CHALMERS
UNIVERSITY OF TECHNOLOGY

Control design for power assisted bicycle trailer

Minimizing trailer resistance for rider through parameter and force estimation

Master's thesis in Systems, Control, and Mechatronics

Andreas Axén Krüger
Daniel Dresdner

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2024
www.chalmers.se

Master's thesis 2024

Control design for power assisted bicycle trailer

Minimizing trailer resistance for rider through parameter and force estimation

Andreas Axén Krüger

Daniel Dresdner

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Systems and Control
Automatic Control Group
Chalmers University of Technology
Gothenburg, Sweden 2024

Control design for power assisted bicycle trailer
Minimizing trailer resistance for rider through parameter and force estimation
Andreas Axén Krüger, Daniel Dresdner

© Andreas Axén Krüger, Daniel Dresdner, 2024.

Supervisor: Jonathan Lock, Elysics AB
Examiner: Torsten Wik, Automatic Control Group, Department of Electrical Engineering

Master's Thesis 2024
Department of Electrical Engineering
Division of Systems and Control
Automatic Control Group
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Cyclist towing a trailer, AI generated

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2024

Control design for power assisted bicycle trailer
Minimizing trailer resistance for rider through parameter and force estimation

Andreas Axén Krüger
Daniel Dresdner
Department of Electrical Engineering
Chalmers University of Technology

Abstract

This thesis presents the design and implementation of a controller for an electric trailer attached to a bicycle, aiming to ensure that the cyclist does not experience any additional load arising from the trailer. Our presented controller's objective is to minimize the force between the bicycle and the trailer, regardless of load variations, wind conditions, or road terrain.
The studied trailer's rigid mechanical construction results in a system that does not display any meaningful dynamics with respect to the applied wheel torque and the resultant force generated at the tow hitch. This static behavior enables the formulation of the controller as a static optimization problem. By applying Newtonian physics, we derive a motion model, leading to the development of a model-based controller capable of real-time control.
Additionally, an online parameter estimation algorithm is developed, where the gradient of the model serves as a weight to penalize the relative change of each parameter. By assigning different convergence rates during runtime to the parameters — trailer mass, rolling resistance coefficient, and wind drag coefficient — the estimation process can more accurately reflect real-world conditions.
The developed controller was evaluated against conventional control regulators, particularly proportional and integral controllers, using a set of initial parameters. The results are promising, indicating that the online parameter estimation method is effective with an model error of 12% when compared to the true sensor value.

Keywords: Parameter estimation, force control, MIMO, optimization, quadratic programming, model based.

Acknowledgements

First of all we would like to thank the entire staff of Elysics for their hospitality and generosity, allowing us to use their office space and workshop throughout the project.

Secondly, a big thank you to our examiner Torsten Wik for giving us the opportunity to work on this project and showing such a great interest.

Finally, we would especially like to thank our supervisor Jonathan Lock for sharing his valuable time and knowledge, along with many interesting discussions. It has been a tremendous help and we have learned a lot together.

Andreas Axén Krüger, Daniel Dresdner, Gothenburg, June 2024

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

DC	**D**irect **C**urrent
DoF	**D**egree **o**f **F**reedom
FIR	**F**inite **I**mpulse **R**esponse
FOC	**F**ield **O**riented **C**ontrol
IIR	**I**nfinite **I**mpulse **R**esponse
IMU	**I**nertial **M**easurement **U**nit
MCU	**M**otor **C**ontrol **U**nit
MIMO	**M**ultiple **I**nput **M**ultiple **O**utput
OSQP	**O**perator **S**plitting **Q**uadratic **S**olver
PID	**P**roportional **I**ntegral**D**erivative
PMSM	**P**ermanent **M**agnet **S**ynchronous **M**otor
PWM	**P**ulse **W**idth **M**odulation
QP	**Q**uadratic **P**rogramming
RPM	**R**evolutions **P**er **M**inute
SISO	**S**ingle **I**nput **S**ingle **O**utput
UART	**U**niversal **A**synchronous **R**eciever **T**ransmitter
WiFi	**W**ireless **F**idelity

Nomenclature

Below is the nomenclature of indices, sets, parameters, and variables that have been used throughout this thesis.

k	Index for time step
m	Trailer mass $[kg]$
μ	Rolling resistance coefficient
λ	Wind drag coefficient
$\hat{\lambda}$	Estimated Wind drag coefficient
$\hat{\mu}$	Estimated Rolling resistance coefficient
$\hat{m}$	Estimated trailer mass $[kg]$
$K_{1..m}$	Minimization weights used for control signals
$\bar{K}_{1..m}$	Minimization weights used for parameter estimation
x	Decision variable
$\dot{v}$	Linear acceleration $[m/s^2]$
v	Linear Velocity $[m/s]$
ω_t	Angular velocity $[rad/s]$
I	Inertia $[kg/m^2]$
φ	Road incline $[rad]$
b	Number of FIR filter coefficients
$\bar{C}(\dot{v}, v, \phi)$	Parameter estimation cost function
$C(\dot{v}, v, \phi)$	Controller cost function
N	Number of samples
P	Controller quadratic terms
q	Controller linear terms
$\bar{P}$	Parameter estimation quadratic terms

$\bar{q}$	Parameter estimation linear terms
K_p	Proportional gain
K_i	Integral gain
K_d	Derivative gain
$e(t)$	Error term used by PID-controller
M_t	Moment of inertia [N]
r	Reference value [N]
F_{tot}	sum of total forces [N]
$F_{straight}$	Forces acting latitudinal [N]
F_{side}	Forces acting longitudinal [N]
F_s	Force measured by the force sensor [N]
$\hat{F}_s$	Estimation of the force measured by the force sensor [N]
F_a	Force induced by acceleration [N]
F_r	Rolling resistance [N]
F_w	Wind drag [N]
F_p	Parallel forces due to road incline [N]
F_t	Tractive forces [N]

Contents

List of Acronyms **ix**

Nomenclature **xi**

List of Figures **xv**

List of Tables **xvii**

1 Introduction **1**
1.1 Ethics & Sustainability 2

2 System description **3**

3 Theory **5**
3.1 Digital signal processing 5
3.2 Quadratic problems and optimization 6
3.3 Control Theory 6
3.3.1 PID control 6

4 Method **9**
4.1 Mathematical model 9
4.2 Estimating lateral and longitudinal forces 12
4.3 Road incline estimation 13
4.4 Parameter estimation 14
4.4.1 Defining the cost function 14
4.4.1.1 Converting the cost function to a QP problem 15
4.4.2 Optimal filter length for cost function input signals 16
4.4.3 Optimal minimization weights used for cost function 17
4.5 Control design 18
4.5.1 Control using model based optimization 18
4.5.2 PI controller 19
4.6 Evaluation 19

5 Results **21**
5.1 Estimating trailer forces 21
5.2 Parameter estimation 22
5.3 Control design 24

6 Discussion **27**
6.1 Evaluation of research questions . . . 28
6.2 Future work . . . 29
6.3 Summary . . . 29

Bibliography **31**

A Appendix 1: Equations **I**
A.1 Rotation matrix . . . II
A.2 Estimation cost function . . . II
A.3 Control cost function . . . III

B Appendix 2: System description **V**
B.1 System requirements . . . VI
B.2 Hardware components . . . VI
B.2.1 Electrical circuit diagram . . . VI
B.2.2 STM32 Microcontroller . . . VIII
B.2.3 ESP32 Microcontroller . . . VIII
B.2.4 Motors . . . VIII
B.2.5 Motor Control Unit . . . IX
B.2.6 Batteries . . . IX
B.2.7 Sensor specifications . . . IX
B.2.7.1 HX711 - Force sensor . . . IX
B.2.7.2 LIS3DH - Accelerometer . . . X
B.2.8 3D printed mechanical components . . . X
B.2.8.1 Load cell housing . . . X
B.2.8.2 Dead man's grip . . . XI
B.3 Software architecture . . . XII
B.3.1 Communication overview . . . XII
B.3.1.1 UART protocol . . . XII
B.3.1.2 User communication and data logging . . . XII
B.3.1.3 Motor control with MCU communication . . . XIV
B.3.2 Optimization solver - OSQP . . . XIV
B.3.3 Main program overview . . . XIV
B.3.3.1 Control loop overview . . . XV
B.3.3.2 Fault handling . . . XVI
B.3.4 Sensor readings and calibration . . . XVII
B.3.4.1 Force sensor . . . XVII
B.3.4.2 Accelerometer . . . XVII
B.3.5 Velocity and acceleration . . . XVIII
B.4 Discussion of system requirements . . . XVIII
B.4.1 Safety requirements . . . XVIII
B.4.2 Mechanical parts for isolating sensor readings . . . XVIII
B.4.3 Communication protocols and CLI interaction . . . XIX
B.5 Future work . . . XIX

List of Figures

2.1 The complete prototype. 3
2.2 System overview including both hardware and software. 4

4.1 Simplification of the forces acting on the trailer, from an overhead view. 10
4.2 Local coordinate frame at the mass center of the trailer. 12
4.3 Measured force and the calculated estimation. A rolling average filter with 30 samples was used to reduce noise for the plot. 13
4.4 Estimated road incline. The incline between 22 and 31s is known to be 5°. 14
4.5 Comparing the output signals when using both a short and long filter. 16
4.6 Optimum for the amount of filter coefficients. 17

5.1 Measured force compared to estimated force using estimated parameters close to the true values. A rolling average filter with 30 samples was used to reduce noise and increase visibility of the results. 22
5.2 Measured force vs estimated force when encountering potholes that is not included in the model. A rolling average filter with 30 samples was used to reduce clutter and increase visibility of the results. 22
5.3 Mass estimation over time with an initial mass of 40 Kg. 23
5.4 Plot showing the estimated friction and drag coefficients over time. . 24

A.1 Force sensor readings without the mechanical mounting I
A.2 Force sensor readings with the mechanical mounting I

B.1 Circuit diagram of electrical hardware. VII
B.2 Circuit diagram of dead man's switch. VII
B.3 Hub motors mounted on the trailer. VIII
B.4 Resulting 3d printed housing for the load cell. X
B.5 Mechanical housing for the load cell, isolating translational forces and removes rotational forces. Marked numbers: 1: 3d-part(1), 2: slider with ball bearings, 3: 3d-part(2), 4: load cell, 5: bar connected to the trailer, 6: bar connected to end of bar and the hinge. XI
B.6 dead man's switch to cut the main power to the trailer. Marked numbers: 1: Switch, 2: reset pin, 3: dead mans cord, 4: switch housing. XI
B.7 Communication flow chart. XII
B.8 Command line interfaces. XIII
B.9 Main program loop. XV

B.10 Finite state machine. XVI

List of Tables

4.1 List of initial parameters used. 12
4.2 Calculated cost function weights. 18
4.3 Test cases used for evaluation. 19

5.1 Parameter values close to the true values. 21
5.2 Parameter values close to the true values. 23
5.3 Root mean square(RMS) of measured force with different controllers and the reduction in force between using no controller and one of the controllers. 24

B.1 Subsystems and operating area. V
B.2 Motor specification. IX
B.3 Force sensor specifications. IX
B.4 Accelerometer specifications. X
B.5 Table of MCU commands. XIV

1
Introduction

Electric bicycles are a great option for commuting, offering an eco-friendly and efficient alternative to traditional vehicles, such as cars or public transport. However, their cargo capacities are limited in comparison. A recent field study highlights this as one of the factors limiting bicycle use and emphasizes the desire for a practical solution [7]. Notably, when participants were asked about the potential of an electrically powered trailer, the majority indicated a willingness to invest in such a solution. The addition of an electric bicycle trailer[1] benefits from making bicycles a more versatile and attractive option for a wider range of users, thereby providing sustainable commuting solutions.

