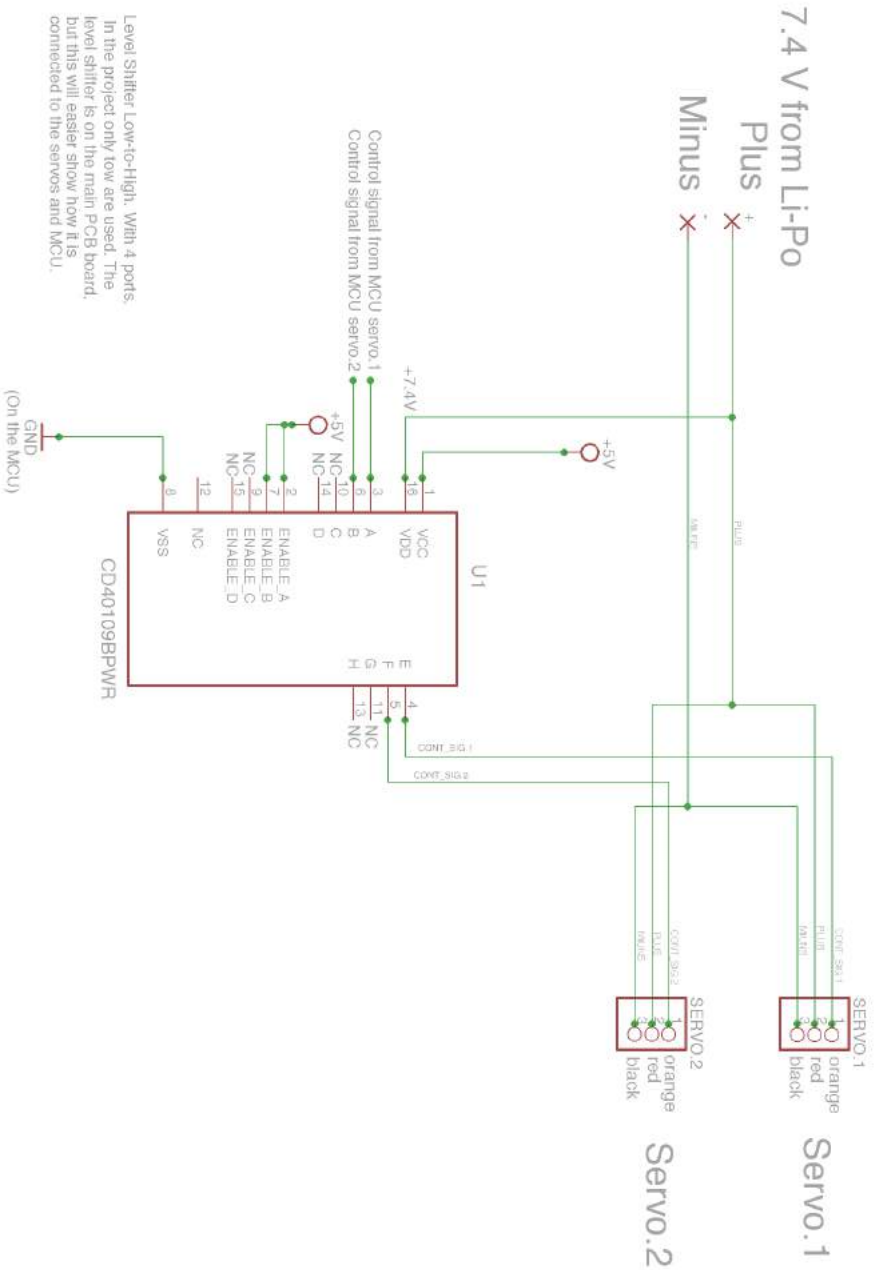
B.6 On-board Slave MCU, schematics



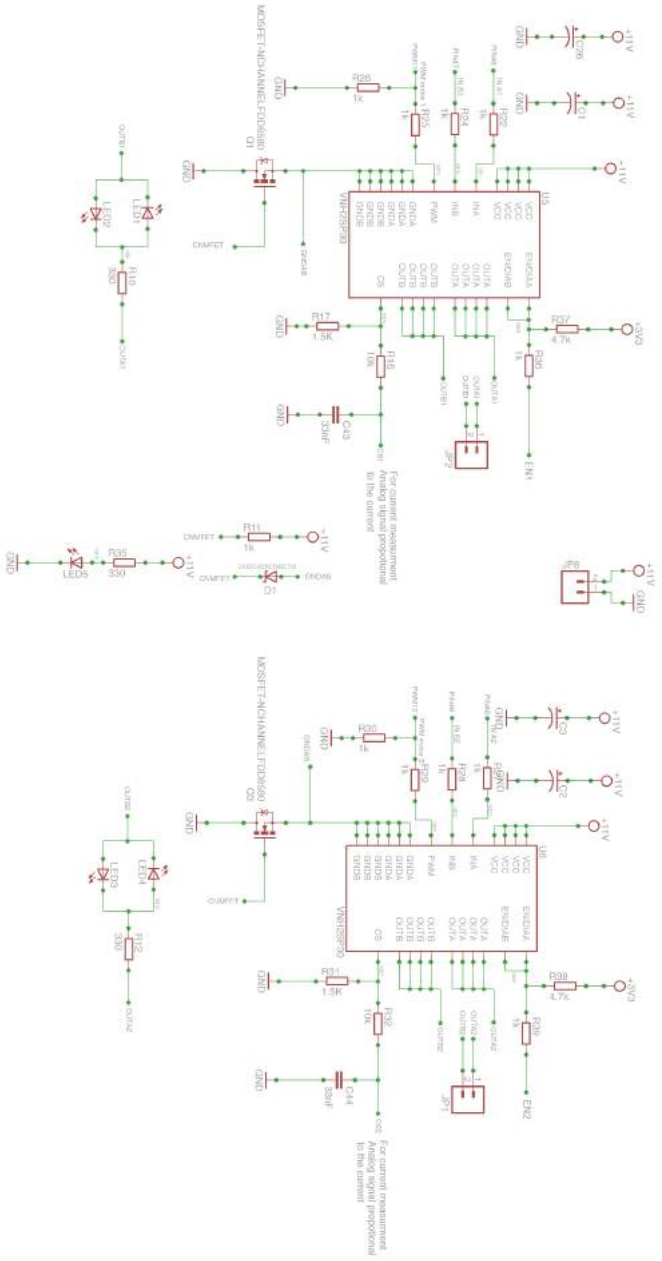
B.7 Rotary magnet encoder, schematics



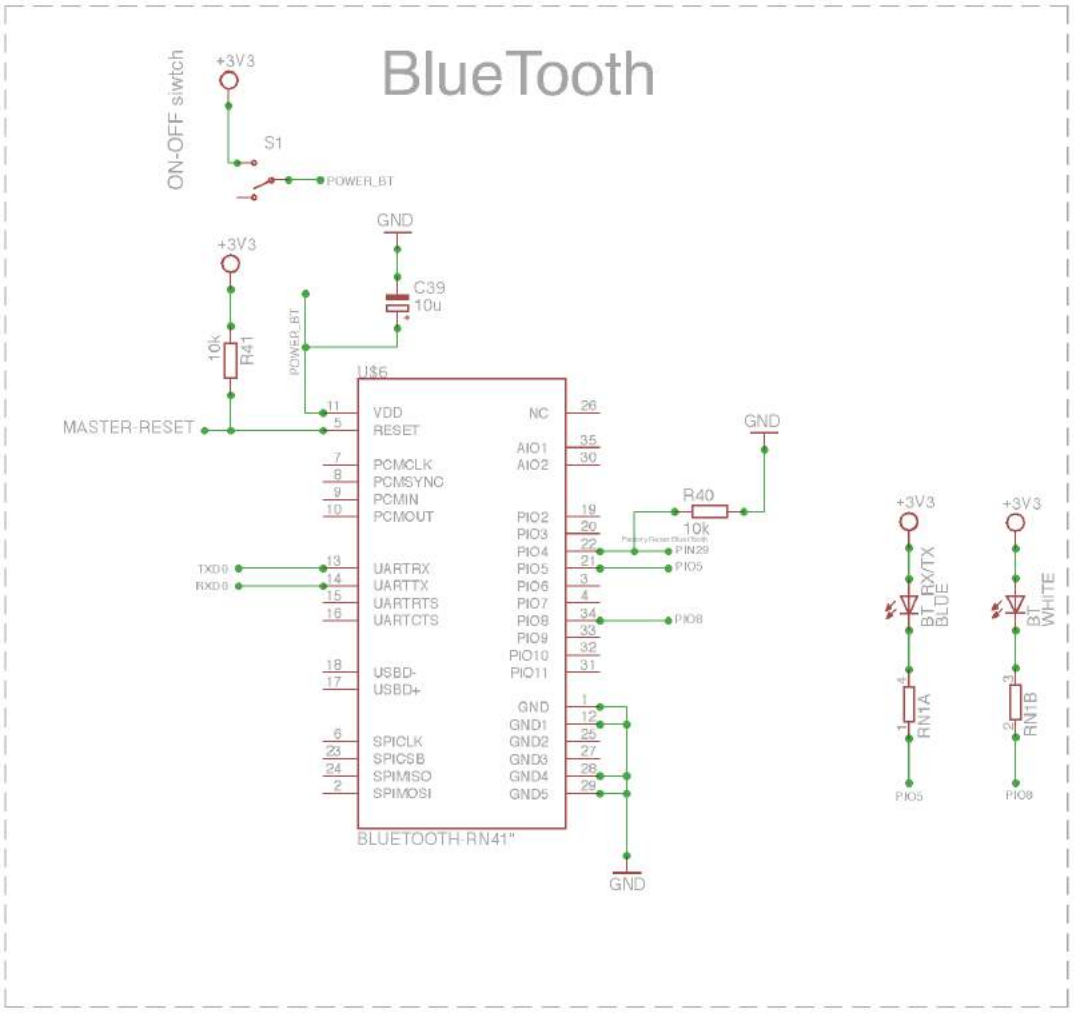
B.8 Servo controller, schematics



B.9 Motor controller, schematics



B.10 BlueTooth, schematics



B.11 Main PCB

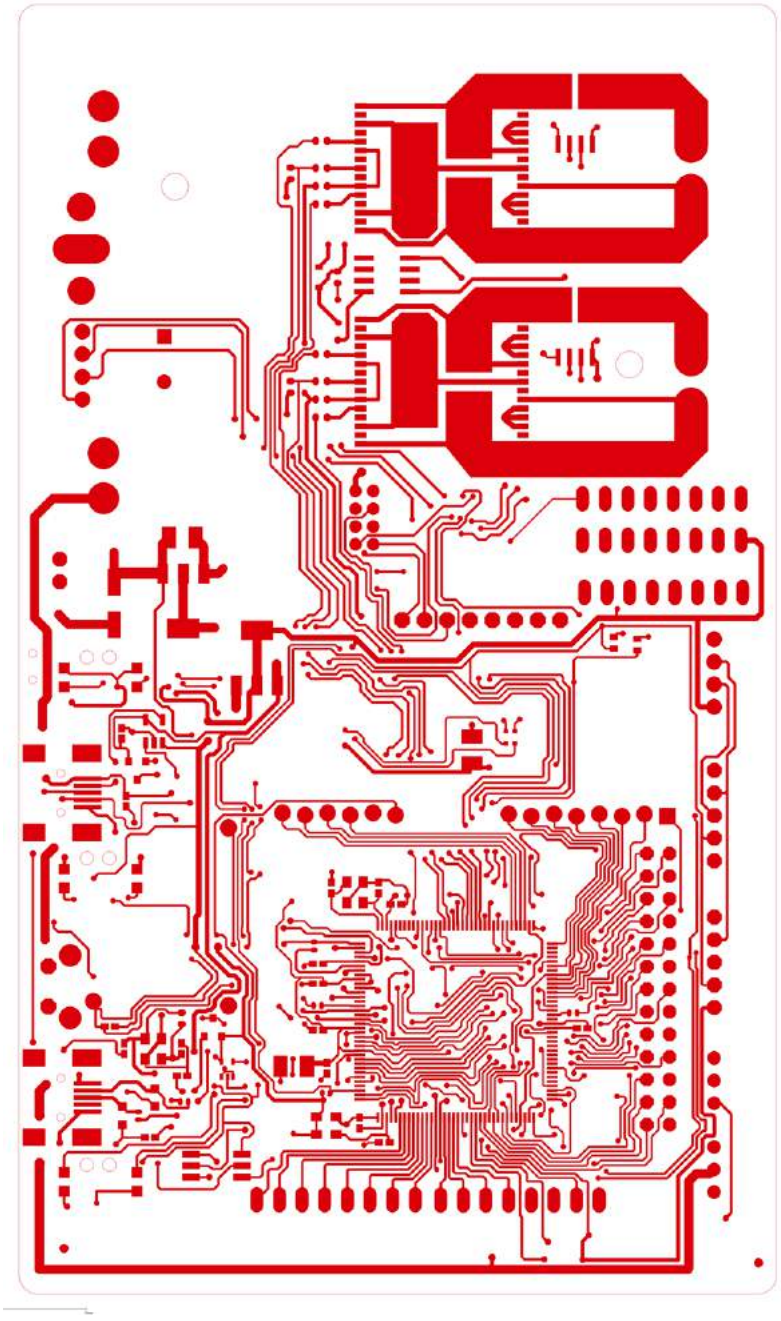


Figure B.1: Top copper Layer

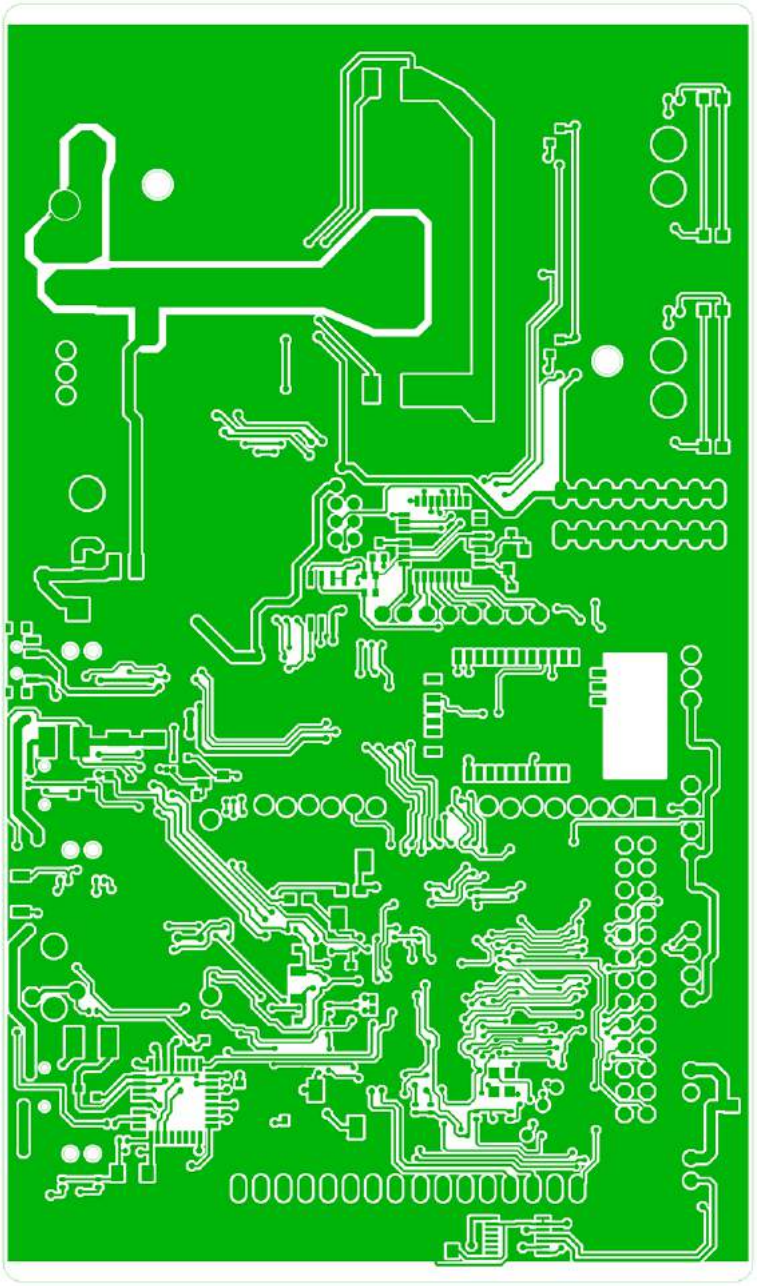


Figure B.2: bottom copper Layer

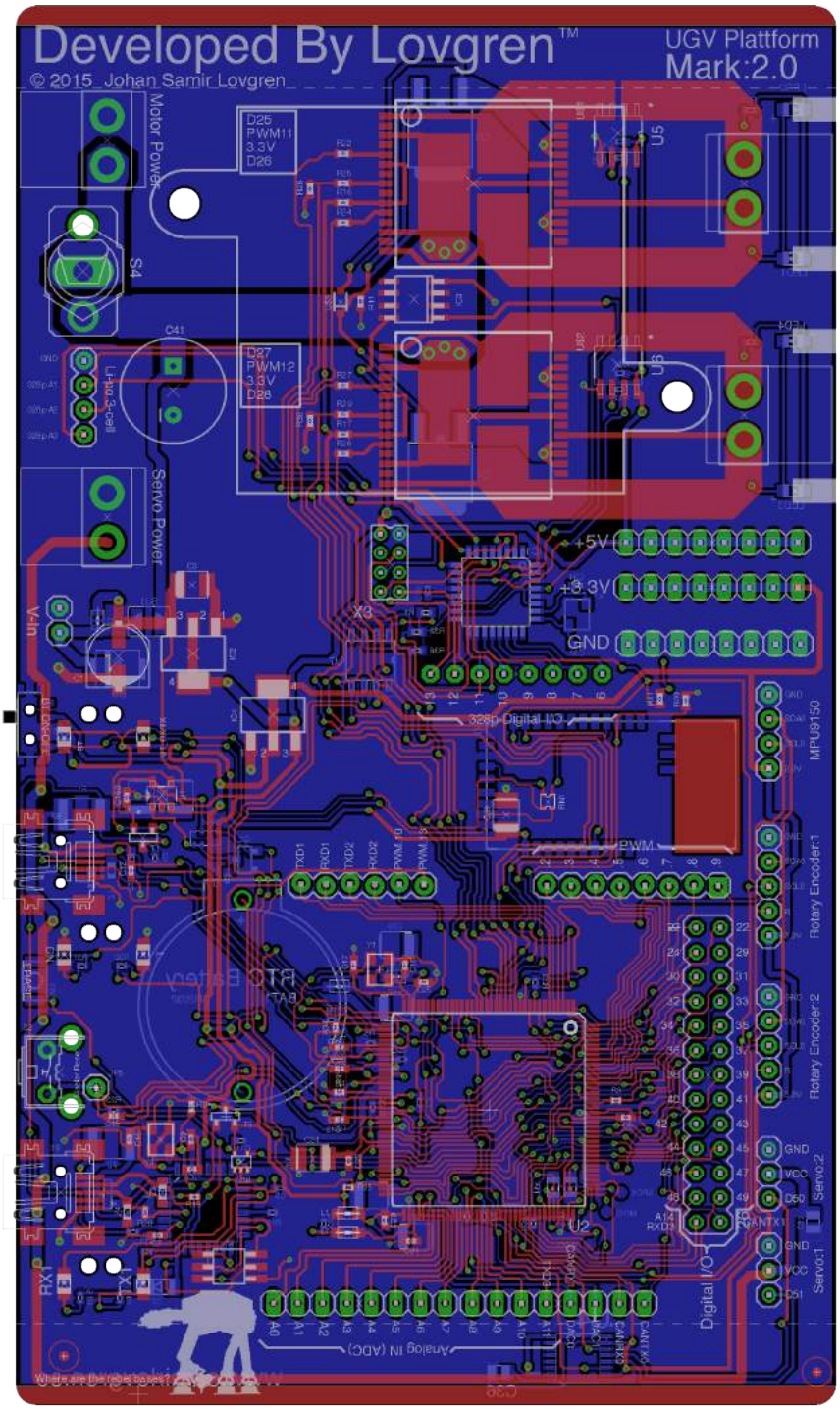


Figure B.3: All layers and silkscreen

C

Appendix C

C.1 MPU Test Code

```
// MPU-9150 Accelerometer + Gyro + Compass
//
// By Samir Lovgren
// June 2014
//
//
// Documentation:
// - The InvenSense documents:
//   - "MPU-9150 Product Specification Revision 4.0",
//     PS-MPU-9150A.pdf



#include <Wire.h>

// Register names according to the datasheet.
// According to the InvenSense document
// "MPU-9150 Register Map and Descriptions Revision 4.0",

//----------[INTERNAL Adress to the MPU9150]----------//

#define MPU9150_SELF_TEST_X      0x0D   // R/W
#define MPU9150_SELF_TEST_Y      0x0E   // R/W
#define MPU9150_SELF_TEST_X      0x0F   // R/W
#define MPU9150_SELF_TEST_A      0x10   // R/W
