NTNU
Norwegian University of
Science and Technology

Autonomous Bicycle

The First Self Balanced Ride

Dag Christian Ånnestad

Master of Science in Engineering Cybernetics
Submission date: September 2011
Supervisor: Amund Skavhaug, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

NTNU
Norwegian University of Science and Technology

Faculty of Information Technology, Mathematics and Electrical Engineering
Department of Engineering Cybernetics

Master's thesis

Name of candidate: Dag Christian Ånnestad
Program: Engineering Cybernetics, Adapted Computer Systems
Assignment Title: Autonomous Bicycle - The First Self Balanced Ride
Assignment Text:

The Department of Engineering Cybernetics at the Norwegian University of Science and Technology wish to complete the work of developing the Autonomous Bicycle. The work shall be done with emphasis on development of a total system, in such a way that future additions are practically possible.

The assignment consists of the following:

1. The candidate shall familiarize himself with earlier work.
2. Identify and implement necessary changes to the framework.
3. Make necessary changes to the framework in order to complete the task.
4. Identify the parameters and implement a model for the bicycle.
5. Identify suitable control algorithms and compare through simulations.
6. Implement a suitable controller for self balance on the physical bicycle.

Assignment given: May 6. 2011
Submission date: September 26. 2011
Performed at the Department of Engineering Cybernetics
Supervisor: Associate Professor Amund Skavhaug

Trondheim, May 6. 2011

Associate Professor Amund Skavhaug

Preface

My contribution to the Autonomous Bicycle is finished and the Autonomous Bicycle is finally capable of performing self balanced rides. I am looking forward to begin the next chapter of my life. Throughout this thesis I have enjoyed the versatility of the challenges associated with the project. Especially the ability to test how theory works in a real world application. I would like to thank Amund Skavhaug for giving me this project, as a friendly, helpful and inspiring supervisor. I would like to thank the people at the Department of Engineering Cybernetics mechanical workshop for the fast help producing new brackets during the steer gearbox breakdown. And at last but not least thank my girlfriend Solveig for help during some of the field test and keeping up with my many late hours. Finally I would like to recommend the Institute of Engineering Cybernetics to assign an equally large budget next year, so that the gearbox could be changed and the Autonomous Bicycle in the future could be completed with zero forward speed balance.

Trondheim, September 26. 2011

Dag Christian Ånnestad

Summary

The idea of an autonomous bicycle originates from Jens G. Balchen who wanted to make an unmanned autonomous bicycle. The idea was picked up by Amund Skavhaug who extended the idea with the concept of using an inverted pendulum to simulate a leaning rider. The previous attempts to develop a bicycle capable of performing an autonomous ride has so far all ended in failure. The main reason for the Department of Engineering Cybernetics is to develop such a bicycle is for use in recruitment and motivation of students. The main goal of this thesis is to develop a bicycle that after the implementation of a suitable control is capable of performing an autonomous ride.

The goal of this thesis is to create a controller making the bicycle capable of performing the first self balanced ride. The focus is not on implementing the most advanced controller but creating a system actually capable of performing this first ride. An equally important focus is that the framework delivered at the end of this thesis is capable of handling the further development towards a fully autonomous bicycle.

The author has during this thesis performed the additions needed in order to be able to implement a self balancing controller. The parameters of the real bicycle were measured and used to create a simulation environment of the bicycle in Simulink. Several controllers were simulated in Simulink, before a controller were implemented on the physical bicycle.

The physical bicycle delivered as a part of this thesis consists of a fully functional framework both capable and ready for the further development. A self balancing controller is implemented on the bicycle and the bicycle has performed it's first self balance ride.

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Problem . . . 1
  - 1.2.1 Leaving Out The Pendulum . . . 2
- 1.3 Previous Work . . . 2
- 1.4 Outline . . . 3
- 1.5 Project Delivery . . . 3

2 Background Theory **5**
- 2.1 The Whipple Bicycle Model . . . 5
  - 2.1.1 Bicycle Model Without Rider . . . 6
- 2.2 Control Theory . . . 13
  - 2.2.1 State-Space Representation . . . 13
  - 2.2.2 Proportional(P) To Proportional-Integral-Derivative(PID) Control . . . 13
  - 2.2.3 Linear-Quadratic Regulator(LQR) Design . . . 15
  - 2.2.4 Kalman Estimator . . . 16

3 Description Of The Existing Bicycle **19**
- 3.1 Hardware . . . 20
- 3.2 Software . . . 21

4 Modifications And Additions To The Framework **23**
- 4.1 Incremental Encoder . . . 23
- 4.2 Simulink Model . . . 25
  - 4.2.1 Scaling And Calibration Of Analog Input And Output 25
  - 4.2.2 Edge Limiting The Steering And Pendulum Motor . . 27
  - 4.2.3 Orientation Changes To The IMU Roll Signal . . . 29
  - 4.2.4 Free Rotation Of The IMU Yaw Signal . . . 29
  - 4.2.5 Total Simulink Model . . . 32

5 The Catastrophe **33**
5.1 Fixing The Existing Gearbox 33
5.2 New Gearbox And Transmission 33

6 Measurement of Bicycle Parameters **37**
6.1 Wheel Radius 38
6.2 Tube Angles 38
6.3 Trail 39
6.4 Wheelbase 39
6.5 Mass 39
6.6 Center of Mass 39
6.6.1 Wheels 40
6.6.2 Rear Body Frame 40
6.6.3 Front Handlebar Frame 41
6.7 Moment of Inertia 42
6.7.1 Torsional Pendulum 42
6.7.2 Compound Pendulum 43
6.7.3 Wheels 44
6.7.4 Rear Body Frame 45
6.7.5 Front Handlebar Frame 47

7 Simulation **49**
7.1 Simulation Environment 49
7.1.1 Bicycle Model 49
7.1.2 Environment 50
7.1.3 Process And Measurement Noise 51
7.2 Controller Simulations 52
7.2.1 Proportional(P) Control 52
7.2.2 Linear-Quadratic Regulator(LQR) Control 53
7.2.3 Linear-Quadratic-Gaussian(LQG) control 55
7.2.4 Discussion And Comparison Of The Simulated Controllers 60

8 Real World Self Balancing **61**
8.1 Propulsion Speed Controller 61
8.2 Self Balance With A P Controller 62

9 Roadmap For Further Work **67**
9.1 Braking Capabilities 67
9.1.1 Motor Brake 67
9.1.2 Bicycle Brakes 68

9.2 Verifying And Updating The Model . . . 68
9.3 Leaning Rider - Low To Zero Forward Speed Balance . . . 68
9.4 Autopilot . . . 68
9.4.1 Without Any Additions . . . 69
9.4.2 With GPS . . . 69
9.4.3 With Computer Vision . . . 69
9.5 Collision Avoidance . . . 69

10 Discussion **71**
10.1 The Catastrophe . . . 71
10.2 Measurement Of Bicycle Parameters . . . 71
10.3 Simulations . . . 71
10.4 Real World Implementation . . . 72
10.5 Other . . . 72

11 Conclusion **73**

A Matlab Calculations **77**
A.1 Center Of Mass . . . 77
A.2 Moment Of Inertia . . . 78
A.3 List Of Parameters Needed By The Model . . . 80
A.4 Creating Model . . . 80
A.5 Observer And LQR-Controller . . . 82

B Incremental Decode PCB Board **85**
B.1 Incremental Decode Board Copper . . . 85
B.2 Incremental Decode Board Schematic . . . 86

C Steering Motor And Gear Data **87**
C.1 Steering Motor . . . 87
C.2 Steering Planetary Gear . . . 88
C.3 Steering Transmission Gears . . . 88
C.3.1 45 Teeth Spur Gear . . . 88
C.3.2 120 Teeth Spur Gear . . . 89
C.4 Total Steering Performance . . . 89

D How To Set Up A Host System **91**

E How To Start The Bicycle **93**

Acronyms And Abbreviations

AUTOSIM	Software simulation environment
C/C++	Programming language
GPS	Global Positioning System
I/O	Input and Output
IMU	Inertial Measurement Unit
LiPo	Lithium ion polymer
LQ	Linear Quadratic
LQG	Linear Quadratic Gaussian
LQR	Linear Quadratic Regulator
MATLAB	Software suite from Mathworks
Mti	IMU form xSense
PCB	Printed Circuit Board
PID	Proportional-Integral-Derivative
Putty	Terminal interface for windows
QNX	Real time operating system
RS-232	Communications interface
RTW	Real Time Workshop
Simulink	Simulation environment included in MATLAB
SPACAR	Software simulation environment
xmega	Micro-controller from Atmel

Table 1: Acronyms and abbreviations

Chapter 1

Introduction

1.1 Motivation

The work on developing an autonomous bicycle with control of an inverted pendulum, the steering angle and the propulsion speed started back in the 1980's, when Amund Skavhaug picked up an idea from Jens G. Balchen of developing an autonomous bicycle. The last few years there has been development on the bicycle from the 1980's by Loftum[1], Bjermeland[2], Fossum[3], Sølvberg[4], Brekke[5] and Hatlevoll[6] with various results, but no success developing an autonomous bicycle. The author of this thesis performed a complete rebuild of the bicycle during a project delivered this spring. With this as a very good starting point the author would like to complete this framework with the result of a bicycle capable of performing a self balance ride. The main reason for the Department of Engineering Cybernetics to develop such a bicycle is for use in both recruitment and motivation of students. The concept of controlling a bicycle with an inverted pendulum simulating a leaning rider is also an interesting subject, due to the fact that this is not done by anyone previously[1].

1.2 Problem

The goal of this thesis is to implement a self balancing controller for the bicycle without the use of the inverted pendulum. The task should be performed with a theoretical approach, by first creating a model and perform simulations before taking the step out into the real world. The following list gives an overview of the steps towards this goal:

[1] At least not to the knowledge of the author of this thesis.

- If needed, performing additions and upgrades to the framework.
- Implement basic protection features in software.
- Implement scaling and calibration of the inputs and outputs to the Simulink model.
- Measure the physical bicycle parameters for use during the implementation of a model.
- Implement a model of the real bicycle in Simulink.
- Create and simulate different controllers for the Simulink model.
- Implement a speed controller on the real bicycle.
- Implement a self balancing controller on the physical bicycle.

A secondary and at least equally important goal is to deliver a complete system ready for the further development towards a fully Autonomous Bicycle.

1.2.1 Leaving Out The Pendulum

The inverted pendulum currently attached to the bicycle has a mechanical defect in the form of a rather large backlash of 2 degrees. The control of the inverted pendulum would with this large backlash either be very relaxed or cause the pendulum to oscillate. Both results would be unable of balancing the bicycle as a relaxed controller would be to slow, while the oscillating controller would make the bicycle even more unstable. The reason for not changing the gearbox during the previous project were due to budget limitations[2]. The inverted pendulum was therefore left out of the problem in this thesis. This to prove that it is possible to create a self balance bicycle and hopefully motivate the Department of Engineering Cybernetics to fund the addition of a new gearbox for the inverted pendulum.

1.3 Previous Work

A lot of work has been done on the bicycle over the last few years, the project was first initiated back in the 80's, but this thesis is focusing on the work done since Loftum[1] and Bjermeland[2] started up the work again

[2]The cost of a new gearbox for the inverted pendulum with the needed specifications would cost the entire budget.

in 2006. Bjermeland[2] focused on working with the bicycle dynamics and mathematical modelling, while Loftum[1] developed the instrumentation and computer system. Since 2006 the computer system and instrumentation has been further developed and bug fixed by Fossum[3], Sølvberg[4], Brekke[5] and Hatlvoll[6]. After this the author of this thesis has performed a complete rebuild of both the hardware and software framework during a former project[7].

1.4 Outline

Chapter 2 gives a description of the background theory relevant for the completion of this thesis.
Chapter 3 gives a brief description of the hardware and software framework available at the start of this thesis.
Chapter 4 describes the work performed on the hardware and software framework.
Chapter 5 describes the steering planetary gearbox breakdown.
Chapter 6 describes the methods and the work performed for measuring the parameters needed to create a model of the physical bicycle.
Chapter 7 describes the development of a model with the parameters measured in Chapter 6 together with the development and simulation of different controllers for self balance.
Chapter 8 describes the implementation and result of the self balancing controller on the physical bicycle.
Chapter 9 gives a short description on the further work to be performed on bicycle.
Chapter 10 is a discussion about the result and choices made during this thesis.
Chapter 11 is the conclusion of this thesis.

1.5 Project Delivery

In addition to this report as part of the delivery is the complete hardware and software framework delivered together with this thesis. The software is available on the attached CD, together with videos documenting that the Autonomous Bicycle is capable of performing a self balanced ride.

Chapter 2

Background Theory

2.1 The Whipple Bicycle Model

To develop a control system it is helpful to have a good mathematical model of the system. When it comes to bicycles there are different models available, but most used is the Whipple[8] bicycle model. The linear equations derived by Papadopoulos[9] are the most tested equations available today.

- In 1987 Papadopoulos[9] derived the linear equations used in this report.

- In 2004 Meijaard[10] derived the linear equations using another method, with the same result as in [9].

- In 2006 experiments were performed by Kooijman[11] on a real bicycle to validate the linearised equations derived by Papadopoulos[9]. Kooijman[11] refers to a benchmark performed by Papadopoulos, Meijaard and Schwab[12] in 2005, but these equations are the same as in [9].

- In 2007 Meijaard, Papadopoulos, Ruina and Schwab[13] published a benchmark and review of the linear dynamic equations. This publication conclude that the linear equations derived by Papadopoulos[9] are "reliable equations for a well-delineated model for more deeply studying controlled and uncontrolled stability of a bicycle"[13]. They found that both SPACAR and AUTOSIM produced linear equations with coefficient matrices that correspond with the pen-and-paper calculations to the 14. digit.

- There are also numerous other publications available, a brief review is included in the paper published by Meijaard, Papadopoulos, Ruina and Schwab[13].

Chapter 2.1.1 shortly describes the model developed by Whipple[8], explaining the linear equations derived by Papadopoulos[9] and the parameters needed to complete the model. Readers interested in more information about the model are first referred to the main source of this chapter [13] then to Whipple[8], Papadopoulos[9] and Meijaard[10].

2.1.1 Bicycle Model Without Rider

The mechanical model of the bicycle without a rider(or with a static rider) consist of four parts, the rear body frame(which would includes the static rider), front handlebar frame, rear wheel and front wheel. These four parts are connected through rotating joints at the wheel hubs and trough the steering axis as shown in Figure 2.1. The model assumes the following constraints:

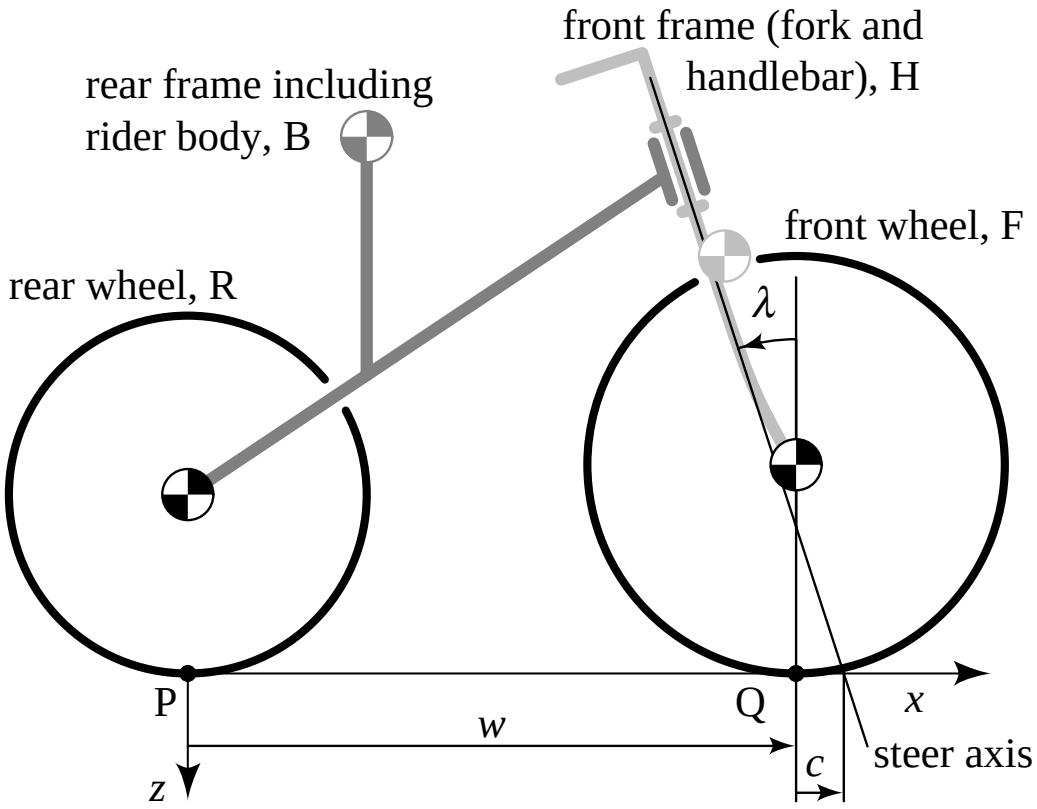


Figure 2.1: The Whipple Bicycle model(figure taken from [13]).

- The model assumes that both wheels have contact with ground, and that the ground is completely horizontal.
- The model assumes symmetry about the xz plane when the bicycle is standing straight up with the steering in neutral position, and circular symmetry of the wheels.
- The model assumes knife-edge wheels.
- The model assumes that if a rider is included it has to be a static rider, in other words assumes no motion of a rider with respect to the rear body frame.
- The model assumes a 100% stiff frame.
- The model assumes no form of damping.
- The model assumes no friction in the joints.
- The model assumes a tyre model with no slip or skidding.

The four objects would without constraints and connections have 24 degrees of freedom, each part would have 3 translational and 3 rotational degrees of freedom. With the constraints and joints given above, 5 degrees of freedom per joint and 1 degree per wheel contact point are removed. The slip and skid constraints does not remove any degrees of freedom as we still are able to move sideways by turning the wheel when moving forward[1]. The bicycle can then be described in the following way(see Figure 2.2 for visualisation):

- The position of the rear wheel contact point $(x_P, 0, y_P)$ relative to the global fixed coordinate system.
- A yaw rotation ψ around the z-axis in this contact point.
- A roll rotation ϕ around the x-axis of the coordinate system fixed in $(x_P, 0, y_P)$ and rotated by ψ.
- A rotation θ_B that is there only to describe the small rotation of the rear body frame during the combined steer and roll to keep both wheels connected to ground at all times.
- A rotation δ between the rear body and front handlebar frame.
- Rotations ϕ_R and ϕ_F describes the rotation of the wheels relative to the rear body and front handlebar frame.

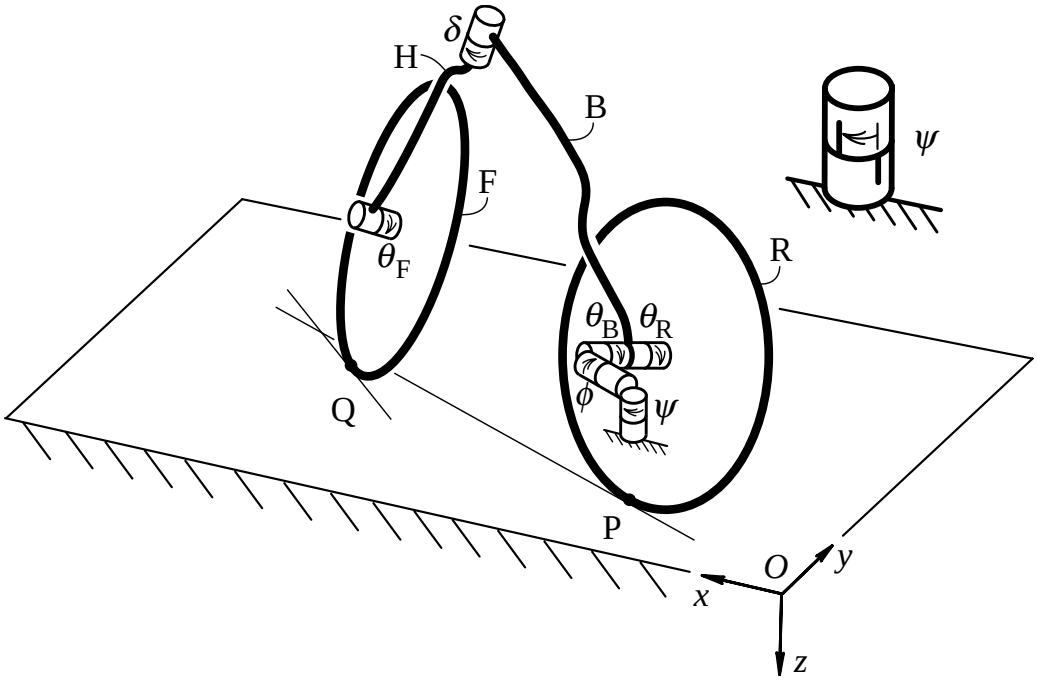


Figure 2.2: Bicycle model showing the rotations used in the linearised equations(figure taken from [13]).

This is reduced even more when we remove the four non-holonomic rolling constraints related to the wheel-to-ground contact points(longitude and lateral for each contact.). The bicycle can then be described by the three following parameters(as seen in Figure 2.2.).:

- The lean rate of the rear body frame ϕ.
- The steering rate δ.
- The rotation rate of the rear wheel θ_R relative to the rear body frame.

Right turn and roll are considered as the positive direction. As well a forward motion is considered the positive direction for θ_R. The above part of this section gave a short description of the mechanical model, while the following will present the linearised equations of motion and the coefficients used by the equations. The first linear equation(Equation (2.1)) is linearised with the assumption of straight forward and upright motion($\delta = \phi = 0$). The equation is valid at any constant speed. This removes any first-order coupling between

[1] And of course backwards, even if this is not relevant at this time

forward motion and the steer and lean. In other words the rotational speed of the bicycle model is not affected by the steering or lean.

$$\left[r_R^2 m_T + I_{Ryy} + \left(\frac{r_R}{r_F}\right)^2 I_{Fyy}\right] \ddot{\theta_R} = T_{\theta_R} \tag{2.1}$$

Second we have the linearised equations of motion for the steer and lean angle(δ and ϕ), which are a set of two coupled second-order, ordinary differential equations with constant coefficients. These two equations are combined to an equivalent set (2.2) with only the external force(T_ϕ and T_δ) on the right-hand side.

$$\mathbf{M}\ddot{\mathbf{q}} + \upsilon\mathbf{C}_1\dot{\mathbf{q}} + [g\mathbf{K}_0 + \upsilon^2\mathbf{K}_2]\mathbf{q} = \mathbf{f} \tag{2.2}$$

The coefficients of the second linearised equations are:

- $\boldsymbol{q} = [\phi, \delta]^T$ which is the time-varying steer and roll angles.
- $\boldsymbol{f} = [T_\phi, T_\delta]^T$ which is the time-varying general steer and lean forces.
- $\boldsymbol{M}$ is the mass moment of inertia matrix, brings the kinetic energy into the model.
- $\boldsymbol{C}_1$ is the damping like matrix that captures the skew-symmetric gyroscopic torques from turning the steer wheel and changing the roll angle. It also captures the inertial reactions due to steer rate.
- $\boldsymbol{K}_0$ is the velocity independent stiffness matrix, brings the potential energy into the model.
- $\boldsymbol{K}_2$ is the velocity dependant stiffness matrix, brings the gyroscopic and centrifugal forces to the model.