Numerous studies have focused on control theory with pedal-assisted bicycles, typically providing feedback from the torque generated at the pedals [6][2]. However, the complexity increases when a trailer is attached to a bicycle, which is not adequately addressed by existing studies that focus solely on a bicycle without a trailer.
Additionally, several studies have addressed trailer/vehicle dynamics for vehicles such as cars and trucks, investigating parameter estimation and trailer motion control. Although the results are promising, these studies often rely on optical sensors, expensive equipment, or machine learning algorithms to estimate dynamic parameters like trailer mass [8][19]. These methods are not cost effective for the scale and complexity of this application.

The aim of this thesis was to develop a prototype and design a controller for a power-assisted trailer that is attached to bicycles. The trailer is suppose to compensate for the load regardless of its weight or the environment. Therefore, an accurate online estimation of the mass, rolling resistance coefficient and aerodynamic drag coefficient was investigated. This, in turn, allowed the estimation of the force required to counteract the resistance forces encountered when towing a bicycle trailer. The project consisted of developing, testing and evaluating the hardware and software required to handle communication, control and online parameter estimation.

[1]The term trailer is generic and could for example imply either a cargo trailer or a trailer for transportation of children. The goal of the prototype is to be generic enough to generate results that are applicable for any kind of bicycle trailer.

The following research questions was stated:

- **Question 1:** Is it advantageous to state the control problem as an online optimization problem in comparison to using a more conventional PID controller?

- **Question 2:** The model parameters of the trailer will have parameters that change over time at different rates. Can these parameters be estimated online and how should the different convergence rates be handled?
- **Question 3:** What sensors are needed for the control of the trailer to be effective and how will removing sensors affect the result?

To verify the system a physical trailer was used with a non-electric bicycle as a test system where different controllers and estimation methods are evaluated.

With the scope of the project set to the construction of a prototype, limitations of the project were defined as:

- Limited to one specific bicycle hinge.
- Road curbs are not modeled.
- Safety of the system is designed for a maximum velocity of 24 km/h.
- Though the model supports going in reverse, this is not part of the verification.
- Stopping or parking at a non-zero incline area is not considered.
- Capacity, weight, size of components and mounting are not taken into consideration.

1.1 Ethics & Sustainability

The importance of sustainable transportation raises the question of increasing the appeal of bicycles and their current lack of storage space. With an attachable trailer, that lack of space is significantly improved. By putting the electronics and power management system on the trailer instead of the bicycle we open up the idea of an electric trailer that can be used by non electric bikes and shared by many users. This reduces the component and battery consumption while still retaining the effect of renewable energy resources as well as the increased capacity of packing space.
Since this prototype will be operating in traffic, safety aspects have been taken into consideration. Ensuring safety during the testing and debugging phases is also crucial for the developers, as it not only protects the team from potential hazards but also ensures the reliability and integrity of the prototype.

2
System description

A significant part of the project has involved designing and implementing both hardware and software components into an operational system (see Figure 2.1).

Figure 2.1: The complete prototype.

Although not essential for understanding the main content of this report, a brief description of the system (see Figure 2.2) is needed to understand the context of the work. A simplified explanation of the system is given outlining the system architecture. The system consist of two microcontrollers asynchronously working together using a fixed deterministic period time of 15 ms. The first microcontroller, ESP32, handles the WiFi communication as well as data logging. Estimation, control and fault handling is done by the second microcontroller, STM32. Sensors are mounted on the trailer and measurements are processed by the STM32 microcontroller before motor commands are sent to the control units. As a safety precaution for the developers a dead man's switch was constructed and added to the trailer to shutdown the system during testing.

An in-depth technical overview is given in the appendix (see Appendix B) and it serves two purposes. First, it is intended for readers with a technical interest in the

project, offering explanations of the design, implementation, and operation of the trailer. Second, it serves as a resource for future developers who may continue work on this project.

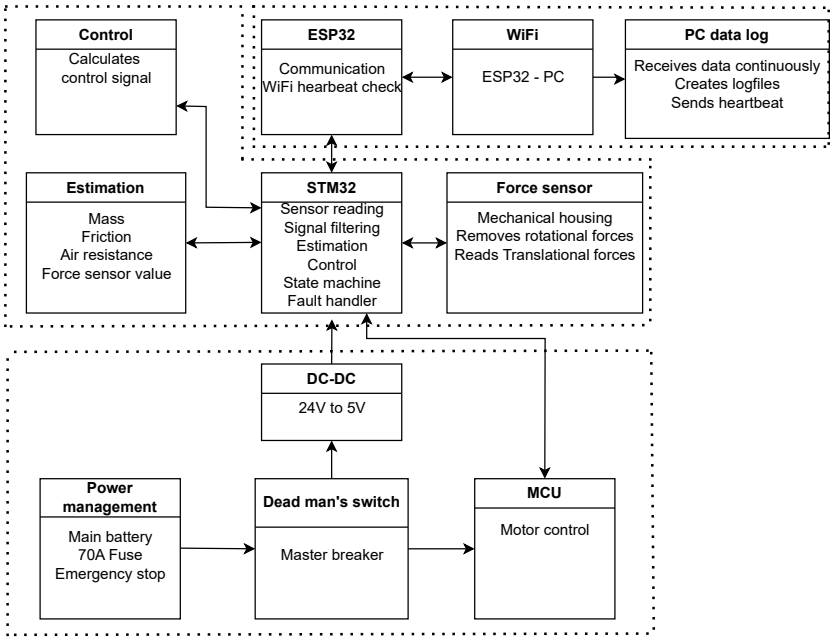


Figure 2.2: System overview including both hardware and software.

3
Theory

In this section we present a brief introduction to concepts and theories used in later chapters of the thesis.

3.1 Digital signal processing

The following section explains the basic theory of filters and how to implement digital filtering in embedded systems. A digital filter can be used to obtain an enhanced signal by processing a sampled discrete-time signal. Coefficients used by a digital filter can be symmetric, meaning they are mirrored around their center point. Linear discrete-time filters can be divided into two categories, finite impulse response (FIR) filters and infinite impulse response (IIR) filters. FIR filters only make use of the signals input values and are also known as a nonrecursive filter, while IIR filters also use previous output values and are therefore known as recursive filters [10]. Recursive filters have the benefit of needing fewer coefficients and are faster to execute than nonrecursive filters while achieving similar results. However, due to potential floating point inaccuracies recursive filters run the risk of becoming unstable. Nonrecursive filters, on the other hand, are guaranteed to be stable and, with symmetrical coefficients, have a linear phase response.
The corresponding transfer function of a linear time-invariant filters can be expressed using the *Z*-transform as

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + ... + b_N z^{-N}}{1 + a_1 z^{-1} + a_2 z^{-2} + ... + a_M z^{-M}}, \tag{3.1}$$

where $H(z)$ is the transfer function, $X(z)$ is the input and $Y(z)$ the output. Equation 3.1 describes a recursive filter, but if the denominator is set to one, a nonrecursive filter is obtained. The order of the filter is defined as $max(N, M)$. Using the inverse *Z*-transform, the following difference equation can be derived from Equation (3.1):

$$y[k] = -\sum_{k=1}^{M} a_k y[n-k] + \sum_{k=0}^{N} b_k x[n-k]. \tag{3.2}$$

From Equation (3.2) we can see that by only taking the sum of the product of the filter coefficients, b_k, and the input signals we obtain a realization of a FIR filter and by including the feedback outputs an IIR filter is obtained.
When using a nonrecursive filter with symmetric coefficients a sample delay is always

introduced. This delay is given by

$$\text{sample delay} = \frac{b+1}{2}, \tag{3.3}$$

where b is the number of coefficients that make up the filter. The nature of symmetric nonrecursive filters having constant phase implies that as long as all signals have been filtered using the same number of coefficients they will be delayed by the same number of samples.

3.2 Quadratic problems and optimization

We introduce $\underset{x}{\text{argmin}} f(x)$ as an optimization problem where f is referred to as the *cost function* and $x \in \mathbb{R}^n$ as the *decision variable*.
The goal of optimization is to find the global minima of the cost function. Typically a scalar fitness function is introduced to evaluate how good a solution is, based on a metric that is appropriate for the given problem [12]. Often, constraints are placed on x, limiting its range. These constraints are often entailed by physical limitations.

In this thesis every cost function is quadratic, which is numerically convenient as they are always convex and thus numerically tractable.

A quadratic problem is defined to be of the form

$$\underset{x}{\text{argmin}} \quad \frac{1}{2}\boldsymbol{x^T P x} + \boldsymbol{q^T x} \tag{3.4}$$

$$\text{subject to} \quad \boldsymbol{l} \leq \boldsymbol{Ax} \leq \boldsymbol{u} \tag{3.5}$$

where $\boldsymbol{x} \in \mathbb{R}^{\boldsymbol{n}}$ is the decision variable [5]. The objective function is here defined by a positive semi-definite matrix $\boldsymbol{P} \in \mathbb{S}_+^{\boldsymbol{n}}$ and the column vector $\boldsymbol{q} \in \mathbb{R}^{\boldsymbol{n}}$. The linear constraints are defined by a matrix $\boldsymbol{A} \in \mathbb{R}^{\boldsymbol{m \times n}}$ and vectors $\boldsymbol{l}$ and $\boldsymbol{u}$, that correspond to the lower and upper constraints. An equality constraint can be constructed for an element x_k by setting $\boldsymbol{l} = \boldsymbol{u}$ [15].
This theory is implemented in Sections 4.4 and 4.5.1.

3.3 Control Theory

3.3.1 PID control

A common control algorithm is the proportional, integral and derivative (PID) controller. This simple controller is often effective and and relies on the measured error between a reference value and an measured output [9]. The PID controller works by creating an error term, $e(t)$, and using the three parts of the PID, proportional gain, integral gain and derivative gain applied to the error term, to calculate a control signal

$$u(t) = K_p \cdot e(t) + K_i \cdot \int_{t_0}^{t} e(t)dt + K_d \cdot \frac{de(t)}{dt}. \tag{3.6}$$

Here the proportional gain K_p utilizes the magnitude of the error, the integral gain K_i acts on the cumulative error and the derivative gain K_d is applied to the rate of change of $e(t)$.
Potential drawbacks of the PID controller are that it does not directly handle multiple-input-multiple-output (MIMO) systems and non-linearities.