#define MPU9150_SMPLRT_DIV       0x19   // R/W
#define MPU9150_CONFIG           0x1A   // R/W
#define MPU9150_GYRO_CONFIG      0x1B   // R/W
#define MPU9150_ACCEL_CONFIG     0x1C   // R/W
#define MPU9150_FF_THR           0x1D   // R/W
#define MPU9150_FF_DUR           0x1E   // R/W
#define MPU9150_MOT_THR          0x1F   // R/W
#define MPU9150_MOT_DUR          0x20   // R/W
#define MPU9150_ZRMOT_THR        0x21   // R/W
#define MPU9150_ZRMOT_DUR        0x22   // R/W
#define MPU9150_FIFO_EN          0x23   // R/W
#define MPU9150_I2C_MST_CTRL     0x24   // R/W
#define MPU9150_I2C_SLV0_ADDR    0x25   // R/W
#define MPU9150_I2C_SLV0_REG     0x26   // R/W
#define MPU9150_I2C_SLV0_CTRL    0x27   // R/W
#define MPU9150_I2C_SLV1_ADDR    0x28   // R/W
#define MPU9150_I2C_SLV1_REG     0x29   // R/W
#define MPU9150_I2C_SLV1_CTRL    0x2A   // R/W
#define MPU9150_I2C_SLV2_ADDR    0x2B   // R/W
#define MPU9150_I2C_SLV2_REG     0x2C   // R/W
#define MPU9150_I2C_SLV2_CTRL    0x2D   // R/W
#define MPU9150_I2C_SLV3_ADDR    0x2E   // R/W
#define MPU9150_I2C_SLV3_REG     0x2F   // R/W
#define MPU9150_I2C_SLV3_CTRL    0x30   // R/W
#define MPU9150_I2C_SLV4_ADDR    0x31   // R/W
#define MPU9150_I2C_SLV4_REG     0x32   // R/W