To describe the different coefficients of the linearised equations below, we need the 25 parameters described in Table 2.1. The measurement and estimation of these parameters are described in Chapter 6. The equations and notations are the same as found in [13], with subscript R corresponding to the rear wheel, F the front wheel, B the rear body frame and H the front handlebar frame. In addition to this T is used for the total system and A for the front assembly(front handlebar frame and front wheel.). All calculations consider the bicycle in the upright straight forward position(neutral position).
The definition of the total mass of the system along with the center of this mass.

$$m_T = m_R + m_B + m_H + m_F \tag{2.3}$$

Parameter	Symbol
Wheel base	ω
Trail	c
Steer axis tilt	λ
Gravity	g
Forward speed	v
Rear wheel	R
Radius	r_R
Mass	m_R
Moments of inertia	$(I_{Rxx}, I_{Ryy}, I_{Rzz})$
Rear body frame	B
Center of mass	(x_B, z_B)
Mass	m_B
Moments of inertia	$\begin{bmatrix} I_{Bxx} & 0 & I_{Bxz} \\ 0 & I_{Byy} & 0 \\ I_{Bxz} & 0 & I_{Bzz} \end{bmatrix}$
Front handlebar frame	H
Center of mass	(x_H, x_H)
Mass	m_H
Moments of inertia	$\begin{bmatrix} I_{Hxx} & 0 & I_{Hxz} \\ 0 & I_{Hyy} & 0 \\ I_{Hxz} & 0 & I_{Hzz} \end{bmatrix}$
Front wheel	F
Radius	r_F
Mass	m_F
Moments of inertia	$(I_{Fxx}, I_{Fyy}, I_{Fzz})$

Table 2.1: Description of the 25 bicycle parameters needed to calculate the coefficients for the linearised equations(Parameters g and v are used in the model but not depending on the bicycle, I_{Byy} and I_{Hyy} are not used by the model. These four parameters comes in addition to the count of 25.).

$$x_T = \frac{x_B m_B + x_H m_H + \omega m_F}{m_T} \tag{2.4}$$

$$z_T = \frac{-r_R m_R + z_B m_B + z_H m_H - r_F m_F}{m_T} \tag{2.5}$$

The definition of the total mass moment and product of inertia with respect to the rear wheel contact point:

$$I_{Txx} = I_{Rxx} + I_{Bxx} + I_{Hxx} + I_{Fxx} + m_R r_R^2 + m_B z_B^2 + m_H z_H^2 + m_F r_F^2 \tag{2.6}$$

$$I_{Txz} = I_{Bxz} + I_{Hxz} + m_B x_B z_B + m_H x_H z_H \tag{2.7}$$

The moment of inertia for the wheels are the same in x and z due to symmetry about the rotational axis giving:

$$I_{Rzz} = I_{Rxx}, \quad I_{Fzz} = I_{Fxx} \tag{2.8}$$

Continue to define the total mass moment and product of inertia, around the rear wheel contact point z-axis:

$$I_{Tzz} = I_{Rzz} + I_{Bzz} + I_{Hzz} + I_{Fzz} + m_B x_B^2 + m_H x_H^2 + m_F \omega^2 \tag{2.9}$$

In a similar way the total mass and center of mass for the front assembly can be defined, relative to the rear wheel contact point:

$$m_A = m_H + m_F \tag{2.10}$$

$$x_A = \frac{x_H m_H + \omega m_F}{m_A}, \quad z_A = \frac{z_H m_H - r_F m_F}{m_A} \tag{2.11}$$

The definition of the mass moment and product of inertia about the front assembly center of mass:

$$I_{Axx} = I_{Hxx} + I_{Fxx} + m_H (z_H - z_A)^2 + m_F (r_F + z_A)^2 \tag{2.12}$$

$$I_{Axz} = I_{Hxz} - m_H (x_H - x_A)(z_H - z_A) + m_F (\omega - x_A)(r_F + z_A) \tag{2.13}$$

$$I_{Azz} = I_{Hzz} + I_{Fzz} + m_H (x_H - x_A)^2 + m_F (\omega - x_A)^2 \tag{2.14}$$

The position of the center of mass of the front assembly is located a small distance in front of the steering axis and can be calculated to be:

$$u_A = (x_A - \omega - c)\cos\lambda - z_A \sin\lambda \tag{2.15}$$

For the front assembly the moment of inertia about the steering axis and the product of inertia about the global z and x axes due to acceleration about the steer axis.

$$I_{A\lambda\lambda} = m_A u_A^2 + I_{Axx} \sin^2\lambda + 2 I_{Axz} \sin\lambda \cos\lambda + I_{Azz} \cos^2\lambda \tag{2.16}$$

$$I_{A\lambda x} = -m_A u_A z_A + I_{Axx} \sin\lambda + I_{Axz} \cos\lambda \tag{2.17}$$

$$I_{A\lambda z} = m_A u_A x_A + I_{Axz} \sin\lambda + I_{Azz} \cos\lambda \tag{2.18}$$

The mechanical trail ratio:

$$\mu = \frac{c}{\omega} \cos\lambda \tag{2.19}$$

The gyroscopic coefficients for the rear and front wheel:

$$S_R = \frac{I_{Ryy}}{r_R}, \quad S_F = \frac{I_{Fyy}}{r_F}, \quad S_T = S_R + S_F \tag{2.20}$$

A frequently appearing static moment term(In conjunction with the forces due to change in position of the center of mass for the front handlebar frame during lean or steer.):

$$S_A = m_A u_A + \mu m_T x_T \tag{2.21}$$

With the above definitions and variables the matrices of the linearised equations are formed. First the mass matrix containing the mass moments of inertia:

$$M_{\phi\phi} = I_{Txx}, \qquad M_{\phi\delta} = I_{A\lambda x} + \mu I_{Txz} \tag{2.22a}$$

$$M_{\delta\phi} = M_{\phi\delta}, \qquad M_{\delta\delta} = I_{A\lambda\lambda} + 2\mu I_{A\lambda z} + \mu^2 I_{Tzz} \tag{2.22b}$$

$$\mathbf{M} = \begin{bmatrix} M_{\phi\phi} & M_{\phi\delta} \\ M_{\delta\phi} & M_{\delta\delta} \end{bmatrix} \tag{2.23}$$

The gravity dependent stiffness matrix:

$$K_{0\phi\phi} = m_T z_T, \qquad K_{0\phi\delta} = -S_A \tag{2.24a}$$

$$K_{0\delta\phi} = K_{0\phi\delta}, \qquad K_{0\delta\delta} = -S_A \sin\lambda \tag{2.24b}$$

$$\mathbf{K}_0 = \begin{bmatrix} K_{0\phi\phi} & K_{0\phi\delta} \\ K_{0\delta\phi} & K_{0\delta\delta} \end{bmatrix} \tag{2.25}$$

The velocity dependent stiffness matrix:

$$K_{2\phi\phi} = 0, \qquad K_{2\phi\delta} = \frac{S_T - m_T z_T}{\omega} \cos\lambda \tag{2.26a}$$

$$K_{2\delta\phi} = 0, \qquad K_{2\delta\delta} = \frac{S_A + S_F \sin\lambda}{\omega} \cos\lambda \tag{2.26b}$$

$$\mathbf{K}_2 = \begin{bmatrix} K_{2\phi\phi} & K_{2\phi\delta} \\ K_{2\delta\phi} & K_{2\delta\delta} \end{bmatrix} \tag{2.27}$$

And finally the 'damping'[2] matrix:

$$C_{1\phi\phi} = 0, \qquad C_{1\phi\delta} = \mu S_T + S_F \cos\lambda + \frac{I_{Txz}}{\omega}\cos\lambda - \mu m_T z_T \tag{2.28a}$$

$$C_{1\delta\phi} = -(\mu S_T + S_F \cos\lambda), \qquad C_{1\delta\delta} = \frac{I_{A\lambda z}}{\omega}\cos\lambda + \mu\left(S_A + \frac{I_{Tzz}}{\omega}\cos\lambda\right) \tag{2.28b}$$

$$\mathbf{C}_1 = \begin{bmatrix} C_{1\phi\phi} & C_{1\phi\delta} \\ C_{1\delta\phi} & C_{1\delta\delta} \end{bmatrix} \tag{2.29}$$

2.2 Control Theory

Some of the theory described here could be considered basic knowledge for control engineers. But it is include here in order to make this report a complete starting point for people not familiar with the theory used. The theory in this chapter is mostly taken from [14], [15] and [16]. In the event of differences, the notations found in [16] are used.

2.2.1 State-Space Representation

State-Space representation is a way to describe a physical system with a set of input, output and state variables as a mathematical model. Linear time-invariant systems can be written on the form shown in Equation (2.30), an illustration can be found in Figure 2.3:

$$\dot{x} = Ax + Bu \tag{2.30a}$$

$$y = Cx + Du \tag{2.30b}$$

2.2.2 Proportional(P) To Proportional-Integral-Derivative(PID) Control

One of the most basic controller available, easily described in Figure 2.4. The sketch describes the complete PID controller, but any controller from the P to the PID can be described by the same sketch. To get the desired controller, the unwanted part(s) could easily be removed. For exampled removing the I part would result in a PD controller. It should be noted that during an

[2]There is no real damping in the model, but there is the damping like skew symmetric gyroscopic torque from steer and lean rates

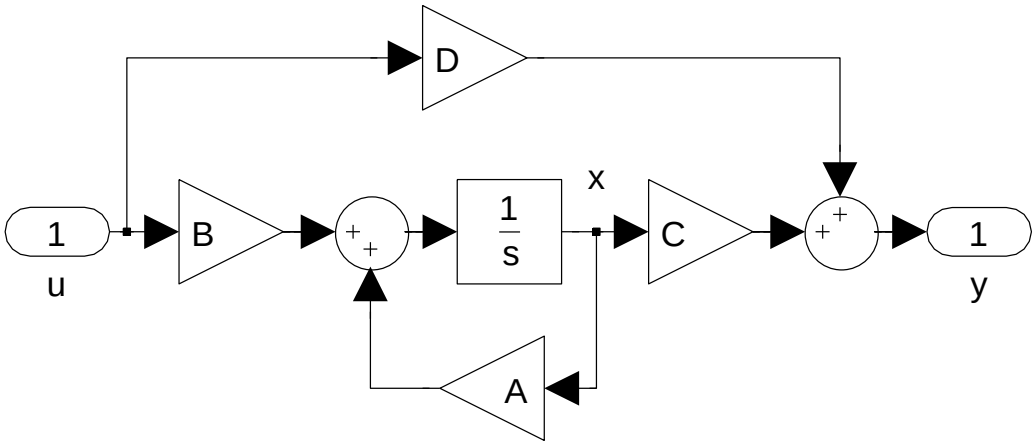


Figure 2.3: State-Space representation

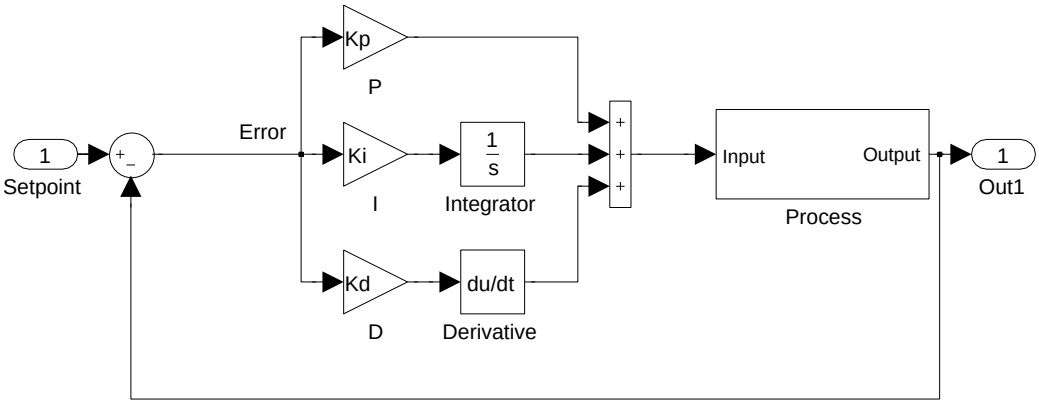


Figure 2.4: PID Controller

implementation, some topics needs to be addressed. First to be noted is the integral windup, occurring when a large change in set-point generates a integral term larger than the maximum control signal to the process. The system then overshoots and increases until the integral term is unwounded, causing oscillations in the process. This should be addressed by implementing some form for anti-windup, limiting the maximum integral value. Second to be noted is a problem with the derivative term, small amounts of measurement noise could cause rather large changes to the output. Some method for removing noise should be considered during controller design, for example low-pass filtering.

2.2.3 Linear-Quadratic Regulator(LQR) Design

The LQR regulator is a solution to the Linear Quadratic(LQ) problem, this is a problem described by a set of linear differential equations and a quadratic cost function. The LQR algorithm is basically an algorithm that finds the coefficient matrix(K) for a state-feedback controller(see Figure 2.5), based on the supplied cost function weighting factors(Q, R and N). For a continuous

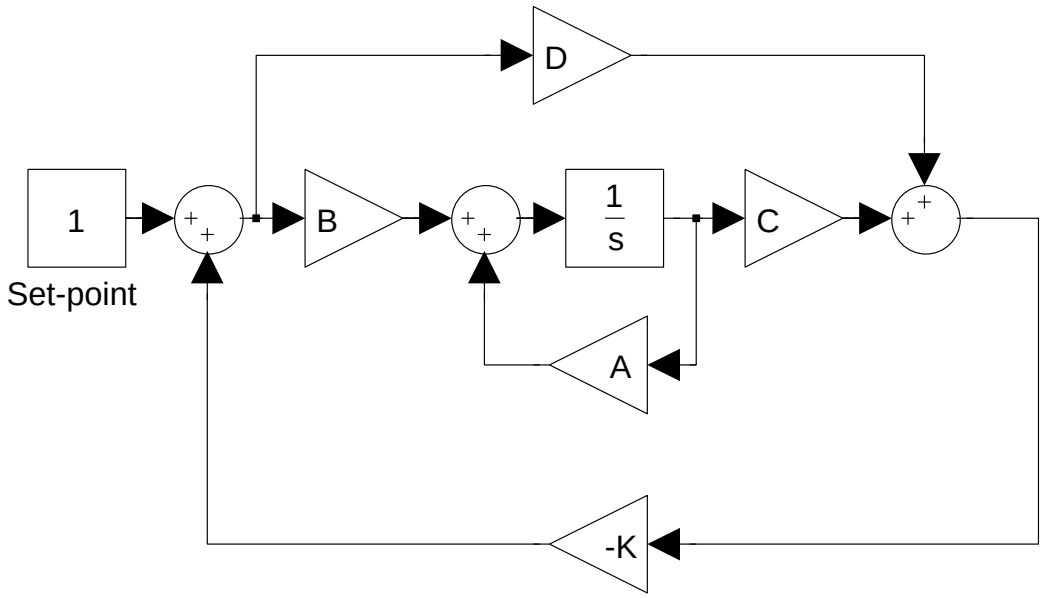


Figure 2.5: State-feedback control of a state-space model

time system described by:

$$\dot{x} = Ax + Bu \tag{2.31}$$

A optimal gain matrix K that minimises the quadratic cost function,

$$J(u) = \int_0^\infty (x^T Q x + u^T R u + 2x^T N u)\, dt \text{ where, } u = -Kx \tag{2.32}$$

is calculated by the following equation:

$$K = R^{-1}(B^T S + N^T) \tag{2.33}$$

Where S is found by solving the continuous time Riccati equation:

$$A^T S + SA - (SB + N)R^{-1}(B^T S + N^T) + Q = 0 \tag{2.34}$$

The optimal gain matrix K can be calculated by the following MATLAB function:

```
[K,S,e] = LQR(A,B,Q,R,N)
```

The LQR regulator has the advantage of being able to handle multiple inputs, where the algorithm finds the optimal state-feedback matrix(K) that minimizes the undesired behaviour specified in the weighting matrices(Q,R and N). A disadvantage of the regulator is that the controller demands full state feedback, which in most cases are unavailable. An observer is therefore needed to estimate the unmeasured states.

2.2.4 Kalman Estimator

The Kalman Bucy filter is a mathematical method that can be used to estimate the unmeasured states and produce estimates that are closer to the true value than the measurement. The Kalman Bucy filter provides the optimal solution to the continuous estimation problem:

$$\dot{x} = Ax + Bu + Gw \tag{2.35a}$$
$$y = Cx + Du + Hw + v \tag{2.35b}$$

The white process and measurement noise satisfies:

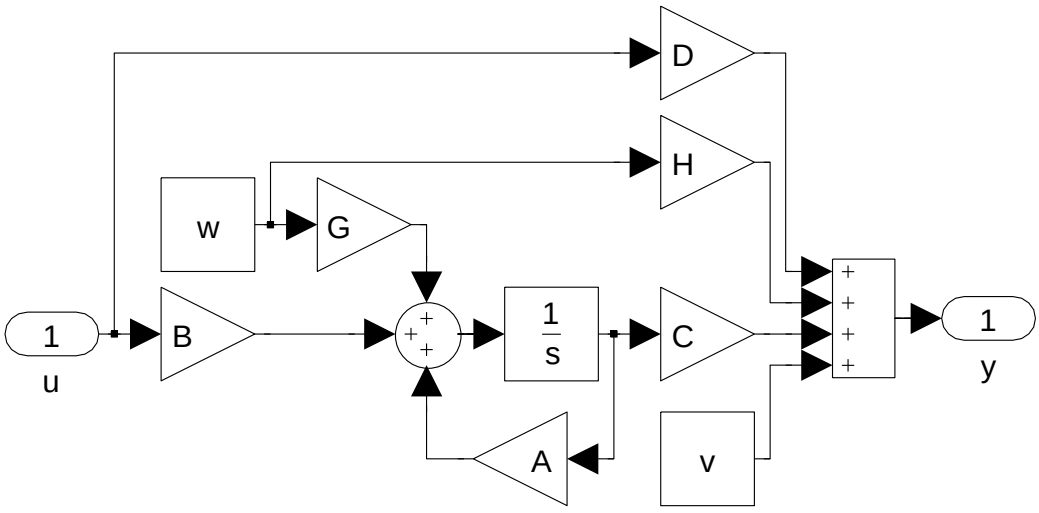


Figure 2.6: Sketch of the Continuous State-Space model used in the estimation problem

$$E(w) = E(v) = 0, \quad Q = E(ww^T), \quad R = E(vv^T), \quad N = E(wv^T) \tag{2.36}$$

Where Q, R and N are the covariances of the noise. The optimal Kalman Bucy filter for estimating the process states are described by the following equation:

$$\dot{\hat{x}} = A\hat{x} + Bu + L(y - C\hat{x} - Du) \tag{2.37}$$

The Kalman Bucy filter gain is calculated by solving the following Riccati

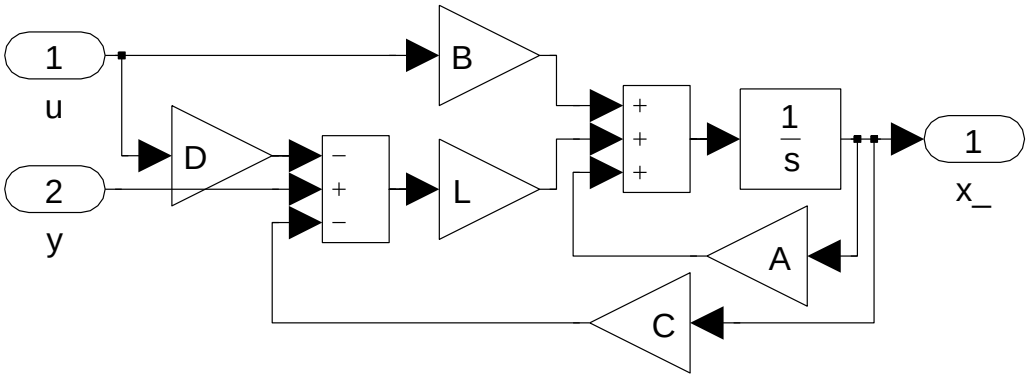


Figure 2.7: Sketch of the Kalman Bucy Filter

equation:

$$L = \left(PC^T + \bar{N}\right)\bar{R}^{-1} \tag{2.38}$$

Where:

$$\bar{R} = R + HN + N^T H^T + HQH^T \tag{2.39}$$

$$\bar{N} = G\left(QH^T + N\right) \tag{2.40}$$

$$P = \lim_{t\to\infty} E\left(\{x - \hat{x}\}\{x - \hat{x}\}^T\right) \tag{2.41}$$

Chapter 3

Description Of The Existing Bicycle

This chapter describes the Autonomous Bicycle at the start of this thesis. The chapter describes the state of both physical hardware and software. This chapter is mainly based on an earlier project report[7] written by the author of this thesis. Readers interested in a more detailed description and specifications are referred to [7].

Figure 3.1: The bicycle frame

3.1 Hardware

The bicycle consists of a modern bicycle frame, as seen in Figure 3.1. The frame is equipped with the following additional hardware:

- A propulsion motor capable of accelerating the bicycle at a rate of about $2 - 3m/s^2$, with a tested top speed of more than $6.4m/s$ and a theoretical top speed of about $8m/s$ without implementing gear shifting.
- A steering motor capable of turning the steer with a rate of $317°/s$ with an acceleration above $20000°/s^2$.
- A motorised inverted pendulum, simulating a leaning rider. The possible rotational speed of the pendulum is $244°/s$ with acceleration similar to the steering motor. The gear of this motor however has a backlash of about 2°, which is a bit too much to be able to use the pendulum for stability control.
- Precision linear potentiometers for measuring steer and pendulum angle.
- Tachometer generators for measuring motor speeds.
- XSens MTi IMU for measuring relative movement.
- A computer system case containing:
  - One or two $24V$ $3300mAh$ lithium-ion polymer batteries.
  - A power distribution Board, with power buttons and emergency stop inputs.
  - Power buttons and emergency stop button.
  - A power converter for the 5 and 12 volt equipment.
  - Three $25A$ motor drivers.
  - Wireless router for host-target communication.
  - Embedded target computer.
  - Analog I/O-card, for measuring sensor input.
  - I/O connection card.