4
Method

This section will cover the method used to derive the models used for the control design, parameter estimation and simulations.

- Firstly a mathematical model describing the forces of the trailer is presented.
- Then, the force sensor measurement and road incline are discussed, which will act as the base for the remaining work.
- An approach for estimating time varying parameters, such as trailer mass and road friction is then shown.
- Lastly, the control design is presented.

4.1 Mathematical model

A mathematical model is necessary to describe the behaviour of the system and in turn enable a model-based control design. The aim is to develop a model that describes the resulting forces acting on the trailer and its behaviour accordingly. The model has to be computationally fast enough for online execution and at the same time sufficiently accurate for a model-based control.

We model the trailer's motion using Newtonian physics, investigating the forces acting on it. Tractive forces (F_t) propel the vehicle forward while resistive forces ($F_{resistance}$) resist the motion, giving the total force

$$F_{tot} = \sum F_t - \sum F_{resistance}. \tag{4.1}$$

$F_{resistance}$ is defined as the sum of rolling resistance (F_r), aerodynamic drag (F_w), parallel force (F_p) and acceleration resistance (F_a).

- Rolling resistance is an effect of tyre material, ground and weight, contact and is determined by a friction coefficient (μ) in combination with the road incline($\widetilde{\varphi}$)
$$F_r = \mu m g \cos(\varphi) \tag{4.2}$$
where m is the trailer mass and g is the gravitational constant.
- Aerodynamic drag is defined by the affected area perpendicular to the direction of motion (A_f), the air density (ρ), and the drag coefficient (C_d), and is proportional to the velocity squared (v^2). For simplicity, the constant drag terms, i.e. $\frac{1}{2}\rho C_d A_f$, are lumped together to form λ.
$$F_w = \frac{1}{2}\rho C_d A_f v^2 = \lambda v^2. \tag{4.3}$$

- The force parallel to the ground is determined by the mass (m) and incline of the ground (φ), of which it is parallel,

$$F_p = mg \sin(\varphi). \tag{4.4}$$

- Inertial forces is determined by the mass (m) and the acceleration (a) of the object, i.e.

$$F_a = ma. \tag{4.5}$$

F_{tot} can be divided into two components, $F_{straight}$ and F_{side} (see Figure 4.1). $F_{straight}$ is given by combining Equations (4.1) to (4.5) to get

$$F_{straight} = F_t - F_w - F_p - F_r - F_a, \tag{4.6}$$

where the tractive force F_t is the sum of the left and right wheel contributions, F_{wl} and F_{wr}, respectively. The side force acting on the system can be determined by analyzing the moment of inertia (M_t) applied to the trailer around its center of mass, and converting the torque generated to a force applied to the hitch. Doing this gives the relation

$$M_t = \ell F_{side} \tag{4.7}$$

where ℓ defines the length from the mass center to the hitch connection.

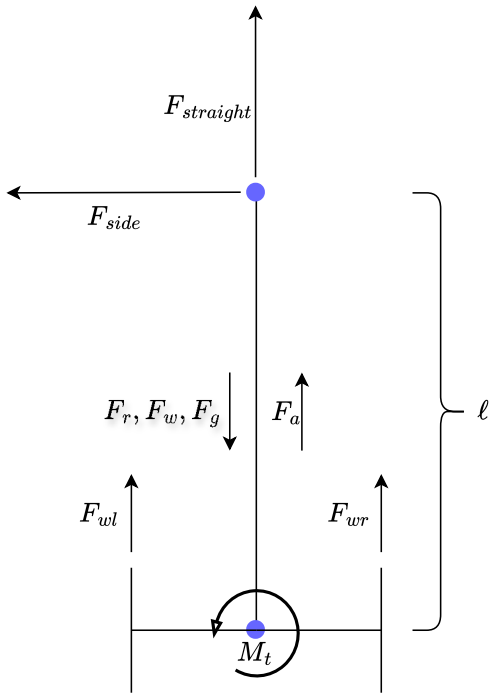


Figure 4.1: Simplification of the forces acting on the trailer, from an overhead view.

By combining Equation (4.6) and rearranging Equation (4.7) the model is given by

$$F_{straight} = F_{wl} + F_{wr} - F_a - F_w - F_p - F_r \tag{4.8}$$

Equation (4.8) describes the force in the direction of motion when an external force is applied. This corresponds to the trailer being towed. By adding the torque casued by the forces (F_{wl} and F_{wr}) we can fully describe also the lateral forces. This results in

$$F_{side} = \frac{M_t}{\ell} - \frac{d}{2\ell}(F_{wr} - F_{wl}) \tag{4.9}$$

Examining Equations (4.8) and (4.9) reveals that if all the forces are known, we can obtain both $F_{straight}$ and F_{side} immediately, without any dynamics.

For testing and simulation, a default set of constant parameters are used (see Table 4.1). The mass was measured using a scale. The drag coefficient was estimated by assuming the trailer to be a rectangular shape and approximating the frontal area. The rolling resistance coefficient was determined by pulling the trailer on a flat surface and read the force measured by the sensor.

Table 4.1: List of initial parameters used.

Mass, (m)	55 $[kg]$
Rolling resistance coefficient, (μ)	0.03 [-]
Drag coefficient, (λ)	0.15 [-]

4.2 Estimating lateral and longitudinal forces

A local coordinate frame following the convention of roll, pitch and yaw is introduced at the mass center (MC) of the trailer (see Figure 4.2), where the direction of forward motion is defined along the x-axis. By having mounted the force sensor (F_s) onto the mechanical mounting system, described in Section B.2.8, along the trailer connection arm, all forces along the x-axis are isolated and picked up by F_s.

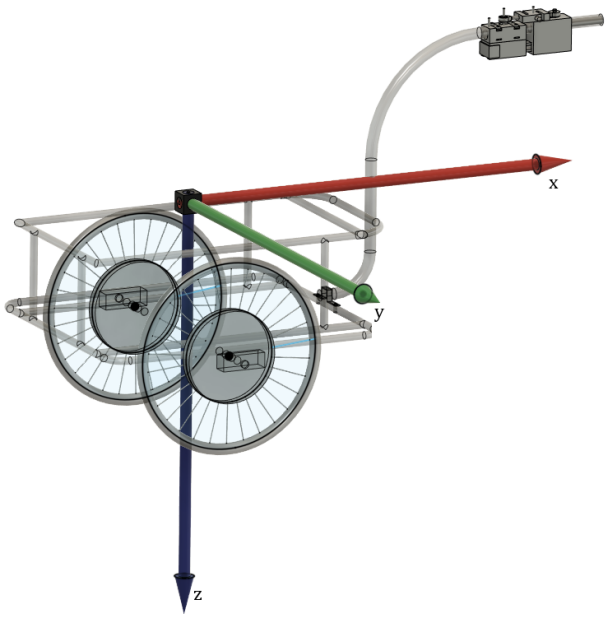


Figure 4.2: Local coordinate frame at the mass center of the trailer.

By using Equation (4.8) and Equation (4.9) an estimation of F_s and F_{side} is given by

$$\hat{F}_s = F_{wl} + F_{wr} - F_a - W_r - F_w - F_g \tag{4.10}$$

$$\hat{F}_{side} = \frac{M_t}{\ell} - \frac{d}{2\ell}(F_{wr} - F_{wl}) \tag{4.11}$$

where $\hat{F}_s$ is the estimated force applied along the x-axis on the true force sensor and $\hat{F}_{side}$ is the force applied sideways. To verify the accuracy of the force estimation, a test was performed on the physical system where both the measured force and the estimated force data were logged (see Figure 4.3).

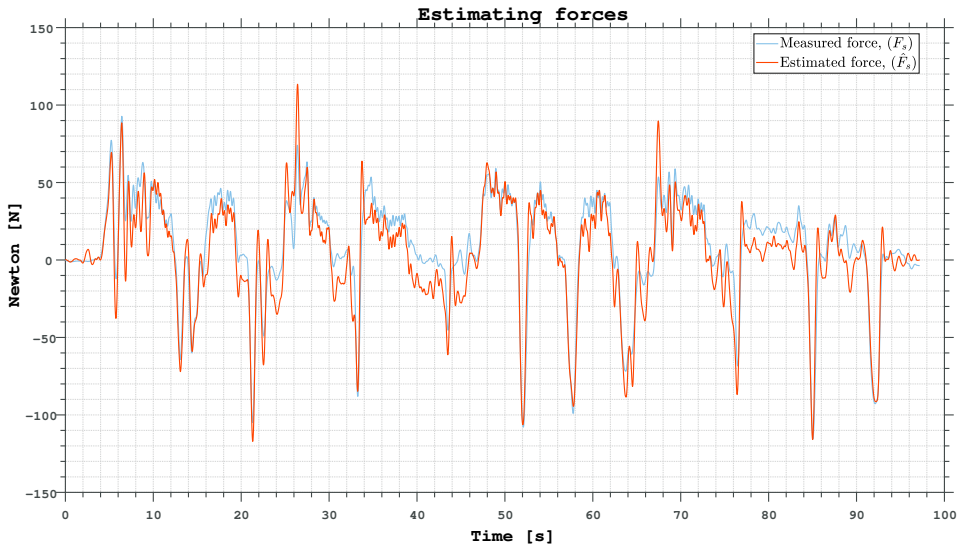


Figure 4.3: Measured force and the calculated estimation. A rolling average filter with 30 samples was used to reduce noise for the plot.

4.3 Road incline estimation

The complete model utilises the angle of the road's incline. Using the accelerometer mounted on the trailer, values of acceleration in the x, y and z directions are acquired. Using the data acquired from the accelerometer in combination with the linear acceleration along the direction of motion obtained by differentiating the trailer velocity as reported by the motor controllers, we can calculate the pitch (φ) of the trailer and in turn estimate road incline as

$$\varphi = \arctan\left(\frac{A_x}{\sqrt{A_y^2 + A_z^2}}\right). \tag{4.12}$$

Here A_x, A_y and A_z are the rotated accelerometer readings (see Appendix A.1 for details). The accelerometer data is filtered using a low-pass FIR filter with 85 taps to smooth the data and to isolate road incline from speed bumps. The estimated road incline was verified by moving the trailer along a road with a known incline of 5 degrees (see Figure 4.4). Verification of the estimator shows that the incline of the road can be accurately estimated.