#define MPU9150_I2C_SLV4_DO      0x33   // R/W
#define MPU9150_I2C_SLV4_CTRL    0x34   // R/W
#define MPU9150_I2C_SLV4_DI      0x35   // R
#define MPU9150_I2C_MST_STATUS   0x36   // R
#define MPU9150_INT_PIN_CFG      0x37   // R/W
#define MPU9150_INT_ENABLE       0x38   // R/W
#define MPU950_DMP_INT_STATUS    0x39  // Check DMP interrupt
#define MPU9150_INT_STATUS       0x3A   // R
#define MPU9150_ACCEL_XOUT_H     0x3B   // R
#define MPU9150_ACCEL_XOUT_L     0x3C   // R
#define MPU9150_ACCEL_YOUT_H     0x3D   // R
#define MPU9150_ACCEL_YOUT_L     0x3E   // R
#define MPU9150_ACCEL_ZOUT_H     0x3F   // R
#define MPU9150_ACCEL_ZOUT_L     0x40   // R
#define MPU9150_TEMP_OUT_H       0x41   // R
```

C.2 MPU Test Code

```
#define MPU9150_TEMP_OUT_L       0x42  // R
#define MPU9150_GYRO_XOUT_H      0x43  // R
#define MPU9150_GYRO_XOUT_L      0x44  // R
#define MPU9150_GYRO_YOUT_H      0x45  // R
#define MPU9150_GYRO_YOUT_L      0x46  // R
#define MPU9150_GYRO_ZOUT_H      0x47  // R
#define MPU9150_GYRO_ZOUT_L      0x48  // R
#define MPU9150_EXT_SENS_DATA_00  0x49  // R
#define MPU9150_EXT_SENS_DATA_01  0x4A  // R
#define MPU9150_EXT_SENS_DATA_02  0x4B  // R
#define MPU9150_EXT_SENS_DATA_03  0x4C  // R