The physical state of the bicycle is at a satisfying level with the exception of the backlash in the pendulum gearbox and a noisy tachometer generator used for measuring propulsion speed.

3.2 Software

The software is built up as a host target environment, where the bicycles on-board computer is the target. Any computer running windows, equipped with a wireless network can be set up as a host system. The host platform needs to be set up with the following software:

- QNX Software development platform(6.5.0).
- MATLAB with Simulink and RTW.
- RTW additions for QNX support[1].
- Universal library[2] from Diamond systems added to the QNX directory.
- Additionally a terminal interface as Putty may come handy.

With a host system up and running a Simulink model is available from [7], see Figure 3.2. The model is set-up to generate C/C++ code with RTW, compile the generated code for a QNX target and then upload the executable to the target computer. It is possible to run the target system in both standalone mode and in host-target mode with on the fly parameter tuning and value reading from the host computer.

[1]Supplied on the CD attached to [7].
[2]Version 6.02a, also found on CD attached to [7].

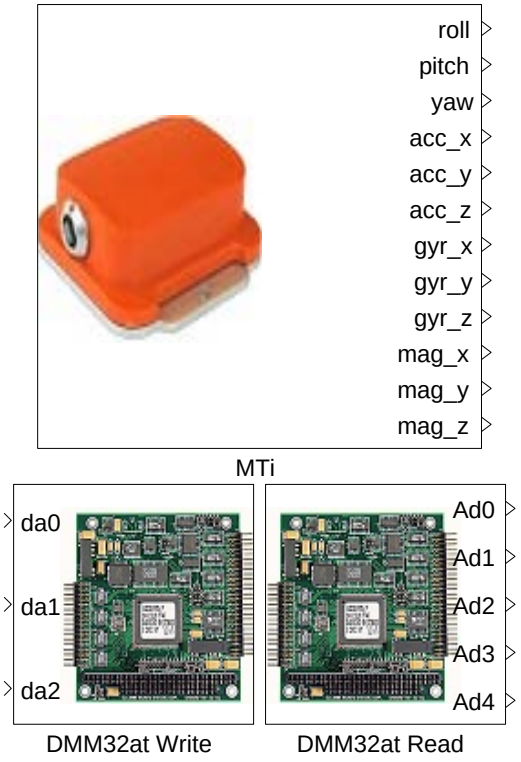


Figure 3.2: Simulink model containing three s-functions for input and output

Chapter 4

Modifications And Additions To The Framework

This chapter describes the modifications performed on the bicycle and the additions made to the Simulink model prior to the implementation of a control system.

4.1 Incremental Encoder

The propulsion tachometer generator was removed and an incremental encoder was attached for measuring the forward speed, see Figure 4.1. The transmission between the encoder and the motor was also changed from the rubber band used on the tachometer generator to a spur gear 1:1 transmission. The specifications of the encoder can be found in Table 4.1.

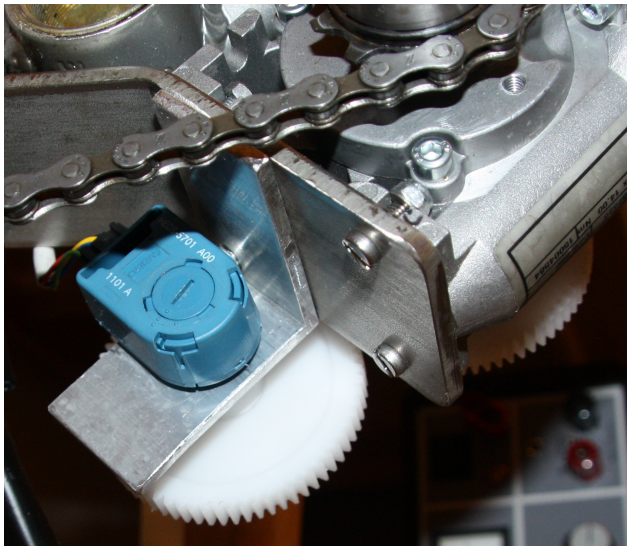


Figure 4.1: Incremental encoder mounted to the propulsion motor

Producer	Avago Technologies
Type	Optical Encoder
Number of channels	2
Output	TTL Quadrature
Max. rotational speed	2000rpm
Supply voltage	4.5V to 5.5V
Number of pulses	500 ppr
Rotational life cycles	12000000
Operating temperature	$-20°C$ to $+85°C$
Supply current	17mA to 40mA
High level output voltage	Min. 2.4V
High level output current	-40μA Max.
Low level output voltage	Max. 0.4V
Low level output current	3.2mA

Table 4.1: Specifications for the incremental encoder

To be able to read the quadrature signal from the encoder, a printed circuit board(PCB) was designed featuring an Atmel xmega, see Appendix B. The board is equipped with a $3.3V$ linear regulator and level transceivers. The board reads the quadrature signal and outputs an analog voltage signal to be read by the analog I/O card. Figure 4.2 gives a short visualisation of how an incremental encoder works and shows the quadrature signal generated.

The software written for the xmega can be found on the CD attached to this thesis.

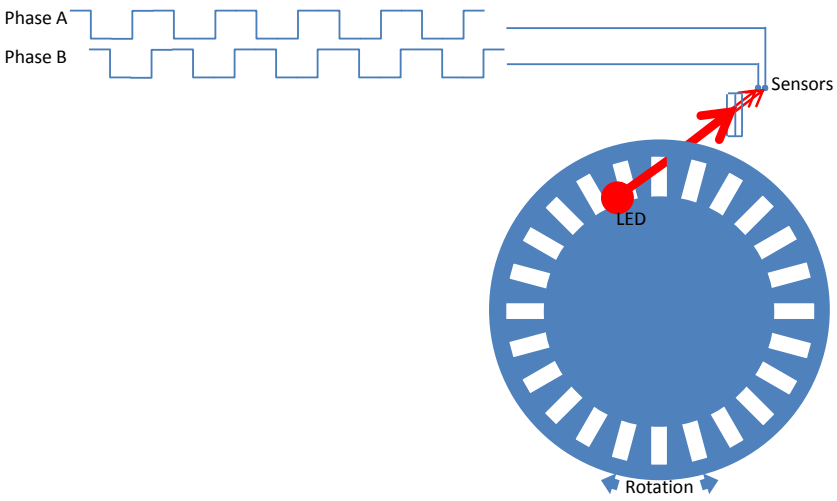


Figure 4.2: Brief visual description of an incremental encoder and the produced quadrature signal

4.2 Simulink Model

This section describes the changes done to the Simulink model before implementing a controller. This includes tasks as scaling, calibration, edge limiting and signal orientation changes.

4.2.1 Scaling And Calibration Of Analog Input And Output

Scaling and calibration of the analog input and outputs were implemented into the Simulink model. Figure 4.3 shows the simulink diagram for scaling the motor output signals, it also includes saturation to ensure that the value written is in the 0 to 4095 integer range. Figure 4.4 shows the Simulink diagram for scaling and calibration of the sensor readings. The gain is set to scale the read values into degrees[1] for the steer and pendulum. The steer and pendulum values were calibrated with the neutral position corresponding to a angle of 0. While the propulsion reading were scaled to correspond to the forward speed in m/s.

[1]During the implementation of a controller or observer based on the linearised model, the values needs to be converted to radians.

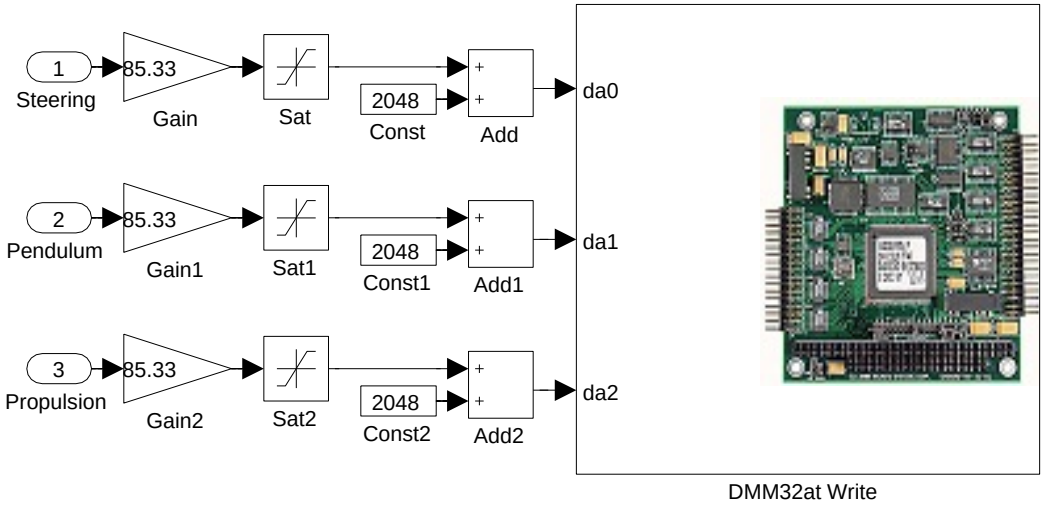


Figure 4.3: Analog output scaling

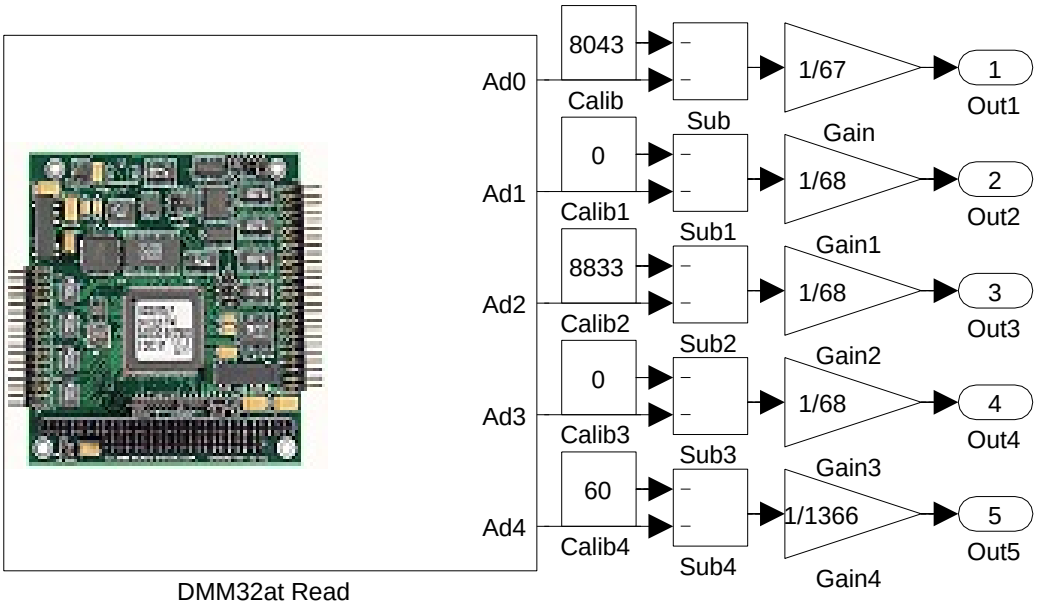


Figure 4.4: Analog Input scaling and calibration

4.2.2 Edge Limiting The Steering And Pendulum Motor

The steer and pendulum movement are limited by the geometry of the bicycle. To protect the steer and pendulum from hitting the bicycle due to faults in the controller design, a progressive edge limiter were implemented, as shown in Figure 4.5. The progressive edge limiter gradually limits the steer or pendulum to move outside the -60 to 60 degree region. This is performed from ±60 to ±90 degrees by gradually giving more control to the edge limiter. The functionality is described by the following pseudo code, a plot of the output compared to the input and the measured angle can be seen in Figure 4.6.

---

if $MeasAngle > 60$ **then**
  $Ratio = (MeasAngle - 60)/30$
  if $Ratio > 1$ **then**
    $Ratio = 1$
  end if
else if $MeasAngle < -60$ **then**
  $Ratio = (MeasAngle + 60)/30$
  if $Ratio < -1$ **then**
    $Ratio = -1$
  end if
else
  $Ratio = MeasAngle$
end if
if $Torque > 24$ **then**
  $TorqueSat = 24$
else if $Torque < -24$ **then**
  $TorqueSat = -24$
else
  $TorqueSat = Torque$
end if
$MotorTrust = TorqueSat * (1 - abs(Ratio)) - 24 * Ratio$

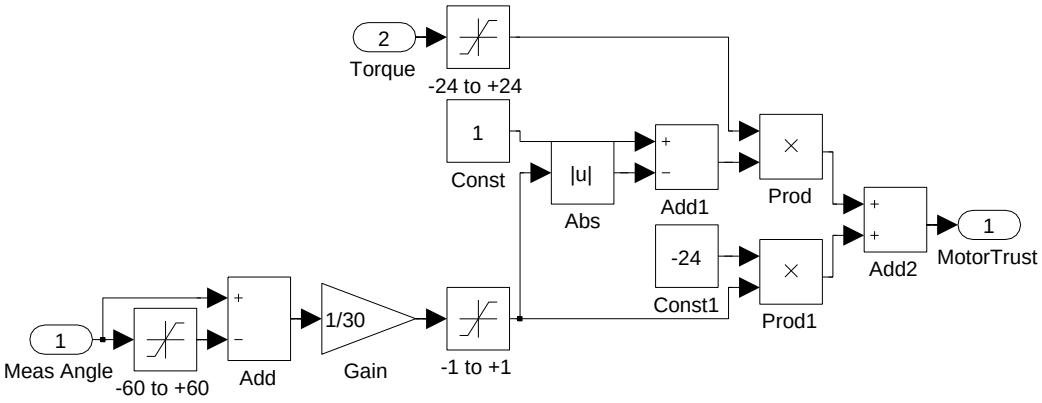


Figure 4.5: Simulink diagram of the progressive edge limiter

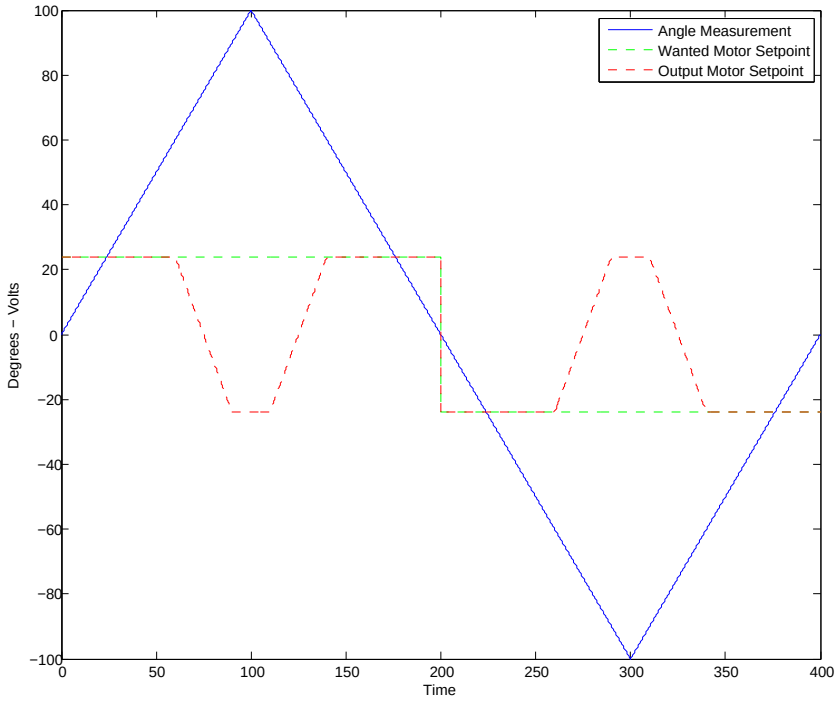


Figure 4.6: Plot showing the output of the edge limiter versus the input and measured angle. It is shown by the plot that in the ±60 to ±90 region, the output becomes gradually independent of the input value as it gets more dependent of the measured angle.

4.2.3 Orientation Changes To The IMU Roll Signal

The IMU roll angle outputs a signal from -180 to 180 degrees, during a 360 degree rotation the signal wraps around from 180 to -180. This wraparound would normally not occur on the roll angle, but due to the mounting direction the wraparound happens when the bicycle changes from a left lean to a right lean and the other way around. To encounter this the logic shown in pseudo code below for flipping the signal was implemented in Simulink, see Figure 4.7.

---

if $ReadAngle > 0$ **then**
  $Angle = ReadAngle - 180$
else
  $Angle = ReadAngle + 180$
end if

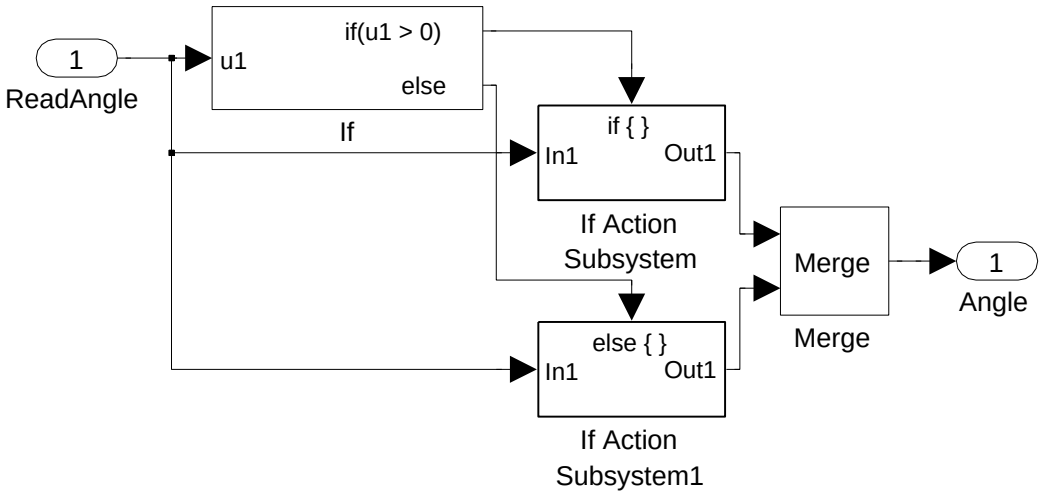


Figure 4.7: Simulink diagram showing the change of roll orientation logic.

4.2.4 Free Rotation Of The IMU Yaw Signal

The IMU yaw angle outputs a signal from -180 to 180 degrees, during a 360 degree rotation the signal wraps around from 180 to -180. To use this measurement we need to remove the wraparound, since this could cause serious problems in a control loop. The simplest way to remove this is to add 360 degrees to the signal for every wraparound from +180 to -180 and subtract 360 degrees for every wraparound from -180 to +180. The implementation

does not take care of variable overflow since this is not a problem in the foreseen future, since the offset is reset to zero every program start. It is possible to create a free rotation overflow safe function, but this is more complex as it is needed to be handled simultaneously in a controller[2]. Pseudo code of the free rotation logic can be found below, see Figure 4.8 for the Simulink diagram. A plot showing the response of the logic can be seen in Figure 4.9.

---

if $(LastReadAngle - ReadAngle) > 180$ **then**
  $Offset = Offset + 360$
else if $(LastReadAngle - ReadAngle) < -180$ **then**
  $Offset = Offset - 360$
end if
$Angle = Offset + ReadAngle$

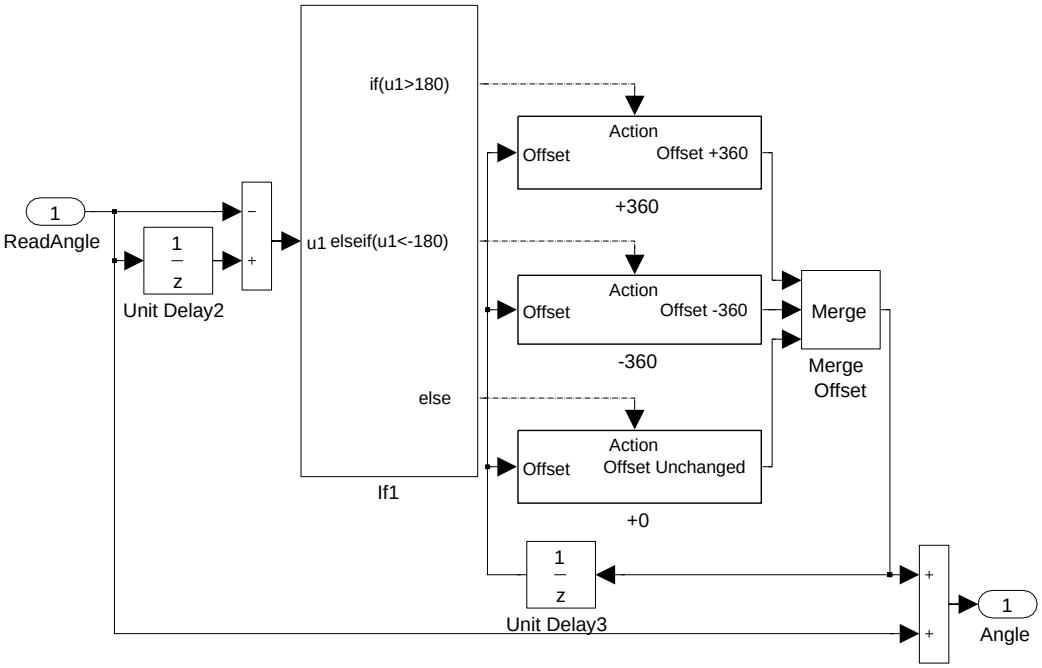


Figure 4.8: Simulink diagram showing the free rotation logic.