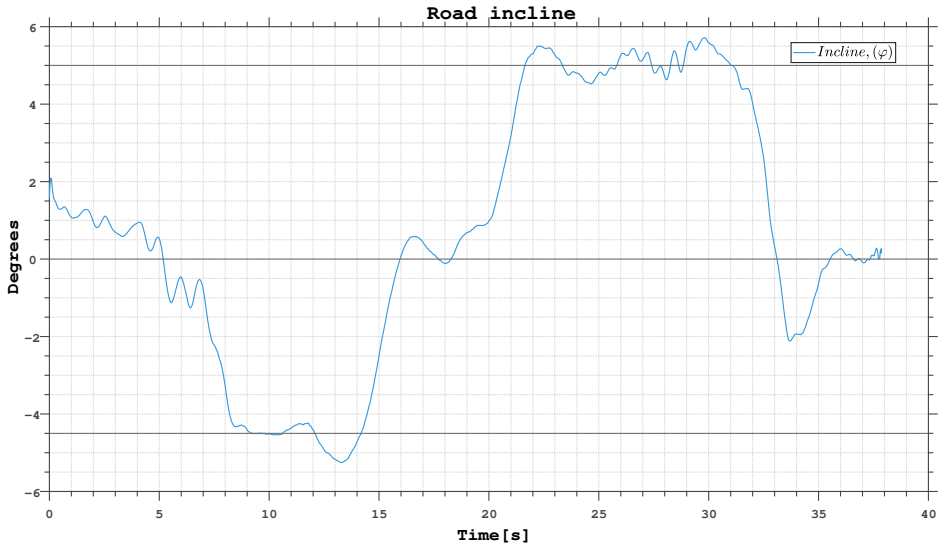


Figure 4.4: Estimated road incline. The incline between 22 and 31s is known to be 5°.

4.4 Parameter estimation

The force sensor measurements are used to estimate the time-varying parameters mass, wind drag coefficient and the rolling resistance coefficient. This is achieved by comparing the error (e) between the force measurement and the estimated force, defined as

$$e = F_s - \hat{F}_s. \tag{4.13}$$

The error squared serves as a basis for a cost function with acceleration, velocity and road incline as input and parameter estimates as the optimal output.
The inputs are filtered to reduce noise and improve performance of the estimator. To guarantee linear phase and the same amount of time delay for all signals a symmetric FIR filter was used. The optimal amount of filter coefficients that improves performance without using an unnecessary large number of coefficients was calculated (see Section 4.4.2).

4.4.1 Defining the cost function

A cost function is expressed as a first step to pose the parameter estimation as an optimization problem. The used cost function is defined by

$$\min_{\{m,\lambda,\mu\}} \bar{C} = \min_{\{m,\lambda,\mu\}} \bar{K}_1(F_s - \hat{F}_s)^2 + \bar{K}_2(\dot{m})^2 + \bar{K}_3(\dot{\lambda})^2 + \bar{K}_4(\dot{\mu})^2, \tag{4.14}$$

where

$$\dot{m} = \frac{m_k - m_{k-1}}{2}, \quad k = \text{current time step}. \tag{4.15}$$

Note that $\dot{\lambda}$ and $\dot{\mu}$ are defined analogously to $\dot{m}$. $\bar{K}_1, \bar{K}_2, \bar{K}_3, \bar{K}_4$ are penalty weights.

The first term compares the measured force with the estimated force, defined in Equation (4.10), and relates the error to the model parameters. To prevent overfitting and unstable parameters, penalty terms are introduced for each parameter's derivative. The weights of these terms determine how fast a parameter can change. Wind resistance and road friction are assumed to be rapidly changing parameters, as they depend on weather conditions and the road surface. The trailer mass on the other hand is assumed to be a slowly changing parameter, as it is unlikely to change during the movement of the trailer, which suggests a relatively high weight cost. However, an excessive weight would not allow any kind of variance. A method of changing the penalty over time to switch focus from the mass, which is fixed during a run, towards the other parameters such as the rolling friction coefficient was not implemented but suggested as an improvement. This is proposed in Section (6.2).

By changing the value of a weight, one can essentially pause and unpause the estimation of the corresponding parameter during runtime. The desired behaviour is to estimate the parameters as often as possible. At the same time, it should only be valid to estimate a specific parameter when available information is contained within the relevant term of $\hat{F}_s$. The gradient of $\hat{F}_s$,

$$\frac{\partial \hat{F}_s}{\partial(m,\lambda,\mu)} = \begin{bmatrix} \frac{d\hat{F}_s}{dm} \\ \frac{d\hat{F}_s}{d\lambda} \\ \frac{d\hat{F}_s}{d\mu} \end{bmatrix} = \begin{bmatrix} \dot{v} + g(\bar{\mu}\cos(\varphi) + \sin(\varphi)) \\ v^2 \\ g\bar{m}\cos(\varphi) \end{bmatrix}, \tag{4.16}$$

is used to further enhance the effect of estimating a specific parameter, stopping the estimation when the corresponding term approaches zero. Note that $\bar{m}$ and $\bar{\mu}$ are defined as the previously estimated parameter.

The desired behaviour can be achieved by taking the gradient of $\hat{F}_s$ and multiplying each corresponding term of $\bar{C}$ in 4.14 by the corresponding inverse of the gradient. This results in the final weight vector $\bar{\mathbf{K}}$ defined as

$$\bar{\mathbf{K}} = \begin{bmatrix} \bar{K}_1 \\ \bar{K}_2 \\ \bar{K}_3 \\ \bar{K}_4 \end{bmatrix} = \begin{bmatrix} \bar{W}_e & \frac{\bar{W}_m}{\dot{v}+g(\bar{\mu}\cos(\varphi)+\sin(\varphi))} & \frac{\bar{W}_\lambda}{v^2} & \frac{\bar{W}_\mu}{g\bar{m}\cos(\varphi)} \end{bmatrix}^T, \tag{4.17}$$

where $\bar{\mathbf{W}} = \begin{bmatrix} \bar{W}_e & \bar{W}_m & \bar{W}_\lambda & \bar{W}_\mu \end{bmatrix}^T$ are manual tuning weights.

Low values of $\bar{K}_i, i = 2,3,4$, during runtime will result in fast, but possibly jumpy, estimates of the corresponding parameter. This also allows for independent convergence rates for each parameter, e.g., a slow mass update or a fast adaptation of the rolling resistance coefficient.

4.4.1.1 Converting the cost function to a QP problem

To find an optimal solution to Equation (4.14), the cost function can be formulated as a quadratic programming problem (QP) for which there are established algorithms. Here, the Operator Splitting Quadratic Programming (OSQP) solver [14] is

used due to its efficiency on embedded systems, handling of floating point numbers and its ability to detect if the problem is non convex. To rewrite the cost function as a QP problem, the following parameter vector is used

$$\boldsymbol{x} = \begin{bmatrix} m \\ \lambda \\ \mu \end{bmatrix}.$$

By expanding the squares in $\bar{C}$ (see Equation (4.14)) and then gathering the quadratic and linear terms with respect to $\boldsymbol{x}$, the resulting matrix $\bar{\boldsymbol{P}}$ and vector $\bar{\boldsymbol{q}}$ are obtained, see Section 3.2. They are presented in Appendix A.2.

4.4.2 Optimal filter length for cost function input signals

To achieve a good parameter estimation, it is essential to minimize measurement noise from uneven road surfaces and force sensor spikes by filtering input signals. A larger number of filter coefficients is required when the required level of noise reduction increases. One disadvantage of using more coefficients is that they introduce a delay in the resulting signal. Figure 4.5 shows the result of filtering the force sensor data using both a long and a short filter. The blue line represents data filtered with fewer coefficients, while the red line represents data filtered with more coefficients. Notice that the red signal is a lot smoother, but horizontally shifted to the right. When estimating parameters, however, the resulting delay has no direct consequence, unlike when calculating control signals where quick responses are crucial.

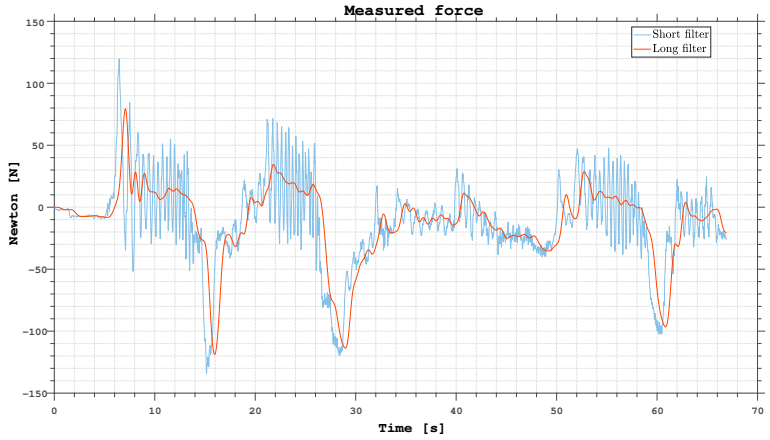


Figure 4.5: Comparing the output signals when using both a short and long filter.

Pattern search (also known as direct search) is used as the optimization algorithm. It requires a fitness function in order to be able to compare each iteration of the optimization sequence. It explores the search space by evaluating the function at a set of points around the current position and moves towards the point with the best value. This is repeated until it converges to an optimal solution [3][13]. The scalar fitness function f is chosen as the mean squared error between the measured force signal F_s and the estimated force $\hat{F}_s$, i.e.,

$$f = \sum_{i=1}^{N} \frac{(F_{s,i} - \hat{F_{s,i}})^2}{N} \tag{4.18}$$

using the fixed parameters in Table 4.1, N is the number of samples for a data set. The optimization is done offline with previously captured data. By using the same data set and treating the number of filter coefficients as a decision variable, the fitness values can be compared to determine the optimal number of filter coefficients (see Figure 4.6).

Given the results presented in Figure 4.6, it was decided that 85 taps should be used for the estimation filters, which entails a delay of 0,645 seconds at a system time period of 15 milliseconds. By matching the filter order when designing the FIR filter used for velocity, acceleration, road incline and force sensor readings, all input signals have the same time delay.

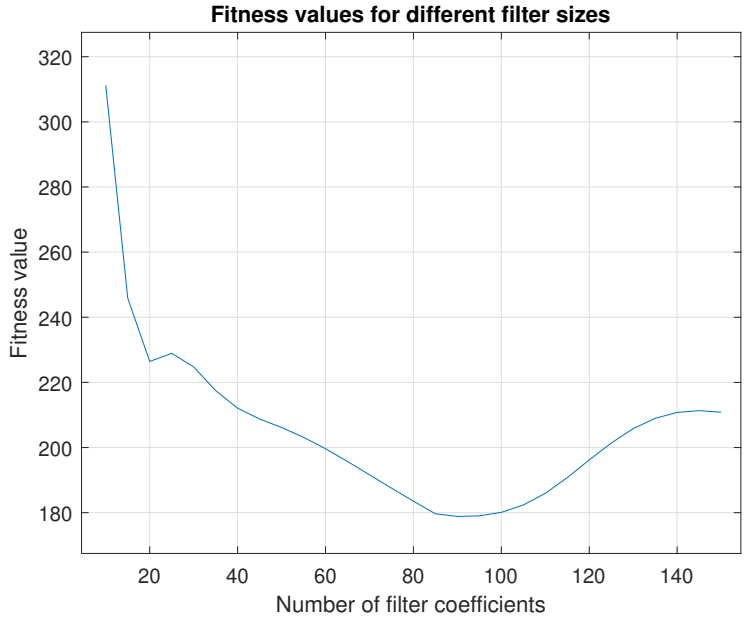


Figure 4.6: Optimum for the amount of filter coefficients.