#define MPU9150_EXT_SENS_DATA_04  0x4D  // R
#define MPU9150_EXT_SENS_DATA_05  0x4E  // R
#define MPU9150_EXT_SENS_DATA_06  0x4F  // R
#define MPU9150_EXT_SENS_DATA_07  0x50  // R
#define MPU9150_EXT_SENS_DATA_08  0x51  // R
#define MPU9150_EXT_SENS_DATA_09  0x52  // R
#define MPU9150_EXT_SENS_DATA_10  0x53  // R
#define MPU9150_EXT_SENS_DATA_11  0x54  // R
#define MPU9150_EXT_SENS_DATA_12  0x55  // R
#define MPU9150_EXT_SENS_DATA_13  0x56  // R
#define MPU9150_EXT_SENS_DATA_14  0x57  // R
#define MPU9150_EXT_SENS_DATA_15  0x58  // R
#define MPU9150_EXT_SENS_DATA_16  0x59  // R
#define MPU9150_EXT_SENS_DATA_17  0x5A  // R
#define MPU9150_EXT_SENS_DATA_18  0x5B  // R
#define MPU9150_EXT_SENS_DATA_19  0x5C  // R
#define MPU9150_EXT_SENS_DATA_20  0x5D  // R
#define MPU9150_EXT_SENS_DATA_21  0x5E  // R
#define MPU9150_EXT_SENS_DATA_22  0x5F  // R
#define MPU9150_EXT_SENS_DATA_23  0x60  // R
#define MPU9150_MOT_DETECT_STATUS  0x61  // R
#define MPU9150_I2C_SLV0_DO      0x63  // R/W
#define MPU9150_I2C_SLV1_DO      0x64  // R/W
#define MPU9150_I2C_SLV2_DO      0x65  // R/W
#define MPU9150_I2C_SLV3_DO      0x66  // R/W
#define MPU9150_I2C_MST_DELAY_CTRL 0x67  // R/W