[2]Can be performed by resetting the added offset simultaneously in both the control loop and the read value

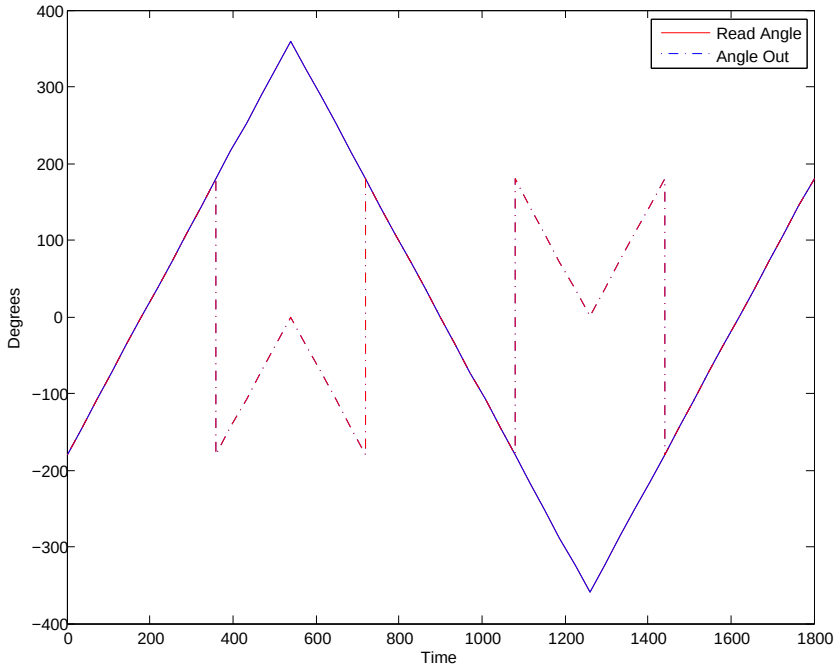


Figure 4.9: Plot showing the result of the free rotation logic. It can be seen that the angle output continues to rotate during a wraparound in both directions.

4.2.5 Total Simulink Model

The various additions to the Simulink described in this chapter were combined and masked in two different subsystems. One for the MTi containing the MTi s-function block, roll orient change and the free 360 degree rotation of the yaw signal. And a Second subsystem containing the rest. The Simulink framework creates can be seen in Figure 4.10.

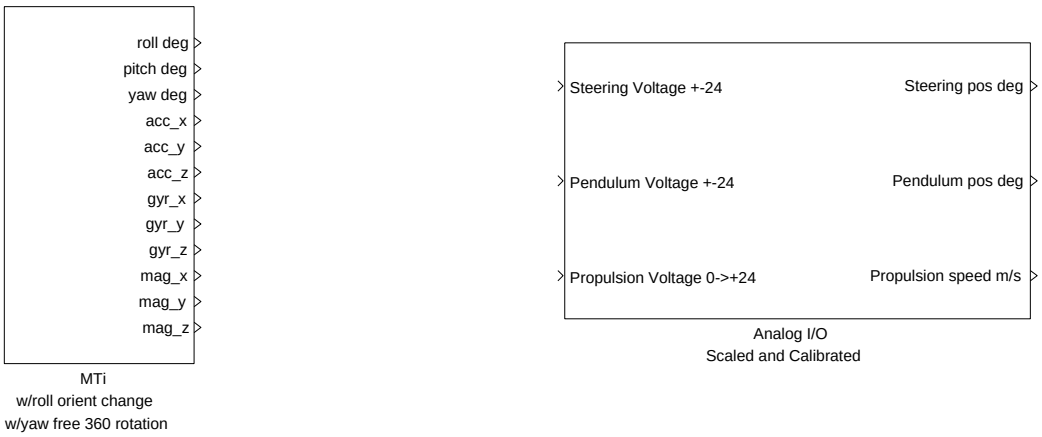


Figure 4.10: The Simulink framework, ready for controller implementation

Chapter 5

The Catastrophe

After one of the first successful self balanced bicycle rides, a major catastrophe occurred. The steer planetary gearbox broke down during a test ride, 10 weeks before the final delivery of the thesis. The task of delivering a self balancing bicycle seemed almost impossible. There could be several reasons for the breakdown, since the total history of the gearbox is unknown. The most likely solution is that the total of the opposite directed forces from the motor and steer were higher than the gearbox could handle. The simplest solution of replacing the gearbox could end up being just a waist of time and money. Because of this a redesign of the total steer gearbox and transmission combination were performed. The gearbox were dismantled to investigate what part of the gearbox that broke down. The outer upper part of the third stage of the planetary gearbox, shown in Figure 5.1, had broken loose from the the output shaft of the gearbox, identified as the small shaft in the center on the image. Due to lack of time and the delivery time of suitable planetary gears, several parallel solutions were set in motion.

5.1 Fixing The Existing Gearbox

As a temporary solution, the current gearbox were fixed by welding the broken joint together. The attempt on fixing the gearbox failed, this can be seen in Figure 5.2. This supports the theory of the joint being exposed to forces above the breakdown rating of the gearbox.

5.2 New Gearbox And Transmission

The previous steer gearbox and transmission solution were not an ideal solution. The planetary gearbox had a gear ratio of 91.12:1, larger than the

Figure 5.1: Image showing the broken third stage of the planetary gearbox.

desired gear ratio. To compensate for this the transmission were built up of two spur gears with a ratio of 1:2. This gave the previous combination a lower combined gear ratio, with the drawback of doubling the forces on the planetary gearbox output shaft. The previous solution had a total gear ratio from motor to steer axis of 45.56:1, which seemed to be sufficient. With more than enough torque to move the steer during rides. There are at least three different solutions to the problem:

1. Ordering a new planetary gearbox with the same gear ratio capable of handling higher forces.

2. Ordering a new planetary gearbox with a ratio about 45:1 and a new transmission without a gear ratio.

3. Ordering a new planetary gearbox with a ratio significantly lower than 45:1 and a transmission with a gear ratio that corresponds to the selected planetary gear ratio.

Since it was a lack of time, several distributors of motors and gears were contacted to see which solutions they could deliver at a short notice. The only distributor capable of delivering a suitable planetary gearbox within the time and budget limits were Stork Drives in Sweden. They could deliver a 15:1 planetary gearbox in 5 weeks. The gearbox were of the same brand and series as the previous one, making us able to reuse the motor without any modifications. To be able to use the planetary gear above a, new transmission

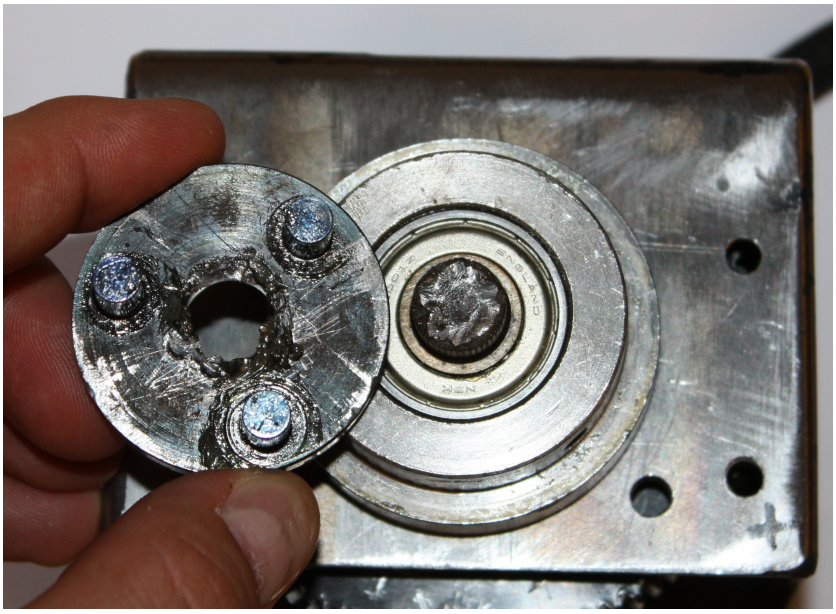


Figure 5.2: Image showing the result of the gearbox fixing, it broke again.

with a gear ratio around 3:1 were needed. Aratron located in Oslo could deliver spur gears with 45 and 120 teeth, in only 3 weeks from the ordering date. This would give a new total gear ratio of 40:1[1], with a slightly lowered output torque. When changing the transmission some modifications were performed on the steer motor bracket and the mechanical department created a new steer potentiometer bracket, based on the previous design. They also helped with milling the new spur gear, so they would fit on the steer and motor axes. The new steer gear and transmission assembly can be seen in Figure 5.3. With the new solution the output shaft of the planetary gearbox is only exposed to $\frac{1}{6}$ of the forces compared to the previous solution, and should be more than capable of handling the torque. Technical data for the new steer motor and gear assembly can be found in Appendix C

[1] 15:1 and 120:45 gives a total ratio of $\frac{15*120}{1*45} = \frac{1}{40}$

Figure 5.3: Image showing the new steer gear and transmission assembly

Chapter 6

Measurement of Bicycle Parameters

This chapter describes the measurment of the different parameters needed for the Whipple bicycle model described in Chapter 2.1. Most of the methods, theory and equations used in this chapter are taken from [17]. The notations however correspond with the ones found in [13]. Figure 6.1 places some of the parameters in a bicycle sketch.

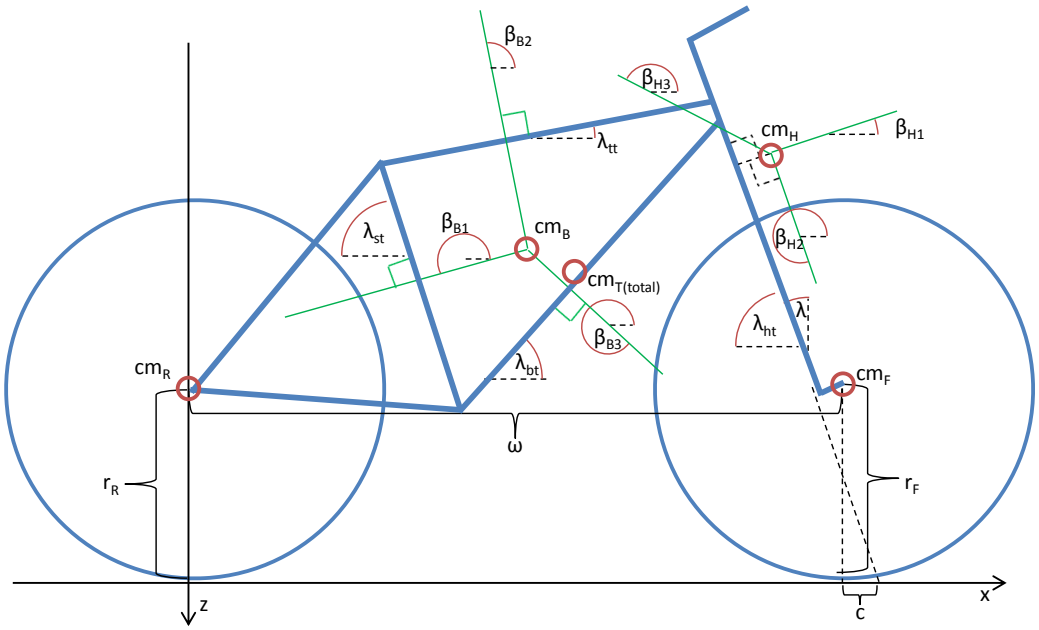


Figure 6.1: Different parameters of the bicycle.

6.1 Wheel Radius

The radius was measured by measuring how long each wheel traversed during 15 rotations. The length measurements were performed with a 50m long tape measure with a resolution of 1mm and an estimated measurement accuracy of ±10mm. The measurements were performed with the tires pumped up to approximately 40psi. The wheel radius were calculated with the following formula where d is the measured length:

$$r = \frac{d}{2\pi n} \tag{6.1}$$

The front wheel traversed length were measured to 30.178m and the rear wheel traversed length were measured to 30.202m. The radius for the front and rear wheel were calculated using Equation (6.1):

$$r_F = 0.3202\text{m} \tag{6.2}$$

$$r_R = 0.3205\text{m} \tag{6.3}$$

6.2 Tube Angles

The angle of the tubes were measured manually with a protractor. The tube angles were measured against a table that was measured to be parallel to ground. The bicycle were standing on the ground with the tires pumped up to approximately 40psi. The protractor had a resolution of 1° and the measurements are considered to have an accuracy of ±1°. The tube angles were measured to:

$$\lambda_{ht} = 70^\circ \tag{6.4}$$

$$\lambda_{tt} = 10^\circ \tag{6.5}$$

$$\lambda_{st} = 74^\circ \tag{6.6}$$

$$\lambda_{bt} = 43^\circ \tag{6.7}$$

The steer axis tilt is the complement to the head angle and can be calculated from the measurement:

$$\lambda = 20^\circ \tag{6.8}$$

6.3 Trail

To calculate trail the fork offset were measured with a vernier caliper with a resolution of 0.05mm. The measurement were performed with the fork lying on a table, supported such as the steer axis were parallel to the table. The distance from the table to the center of the head tube and the distance from the table to the center of the axle axis were measured. The accuracy of the measurements were considered to be ±1mm. The distance from the axle center to the table were measured to 73mm and the distance from the head tube center were measured to $f_o = 35mm$. The trail were calculated from the following formula.

$$c = \frac{r_F sin\lambda - f_o}{cos\lambda} \tag{6.9}$$

$$c = 0.079\text{m} \tag{6.10}$$

6.4 Wheelbase

The wheelbase is the distance between the center of the wheels and were measured directly with a 5m long tape measure with a resolution of 1mm and a accuracy of ±1mm.

$$\omega = 1.051\text{m} \tag{6.11}$$

6.5 Mass

The mass of the front handlebar frame, front wheel and rear wheel were measured with 5kg scale with a resolution of 1g and the accuracy of ±1g. The rear body frame were measured with a digital bathroom scale with a resolution of 100g, the accuracy is assumed to be ±100g.

$$m_H = 3.283\text{kg} \tag{6.12}$$

$$m_F = 2.055\text{kg} \tag{6.13}$$

$$m_R = 2.680\text{kg} \tag{6.14}$$

$$m_B = 23.1\text{kg} \tag{6.15}$$

6.6 Center of Mass

The center of mass for the four parts of the bicycle are calculated in this chapter. The calculations in this chapter were calculated with the help of MATLAB and can be found in Appendix A.1.

6.6.1 Wheels

To comply with the Whipple bicycle model the center of mass of the wheels are assumed to be at the geometric center of the wheels. The center of mass of the wheels in the global coordinate system are described by the following equations:

$$cm_F = \begin{bmatrix} x_F & y_F & z_F \end{bmatrix} = \begin{bmatrix} \omega & 0 & -r_F \end{bmatrix} \tag{6.16a}$$

$$cm_R = \begin{bmatrix} x_R & y_R & z_R \end{bmatrix} = \begin{bmatrix} 0 & 0 & -r_R \end{bmatrix} \tag{6.16b}$$

$$cm_F = \begin{bmatrix} 1.0510 & 0 & -0.3202 \end{bmatrix} \tag{6.17a}$$

$$cm_R = \begin{bmatrix} 0 & 0 & -0.3205 \end{bmatrix} \tag{6.17b}$$

6.6.2 Rear Body Frame

The center of mass for the rear body frame were calculated from measurements performed during the torsion pendulum measurements. The rear body frame were in the measurements hung in three different angles with respectively the top tube, seat tube and the sloped tube parallel to ground, the angles can be seen in Figure 6.1. The frame were hung such that the center of mass would be in the extension of the pendulum arm. The angle of the tubes where measured with a level to ensure that the angle offset from horizontal would be close to zero. The accuracy of the angles are considered to be within $\pm 1^\circ$ from horizontal. The horizontal distance between the rear axle and the extension of the pendulum were measured. Together with the tube angles the position of the center of mass can be calculate. The center of mass is calculated by looking at the pendulum axis as a line in the global xz coordinate system, with a slope m, a z-intercept b and a angle β between the global x-axis and the pendulum arm with positive direction downwards. The slope and z-intercept can be shown to be:

$$m_i = -\tan \beta_i \tag{6.18}$$

$$b_i = -\left(\frac{a_{B_i}}{\cos \beta_i} + r_R \right) \tag{6.19}$$

The horizontal distance a_B and β angles:

$$a_B = \begin{bmatrix} 0.1690 \\ 0.4715 \\ 0.5355 \end{bmatrix} \text{m} \tag{6.20}$$

$$\beta_B = \begin{bmatrix} 196^\circ \\ 100^\circ \\ 313^\circ \end{bmatrix} \tag{6.21}$$

The center of mass can be calculated by finding the intersection between these three lines. This is done by calculating the intersection between two and two lines. Equations for the intersection between line 1 and 2 can be found below, while the equations for the other line intersections can easily be derived from the equation for line 1 and 2.

$$\begin{bmatrix} -m_1 & 1 \\ -m_2 & 1 \end{bmatrix} \begin{bmatrix} x_a \\ z_a \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \tag{6.22}$$

Then the three intersections were averaged to get a more accurate position for the center of mass:

$$x_B = \frac{x_a + x_b + x_c}{3} \tag{6.23}$$

$$z_B = \frac{z_a + z_b + z_c}{3} \tag{6.24}$$

The center of mass is then described by the following:

$$cm_B = \begin{bmatrix} x_B & 0 & z_B \end{bmatrix} \tag{6.25}$$

$$cm_B = \begin{bmatrix} 0.4314 & 0 & -0.6330 \end{bmatrix} \tag{6.26}$$

6.6.3 Front Handlebar Frame

The center of mass for the front handlebar frame was found in a similar way as the rear body frame. The front handlebar frame was hung in two different directions, first with the steer axis horizontal and then with the steer axis vertical. Both directions with the center of mass in the extension of the pendulum arm. The horizontal distance a_H from the extension of the pendulum to the front axle were measured and the β angles:

$$a_H = \begin{bmatrix} 0.429 \\ -0.003 \end{bmatrix} \text{m} \tag{6.27}$$

$$\beta_H = \begin{bmatrix} 20^\circ \\ 290^\circ \end{bmatrix} \tag{6.28}$$

The z-intercept used to calculate the front handlebar frame center of mass is changed to:

$$b_i = w \tan(\beta_i) - r_F - \left(\frac{a_{H_i}}{\cos \beta_i} \right) \tag{6.29}$$

The center of mass is found from only two lines, and the center of mass is found by solving Equation (6.22). The center of mass is given by the following matrix:

$$cm_H = \begin{bmatrix} x_a & 0 & z_a \end{bmatrix} \tag{6.30}$$

$$cm_H = \begin{bmatrix} 0.9015 & 0 & -0.7223 \end{bmatrix} \tag{6.31}$$

6.7 Moment of Inertia

The moment of inertia were measured with the assumption that the bicycle is symmetric about the xz mid-plane. The moment of inertia around the x and z axes was measured using a torsional pendulum, while the moment of inertia around the y-axis were measured with a compound pendulum. The oscillations were measured with a stop watch with a resolution of $\frac{1}{10}$ of a second. The time was measured for a duration between 10 to 30 oscillations depending on the frequency, each measurement were performed three times. All calculations were done with MATLAB and can be found in Appendix A.2.

6.7.1 Torsional Pendulum

A torsion pendulum was created for measuring the moment of inertia around the x and z axes, as seen in figure 6.2. The torsional pendulum consists of three parts, a torsion rod, an upper clamp used to fasten the torsion rod to the roof, and a variation of lower clamp combinations to fasten the bicycle to the torsion rod. The lower clamp is hinged by a bolt and the part being measured were hung in the torsion pendulum with the bolt loose to ensure that the center of mass were align in the extension of the pendulum arm. The tube connected to the pendulum were controlled to be horizontal with a level before tightened the bolt. The angles are assumed to be withing the accuracy of $\pm 1^\circ$. The torsion pendulum were calibrated against a cylindrical rod as seen in figure 6.2, with a known moment of inertia calculated by the following equation:

$$I_{calib} = \frac{m_{calib}}{12}\left(3\,{r_{calib}}^2 + {l_{calib}}^2\right) \tag{6.32}$$

The measured times for $\nu = 20$ oscillation for the calibration rod:

$$t_{calib} = \begin{bmatrix} 54.0 \\ 53.9 \\ 53.8 \end{bmatrix} \text{s} \tag{6.33}$$

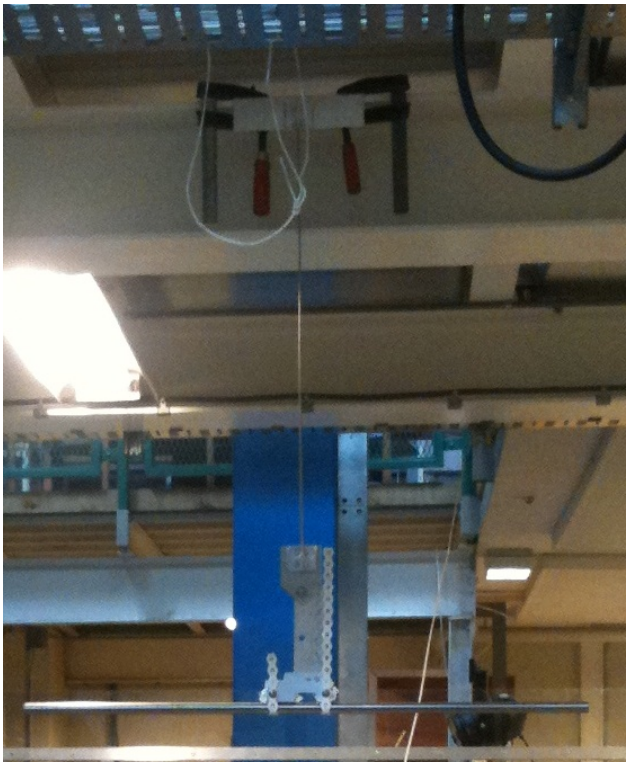


Figure 6.2: The torsion pendulum with the calibration rod mounted

The average oscillation period time can be calculated by the following equation:

$$\overline{T} = \frac{t_1 + t_2 + t_3}{3\nu} \tag{6.34}$$

Together with the known moment of inertia for the calibration rod, the stiffness of the torsional pendulum can be estimated by the following equation:

$$k = \frac{4\, I_{calib}\pi^2}{\overline{T}_{calib}^{\,2}} \tag{6.35}$$

When using the torsional pendulum the moment of inertia around the pendulum axis can be found by the following equation:

$$J = \frac{k\overline{T}^2}{4\pi^2} \tag{6.36}$$

6.7.2 Compound Pendulum

To measure the moment of inertia about the y-axis the principle of a compound pendulum were used. For the wheels a small plastic block with a hole

in it where mounted to the inner rim of the wheel, and then the wheel were hung on a bolt mounted to a staircase. The moment of inertia can then be calculated from the oscillation time with Equation (6.37), where m is the mass and l is the length from the rotation point to the center of mass of the part measured.

$$I_{yy} = \left(\frac{\overline{T}}{2\pi}\right)^2 mgl_c - ml_c^2 \tag{6.37}$$

6.7.3 Wheels

The wheels moment of inertia is quite easy to find as it is assume that the wheels are symmetric about the three orthogonal planes. The moment of inertia around the x and z axes are identical and were measured by hanging the wheel in the torsional pendulum as seen in figure 6.3(a) and 6.3(b). The moment of inertia around the y axis were measured by hanging the wheels as a compound pendulum as seen in Figure 6.4(a) and 6.4(b). The torsional