4.4.3 Optimal minimization weights used for cost function

Pattern search is used offline to determine the optimal weights $\bar{\mathbf{W}}$ (described in Section 4.4.1) used by the cost function $\bar{C}$ (see Equation (4.14)). By using the FIR filter obtained in the previous section to process previously logged data. The data was collected with the intention to include as much information as possible in order to excite all terms in $\hat{F}_s$, and give a fair estimation of all parameters.
This involves:

- Constant acceleration.
- Sudden deceleration.
- Constant velocity.
- Zero velocity.

- Positive road incline.
- Negative road incline.
- Zero road incline.

The fitness function (see Equation (4.13)) is used in the pattern search, optimizing the weights $\bar{\mathbf{W}}$, holding $\bar{\mathbf{W}}$ constant for a complete set of logged data, and repeating the process until the fitness function converges. See Table 4.2 for results determined by the offline pattern search and later used by the online parameter estimation algorithm.

Table 4.2: Calculated cost function weights.

Weight	Value
$\bar{W}_e$	3.701e-03
$\bar{W}_m$	1000
$\bar{W}_\lambda$	210.0
$\bar{W}_\mu$	160.0

4.5 Control design

When designing the controller for the trailer, it is crucial for the signals to have a small time delay, contrary to when estimating the parameters, where we could afford a large time delay (see Section 4.4). All signals are therefore filtered using a low-order, 14-taps, low-pass FIR filter to minimize the time delay.

4.5.1 Control using model based optimization

The quadratic cost function for the control signals is formulated as

$$\min_{\{F_{wl},F_{wr}\}} C = \min_{\{F_{wl},F_{wr}\}} k_1(\hat{F}_s)^2 + k_2(\hat{F}_{side})^2 + k_3(\dot{F}_{wl})^2 + k_4(\dot{F}_{wr})^2, \tag{4.19}$$

where the derivative terms are defined in the same way as Equation (4.15).
The optimization problem aims to minimize the estimated forces given in Equations (4.10) and (4.11) by optimizing with respect to the control signals. To prevent oscillatory behaviour of the motors, penalty weights k_3 and k_4 are introduced to the rate of change of the control signals. By tuning these weights, the response time of the control signals can be altered. This approach encourages smoother transitions in the control signals, contributing to a system with less jerk.
By choosing the vector

$$\boldsymbol{x} = \begin{bmatrix} F_{wr} \\ F_{wl} \end{bmatrix}$$

and repeating the steps in Section 4.4.1.1 on Equation (4.19) one gets the matrix $\boldsymbol{P}$ and the vector $\boldsymbol{q}$, which are presented in Appendix A.3. The resulting controller will be referred to as the $\hat{F}_s$-controller.

4.5.2 PI controller

Simple PI controllers are implemented as a way to benchmark the controller design. The error term is created from force sensor measurements and a reference, $r = 0$. Two variations are implemented where the first one is the PI-controller

$$u(t) = K_p \cdot e(t) + K_i \int_{t_0}^{t} e(t)dt \tag{4.20}$$

and the second one is the P-controller

$$u(t) = K_p e(t). \tag{4.21}$$

When tuning the controllers pattern search is used offline on logged data to find the tuning parameters K_p and K_i that globally minimize the resulting squared error defined as the difference between the force measurement and the reference.

4.6 Evaluation

To experimentally evaluate the parameter estimation and the controller, the trailer was coupled to a non-electric cycle that performed a set of predetermined operations (see Table 4.3). The parameter estimation and controller was first evaluated separately, and then simultaneously. The slope of the ground was included in the tests, as well as disturbances such as debris, road bumps and uneven ground. The evaluation is presented in Chapter 5.

Table 4.3: Test cases used for evaluation.

Test case	Operations
1	1: Flat ground straight path acceleration. 2: Constant velocity. 3: Braking.
2	1: Flat ground straight path acceleration. 2: Downhill. 3: Flat ground straight path constant velocity. 4: Uphill. 5: Braking.

5

Results

This section presents and analyzes the results of the thesis, followed by a discussion of the performance. The aim is to provide an understanding of the results and offer insights into the findings.

5.1 Estimating trailer forces

Tests were performed according to Table 4.3, collecting data for both the measured force (F_s) and the estimated force ($\hat{F}_s$) (as shown in Figure 5.1). In order to focus exclusively on the accuracy of the estimated forces and minimize possible interference from other subsystems, no active controller was used during these tests. When estimating the force, fixed model parameters close to their true value was selected, as can be seen in Table 5.1.

Table 5.1: Parameter values close to the true values.

Parameter	value
Mass	55 [Kg]
Friction coefficient	0.03
Drag coefficient	0.15

Calculating the error

$$\frac{(F_s - \hat{F}_s)_{[RMS]}}{F_{s[RMS]}} = 0.12$$

shows that the estimated force tracks the measured force with a relative RMS error of 12%, which implies that the mathematical model captures the majority of the forces experienced by the trailer.

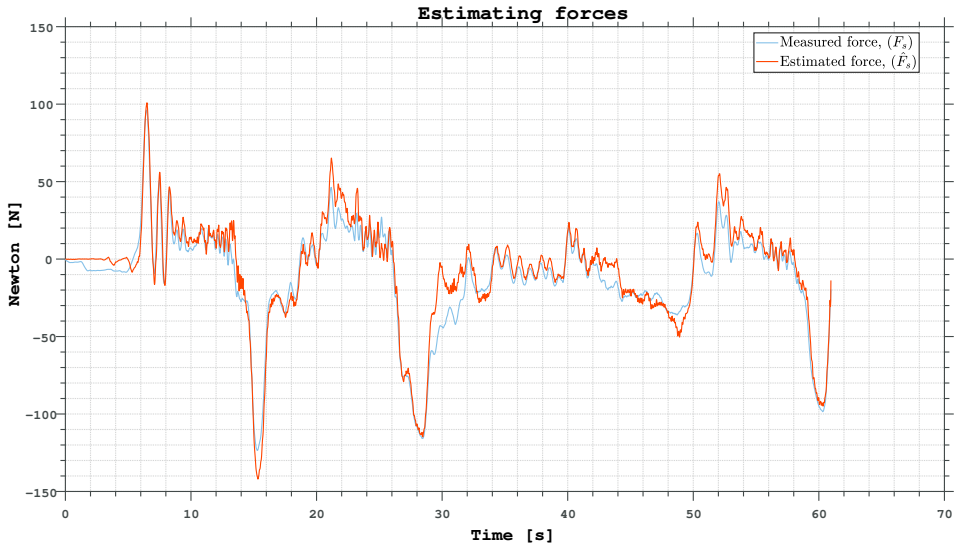


Figure 5.1: Measured force compared to estimated force using estimated parameters close to the true values. A rolling average filter with 30 samples was used to reduce noise and increase visibility of the results.

The physical system was tested on paved bicycle paths where potholes and road bumps were noted and logged. Although the bicycle paths were mostly smooth with occasional gravel, the potholes were large enough for the trailer to drive into them with at least one wheel (see Figure 5.2). This result clearly shows that the model lacks the ability to handle potholes, curbs or speed bumps.

Figure 5.2: Measured force vs estimated force when encountering potholes that is not included in the model. A rolling average filter with 30 samples was used to reduce clutter and increase visibility of the results.

5.2 Parameter estimation

The following results show the resulting model parameters over time while the parameter estimation is performed in real time for a set of initial values (see Table 5.2).

As in the previous tests, no active controller was used in this case to ensure the accuracy of the parameter estimation without interference from other subsystems.

Table 5.2: Parameter values close to the true values.

Parameter	Initial value	True value
Mass	40 [Kg]	55 [Kg]
Friction coefficient	0.01	Changes depending on environment. Typical value range [0.004, 0.05]
Drag coefficient	0.15	Changes depending on environment. Typical value range [0.01, 0.5]

In Figure 5.3, the estimated mass over time can be observed when performing a set of accelerations and braking as well as keeping the system at constant velocity.

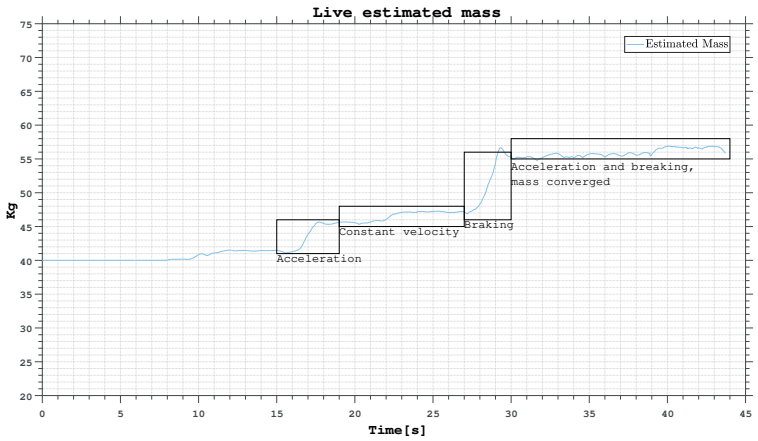


Figure 5.3: Mass estimation over time with an initial mass of 40 Kg.

The figure shows a convergence to around 55 [kg] after 30 seconds, with the estimated value having very small variations of ± 0.5 [kg] even when performing accelerating or braking operations.

The remaining model parameters are shown in Figure 5.4. The true values of the rolling resistance coefficient and wind resistance are difficult to quantify as they depend on environmental conditions such as soil material, wind speed and direction. As discussed in Section 4.4 these parameters will change quickly depending on material and environment. However, their tendency to approach zero values is not physically feasible, and is likely a side effect of the optimization.

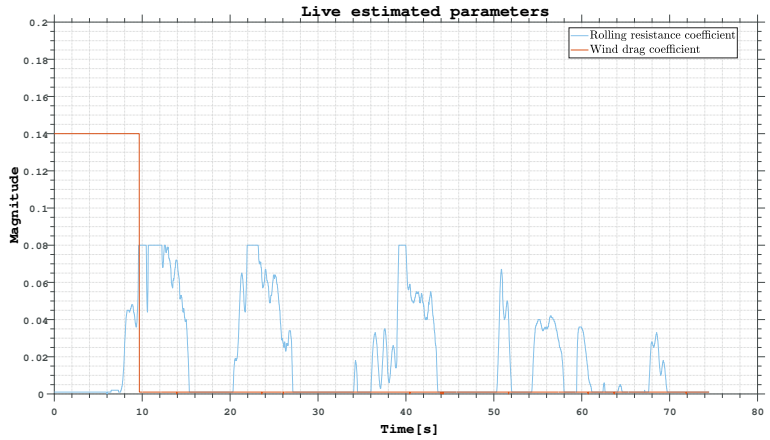


Figure 5.4: Plot showing the estimated friction and drag coefficients over time.