#define MPU9150_SIGNAL_PATH_RESET  0x68  // R/W
#define MPU9150_MOT_DETECT_CTRL   0x69  // R/W
#define MPU9150_USER_CTRL        0x6A  // R/W // Bit 7 enable DMP, bit 3 reset DMP
#define MPU9150_PWR_MGMT_1       0x6B  // R/W // Device defaults to the SLEEP mode
#define MPU9150_PWR_MGMT_2       0x6C  // R/W
#define MPU9150_DMP_BANK         0x6D  // Activates a specific bank in the DMP
#define MPU9150_DMP_RW_PNT       0x6E  // Set read/write pointer to a specific start address in specified
DMP bank
#define MPU9150_DMP_REG          0x6F  // Register in DMP from which to read or to which to write
#define MPU9150_FIFO_COUNTH      0x72  // R/W
#define MPU9150_FIFO_COUNTL      0x73  // R/W
#define MPU9150_FIFO_R_W         0x74  // R/W
#define MPU9150_WHO_AM_I         0x75  // R // Should return 0x68

//MPU9150 Compass
#define MPU9150_CMPS_XOUT_L      0x4A  // R
#define MPU9150_CMPS_XOUT_H      0x4B  // R
#define MPU9150_CMPS_YOUT_L      0x4C  // R
#define MPU9150_CMPS_YOUT_H      0x4D  // R
#define MPU9150_CMPS_ZOUT_L      0x4E  // R
#define MPU9150_CMPS_ZOUT_H      0x4F  // R


// I2C address 0x69 could be 0x68 depends on your wiring.
int MPU9150_I2C_ADDRESS = 0x69;


//Variables where our values can be stored
```