(a) Front wheel (b) Rear wheel

Figure 6.3: The torsion pendulum with the wheels mounted

and compound pendulum measurements were performed measuring the time for $\nu = 30$ oscillations:

$$t_{F_{xz}} = \begin{bmatrix} 54.6 \\ 54.5 \\ 54.7 \end{bmatrix} \text{ s} \tag{6.38}$$

$$t_{R_{xz}} = \begin{bmatrix} 55.9 \\ 55.8 \\ 55.7 \end{bmatrix} \text{ s} \tag{6.39}$$

(a) Front wheel

(b) Rear wheel

$$t_{F_y} = \begin{bmatrix} 44.1 \\ 44.1 \\ 44.2 \end{bmatrix} \text{s} \tag{6.40}$$

$$t_{R_y} = \begin{bmatrix} 42.0 \\ 41.9 \\ 42.1 \end{bmatrix} \text{s} \tag{6.41}$$

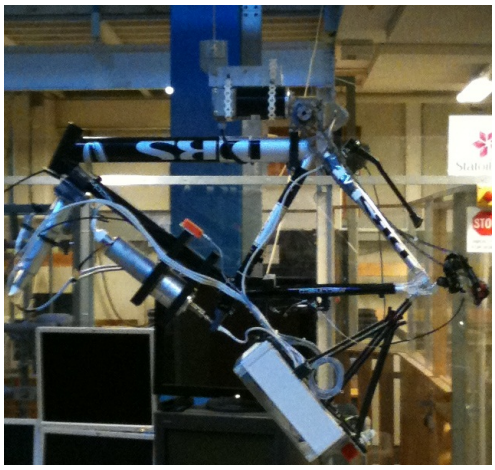
The moment of inertia for the x and z axis can be calculated with Equation (6.34) and (6.36). While the moment of inertia for the y axis can be calculated with Equation (6.34) and (6.37) with the length of the pendulum $l_{c_{fw}} = l_{c_{rw}} =$ 0.265m, the measured distance from the rotational axis of the compound pendulum to the center of the wheel. The moment of inertia for the front(I_F) and the rear(I_R) wheel:

$$I_F = \begin{bmatrix} 0.0758 & 0.1577 & 0.0758 \end{bmatrix} \tag{6.42}$$

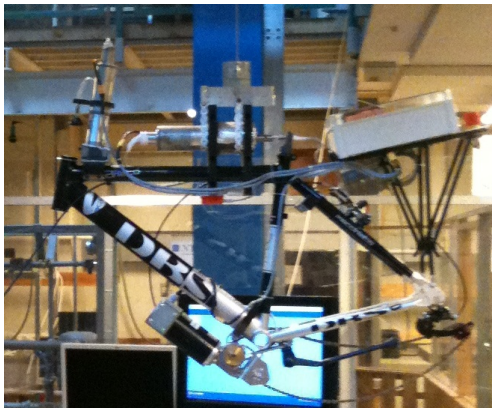
$$I_B = \begin{bmatrix} 0.0726 & 0.1485 & 0.0726 \end{bmatrix} \tag{6.43}$$

6.7.4 Rear Body Frame

For the rear body frame only the moment of inertia about the x and z axis were needed. The rear body frame moment of inertia were measured with the torsional pendulum, the rear body frame were hung in three different angles perpendicular to the top, seat and bottom tubes. The measurements were performed measuring the time for $\nu = 30$ oscillations, the β angles are the same as for the center of mass calculations and can be found in Matrix



(c) Top tube torsional pendulum



(d) Bottom tube torsional pendulum

Figure 6.4: Measurements performed on the rear body frame

(6.21).

$$t_{B_{st}} = \begin{bmatrix} 77.1 \\ 77.1 \\ 77.2 \end{bmatrix} \text{s} \tag{6.44}$$

$$t_{B_{tt}} = \begin{bmatrix} 84.1 \\ 84.0 \\ 84.1 \end{bmatrix} \text{s} \tag{6.45}$$

$$t_{B_{bt}} = \begin{bmatrix} 82.2 \\ 81.8 \\ 82.0 \end{bmatrix} \text{s} \tag{6.46}$$

The moment of inertia about the pendulum axes can be calculated by Equation (6.34) and (6.36). The moment of inertia about the global x and z axes can be calculated by formulating the relation between the inertial frames:

$$\mathbf{J}_i = \mathbf{R}_i \mathbf{I} \mathbf{R}_i^T \tag{6.47}$$

$\mathbf{J}_i$ is the moment of inertia about the pendulum axes. $\mathbf{I}$ is the moment of inertia about the global reference plain. And $\mathbf{R}_i$ is the rotational matrix relating the frames. The $\mathbf{I}$ and $\mathbf{R_i}$ matrices are reduced to 2×2 matrices since the y axis of the two frames are the same and the two frames are assumed lateral symmetric:

$$\mathbf{I} = \begin{bmatrix} I_{xx} & I_{xz} \\ I_{xz} & I_{zz} \end{bmatrix} \tag{6.48}$$

$$\mathbf{R}_i = \begin{bmatrix} \cos\beta_i & -\sin\beta_i \\ \sin\beta_i & \cos\beta_i \end{bmatrix} \tag{6.49}$$

The complete $\mathbf{J}_i$ can be calculated, but only the first entry is needed:

$$J_i = \cos^2\beta_i I_{xx} - 2\sin\beta_i\cos\beta_i I_{xz} + \sin^2\beta_i I_{zz} \tag{6.50}$$

This allows us to form:

$$\begin{bmatrix} J_1 \\ J_2 \\ J_3 \end{bmatrix} = \begin{bmatrix} \cos^2\beta_1 & -2\sin\beta_1\cos\beta_1 & \sin^2\beta_1 \\ \cos^2\beta_2 & -2\sin\beta_2\cos\beta_2 & \sin^2\beta_2 \\ \cos^2\beta_3 & -2\sin\beta_3\cos\beta_3 & \sin^2\beta_3 \end{bmatrix} \begin{bmatrix} I_{xx} \\ I_{xz} \\ I_{zz} \end{bmatrix} \tag{6.51}$$

The moment of inertia can then be described by the following matrix:

$$I = \begin{bmatrix} I_{xx} & 0 & I_{xz} \\ 0 & I_{yy} & 0 \\ I_{xz} & 0 & I_{zz} \end{bmatrix} \tag{6.52}$$

The inertia for the rear body frame, I_{yy} is set to zero since it is not used by the linearised equations:

$$I_B = \begin{bmatrix} 1.3083 & 0 & 0.0421 \\ 0 & 0 & 0 \\ 0.0421 & 0 & 1.5411 \end{bmatrix} \tag{6.53}$$

6.7.5 Front Handlebar Frame

The moment of inertia for the front handlebar frame is found by using the torsional pendulum for finding the moment of inertia about the x and z axes. The torsional measurements and calculations are much the same as for the rear wheel with the following β_H angles, t_H times and ν_H number of oscillations:

$$\nu_H = \begin{bmatrix} 20 \\ 30 \\ 20 \end{bmatrix} \text{ times} \tag{6.54}$$

$$\beta_H = \begin{bmatrix} 20^\circ \\ 290^\circ \\ 147^\circ \end{bmatrix} \tag{6.55}$$

$$t_{H_1} = \begin{bmatrix} 55.2 \\ 55.3 \\ 55.3 \end{bmatrix} \text{ s} \tag{6.56}$$

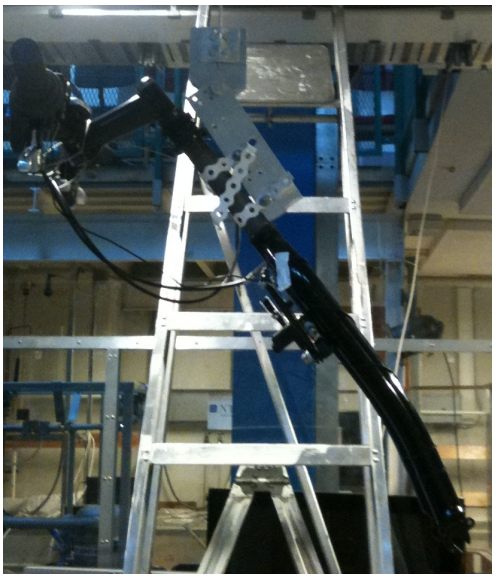
$$t_{H_2} = \begin{bmatrix} 40.1 \\ 40.0 \\ 40.2 \end{bmatrix} \text{ s} \tag{6.57}$$

$$t_{H_3} = \begin{bmatrix} 46.0 \\ 46.0 \\ 46.1 \end{bmatrix} \text{ s} \tag{6.58}$$

The moment of inertia for the front handlebar frame can be written on the form seen in Equation (6.52), I_{Hyy} is set to zero as it us not used by the linearised equations:

$$I_H = \begin{bmatrix} 0.1727 & 0 & -0.0169 \\ 0 & 0 & 0 \\ -0.0169 & 0 & 0.0337 \end{bmatrix} \tag{6.59}$$

(a) Torsion pendulum $\beta = 290°$



(b) Torsion pendulum $\beta = 147°$

Figure 6.5: Front handlebar frame measurements

Chapter 7

Simulation

7.1 Simulation Environment

This section describes the Simulink simulation model developed during this thesis.

7.1.1 Bicycle Model

The bicycle model described in Chapter 2.1 were modelled with the parameters found in Chapter 6. The parameters measured, estimated and calculated in Chapter 6 are found in Appendix A.3. The equations were calculated with the help of MATLAB, with the code found in Appendix A.4. The code calculates the 4 matrices below, needed to complete the model described by Equation (2.2).

$$\mathbf{M} = \begin{bmatrix} 13.3618 & 0.8151 \\ 0.8151 & 0.2094 \end{bmatrix} \tag{7.1a}$$

$$\mathbf{K}_0 = \begin{bmatrix} -18.7346 & -1.2343 \\ -1.2343 & -0.4222 \end{bmatrix} \tag{7.1b}$$

$$\mathbf{K}_2 = \begin{bmatrix} 0 & 17.6051 \\ 0 & 1.2454 \end{bmatrix} \tag{7.1c}$$

$$\mathbf{C}_1 = \begin{bmatrix} 0 & 10.0187 \\ -0.5033 & 1.0098 \end{bmatrix} \tag{7.1d}$$

Equation (2.2) can be arranged in the following form:

$$\ddot{q} = \mathbf{M}^{-1}\left(\mathbf{f} - v\mathbf{C}_1\dot{\mathbf{q}} - [g\mathbf{K}_0 + v^2\mathbf{K}_2]\mathbf{q}\right) \tag{7.2}$$

Equation (7.2) was used to create a model of the bicycle in Simulink, as seen in Figure 7.1.

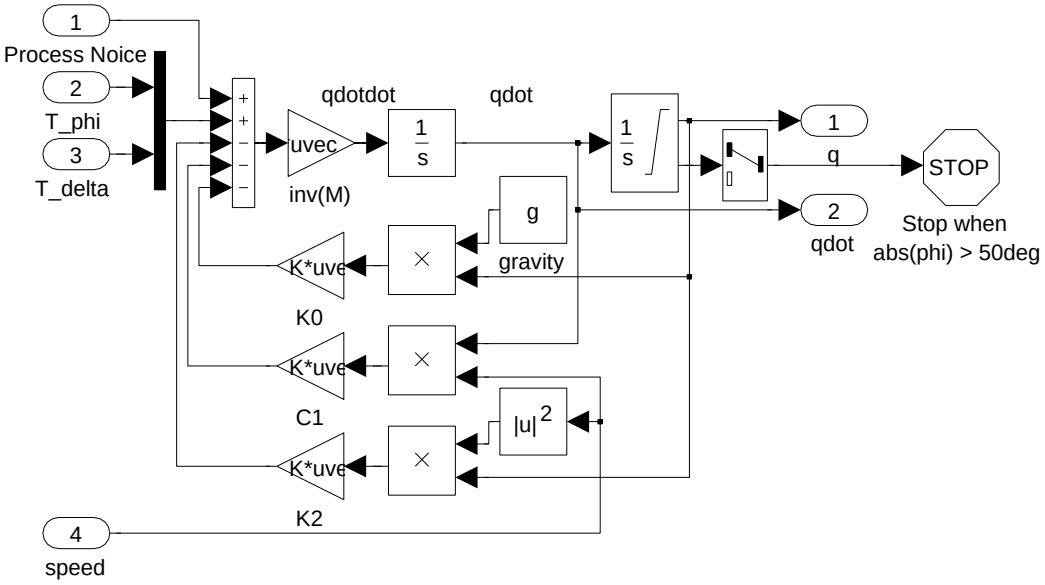


Figure 7.1: Simulink diagram of the bicycle model described in Equation (7.2).

7.1.2 Environment

A simulation environment were built around the model implemented in section 7.1.1, as seen in Figure 7.2. This environment includes real captured measurement noise and simulated process noise(See Chapter 7.1.3.). The

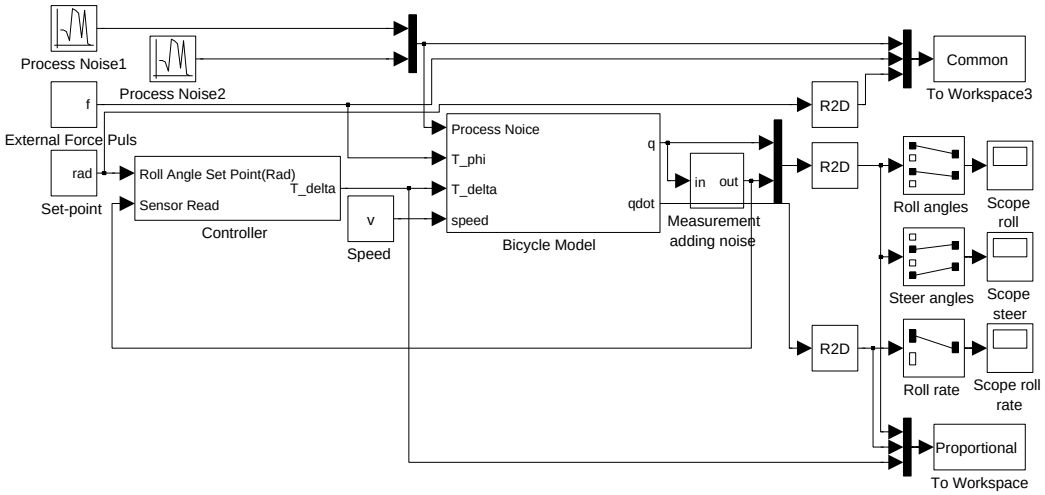


Figure 7.2: Simulink diagram of the simulation environment

simulator has the following features:

- Possible to turn measurement noise on and off.
- Possible to simulate external force on the roll angle, force magnitude, start-time and duration are user selectable.
- Possible to select different set-point signals from a pop-up menu.
- Saves set-point, T_ϕ, T_δ, states and the measured signals to workspace.

7.1.3 Process And Measurement Noise

For use in the simulation environment described in Chapter 7.1.2, a series of measurement noise samples were recorded. This recording was performed by capturing a $60s$ sample series from the physical bicycle, while standing still. This was done by connecting the scaled and calibrated inputs described in Chapter 4.2 to a Simulink "To Workspace" block. The mean of each captured signal were subtracted from the corresponding signal to form each captured noise sample. For the roll, yaw and pitch signals, the small back and forth drift is kept to have a signal as close to the real world as possible. Figure 7.3 shows the noise measurement for the steer and the roll angle measurements. The measurement variance for the steer and roll angle measurement were calculated with the help of MATLAB function 'COV'[1]:

$$E\text{roll meas noise} = 0.0180 \tag{7.3}$$

$$E\text{steer meas noise} = 0.0115 \tag{7.4}$$

The simulation environment uses the following variance for process noise:

$$E\text{roll process noise} = 0.2 \tag{7.5}$$

$$E\text{steer process noise} = 0 \tag{7.6}$$

[1]MATLAB function 'COV(x)' calculates the variance from a vector 'x'.

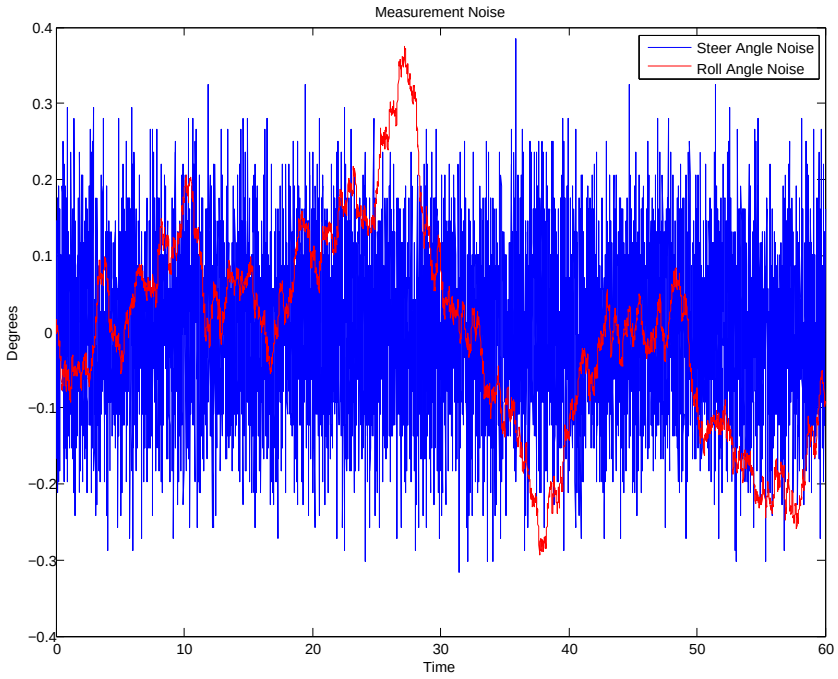


Figure 7.3: Plot showing the real captured measurement noise for the steer and roll angle measurement.

7.2 Controller Simulations

This section describes the simulation of different controllers.

7.2.1 Proportional(P) Control

A controller containing two P regulators in cascade as shown in Figure 7.4 was simulated. The result of the simulation can be found in Figure 7.5. From the result it can be seen that a simple P controller is quite robust and capable of balancing the simulated bicycle.

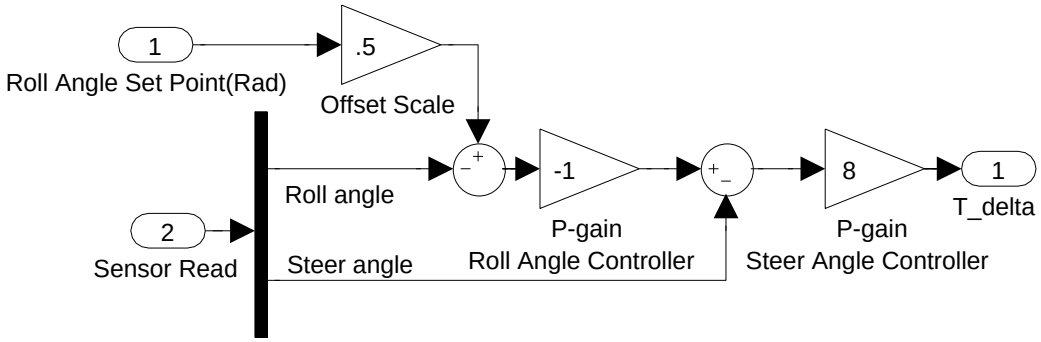


Figure 7.4: Simulink diagram of two P controllers, an inner loop for the steering position and an outer loop for the bicycle roll angle

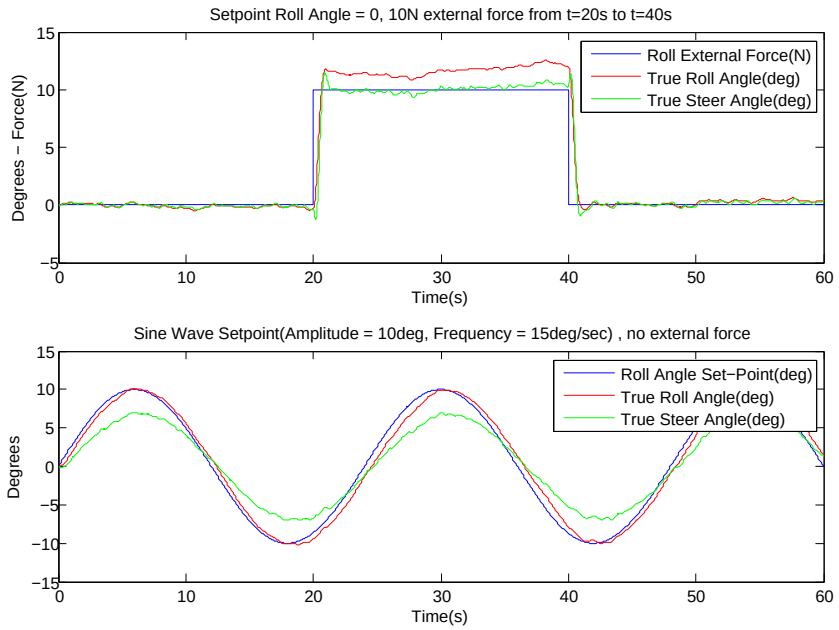


Figure 7.5: Simulation plots for the P controller

7.2.2 Linear-Quadratic Regulator(LQR) Control

To be able to solve the LQ problem and create a LQR regulator, the system has to be described on state-space form, as described in chapter 2.2.1. A linear system as the one described by Equation (2.2), can be written on

state-space form by defining $x_1 = q$ and $x_2 = \dot{q}$ giving:

$$\begin{pmatrix} \dot{x_1} \\ \dot{x_2} \end{pmatrix} = \begin{pmatrix} x_2 \\ \mathbf{M}^{-1}(x_1)\left[-f(x_1, x_2) + u\right] \end{pmatrix} [14]^2 \tag{7.7}$$

This gives us the following matrices for representing Equation (2.2) on state-space form:

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \mathbf{M}^{-1}\left(-\mathbf{K}_0 - (\mathbf{K}_2 * v^2)\right) & & \mathbf{M}^{-1}\left(-\mathbf{C}_1 v\right) & \end{bmatrix} \tag{7.8}$$

$$B = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ \mathbf{M}^{-1} & \end{bmatrix} \tag{7.9}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.10}$$

$$D = 0 \tag{7.11}$$

The optimal gain matrix K was found using MATLAB(as described in Chapter 2.2.3), for the following R and Q weighting matrices, the matrices penalises offsets on the roll angle. MATLAB code for calculating the LQR controller can be found in Appendix A.5.

$$R_{LQR} = \begin{bmatrix} \frac{1}{500} \end{bmatrix} \tag{7.12}$$

$$Q_{LQR} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{500} & 0 & 0 \\ 0 & 0 & \frac{1}{500} & 0 \\ 0 & 0 & 0 & \frac{1}{500} \end{bmatrix} \tag{7.13}$$

The controller was implemented in Simulink as shown in Figure 7.6. The environment were changed to input the four model states(q_ϕ, q_δ, $\dot{q}_\phi$ and $\dot{q}_\delta$) instead of the simulated measurements to the controller. From the simulation results found in Figure 7.7, it can be seen that the generated gain matrix K is capable of stabilising the bicycle. There is a small linear offset with respect to the roll angle set-point, but this can easily be removed by adjusting the scale offset.