5.3 Control design

Three different controllers were compared using the RMS of the force sensor defined as

$$RMS = \sqrt{\frac{1}{N}\sum_{i}^{N} x_i^2} \quad \text{, where} \quad \begin{matrix} N = \text{number of measurements} \\ x_i = \text{measured force} \end{matrix} \tag{5.1}$$

as the performance metric, and comparing them to the RMS of a case where no controller was used. The $\hat{F}_s$-controller, defined in Section 4.5.1, was compared with the P and PI controllers. All controllers were tested a trailer mass of 55kg, as well as additional tests with 20 kg weight added to the trailer for both the PI-controller and the $\hat{F}_s$-controller (see Table 5.3 for the results).
With well tuned parameters the PI-controller will perform as well as the suggested $\hat{F}_s$-controller. When the trailer weight is changed the PI-controller will perform significantly worse without changing its tuning parameters. The $\hat{F}_s$-controller maintains consistent performance at varying trailer weights.

Table 5.3: Root mean square(RMS) of measured force with different controllers and the reduction in force between using no controller and one of the controllers.

Controller	Mass [Kg]	RMS [N]	Reduction	Behaviour
No controller	55	48.2	-	-
P-controller	55	25.44	47.46%	Oscillates
PI-controller	55	24.26	49.9%	Responsive
$\hat{F}_s$-controller	55	23.3	51.88%	Responsive
PI-controller	75	29.95	31.94%	Unresponsive and low compensation
$\hat{F}_s$-controller	75	24.19	50.04%	Responsive

The $\hat{F}_s$-controller and estimator were tested simultaneously, yielding mixed results.

While the controller behaved correctly, the parameter convergence rate was too slow compared to the results in Section 5.2, resulting in inaccurate model parameters.

6
Discussion

Analyzing the results in Section 5.1, it is clear that the model that calculates the forces is a valid method. It can model well-maintained asphalt roads including various inclines with an error of 12%. However, looking at Figure 5.2 it is clear that the derived model lacks information and that there are improvements that can be made regarding potholes, curbs and speed bumps.

The results obtained from the estimator in Section 5.2 show convergence towards the true parameter values, which is particularly evident in the convergence of mass depicted in Figure 5.3 after 2-3 sections of acceleration or braking. Especially during deceleration, the system is exposed to larger forces compared to acceleration, leading to more pronounced changes in the calculated mass. These observations suggest that the dynamic nature of the estimator works as expected, showing that the system reacts to the movements of the trailer.
The wind resistance coefficient converged to small values around 0.01 (see Figure 5.4), which will drive the corresponding term in Equation 4.10 to zero. Therefore, the wind resistance will mostly be negligible when it comes to estimating the total force. This is not unreasonable given the low speed of the bicycle and the fact that the cyclist is in front of the trailer and takes the major part of the wind resistance.
Rolling resistance μ is a function of both the mass and the gradient of the road, resulting in a large force contribution. The results show that μ mostly stays within its interval of expected values and changes rapidly when driving on asphalt, smooth concrete or gravel, which agrees with our assumption of it being a fast changing parameter.

Regarding the controller in Section 5.3 , it can be seen that the $\hat{F}_s$ controller is consistent with different loads compared to the PI controller, which does not work when the trailer parameters deviate from the parameters chosen when tuning the PI controller gains. However, when combining the controller and the parameter estimation, the convergence rate of the estimated parameters decreases, leading to inaccurate estimates over longer time periods. This suggests that the controller is robust under changing loads, but that further fine-tuning is required to achieve accurate estimates while the controller is active.
The low reduction in force measurements presented in Table 5.3 does not reflect the experience of the cyclist when the system was tested. This discrepancy could possibly be due to unwanted force measurements caused by errors in the mechanical design of the force sensor mount or stick-slip phenomena, but this has not been investigated further due to time constraints.

6.1 Evaluation of research questions

Question 1: Is it adventageous to state the control problem as an online optimization problem in comparison to using a more conventional PID controller?

To calculate the control signals by formulating the control problem as an optimization problem to be minimized performs similarly to a conventional PI controller (see Table 5.3). The use of a model where parameters such as trailer mass can be updated during driving provides significant performance advantages over the PI controller in terms of practical use of the trailer where the trailer load can vary over time and use. As shown in Section 5.3, the proposed controller performs the same when the trailer mass changes while the PI controller performs 64% worse compared to the trailer mass for which the controller was set.

Question 2: Can the model parameters be simulated online and how should the different convergence rates be handled?

In order to perform real-time parameter estimation, processing power is required to solve optimization problems online at each time-step. With the implemented estimation method efficient solvers such as OSQP can be used for fast calculations. In combination with utilizing the model gradient and the calculated error between force estimation and measurements, this results in the given model parameters being estimated and converging at a rate that can be tuned by the user (see Figure 5.1). While the parameter estimation gives values close to the real values when used in isolation (no active controller), the convergence speed is slow when used in combination with the controller, leading to estimation errors in the force estimation. This is belive to be because the online parameter estimation needs better tuning.

Question 3: What sensors are needed for the control of the trailer to be effective and how will removing sensors affect the result?

The trailer is equipped with sensors that measure the angular velocity of the wheel and a one DoF load cell that measures the force exerted between the trailer and the towing bicycle. To design a controller that compensates for the weight of the trailer, the force measured from the force sensor is estimated, resulting in the force sensor being redundant. However, to perform a real-time parameter estimation, a comparison between the estimation and a true value is required and for this both the angular velocity sensor, and the force sensor are used, which means that both sensors are required for parameter estimation.
While the force sensor is not needed for the control design another solution is needed to perform live parameter estimation without the sensor. Removing the force sensor would result in lower production costs and less points of failure. However, at this point of development both sensors attached to the trailer are required to perform

the suggested controller.

6.2 Future work

This section presents suggestions for modifications and additional work that can improve the trailer. Such modifications are intended to increase functionality, efficiency and performance. The aim is to provide a guideline for future improvements and to encourage further development of the trailer.

- Since the model and software are designed with F_{side} as a variable, implementing F_{side} to further improve the controller should mostly be a matter of validation. This improvement will especially be noticeable in sharp, slow turns.
- To enhance robustness, addressing special cases like navigating curbs and rough terrain are necessary. This means collecting and analyzing data from such scenarios and then making adjustments to the model or signal processing methods accordingly.
- To distinguish between acceleration and deceleration, two methods can be used: implementing different models for each case or reducing the weights of the derivative terms of the control signals during braking. By doing this, we can enable more responsive braking and thus increase safety and improve overall user experience.
- Introducing new control modes, such as an active mode where the trailer applies more force than necessary to push the bike forward, which would reduce the power needed by the cyclist to propel the bike. This adjustment could be implemented by adding an offset to the cost function. However, implementing this feature would require careful consideration to ensure that the trailer remains safe to use.
- Include a cost that increases over time for the estimated mass of the trailer to switch the focus towards the remaining parameters to be estimated.

6.3 Summary

This thesis has covered the development of an electric bicycle trailer. The thesis began with giving a basic understanding of the system with a system description serving the purpose of documentation for future work. It then continued by explaining the methodology of deriving the models used for control design as well as describing the algorithm used for online parameter estimation. Verification was done on the physical system both for the parameter estimation and the model based controller, which both work as intended when used separately. The prototype was designed with a framework in mind to guide the development through system requirements and research questions.
The results are promising, with the prototype operating with good accuracy on well-maintained asphalt roads, with a few edge case exceptions. There are still issues with the handling of potholes, curbs and speed bumps, which should be considered

to make the ride smoother. Continued work on simultaneous parameter estimation and control should also be focused in future work.

Bibliography

[1] *Accelerometer, STMicroelectronics - LIS3DHH, datasheet.* URL: https://www.st.com/content/ccc/resource/technical/document/datasheet/c9/16/f2/79/ca/31/47/89/DM00096789.pdf/files/DM00096789.pdf/jcr:content/translations/en.DM00096789.pdf.

[2] Zhu Deyi, Zhang Kai, and Yin Dejun. "A New Torque Control Approach for Electric Power Assisted Bicycle Based on Model-Following Control". In: *2017 International Conference on Information, Communication and Engineering (ICICE)*. 2017, pp. 36–39. DOI: 10.1109/ICICE.2017.8479205.

[3] Robert Hooke and T. A. Jeeves. "" Direct Search" Solution of Numerical and Statistical Problems". In: *J. ACM* 8.2 (1961), pp. 212–229. ISSN: 0004-5411. DOI: 10.1145/321062.321069.

[4] *Load cell amplifier, SparkFun Electronics - HX711 datasheet.* URL: https://www.electrokit.com/upload/product/41013/41013925/hx711_english.pdf.

[5] Stephen J. Wright Jorge Nocedal. *Numerical Optimization*. 2009. ISBN: 978-1-4939-3711-0.

[6] Zhang Kai and Yin Dejun. "A control approach adaptive to load and road slope for electric power assisted bicycle". In: *2017 36th Chinese Control Conference (CCC)*. 2017, pp. 3414–3418. DOI: 10.23919/ChiCC.2017.8027887.

[7] P. Kasper, N. Jesper, D. Lovisa, F. Benjanmin, and M. Christopher. *En självgående cykelvagn*. BSc Thesis. 2022.

[8] Amin Korayem, Amir Khajepour, and Baris Fidan. "A Review on Vehicle-Trailer State and Parameter Estimation". In: *IEEE Transactions on Intelligent Transportation Systems* 23 (July 2022), pp. 5993–6010. DOI: 10.1109/TITS.2021.3074457.

[9] Bengt Lennartson. *Reglerteknikens grunder*. Studentlitteratur, 2002. ISBN: 9789144024165.

[10] Tan Li. *Digital Signal Processing : Fundamentals and Applications*. Academic Press Inc, 2007. ISBN: 9780080550572.

[11] *Selden manual - Motor control unit datasheet.* URL: https://support.seldenmast.com/files/1715933884/597-283-E.pdf.

[12] Biologically inspired optimization methods. *Wahde, M.* WIT Press, 2008. ISBN: 9781845643447.

[13] *Pattern search - Matlab documentation*. URL: https://se.mathworks.com/help/gads/patternsearch.html. (accessed: 22.05.2024).

[14] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. "OSQP: an operator splitting solver for quadratic programs". In: *Mathematical Programming Computation* 12.4 (2020), pp. 637–672. DOI: 10.1007/s12532-020-00179-2.

[15] Bartolomeo Stellato and Goran Banjac. *OSQP documentation*. URL: https://osqp.org/docs/release-0.6.3/index.html#. (accessed: 05.06.2024).