C.3 MPU Test Code

```
double cmps[3];
double accl[3];
double gyro[3];
double temp;



unsigned long lastCmdTime = 0;

void setup()
{
  // Initialize the Serial Bus for printing data.
  Serial.begin(115200);

  // Initialize the 'Wire' class for the I2C-bus.
  Wire.begin();

  // Clear the 'sleep' bit to start the sensor.
  MPU9150_writeSensor(MPU9150_PWR_MGMT_1, 0);

  MPU9150_setupCompass();
}


void loop()
{
// Compass, Gyro, Acceleration.  In x,y,z


cmps [0] = MPU9150_readSensor(MPU9150_CMPS_XOUT_L,MPU9150_CMPS_XOUT_H);
cmps [1] = MPU9150_readSensor(MPU9150_CMPS_YOUT_L,MPU9150_CMPS_YOUT_H);
cmps [2] = MPU9150_readSensor(MPU9150_CMPS_ZOUT_L,MPU9150_CMPS_ZOUT_H);


gyro [0] = MPU9150_readSensor(MPU9150_GYRO_XOUT_L,MPU9150_GYRO_XOUT_H);
gyro [1] = MPU9150_readSensor(MPU9150_GYRO_YOUT_L,MPU9150_GYRO_YOUT_H);
gyro [2] = MPU9150_readSensor(MPU9150_ZYRO_YOUT_L,MPU9150_GYRO_ZOUT_H);

accl [0] = MPU9150_readSensor(MPU9150_ACCEL_XOUT_L,MPU9150_ACCEL_XOUT_H);
accl [1] = MPU9150_readSensor(MPU9150_ACCEL_YOUT_L,MPU9150_ACCEL_YOUT_H);
accl [2] = MPU9150_readSensor(MPU9150_ACCEL_ZOUT_L,MPU9150_ACCEL_ZOUT_H));



//if  you want to plot all data from the IMU, just add it!


//converting the raw data from the gyro to rad/sec

  Serial.print(gyro [0]* (250/32768)*(pi/180))
  Serial.print(" ,");
  Serial.print(gyro [1]* (250/32768)*(pi/180))
  Serial.print(" ,");
  Serial.print(gyro [2]* (250/32768)*(pi/180)))
  Serial.print(",");
 Serial.println( millis() ); //to get the time stamp right

  delay(20); //nessesary, else serial port wont manage all data if we dont have any delay

}// end main loop
```

C.4 MPU Test Code

```
//--------------------[functions]------------------------------//
void MPU9150_setupCompass(){
 MPU9150_I2C_ADDRESS = 0x0C;     //change Adress to Compass

 MPU9150_writeSensor(0x0A, 0x00); //PowerDownMode
 MPU9150_writeSensor(0x0A, 0x0F); //SelfTest
 MPU9150_writeSensor(0x0A, 0x00); //PowerDownMode

 MPU9150_I2C_ADDRESS = 0x69;     //change Adress to MPU

 MPU9150_writeSensor(0x24, 0x40); //Wait for Data at Slave0
 MPU9150_writeSensor(0x25, 0x8C); //Set i2c address at slave0 at 0x0C
 MPU9150_writeSensor(0x26, 0x02); //Set where reading at slave 0 starts
 MPU9150_writeSensor(0x27, 0x88); //set offset at start reading and enable
 MPU9150_writeSensor(0x28, 0x0C); //set i2c address at slv1 at 0x0C
 MPU9150_writeSensor(0x29, 0x0A); //Set where reading at slave 1 starts
 MPU9150_writeSensor(0x2A, 0x81); //Enable at set length to 1
 MPU9150_writeSensor(0x64, 0x01); //overvride register
 MPU9150_writeSensor(0x67, 0x03); //set delay rate
 MPU9150_writeSensor(0x01, 0x80);

 MPU9150_writeSensor(0x34, 0x04); //set i2c slv4 delay
 MPU9150_writeSensor(0x64, 0x00); //override register
 MPU9150_writeSensor(0x6A, 0x00); //clear usr setting
 MPU9150_writeSensor(0x64, 0x01); //override register
 MPU9150_writeSensor(0x6A, 0x20); //enable master i2c mode
 MPU9150_writeSensor(0x34, 0x13); //disable slv4
}


//-----------------[Read TOW from the sensor]-----------------//
int MPU9150_readSensor(int addrL, int addrH){
 Wire.beginTransmission(MPU9150_I2C_ADDRESS);
 Wire.write(addrL);
 Wire.endTransmission(false);
 Wire.requestFrom(MPU9150_I2C_ADDRESS, 1, true);
 byte L = Wire.read();
 Wire.beginTransmission(MPU9150_I2C_ADDRESS);
 Wire.write(addrH);
 Wire.endTransmission(false);
 Wire.requestFrom(MPU9150_I2C_ADDRESS, 1, true);
 byte H = Wire.read();

 return (H<<8)+L;
}

//-----------------[Read ONE from the sensor]-----------------//
int MPU9150_readSensor(int addr){
 Wire.beginTransmission(MPU9150_I2C_ADDRESS);
 Wire.write(addr);
 Wire.endTransmission(false);
 Wire.requestFrom(MPU9150_I2C_ADDRESS, 1, true);
 return Wire.read();
}

//-----------------[Write to the sensor]-----------------//
int MPU9150_writeSensor(int addr,int data){
 Wire.beginTransmission(MPU9150_I2C_ADDRESS);
 Wire.write(addr);
 Wire.write(data);
 Wire.endTransmission(true);
 return 1;
}
```