[2]Note that in the equation u corresponds to the general steer and lean forces(f) in Equation (2.2), while f corresponds to the $\mathbf{K}_0$, $\mathbf{K}_2$ and $\mathbf{C}_1$ matrices.

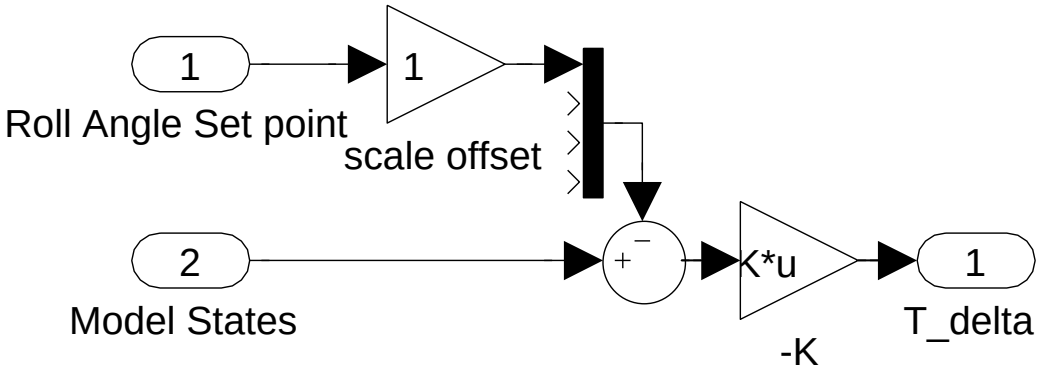


Figure 7.6: Simulink diagram of the ideal LQR controller, all states available directly from the model

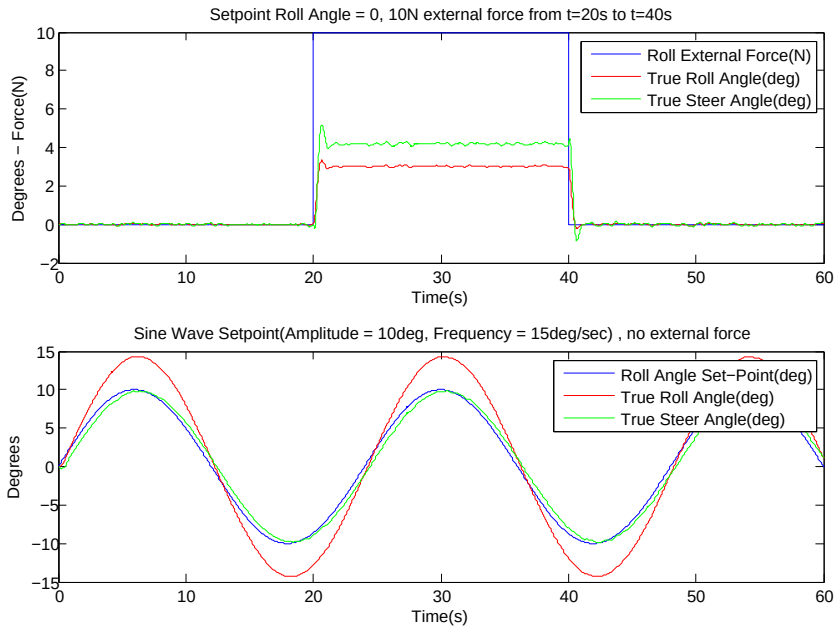


Figure 7.7: Simulation plots for the LQR controller

7.2.3 Linear-Quadratic-Gaussian(LQG) control

The LQR controller created in the previous sub-section needs access to all process states. All states are not measured on the physical bicycle and the states needs to be estimated with an observer. The combination of using a Kalman filter for estimating the states and a LQR controller is called LQG. A Kalman Bucy filter as described in Chapter 2.2.4 is suitable as an observer

for estimating the four states from the simulated measurements. The Kalman Bucy filter was implemented in simulink as shown in Figure 7.8 and is used to estimate the states used by the LQR controller implemented in Chapter 7.2.2.

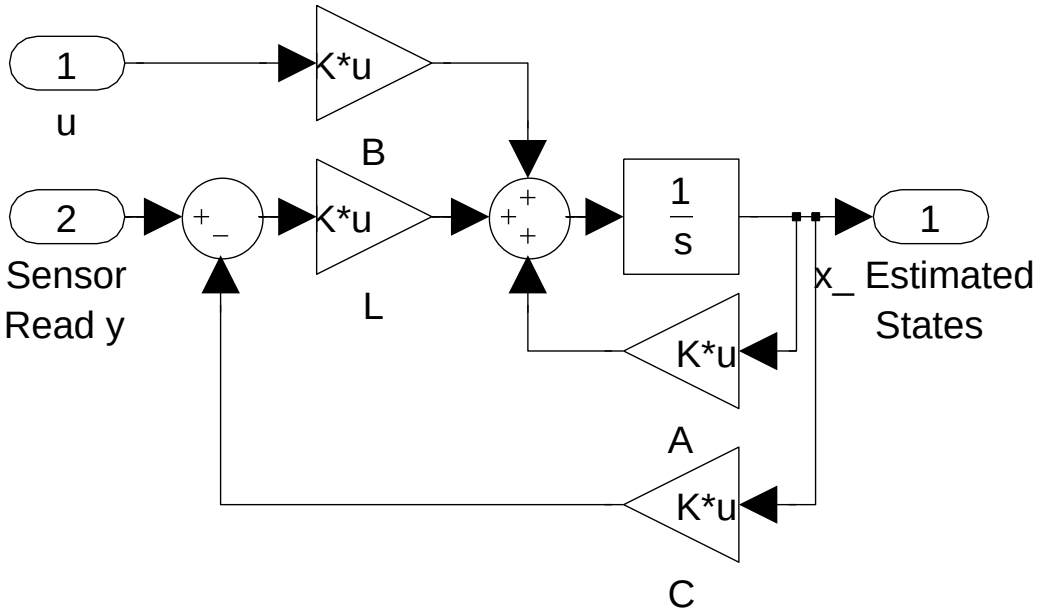


Figure 7.8: Simulink diagram showing the Kalman Bucy filter

The Kalman Bucy filter was created with the covariance matrices found in Equation (7.14) and (7.15). For use in the Kalman Bucy filter design the variance from Chapter 7.1.3 was used as a starting point. The variance for the process steer state were tuned from 0 to 4 to get the observer to be able to estimate the correct steer rate. Attempts on tuning the noise variance to be able to estimate the correct roll rate were performed without any notable significance on the result. Before tuning the process noise variance the controller handled external forces very badly, as seen in Figure 7.9. The simulation result of the LQG controller after tuning the process noise variance can be seen in Figure 7.10. Figure 7.12 compares the estimated states with

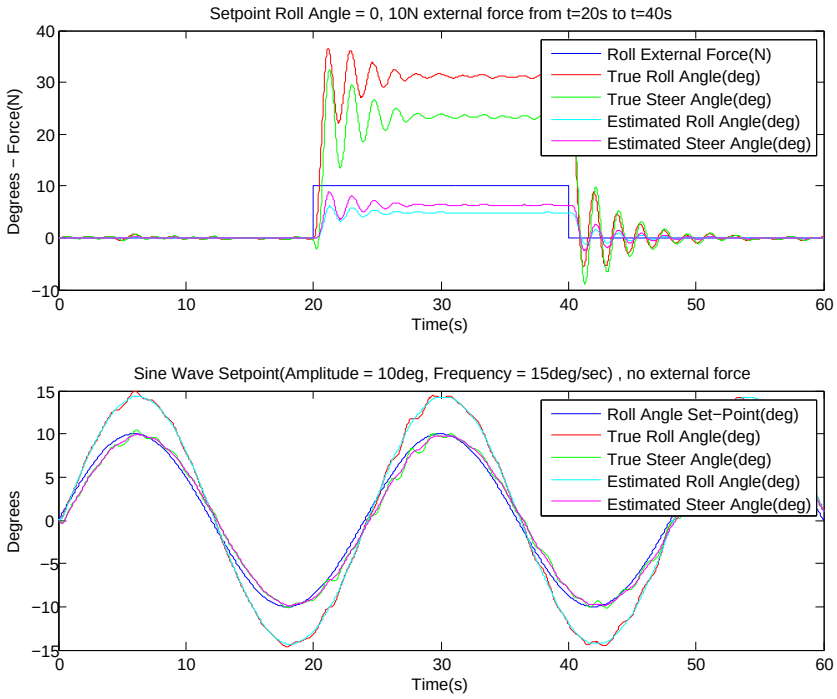


Figure 7.9: Simulation plots for the LQG controller before tuning process noise variance

the true states after the process noise variance tuning. MATLAB code for generating the Kalman Bucy filter can be found in Appendix A.5.

$$Q_{Kalman} = \begin{bmatrix} 0.2 & 0 \\ 0 & 4 \end{bmatrix} \tag{7.14}$$

$$R_{Kalman} = \begin{bmatrix} 0.0180 & 0 \\ 0 & 0.0115 \end{bmatrix} \tag{7.15}$$

After tuning the Kalman Bucy filter the LQR regulator was tuned by increasing the weighting of roll angle in the Q_{LQR} matrix. The result of this tuning can be seen in Figure 7.11.

$$Q_{LQR} = \begin{bmatrix} 1000 & 0 & 0 & 0 \\ 0 & \frac{1}{500} & 0 & 0 \\ 0 & 0 & \frac{1}{500} & 0 \\ 0 & 0 & 0 & \frac{1}{500} \end{bmatrix} \tag{7.16}$$

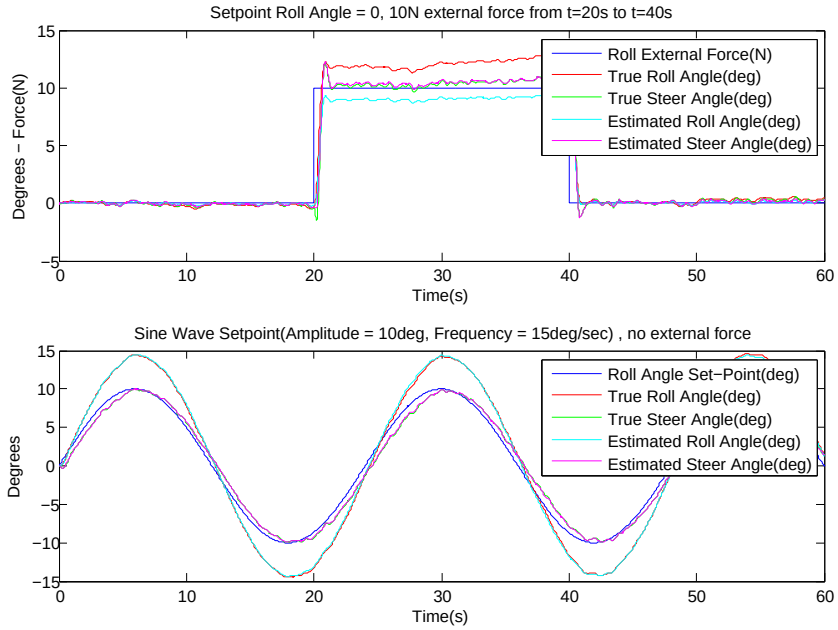


Figure 7.10: Simulation plots for the LQG controller after tuning process noise variance

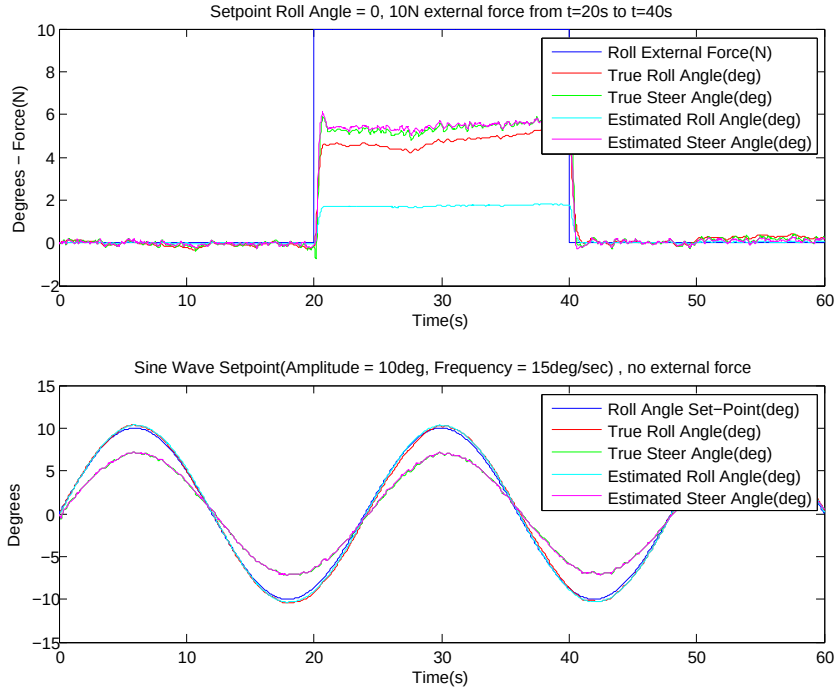


Figure 7.11: Simulation plots for the LQG controller after tuning both process noise variance and the LQR regulator

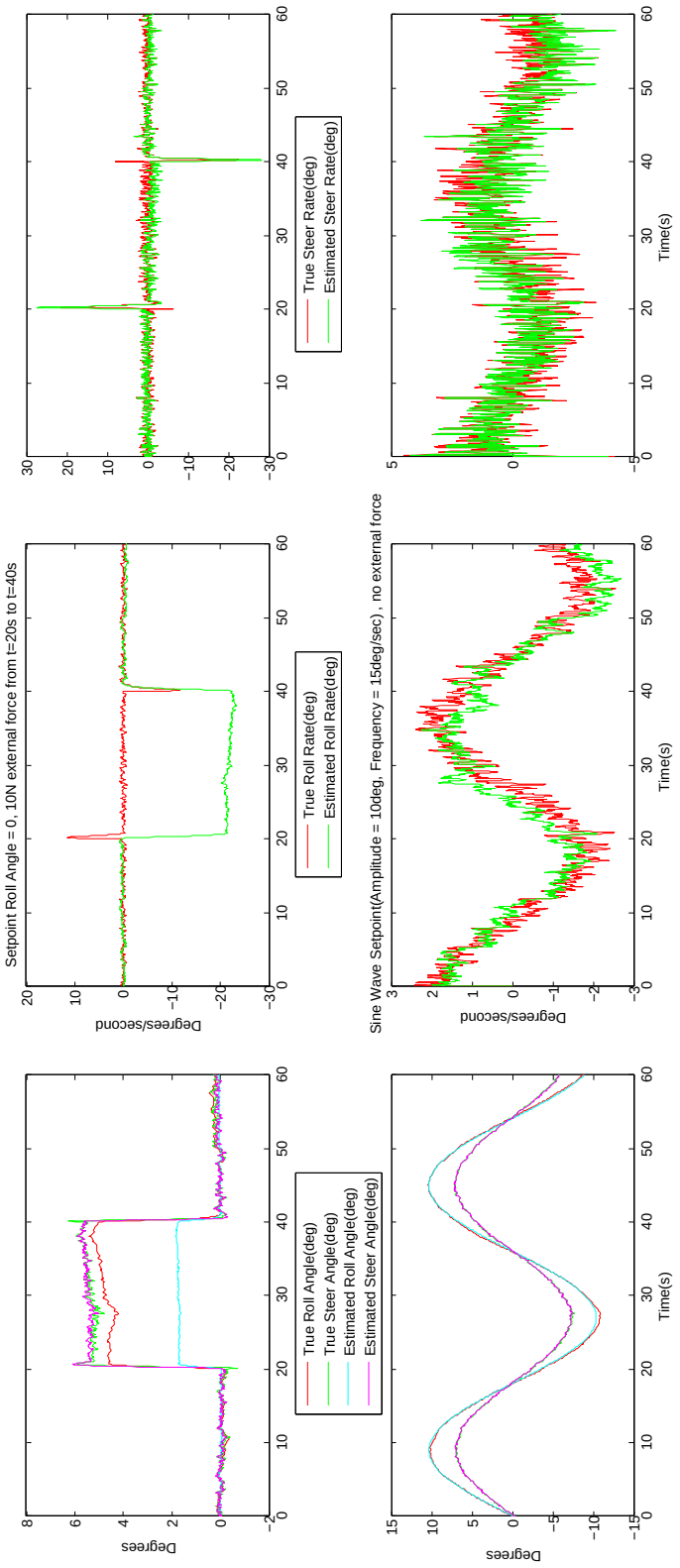


Figure 7.12: Simulation plots comparing the true states versus the estimated states

7.2.4 Discussion And Comparison Of The Simulated Controllers

It can be seen from the results from the simulations, that both the P and the LQG controller are capable of balancing the simulated bicycle. It can be seen by comparing the simulation plots that LQG controller deals with external forces better than the P controller. It could be able to tune the LQG controller even more, but there is no reason for doing so since the LQG controller would need to be tuned before implemented on the physical bicycle. The LQG controller has the advantage over the P controller when it comes to flexibility in the term of handling multiple inputs. While the P controller has the advantage of being very simple and not dependant on a mathematical model of the system.

Chapter 8

Real World Self Balancing

This chapter describes the work performed to implement a self balancing controller on the physical bicycle, and the implementation of a controller for the propulsion speed. The P and LQG controller simulated in Chapter 7.2 were both capable of balancing the simulated bicycle. The P-controller has the advantage of being able to control the bicycle with feedback directly from the measured angles. The model and observer of the LQG controller needs to be verified before being used, and most likely tuned before being capable of balancing the bicycle. One way of verifying and tuning the observer would be to use motor set-point and measurements from a self balanced ride performed with the P controller. By logging the output of the controller together with the measurements and see if the observer is capable of estimating correct values. It were selected to first try out the P controller and than implement the LQG controller depending on the performance of the P controller.

8.1 Propulsion Speed Controller

The bicycle has a free rotation hub on the rear wheel and not equipped with brakes. This gives no possibility to physically slow down the bicycle, except waiting for the forward friction and air resistance to slow it down. Since there is no need for braking capabilities during this thesis, the choice of selecting which method for implementing braking capabilities are left for the future. This slightly complicates the implementation of a speed controller. The controller implemented is a PI controller without feedback on the proportional part. The controller is implemented with two sets of parameters, one set used when running and one set used during the start to reduce the time used for accelerating to the desired speed. The regulator is implemented in Simulink as two different regulators with a switch changing from the first to the second

controller just before the set point speed is reached. An illustration of the controller can be seen in Figure 8.1. The controller was tuned with a person running alongside the bicycle for stability.

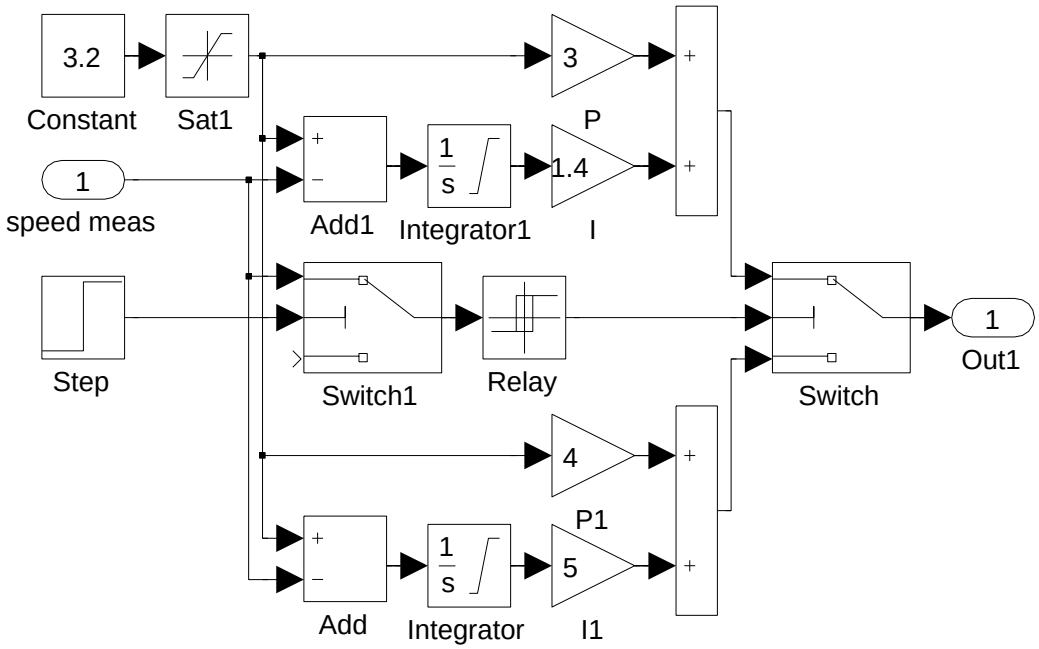


Figure 8.1: Simulink diagram of the propulsion speed controller.

8.2 Self Balance With A P Controller

The P controller simulated in Chapter 7.2.1 was implemented into the Simulink framework as seen in Figure 8.2 and 8.4. In the simulation model the steering position is controlled by applying an external torque on the steer axis, while on the physical bicycle the the voltage applied to the motor is controlled. The steer angle controller were initially tuned standing still, before the first test ride to compensate for the differences between the model and the physical bicycle. The response of the steer angle controller when standing still can be seen in Figure 8.3. After tuning the steer angle controller, the roll angle controller was tuned to balance the bicycle. During one of the first rides trying to tune the controller a catastrophe occurred, the steering planetary gearbox broke down. Chapter 5 describes the gear box breakdown and the work of replacing it. Due to this event there was only enough time to implement one controller, the P controller were able to balance the bicycle. It can be seen in Figure 8.6 that a P gain of 2 results in satisfying balance of the bicycle.