[16] *STM32f407g-disc1, STMicroelectronics - datasheet*. URL: https://www.st.com/content/ccc/resource/technical/document/datasheet/ef/92/76/6d/bb/c2/4f/f7/DM00037051.pdf/files/DM00037051.pdf/jcr:content/translations/en.DM00037051.pdf.

[17] *Load cell, Sparkfun electronics - TAS501, datasheet*. URL: https://cdn.sparkfun.com/assets/parts/1/2/2/3/8/TAS501.pdf.

[18] *Technical documentation*. URL: https://github.com/andreaskruger/Master-thesis-Control-and-parameter-estimation-for-a-electric-trailer. (accessed: 05.06.2024).

[19] Hongchao Zhao, Shunbo Zhou, Wen Chen, Zhiqiang Miao, and Yun-Hui Liu. "Modeling and Motion Control of Industrial Tractor–Trailers Vehicles Using Force Compensation". In: *IEEE/ASME Transactions on Mechatronics* 26.2 (2021), pp. 645–656. DOI: 10.1109/TMECH.2021.3057898.

A

Appendix 1: Equations

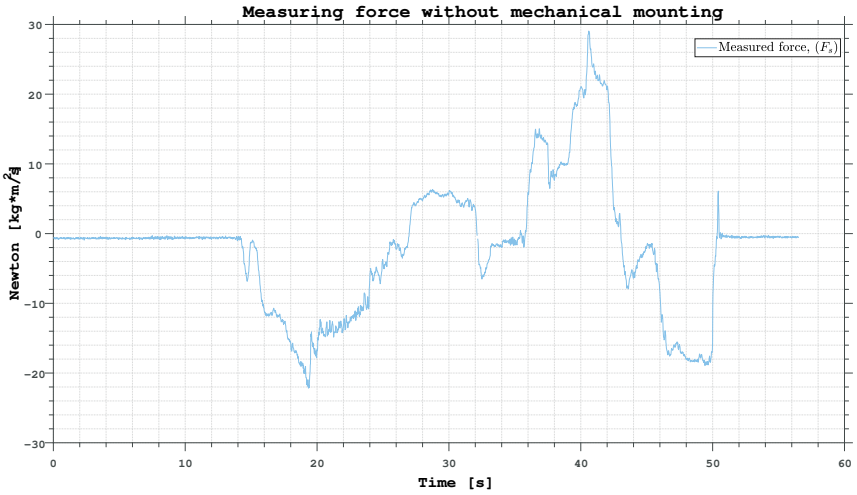


Figure A.1: Force sensor readings without the mechanical mounting

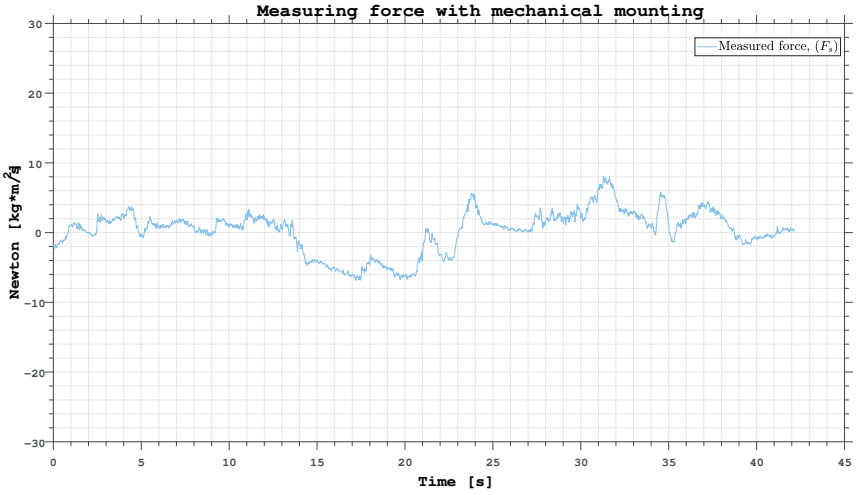


Figure A.2: Force sensor readings with the mechanical mounting

A.1 Rotation matrix

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix}, \quad R_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{A.1}$$

$$R_{xy}(\theta) = R_x(\theta)R_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ \sin(\theta)^2 & \cos(\theta) & -\cos(\theta)\sin(\theta) \\ -\cos(\theta)\sin(\theta) & \sin(\theta) & \cos(\theta)^2 \end{bmatrix} \tag{A.2}$$

A.2 Estimation cost function

$$\bar{\boldsymbol{P}} = \begin{bmatrix} \bar{p}_{11} & \bar{p}_{12} & \bar{p}_{13} \\ \bar{p}_{21} & \bar{p}_{22} & \bar{p}_{23} \\ \bar{p}_{31} & \bar{p}_{32} & \bar{p}_{33} \end{bmatrix}, \tag{A.3}$$

$$\tag{A.4}$$

where

$$\begin{aligned} \bar{p}_{11} &= \bar{K}_2 + \bar{K}_1 \left(g^2 \sin(\phi)^2 + 2\, g\, \dot{v}\, \sin(\phi) + \dot{v}^2 \right) \\ \bar{p}_{12} &= \bar{K}_1 \left(2\, v^2\, \dot{v} + 2\, g\, v^2\, \sin(\phi) \right) \\ \bar{p}_{31} &= \bar{K}_1 \left(2\, \hat{m}\, \cos(\phi)\, \sin(\phi)\, g^2 + 2\, \hat{m}\, \dot{v}\, \cos(\phi)\, g \right) \\ \bar{p}_{21} &= \bar{p}_{12} \\ \bar{p}_{22} &= \bar{K}_1\, v^4 + \bar{K}_3 \\ \bar{p}_{23} &= 2\, g\, \bar{K}_1\, \hat{m}\, v^2\, \cos(\phi) \\ \bar{p}_{31} &= \bar{p}_{13} \\ \bar{p}_{32} &= \bar{p}_{23} \\ \bar{p}_{33} &= \bar{K}_1\, g^2\, \hat{m}^2 \cos(\phi)^2 + \bar{K}_4 \end{aligned} \tag{A.5}$$

$$\bar{\boldsymbol{q}} = \begin{bmatrix} -\bar{K}_1(2F_s\dot{v} + 2F_s g \sin(\phi)) - 2\bar{K}_2 m_{prev} \\ -2F_s\bar{K}_1 v^2 - 2\bar{K}_3\lambda_{prev} \\ -2F_s g\bar{K}_1\hat{m}\cos(\phi) - 2\bar{K}_4\mu_{prev} \end{bmatrix}, \tag{A.6}$$

$$\bar{\boldsymbol{x}} = \begin{bmatrix} m \\ \lambda \\ \mu \end{bmatrix} \tag{A.7}$$

A.3 Control cost function

$$\boldsymbol{P} = \begin{bmatrix} k_1 + k_2 + \frac{k_4}{4} & k_1 - k_2 \\ k_1 - k_2 & k_1 + k_2 + \frac{k_3}{4} \end{bmatrix}, \tag{A.8}$$

$$\boldsymbol{q} = \begin{bmatrix} 2\hat{F}k_1 + 2\hat{F}_{side}k_2 - \frac{F_{wr,prev}k_4}{2} \\ 2\hat{F}k_1 - 2\hat{F}_{side}k_2 - \frac{F_{wl,prev}}{2} \end{bmatrix} \tag{A.9}$$

B

Appendix 2: System description

The trailer consists of both a hardware part and a software part that were designed and implemented during this project. This chapter aims to provide a complete overview of the system with descriptions of all its subsystems (see Figure 2.2 illustrating a system overview). Descriptions of hardware, software architecture, system requirements and security measures are given in this chapter.

The system consists of two main parts, hardware and software, which are divided into the three subsystems communication, main loop and motor control unit (MCU). The hardware constructed during the project handles the power distribution to the onboard microcontrollers as well as the two MCUs. For safety all power are routed through an emergency stop and a dead man's switch. The software handles sensor readings, signal filtering, WIFI communication, parameter estimation and motor control. To further increase safety during development, a signal was sent over WiFi with a fixed periodic timer, and if no signal was received, an error was triggered. This is referred to as a heartbeat function.

The sub systems are listed in Table B.1.

Table B.1: Subsystems and operating area.

Subsystem	Operating area
ESP32	WiFi communication Heartbeat(WiFi fail-safe) Commands sent to the trailer(WiFi)
STM32	Sensor readings Signal filtering Estimation Control algorithm MCU communication
MCU	Motor control

For a more detailed description and technical documentation of the system visit the project repository [18]. Comments and function descriptions of the code exists in the repository as well as the code base, environments for simulation and WIFI communication scripts for logging data.

B.1 System requirements

Defining comprehensive system requirements laid the foundation for the development of a robust and functional prototype. This section defines the necessary specifications and essential features that the electric bicycle cart prototype had to include.

Hardware requirements:

- Safety in the form of fuses, emergency stop, dead man's switch.
- A mechanical part was required in order to ensure that the force sensor measures forces in one direction only.
- Robust mounting that can withstand rigorous testing while driving outdoors.
- Cable management resulting in safer usage.

Software requirements:

- Wireless bidirectional communication, specifically being able to receive and log data as well as commands.
- Main program implemented for a user friendly and maintainable software architecture.
- Maintain a system period of below 15ms.

B.2 Hardware components

This section aims to provide a basic understanding of the hardware that was created for the electric trailer. Energy management, charging, mounting of sensors and electronics are all given a brief description to enable future work. The hardware was assembled for easy access to all relevant parts of the trailer used during the development of this project.

B.2.1 Electrical circuit diagram

Figure B.1 shows the circuit diagram for the electrical system. The STM32 side is isolated from the MCU side using opto-isolators. This precaution is necessary to prevent ground loops and potential communication failure between the STM32 microcontroller and the motors, as sharing the same ground between these components could lead to signal corruption. The separation is represented as a dotted line crossing the opto isolators. Note that the diagram illustrates the connection of a single MCU. In reality, this connection is implemented twice, once per MCU.

Note also that two pins are connected to **UART**, which is used for software debugging.

Figure B.2 shows how the dead man's handle is constructed and connected to a larger solenoid to quickly cut the power supply to the MCU if the operator encounters problems.