C.5 MCU to Matlab code

```
clear all
clc
pause(1);
pi = 3.14;
%Defined Properties

serialPort = serial ('/dev/tty.UGV_Crab-C506-RNI-SPP', 'BaudRate' ,115200);
pause(1);

%For the live plot
plotTitle = 'Live Plot From IMU';  % plot title
xLabel = 'Elapsed Time (s)';    % x-axis label
yLabel = 'Data';              % y-axis label
plotGrid = 'on';              % 'off' to turn off grid
min = -10;                    % set y-min
max =  10 ;                   % set y-max
scrollWidth = 10;             % display period in plot, plot entire data log if <= 0
delay = 0.001;                % make sure sample faster than resolution

%Define Function Variables
time = 0;
data = 0;
count = 0;

compX = 0;
compY = 0;
compZ = 0;
gyrX = 0;
gyrY = 0;
gyrZ = 0;
accX = 0;
accY = 0;
accZ = 0;
t = 0;

compX_in = 0;
compY_in = 0;
compZ_in = 0;
gyrX_in = 0;
gyrY_in = 0;
gyrZ_in = 0;
accX_in = 0;
accY_in = 0;
accZ_in = 0;
t_in = 0;

%-----------------[Set up Plot]----------------------%
plotGraph = plot(time,gyrZ,'-mo',...
```

C.6 MCU to Matlab code

```
            'LineWidth',1,...
            'MarkerEdgeColor','k',...
            'MarkerFaceColor',[.49 1 .63],...
            'MarkerSize',3);

title(plotTitle,'FontSize',25);
xlabel(xLabel,'FontSize',15);
ylabel(yLabel,'FontSize',15);
axis([0 10 min max]);
grid(plotGrid);

%Open Serial COM Port
s = serial(serialPort);
disp('Close Plot to End Session');
fopen(s);

tic




%Logg andf Plot starts here. Creating a window, and when the window is closed
by the operator the logging stops!

%------------------------------------------------------------------------------------------------%
while ishandle(plotGraph) %Loop when port is open is Active

   %Read the data from the serial port and then stors the data
   dat = fscanf(s,'%s'); %Read Data from Serial as double
   [compX_in compY_in compZ_in gyrX_in gyrY_in gyrZ_in accX_in accY_in accZ_in
t_in] = strread(dat, '%d %d %d %d %d %d %d %d %d %d', 'delimiter', ',')



  compX = [compX; compX_in];
  compY = [compY; compY_in];
  compZ = [compZ; compZ_in];

  gyrX = [gyrX; gyrX_in];
  gyrY = [gyrY; gyrY_in];
  gyrZ = [gyrZ; gyrZ_in];

  accX = [accX; accX_in];
  accY = [accY; accY_in];
  accZ = [accZ; accZ_in];

  %t = [t; t_in];
```

C.7 MCU to Matlab code

```
        count = count + 1;
        time(count) = toc;    %Extract Elapsed Time


 gyrY_in = gyrY_in*(250.0/32768.0)*(3.14/180);


%-----------------------------------------------%
%This is live plot of ONE of the datas... uncomment if you dont want a live plot,
%slows down the script

    if( ~isempty(dat) && isfloat(gyrY_in) ) %Make sure Data Type is Correct
      %  count = count + 1;
      %  time(count) = toc;    %Extract Elapsed Time
       data(count) = gyrY_in;
        %Extract 1st Data Element make it to radians

       %Set Axis according to Scroll Width
       if(scrollWidth > 0)
       set(plotGraph,'XData',time(time > time(count)-
scrollWidth),'YData',data(time > time(count)-scrollWidth));
       axis([time(count)-scrollWidth time(count) min max]);
       else
       set(plotGraph,'XData',time,'YData',data);
       axis([0 time(count) min max]);
       end

       %delay soMATLAB can Update Plot
       pause(delay);
    end % end liveplot
  %-----------------------------------------------%



   pause(delay);

end %en while window is opend


 %-------------------------------------------------------------------------------------------------%

%Close Serial COM Port and Delete useless Variables
fclose(s);
delete(s);
clear count dat delay max min plotGraph plotGrid plotTitle s ...
     scrollWidth serialPort xLabel yLabel;

disp('Session Terminated...');
```