When looking at Figure 8.6 is it important to compare the result with measurement noise, IMU back and forth drift and the actual movement of the bicycle. It can be seen in the video on the CD attached to the thesis that this controller is good enough for real world self balance. The small offset seen in the figure indicates that the bicycle has a small drift to the right. This is by the controller compensated for with a negative movement of the steer[1] in order to balance the bicycle. The offset of the roll angle is probably caused by a combination of the sensor offset, noise, drift and external forces acting on the bicycle.

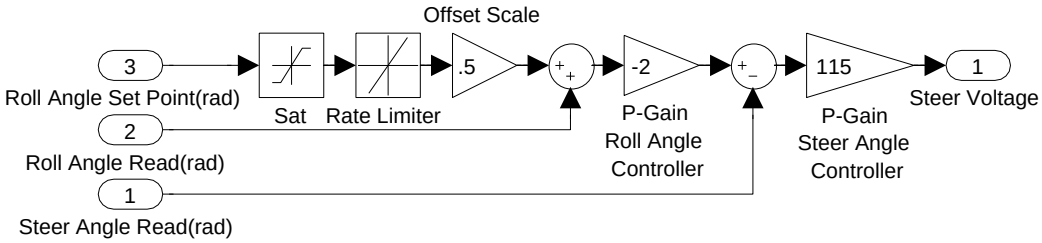


Figure 8.2: Simulink diagram of the controller implemented on the physical bicycle.

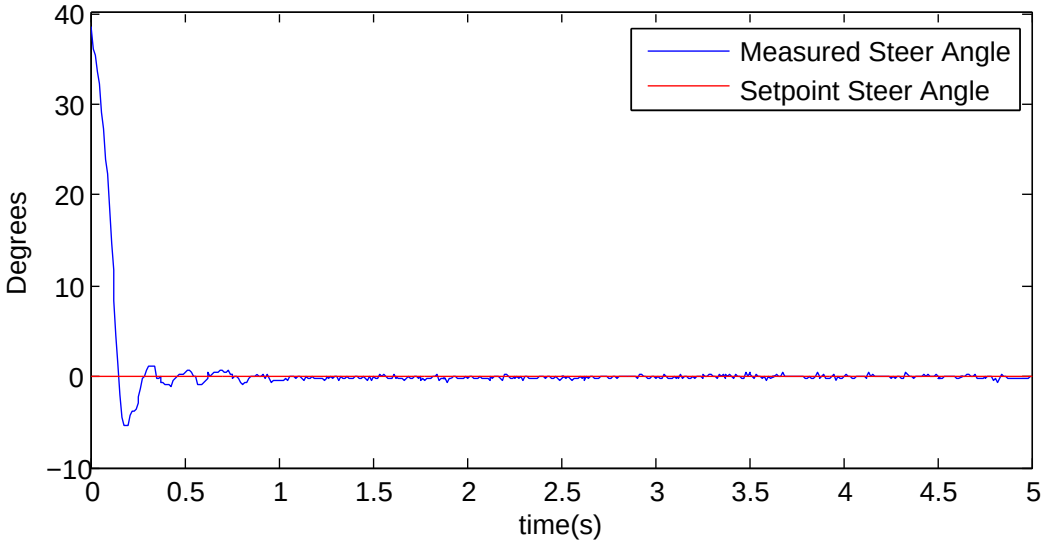


Figure 8.3: Plot showing the steer angle controller response.

[1]The roll and steer measurements are positive in different directions.

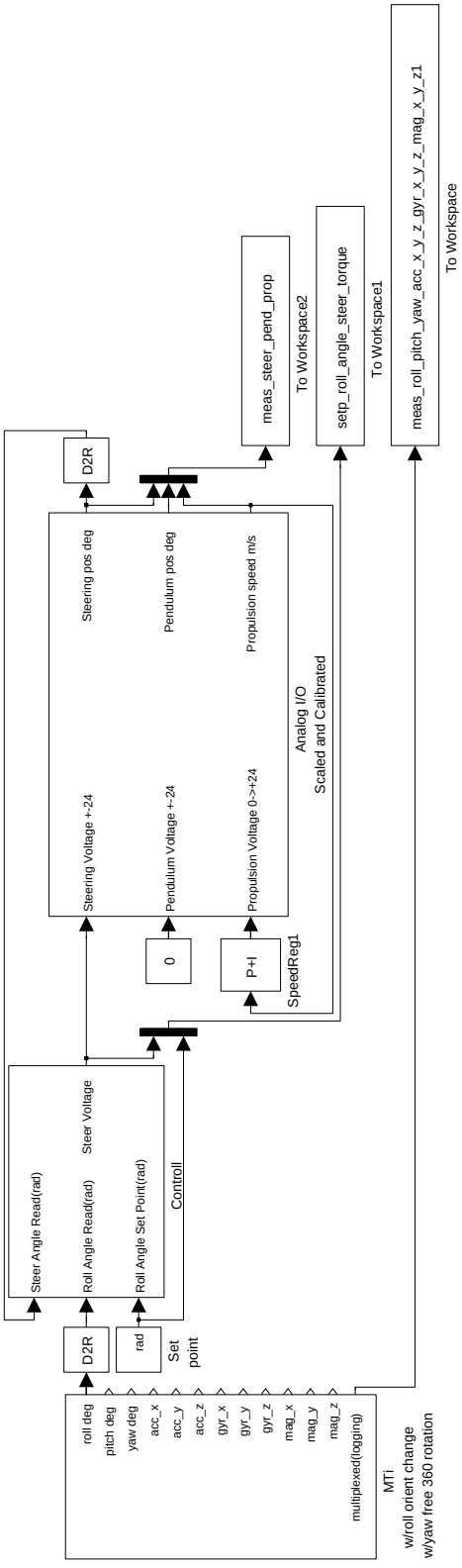


Figure 8.4: The total Simulink diagram for the physical bicycle.

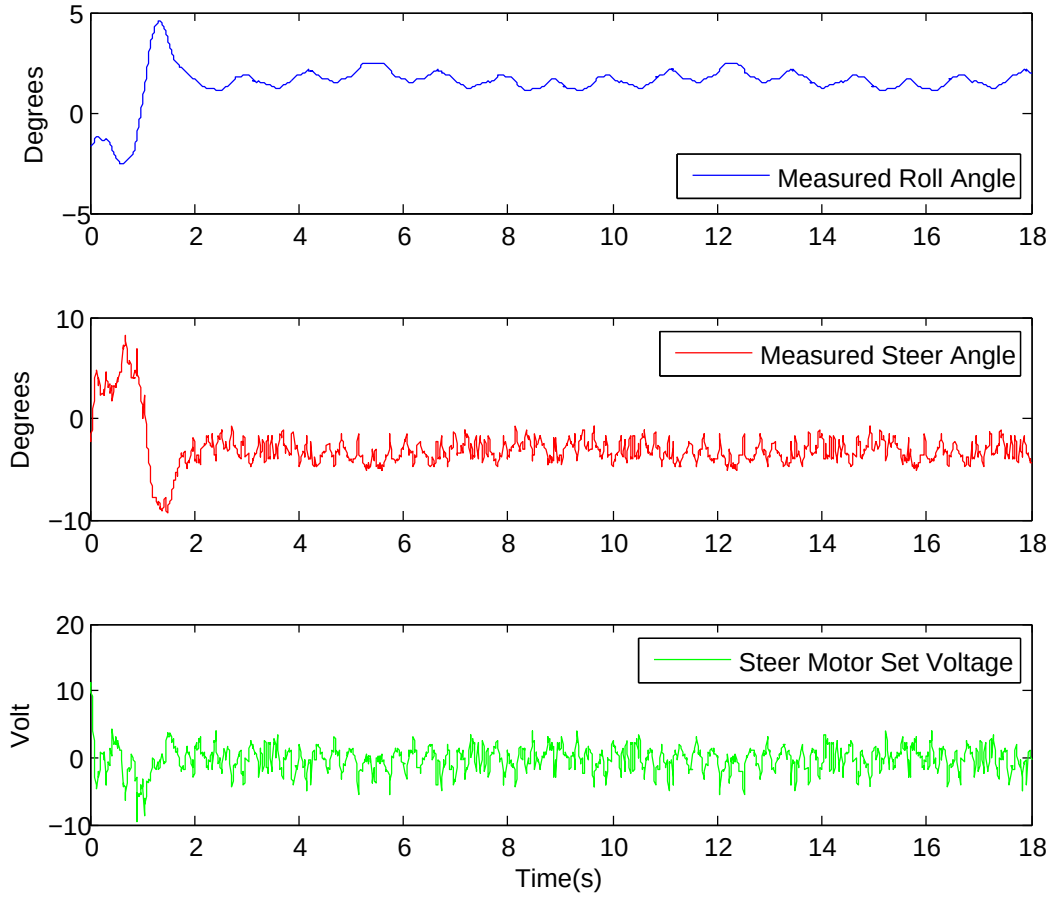


Figure 8.5: Plot showing the roll angle controller response.

Figure 8.6: An image showing the bicycle during one of the test rides.

Chapter 9

Roadmap For Further Work

The bicycle is know finally capable of performing a self balanced ride. While the primary focus of this thesis has been to deliver a working self balanced bicycle, there has been an equally important secondary focus on delivering a complete functional hardware and software framework for the further development on the bicycle. This chapter is a recommendation on how to further develop the bicycle. The various tasks below are described in the order that the author would recommend further development. The chapter includes some possible solutions for solving some of the task, this should not be seen as a blueprint on how it should be solved, but more as an idea of a solution.

9.1 Braking Capabilities

One of the things that might come handy in the development of a low to zero speed controller is the ability to slow down the bicycle. This to better control and measure the actual speed of the bicycle. The author could think of two ways for implementing braking capabilities to the bicycle. By using either the propulsion motor or the bicycle brakes.

9.1.1 Motor Brake

Using the propulsion motor for braking is the simplest solution. To enable this all that needs to be done is to block the free rotation hub, change the lower analog output saturation from 0 to -2048 and implement a new speed controller.

9.1.2 Bicycle Brakes

To use the bicycle brakes there are needed to add servos capable of powering the brakes. Then the motors need to be interfaced to the computer system and at last a new speed controller fusing the propulsion and brake would need to be implemented. The current computer system have no direct method for controlling a brake servo, one solution could be to implement support for the digital outputs on the I/O card and use these to control the brakes.

9.2 Verifying And Updating The Model

The bicycle is now equipped with a functional controller and before performing further development, a verification and update of the model against the physical bicycle should be performed. One way to update the model could be to use MATLAB 'System Identification Toolbox' to identify the state-space matrices from the measured system input and output of the physical bicycle.

9.3 Leaning Rider - Low To Zero Forward Speed Balance

One of the goals for the Autonomous Bicycle to distinguish it from other project is the simulated leaning rider in form of an inverted pendulum. The inverted pendulum currently attached to the bicycle is fully controllable from the Simulink model, but before being used in a balance controller the gearbox should be replaced by a new one without backlash. This kind of gearbox is quite expensive and two possible suppliers are 'Sumitomo Drive Technologies' and 'Harmonic Drive AG'. It is worth to mention that there is a rather long delivery time for these type of gears so a candidate aiming on changing the gearbox is recommended to write both project and thesis on the Autonomous Bicycle in order to have time to implement a controller on the physical bicycle. I would recommend a candidate to use this thesis as a starting point, and use the thesis of Bjermeland[2] as a guidance on how to add the pendulum to the model.

9.4 Autopilot

The implementation of an autopilot could be performed independently of the addition of the leaning rider. The implementation of an autopilot could

be performed with several different approaches. This section describes some different solutions.

9.4.1 Without Any Additions

An basic autopilot capable of controlling the movement of the bicycle base on inertial navigation could be implemented without adding any sensors to the bicycle. This kind of controller would only be capable of controlling the bicycle relative to the starting position, but it might be sufficient to perform short autonomous rides.

9.4.2 With GPS

If a GPS receiver is added to the system it could be possible to create a bicycle capable of following a defined path relevant to the world, not only relevant to the starting position. Due to the accuracy of the GPS signals, this solution would need to perform sensor fusion of the data from both the GPS and the IMU. To add a GPS sensor to the computer system the spare RS-232 port could be used. In order for the Simulink model to talk to the GPS sensor there would be needed to write a driver. The current framework uses device managers for handling the communication with the hardware and an MATLAB s-function that communicates with the device manager. The MTi device manager and MATLAB s-function could be copied and used as a starting point when creating a driver for a GPS.

9.4.3 With Computer Vision

An autopilot could also be created with some kind of vision to follow for example a path in on the ground.

9.5 Collision Avoidance

The last thing i would like to add to the roadmap is the implementation of collision avoidance. A completely autonomous bicycle would need to avoid obstacles. This is a area that the author has little experience with and i would instead of suggesting a solution advise a candidate aiming on implementing collision avoidance to have a look on other thesis's targeting this kind of a challenge.

Chapter 10

Discussion

10.1 The Catastrophe

With the benefit of hindsight is it easy to see that I should have done some other choices regarding the steering during my previous project[7]. The choice of gearing up the output of the planetary gear in the transmission resulted in a broken gearbox, due to large forces acting on the gearbox. If I had not made the mistake causing this catastrophe there would probably have been time to test out the LQG controller or implement some kind of heading control as an extra to the thesis. I believe that with the new transmission solution, it should be impossible to break the new gearbox.

10.2 Measurement Of Bicycle Parameters

There could have been possible to measure the parameters of the bicycle more accurate if better equipment had been available. However the authors believe that the error caused by the equipment used are less then the error caused by the assumptions made in the linearised equations.

10.3 Simulations

It can be seen from the simulations that both the P and the LQG controller are capable of balancing the simulated bicycle. The P controller were initially chosen to see if it were possible at all to balance the bicycle with such a simple controller. An integral or derivative term could have been added to improve the controller, but it was seen that the set-point offset is close to linear and therefore compensation is performed with a gain on the set-point.

The simulation of a LQG controller could be seen as unnecessary when we look at the performance of the P controller on the physical bicycle. The LQG controller however has the ability of handling multiple inputs and outputs and the simulations could be a good starting point when an inverted pendulum is to be added in the controller. Without the gearbox breakdown, we would have experienced the satisfying result of the P controller and the time used on implementing and simulating the LQR and LQG controller could have been used on implementing a heading or path controller.

10.4 Real World Implementation

After the catastrophe with the gearbox a choice of which controller to implement had to be done. It was selected to implement the P controller, this since if before implementing the LQG controller we would need to verify the observer to ensure that it's able to estimate the correct states for the real bicycle. It can be seen in Figure 8.6 that the P controller does such a good job on balancing the bicycle that without the gearbox breakdown, I would most likely developed a heading or path controller as an extension to the thesis instead of developing the LQG controller. It could be discussed that the LQG controller might give a slightly better balance, but it would also add complexity and uncertainty regarding the robustness of the controller. How would for exampled the LQG controller react to changes in the measurement or process noise. It is important to remember that one of the main goals of the thesis was to deliver a framework not only capable of performing a self balance ride, but a framework both capable and ready for further development.

10.5 Other

I should have foreseen that the gearbox could breakdown. I would also say that the choice of leaving the inverted pendulum out of the problem were a correct choice related to the fact that it would probably have caused more problems than help on balancing the bicycle.

Chapter 11

Conclusion

The work performed during this thesis was executed with the goal of having the Autonomous Bicycle conduct it's first self balanced ride. The framework for the Autonomous Bicycle is not only capable of delivering the performance needed to conduct self balancing, but is also capable of delivering the performance needed during further development. The Autonomous Bicycle is now for the first time capable of performing a self balanced ride. The goal of this thesis was not to implement the most advanced controller, but to deliver a self balancing bicycle as a platform for further development.

Bibliography

[1] H. O. Loftum, "Styresystem for kybernetisk sykkel - instrumentering for styring av en tohjuls herresykkel," Master's thesis, Norwegian University of Science and Technology (NTNU), Trondheim, June 2006.

[2] L. Bjermeland, "Modeling, simulation and control system for an autonomous bicycle," Master's thesis, Norwegian University of Science and Technology (NTNU), Trondheim, June 2006.

[3] J. A. Fossum, "Instrumentering og datasystemer for autonom kybernetisk sykkel," tech. rep., Norwegian University of Science and Technology (NTNU), Trondheim, December 2006.

[4] A. Sølvberg, "Cyberbike," Master's thesis, Norwegian University of Science and Technology (NTNU), Trondheim, June 2007.

[5] S. E. Brekke, "Autonomous bicycle," Master's thesis, Norwegian University of Science and Technology (NTNU), Trondheim, September 2010.

[6] D. B. Hatlevoll, "Autonomous bicycle, hardware and software development," tech. rep., Norwegian University of Science and Technology (NTNU), Trondheim, January 2011.

[7] D. C. Ånnestad, "Autonomous bicycle - the rebuild, a fresh start," tech. rep., Norwegian University of Science and Technology (NTNU), Trondheim, April 2011.

[8] J. J. W. Whipple, "The stability of the motion of a bicycle," *Quarterly Journal of Pure and Applied Mathematics*, vol. XXX, 1899.

[9] J. M. Papadopoulos, "Bicycle steering dynamics and self-stability: a summary report on work in progress," technical report, Cornell University, Ithaca, NY, 1987.

[10] J. P. Meijaard, "Derivation of the linearized equations for an uncontrolled bicycle," internal report, University of Nottingham, UK, 2004.

[11] J. D. G. Kooijman, "Experimental validation of a model for the motion of an uncontrolled bicycle," Master's thesis, Delft University of Technology, 2006 The Netherlands, 2006.

[12] A. L. Schwab, J. P. Meijaard, and J. M. Papadopoulos, "A multibody dynamics benchmark on the equations of motion of an uncontrolled bicycle," 2005. ENOC-2005, Eindhoven, Netherlands.

[13] J. P. Meijaard, J. M. Papadopoulos, A. Ruina, and A. L. Schwab, "Linearized dynamics equations for the balance and steer of a bicycle: a benchmark and review," *Proceedings of the Royal Society A - Mathematical, Physical and Engineering Sciences*, vol. 463, no. 2084, pp. 1955–1982, 2007,.

[14] O. Egeland and J. T. Gravdahl, *Modeling and Simulation for Automatic Control*. Marine Cybernetics, 2002. ISBN 82-92356-01-0.

[15] R. G. Brown and P. Y. C. Hwang, *Introduction to random signals and applied kalman filtering*. Wiley, 1997. ISBN 0-471-12839-2.

[16] MathWorks, *MATLAB Function Reference* . http://www.mathworks.se/help/techdoc/ref/f16-6011.html. Checked September 2011.

[17] J. K. Moore, M. Hubbard, D. L. Peterson, A. L. Schwab, and J. D. G. Kooijman, "An accurate method of measuring and comparing a bicycle's physical parameters," in *Bicycle and Motorcycle Dynamics: Symposium on the Dynamics and Control of Single Track Vehicles*, (Delft, Netherlands), October 2010.

[18] Dunkermotoren, *Permanent Magnet DC-Motors*. Dunkermotoren GmbH, http://www.dunkermotoren.de/data/technical_data/motors/pdf/90130_GR 53x58_Seite 30.pdf, 2009. Checked July, 2011.

[19] Dunkermotoren, *Planetary Gearbox*. Dunkermotoren GmbH , http://www.dunkermotoren.de/data/technical_data/gears/pdf/1004_Flyer_PLG52.pdf, 2009. Checked July, 2011.

[20] Huco, *Stainless Steel Spur Gears with hub* . http://www.huco.com/products.asp?p=true&cat=267. Checked July, 2011.