C.8 LM335 Test code

```
//LM335 Temperature sensor for Active Cooling
//by Johan S Lovgren, 2014

int LM335 = A3;
double tempK=0, tempC=0;

//-------------------------------------------[Setup]-------------------------------------------//
void setup(){
pinMode (LM335,INPUT); //set the analog pin named LM335 to a input pin
Serial.begin(57600);    //Setup serial
}//end setup


//-------------------------------------------[Main]-------------------------------------------//

void loop(){

tempK = analogRead(LM335) * 0.004882812 * 100;    // Kelvins
tempC = tempK - 273.15-3;    //Convert from Kelvin to Celsius

//Print all the values to Serial
Serial.print("Celsius: ");
Serial.println(tempC);
delay(100);

}//end main
```

D

Appendix D

D.1 Various images

Here the various images from the project is presented.

Figure D.1: image of the drive axis with the extensions

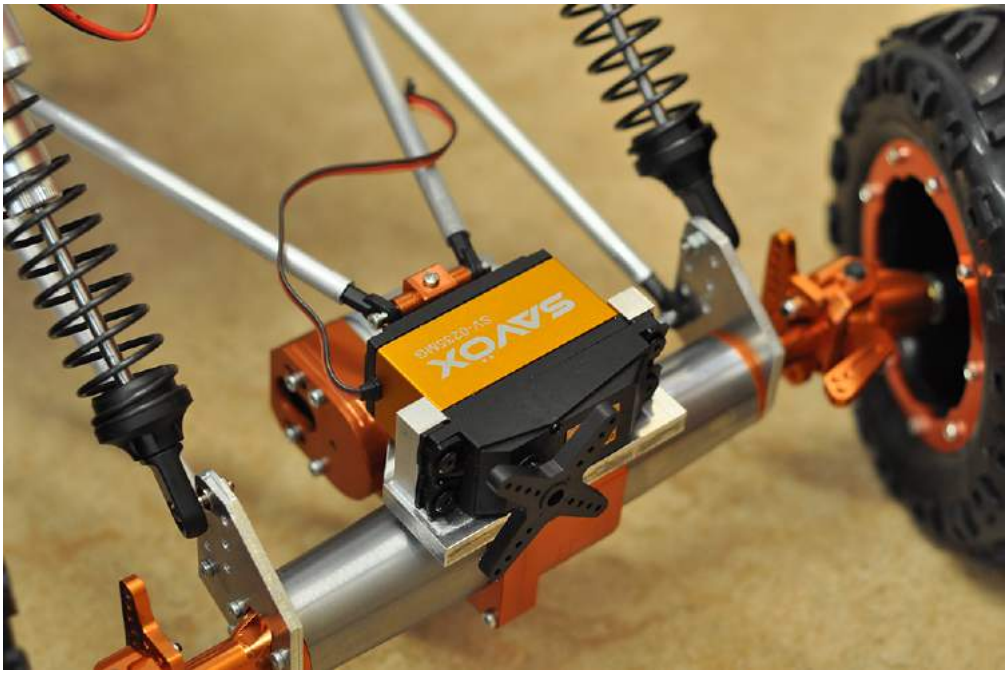


Figure D.2: Servo mounted on the servo-plate

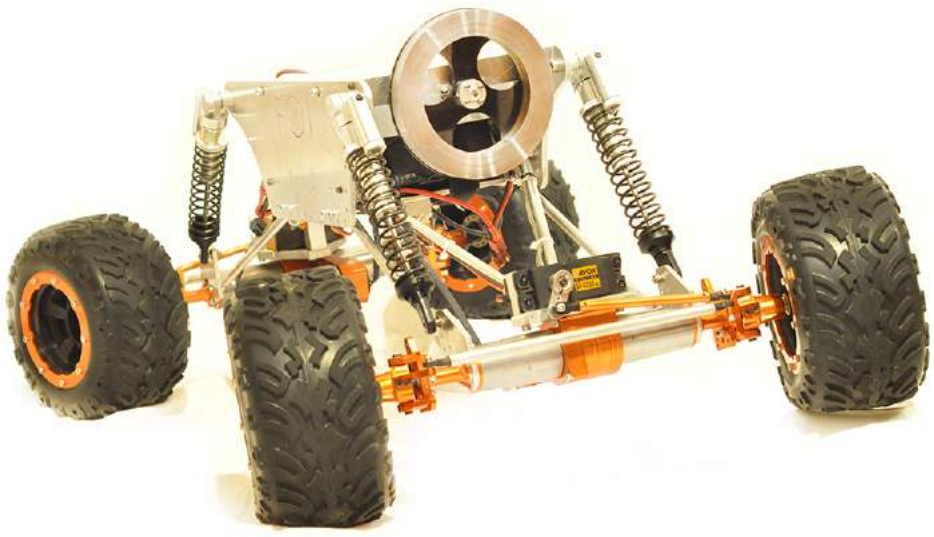


Figure D.3: Complete UGV

Figure D.4: Test run of the UGV with 3kg load

Figure D.5: Test run of the UGV with high COG

Figure D.6: During Rollover-test

Figure D.7: Outdoor test of the UGV

During this test at KTH in Stockholm, Discovery Channel filmed the tests. So it may be broadcast on tv, my 5 seconds of fame.

Figure D.8: The UGV with one of the sensor system

Figure D.9: Johan S Lövgren with the UGV

May the force be with you...