Appendix A

Matlab Calculations

A.1 Center Of Mass

```
%GENERAL PARAMETERS
w      = 1.051;          %m   Wheelbase
r_R    = 0.3205;         %m   Radius rear wheel
r_F    = 0.3202;         %m   Radius front wheel
alpha  = 70*pi/180;      %rad head tube angle
lambda = 20*pi/180;      %rad Steer axis tilt,comp. to head
lambda_st = 74*pi/180;   %rad seat tube angle
lambda_tt = 10*pi/180;   %rad top tube angle
lambda_bt = 43*pi/180;   %rad bottom tube angle
%REAR WHEEL
cm_R = [0 0 -r_R]        %m CM rear wheel in global coord
%FRONT WHEEL
cm_F = [w 0 -r_F]        %m CM front wheel in global coord
%REAR BODY FRAME
a_B = [-0.1690;          %m distance rear axle to pend
        0.4715;
        0.5355];
b_B = [(270*pi/180)-lambda_st;%rad pend angle, beta
       (270*pi/180)+lambda_tt;
       (270*pi/180)+lambda_bt];
b   = -[a_B(1)/cos(b_B(1))+r_R;
        a_B(2)/cos(b_B(2))+r_R;
        a_B(3)/cos(b_B(3))+r_R];
m   = -[tan(b_B(1));
        tan(b_B(2));
        tan(b_B(3))];
xz_a = inv([-m(1) 1;-m(2) 1])*[b(1);b(2)];
xz_b = inv([-m(2) 1;-m(3) 1])*[b(2);b(3)];
xz_c = inv([-m(1) 1;-m(3) 1])*[b(1);b(3)];
```

```
cm_B = [(xz_a(1)+xz_b(1)+xz_c(1))/3 0 ...
    (xz_a(2)+xz_b(2)+xz_c(2))/3]
%FRONT HANDLEBAR FRAME
a_H = [0.429;           %m distance front axle to pend
       -0.003];
b_H = [lambda;          %rad pend angle, beta
       (360*pi/180)-alpha];
b   = [w*tan(b_H(1))-r_F-a_H(1)/cos(beta(1));
       w*tan(b_H(2))-r_F-a_H(2)/cos(beta(2))];
m   = [-tan(b_H(1));
       -tan(b_H(2));];
xz  = inv([-m(1) 1;-m(2) 1])*[b(1);b(2)];
cm_H = [xz(1) 0 xz(2)]  %m cm rear fork in global coord
```

A.2 Moment Of Inertia

```
clear;
%GENERAL PARAMETERS
g         = 9.81;               %N/kg Gravity
w         = 1.051;              %m    Wheelbase
m_R       = 2.680;              %kg   Mass rear wheel
m_H       = 3.283;              %kg   Mass front frame
m_F       = 2.055;              %kg   Mass front wheel
lambda_st = 74*pi/180;          %rad  seat tube angle
lambda_tt = 10*pi/180;          %rad  top tube angle
lambda_bt = 43*pi/180;          %rad  bottom tube angle
%Calculating the k factor from the calib rod meas
m_calib = 2.108;                %kg   Mass calib rod
l_calib = 0.9515;               %m    Length calibrod
r_calib = 0.0100;               %m    Radius calib rod
I_calib = m_calib/12*(3*r_calib^2+l_calib^2);
T_calib = (54+53.9+53.8)/20/3;%s Osc time calib rod
k       = (4*I_calib*pi^2)/(T_calib^2);
%REAR WHEEL
nu_R   = 30;                    %     Number of osc
l_R    = 0.265;                 %m    Length cpend
t_R_xz = (55.9+55.8+55.7)/3/nu_R;%s Average time, tpend
t_R_y  = (42.0+41.9+42.1)/3/nu_R;%s Average time, cpend
I_R_xx = k*t_R_xz^2 / (4*pi^2);  %kgm^2 MOI,x and z axis
I_R_yy = ((t_R_y/(2*pi))^2*(m_R*g*l_R))-(m_R*l_R^2);
I_R_zz = I_R_xx;
I_R    = [I_R_xx I_R_yy I_R_zz]
%REAR FRAME
nu_B   = 10;                    %Number of osc
t_B    = [(77.1+77.1+77.2)/3/nu_B;%s Avg time,seat tube
```

```
30            (84.1+84.0+84.1)/3/nu_B;% top tube
31            (82.2+81.8+82.0)/3/nu_B];% slope tube
32  J_B     = [k*t_B(1)^2/(4*pi^2);   %kgm^2 MOI
33             k*t_B(2)^2/(4*pi^2);
34             k*t_B(3)^2/(4*pi^2)];
35  b_B     = [(270*pi/180)-lambda_st;%rad pend angle, beta
36             (90*pi/180)+lambda_tt;
37             (270*pi/180)+lambda_bt]
38      %Transformation matrix rear frame
39  JI_B    = [cos(b_B(1))^2 -2*sin(b_B(1))*cos(b_B(1)) ...
       sin(b_B(1))^2;
40             cos(b_B(2))^2 -2*sin(b_B(2))*cos(b_B(2)) ...
                  sin(b_B(2))^2;
41             cos(b_B(3))^2 -2*sin(b_B(3))*cos(b_B(3)) ...
                  sin(b_B(3))^2];
42  I_Bt    = JI_B\J_B;
43  I_B     = [I_Bt(1)  0   I_Bt(2);
44             0       0   0;
45             I_Bt(2)  0   I_Bt(3)]
46  %FRONT FRAME
47  cm_H    = [0.9015  0 -0.7223];    %m cm front fork
48  r_F     = 0.3202;                 %m Radius front wheel
49  l_H     = sqrt((cm_H(1)-w)^2+(cm_H(3)+r_F)^2);%m Length cpend
50  nu_H    = [20;
51             30;
52             20;
53             30];
54  t_H_xz  = [(55.2+55.3+55.3)/3/nu_H(1);%s Avg time
55             (40.1+40.0+40.2)/3/nu_H(2);
56             (46.0+46.0+46.1)/3/nu_H(3)];
57  t_H_y   = (44.2+44.3+44.2)/3/nu_H(4);%s Avg time cpend
58  J_H     = [k*t_H_xz(1)^2/(4*pi^2);
59             k*t_H_xz(2)^2/(4*pi^2);
60             k*t_H_xz(3)^2/(4*pi^2);];
61  b_H     = [20*pi/180;             %rad pend angle, beta
62             290*pi/180;
63             147*pi/180];
64      %Transformation matrix front frame
65  JI_H    = [cos(b_H(1))^2 -2*sin(b_H(1))*cos(b_H(1)) ...
       sin(b_H(1))^2;
66             cos(b_H(2))^2 -2*sin(b_H(2))*cos(b_H(2)) ...
                  sin(b_H(2))^2;
67             cos(b_H(3))^2 -2*sin(b_H(3))*cos(b_H(3)) ...
                  sin(b_H(3))^2];
68  I_H_xz  = JI_H\J_H;
69  I_H_y   = ((t_H_y/(2*pi))^2*(m_H*g*l_H))-(m_H*l_H^2);
70  I_H     = [I_H_xz(1) 0     I_H_xz(2);
71             0      I_H_y 0;
72             I_H_xz(2) 0     I_H_xz(3)]
```

```
%FRONT WHEEL
nu_F   = 30;                      %Number of osc
l_F    = 0.265;                   %m length cpend
t_F_xz = (54.6+54.5+54.7)/3/nu_F;%s Avg time, tpend
t_F_y  = (44.1+44.1+44.2)/3/nu_F;%s Avg time, cpend
I_F_xx = k*t_F_xz^2 / (4*pi^2);  %kgm^2 MOI, x and z-axis
I_F_yy = ((t_F_y/(2*pi))^2*(m_F*g*l_F))-(m_F*l_F^2);
I_F_zz = I_F_xx;
I_F    = [I_F_xx I_F_yy I_F_zz]
```

A.3 List Of Parameters Needed By The Model

```
%General params
w      = 1.051;          %Wheelbase(m)
c      = 0.079;          %trail(m)
alpha  = 70*pi/180;      %head angle(rad)
lambda = 20*pi/180;      %Steer axis tilt,comp. to head(rad)
g      = 9.81;           %gravity(N/kg)
%Rear wheel
cm_R   = [0      0      -0.3205]; %rear wheel center of mass
r_R    = 0.3205;                  %rear wheel radius(m)
m_R    = 2.680;                   %rear wheel mass(kg)
I_R    = [0.0758 0.1577  0.0758]; %rear wheel moment of iner
%Rear body frame
cm_B   = [0.4254 0      -0.6427]; %rear frame center of mass
m_B    = 23.1;                    %rear frame mass(kg)
I_B    = [1.3002 0       0.0344;  %rear frame moment of iner
           0      0       0
           0.0344 0       1.5436];
%Front handlebar frame
cm_H   = [0.9015 0      -0.7223]; %front frame center of mass
m_H    = 3.283;                   %front frame mass(kg)
I_H    = [0.1727 0      -0.0169;  %front frame moment of iner
           0      0.1566  0
          -0.0169 0       0.0337];
%Front wheel
cm_F   = [1.0510 0      -0.3202]; %front wheel center of mass
r_F    = 0.3202;                  %front wheel radius(m)
m_F    = 2.055;                   %front wheel mass(kg)
I_F    = [0.0726 0.1485  0.0726]; %front wheel moment of iner
```

A.4 Creating Model

```
clc; clear;
run ../bikeparametercalculations/bicycleparameters
disp('bicycleparameters OK')
%Creating variables from bicycle params, to simpl. reading:
la    = lambda;  %la short for lambda
x_B = cm_B(1); z_B = cm_B(3); x_H = cm_H(1); z_H = cm_H(3);
I_Rxx = I_R(1,1); I_Ryy = I_R(1,2); I_Rzz = I_R(1,3);
I_Bxx = I_B(1,1); I_Bxz = I_B(1,3); I_Bzz = I_B(3,3);
I_Hxx = I_H(1,1); I_Hxz = I_H(1,3); I_Hzz = I_H(3,3);
I_Fxx = I_F(1,1); I_Fyy = I_F(1,2); I_Fzz = I_F(1,3);

%Calculating parameters with equations from chapter 3
m_T   = m_R+m_B+m_H+m_F;                          %(eq.3.3)
x_T   = (x_B*m_B+x_H*m_H+w*m_F)/m_T              %(eq.3.4)
z_T   = (-r_R*m_R+z_B*m_B+z_H*m_H-r_F*m_F)/m_T%(eq.3.5)

I_Txx = I_Rxx+I_Bxx+I_Hxx+I_Fxx+m_R ...
    *r_R^2+m_B*z_B^2+m_H*z_H^2+m_F*r_F^2;%(eq.3.6)
I_Txz = I_Bxz+I_Hxz-m_B*x_B*z_B-m_H ...
    *x_H*z_H+m_F*w*r_F;%(eq.3.7)
     %eq.3.8 performed when listing results from measuring.
I_Tzz = I_Rzz+I_Bzz+I_Hzz+I_Fzz+m_B ...
    *x_B^2+m_H*x_H^2+m_F*w^2;%(eq.3.9)

m_A   = m_H+m_F;                                  %(eq.3.10)
x_A   = (x_H*m_H+w*m_F)/m_A;                      %(eq.3.11)
z_A   = (z_H*m_H-r_F*m_F)/m_A;                    %(eq.3.11)

I_Axx = I_Hxx+I_Fxx+m_H*((z_H-z_A)^2) ...
    +m_F*((r_F+z_A)^2);%(eq.3.12)
I_Axz = I_Hxz-m_H*(x_H-x_A)*(z_H-z_A) ...
    +m_F*(w-x_A)*(r_F+z_A);%(eq.3.13)
I_Azz = I_Hzz+I_Fzz+m_H*((x_H-x_A)^2) ...
    +m_F*((w-x_A)^2);%(eq.3.14)

u_A   = (x_A-w-c)*cos(la)-z_A*sin(la);            %(eq.3.15)
     %L for lambda in I_ALL, I_ALx and I_ALz
I_ALL = m_A*u_A^2+I_Axx*sin(la)^2+2 ...
    *I_Axz*sin(la)*cos(la)+I_Azz*cos(la)^2;%(3.16)
I_ALx = -m_A*u_A*z_A+I_Axx*sin(la)+I_Axz*cos(la);%(eq.3.17)
I_ALz = m_A*u_A*x_A+I_Axz*sin(la)+I_Azz*cos(la);%(eq.3.18)

mu      = c*cos(la)/w;                            %(eq.3.19)

S_R     = I_Ryy/r_R;                              %(eq.3.20)
S_F     = I_Fyy/r_F;                              %(eq.3.20)
S_T     = S_R+S_F;                                %(eq.3.20)

```

```
42 S_A      = m_A*u_A+mu*m_T*x_T;                    %(eq.3.21)
43
44 M        = zeros(2);                              %(eq.3.23)
45 K0       = zeros(2);                              %(eq.3.25)
46 K2       = zeros(2);                              %(eq.3.27)
47 C1       = zeros(2);                              %(eq.3.29)
48
49 M(1,1)   = I_Txx;                                 %(eq.3.22a)
50 M(1,2)   = I_ALx+mu*I_Txz;                        %(eq.3.22a)
51 M(2,1)   = M(1,2);                               %(eq.3.22b)
52 M(2,2)   = I_ALL+2*mu*I_ALz+mu^2*I_Tzz;          %(eq.3.22b)
53
54 K0(1,1) = m_T*z_T;                               %(eq.3.24a)
55 K0(1,2) = -S_A;                                  %(eq.3.24a)
56 K0(2,1) = K0(1,2);                              %(eq.3.24b)
57 K0(2,2) = -S_A*sin(la);                          %(eq.3.24b)
58
59 K2(1,1) = 0;                                     %(eq.3.26a)
60 K2(1,2) = (S_T-m_T*z_T)*cos(la)/w;               %(eq.3.26a)
61 K2(2,1) = 0;                                     %(eq.3.26b)
62 K2(2,2) = (S_A+S_F*sin(la))*cos(la)/w;           %(eq.3.26b)
63
64 C1(1,1) = 0;                                     %(eq.3.28a)
65 C1(1,2) = mu*S_T+S_F*cos(la) ...
       +I_Txz*cos(la)/w-mu*m_T*z_T;%(eq.3.28a)
66 C1(2,1) = -(mu*S_T+S_F*cos(la));                 %(eq.3.28b)
67 C1(2,2) = I_ALz*cos(la)/w+mu*(S_A+I_Tzz*cos(la)/w);%(eq.3.28b)
68 disp('initModel OK'); save whipple g M K0 K2 C1;
69 disp('Parameters saved to whipple'); %clear;
```

A.5 Observer And LQR-Controller

```
1  %clear;
2  load whipple;
3  v = 4; save speed v;%Set propulsion speed
4  %CREATE LQR
5  A = [zeros(2)                    eye(2);
6       inv(M)*(-K0*g-(K2*(v^2)))     inv(M)*(-C1)*v];
7  B = [zeros(2,1);
8       inv(M)*[0 1]'];
9  C = [eye(4)];
10 system = ss(A, B, C, 0);
11 rank(ctrb(A,B))
12 R = [1/500];
13 Q = diag([1000
```

```
                1/500
                        1/500
                                1/500]);
[K, S, e] = lqr(system, Q,R);
save lqr K; disp('Observer & Controller Created')
%CREATE KALMAN BUCY
A = [zeros(2)                   eye(2);
     inv(M)*(-K0*g-(K2*(v^2)))    inv(M)*(-C1)*v];
B = [zeros(2,2);
     inv(M)];
C = [eye(2) zeros(2)];
system = ss(A, B, C, 0);
rank(obsv(A,C))
Q = diag([.2
                4]);%process noice
R = diag([ 0.0180   %measurement noice
                0.0115]);
[kest, L, P] = kalman(system, Q, R);
save kalman L A B C
```

Appendix B

Incremental Decode PCB Board

B.1 Incremental Decode Board Copper

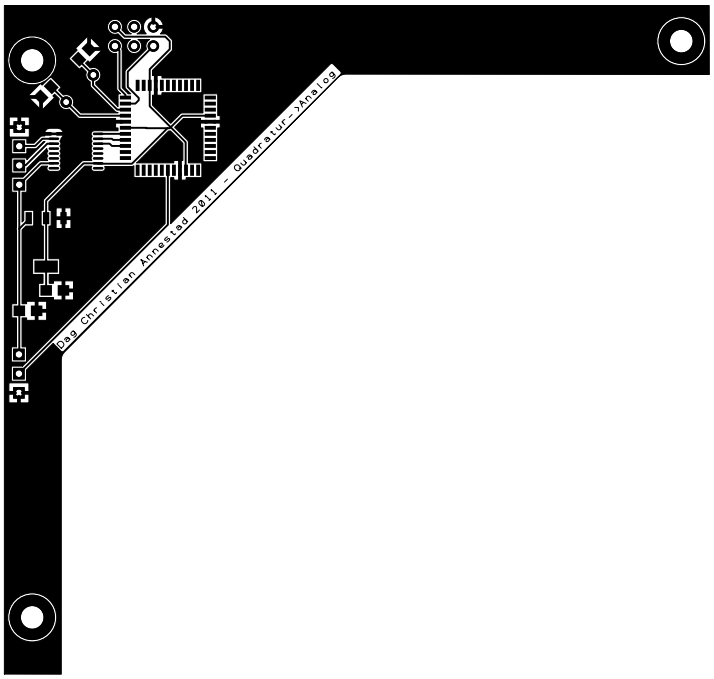


Figure B.1: Incremental decode board copper

B.2 Incremental Decode Board Schematic

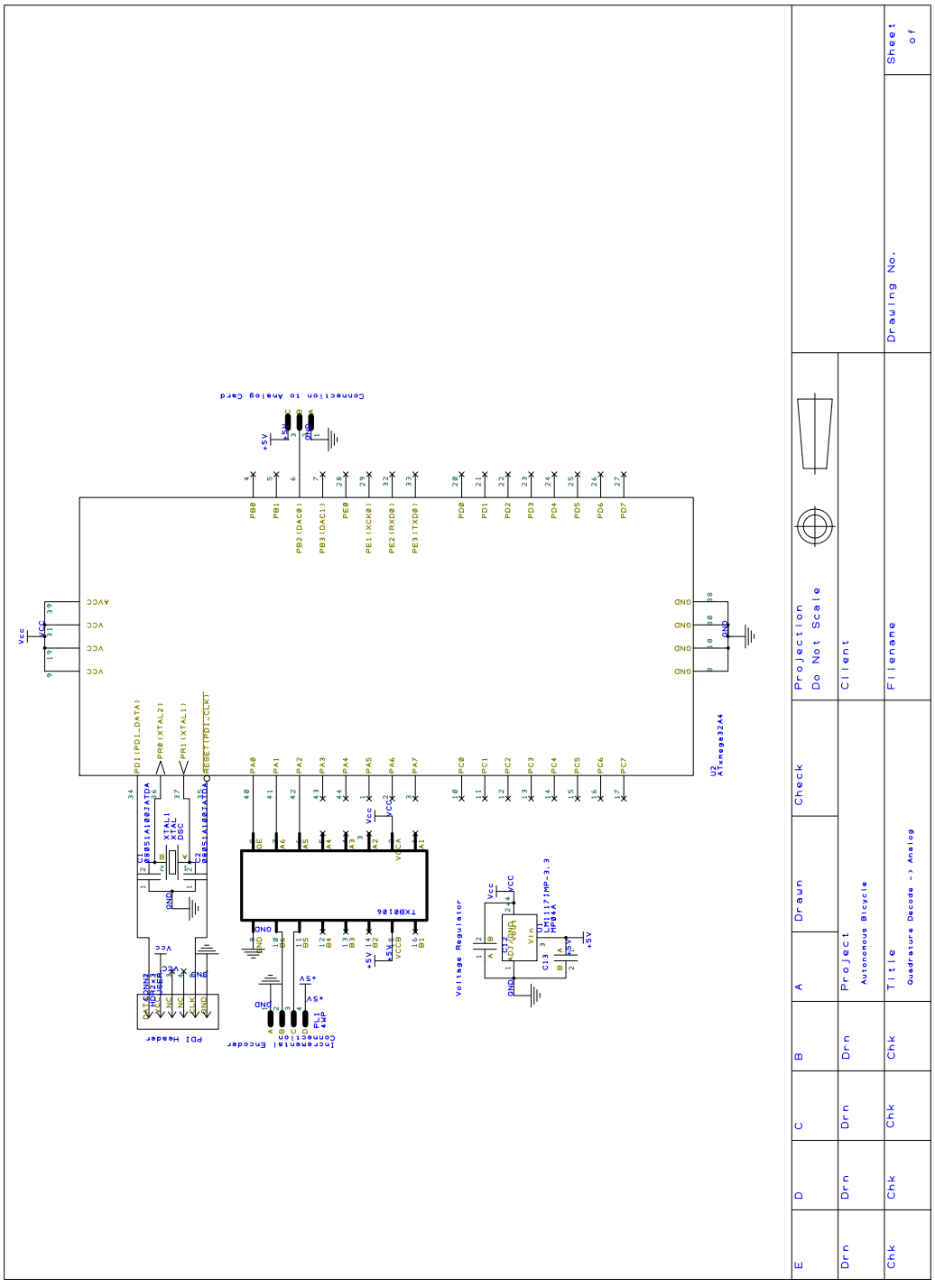


Figure B.2: Incremental decode board schematic

Appendix C

Steering Motor And Gear Data

C.1 Steering Motor

Rated voltage	24	VDC
Continuous rated speed	3000	rpm
Continuous rated torque	17	N cm
Continuous current	2.9	A
Starting torque	143	N cm
Starting current	22.8	A
No load speed	3250	rpm
No load current	0.44	A
Demagnetization current	61	A
Rotor inertia	460	g cm^2
Weight of motor	1160	g

Table C.1: Permanent magnet DC-motor GR 53x58, 60 W[18].

C.2 Steering Planetary Gear

Reduction ratio	15	
Efficiency	0.81	
Number of stages	2	
Continuous torque	800	N cm
Weight of gearbox	720	g
Axial load	500	N
Radial load	350	N

Table C.2: Planetary gearbox PLG 52[19].

C.3 Steering Transmission Gears

C.3.1 45 Teeth Spur Gear

Module	1.0	
Width	6	mm
Number of teeth	45	
Overall diameter	47	mm
Pitch diameter	45	mm
Hub diameter	18	mm
Hub length	6	mm
Bore	6	mm
Efficiency per connection	0.98	
Weight	88	g

Table C.3: Huco 45 teeth spur gear[20].

C.3.2 120 Teeth Spur Gear

Module	1.0	
Width	6	mm
Number of teeth	120	
Overall diameter	122	mm
Pitch diameter	120	mm
Hub diameter	40	mm
Hub length	12	mm
Bore	10	mm
Efficiency per connection	0.98	
Weight	674	g

Table C.4: Huco 120 teeth spur gear[20].

C.4 Total Steering Performance

Theoretical Steer Speed	75[1]	rpm
	450[2]	deg/s
Continuous Output Torque	434[3]	N cm

Table C.5: Total performance of the steer motor gear assembly

[1]Speed calculations = $\frac{\text{Continuous rated speed}}{\text{planetary gear reduction ratio}*\text{transmission gear reduction ratio}} = \frac{3000}{15*\frac{120}{45}}$

[2]Speed calculations = $\frac{\text{deg per round *rpm}}{\text{sec per min}} = \frac{360*75}{60}$

[3]Torque calculation = torque motor $*$ planetary gear reduction ratio $*$ planetary gear efficiency $*$ transmission gear reduction ratio $*$ transmission gear efficiency = $((17*15)*0.81)*(120/45)*.98$

Appendix D

How To Set Up A Host System

The following list describes how to set up a host system:

- Install MATLAB, with Simulink and RTW.
- Install QNX Software Development Platform 6.5.0.
- Install a windows terminal and file transfer interface.
- Copy the files found on the CD from "software/host_ computer_ files/-Matlab/rtw" to "MATLABROOT/rtw".
- Copy the files found on the CD from "software/host_ computer_ QNX650/target" to "QNX650ROOT/target".
- Copy the the complete folder containing the MATLAB model from the CD to the "MATLAB working directory"

Appendix E

How To Start The Bicycle

The following list describes how to start the bicycle:

- Connect a fully charged battery to the bicycle.
- Push the green power button to turn on the computer system.
- Push the red power button to turn on the motor drivers(It is important that the computer system has power when the motor drivers are turned on)..
- Release the red emergency stop button and connect the emergency stop cord.
- Connect to the wireless network ”bikenet” with the password ”JensG-BalchenWeDidIt”
- Open Putty and login with root at ”192.168.1.2”.
- Check that the MTi and analog card is transmitting data(returning a signal with noise):
  - ”cat \dev\mt\orientation\roll”.
  - ”cat \dev\dmm32at\analog\in\ad0”.
- If the MTi or analog card has failed:
  - Type ”slay devc-mt” + ”devc-mt &” and check the driver again.
  - ”slay devc-dmm32at” + ”devc-dmm32at &” and check the driver again.
- Open ”Autonomous_ Bicycle.mdl” in a new MATLAB session.

- Select desired roll angle set-point in the Simulink model
- Hit ”ctrl-b”
- Wait for a menu to pop up.
- Set IP-address to ”192.168.1.2”, select ”1” and set runtime in sec(inf for infinity)
- Wait for Putty to load and start the model on the bicycle.
- Hit ”Connect To Target” button in the Simulink model.
- Hit ”Start real-time code” to start the bicycle.
- Any model uploaded to the bicycle could also be started in standalone mode by typing ”/tmp/./MODELNAME -tf TIMEINSEC” in a terminal to start the selected model.