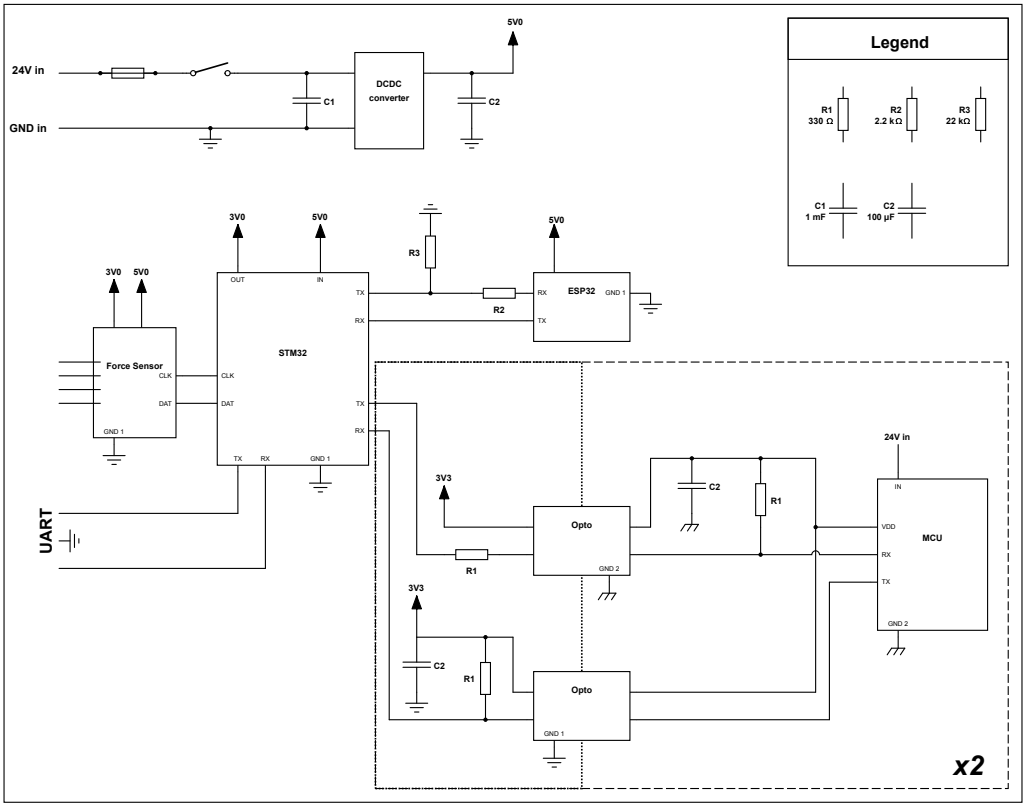


Figure B.1: Circuit diagram of electrical hardware.

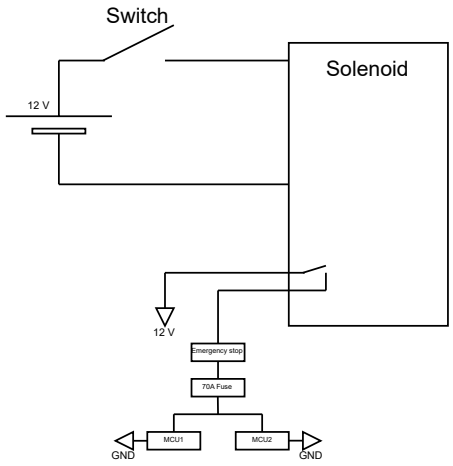


Figure B.2: Circuit diagram of dead man's switch.

B.2.2 STM32 Microcontroller

The discovery kit(STM32F407G-DISC1-Discovery) is a development board based on the STM32F407VG microcontroller created by STMicroelectronics [16]. It has an ARM Cortex-M4 processor with a Floating Point Unit (FPU), which enables fast calculations when using floating point arithmetic's. This is a key feature because of the calculation complexity of online parameter estimation and optimization. Another important feature is the built-in accelerometer that is used for calculating the road incline. The STM32 acts as the main processor in the system, performing actions such as sensor readings, processing of signals, optimization of control signals and calculations used for estimations.

B.2.3 ESP32 Microcontroller

The ESP32-S3-DEVKITC-1-N32R8V was selected due to its integrated WiFi module. This module has important features that enable the ESP32 to communicate over WiFi, which is used for data logging. The ESP32 uses WiFi as well as UART simultaneously to communicate with a PC and the STM32.

B.2.4 Motors

The trailer is equipped with two wheels, each having a permanent magnet synchronous motors(PMSM) using direct drive (see Figure B.3).

Figure B.3: Hub motors mounted on the trailer.

The motors receives commands from the MCU (see TableB.2) for motor specifications.

Table B.2: Motor specification.

Name	Crystalyte - HS2440
Operating voltage	24[V]
Max Amp	40[A]
Torque/A	1.0052[Nm/A]
Poles	23 pairs
Controller type	Sensorless

B.2.5 Motor Control Unit

The Motor Control Unit (MCU), which is an off-the-shelf product, communicates with the STM32 via UART and takes as input a motor torque, controls three phases to the motors, and ensures that torque is outputted by the motors [11]. The STM32 receives messages from the MCU containing RPM, supplied voltage, current and applied torque. The motors are sensorless so the MCU uses field oriented control(FOC) to control the three phases of the motors.

B.2.6 Batteries

With their relative low cost lead-acid batteries were chosen for the prototype. The downside of this type of battery is their size and weight. Two lead-acid 12V batteries are connected in series resulting in 24V supplied to the two MCU's.

B.2.7 Sensor specifications

B.2.7.1 HX711 - Force sensor

The force sensor consists of an s-type load cell and a HX711 amplifier breakout board [4]. The load cell works by having 4 strain gauges in a wheatstone bridge configuration and converts a force to an electrical signal that can be measured [17]. The HX711 breakout board amplifies the electrical signal which is measured and processed by the STM32 (see Table B.3 for sensor specifications).

Table B.3: Force sensor specifications.

Name	TAS501
Type	S-type
Full-scale voltage	±40[mV]
Accuracy	±5%
Amplifier name	HX711
Supply voltage	5[V]

B.2.7.2 LIS3DH - Accelerometer

The STM32 devboard comes with an integrated 3-axis accelerometer [1]. After the STM32 was mounted on the trailer, the coordinate frame of the accelerometer needed to be rotated and calibrated, which is descibed in section B.3.4.2. Specifications for the accelerometer are presented in Table B.4.

Table B.4: Accelerometer specifications.

Name	LIS3DHH
Type	Accelerometer
Axis	3
Noise performance	$45[\mu g/\sqrt{Hz}]$
Data output	16-bit

B.2.8 3D printed mechanical components

The trailer and bicycle are connected via ball joint hinge and the load cell is mounted along the stiff front bar of the trailer. This configuration allows for the load cell to pick up both rotational and bending forces. To eliminate these forces and isolate the translational forces a mechanical housing for the load cell was created (see Figure B.4).

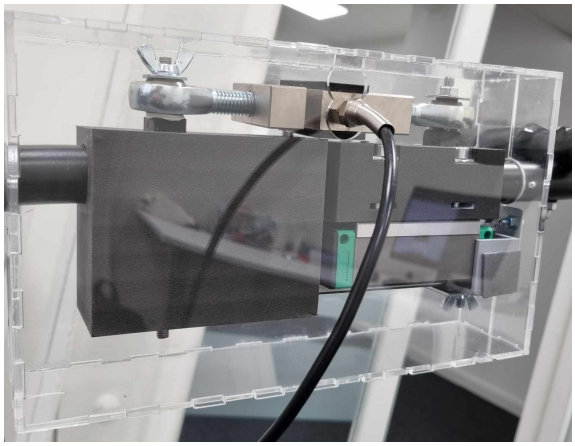


Figure B.4: Resulting 3d printed housing for the load cell.

B.2.8.1 Load cell housing

The front bar of the trailer was cut perpendicular to the bar into two parts and each part was attached to a separate 3d printed part and mounted with screws on a linear slider. This prevents unwanted forces and isolates the translational force.

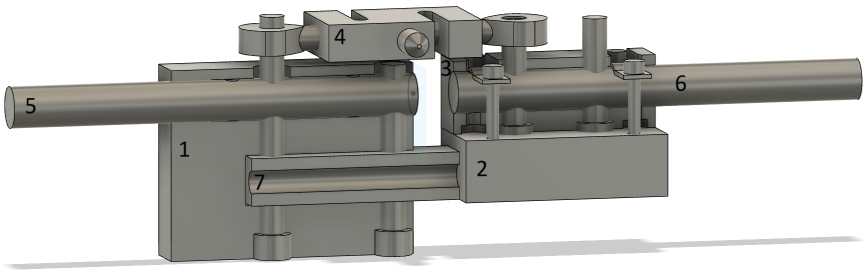


Figure B.5: Mechanical housing for the load cell, isolating translational forces and removes rotational forces. Marked numbers: 1: 3d-part(1), 2: slider with ball bearings, 3: 3d-part(2), 4: load cell, 5: bar connected to the trailer, 6: bar connected to end of bar and the hinge.

To obtain very precise parts a selective laser sintering(SLS) printer was used with a nylon material which yields very stiff parts. To protect the load cell and the linear slider a casing made out of acrylic was created and mounted to encapsulate the parts.

B.2.8.2 Dead man's grip

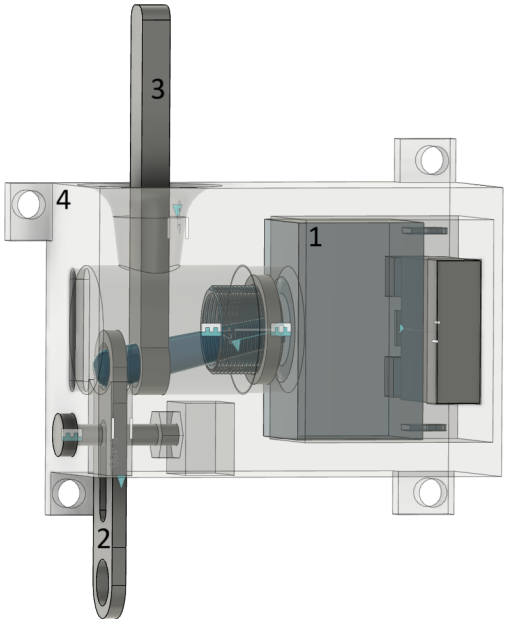


Figure B.6: dead man's switch to cut the main power to the trailer. Marked numbers: 1: Switch, 2: reset pin, 3: dead mans cord, 4: switch housing.

As safety is an important aspect, a dead man's switch was designed with the help of 3d prints and mounted on the trailer. This can be seen in Figure B.6, where numbers are included in the figure to indicate the exact part. A small switch is not

designed or constructed to handle a large amount of current and therefore a larger solenoid is used together with the switch to break the current. The circuit diagram can be seen in Figure B.2. The switch is connected to 5V and to a solenoid and by pulling the switch, the solenoid will break the main current resulting in immediate power loss to the motors. By mounting the switch ("1") in such a way, pulling the dead man's cord ("3") cuts the power. To reset the dead man's switch, the reset pin ("2") is pulled downwards, activating the system power supply.

B.3 Software architecture

This section provides an overview of key concepts to understand the structure of the software. User communication was implemented via a command line interface (CLI) that allows users to interact with the system in real time, while data logging ensures that relevant information is stored. Error handling is crucial to maintain reliability and avoid unexpected and potentially dangerous behaviors during the development of the prototype. Finally, the main program flow is presented to provide an understanding of the underlying logic and reasoning.

B.3.1 Communication overview

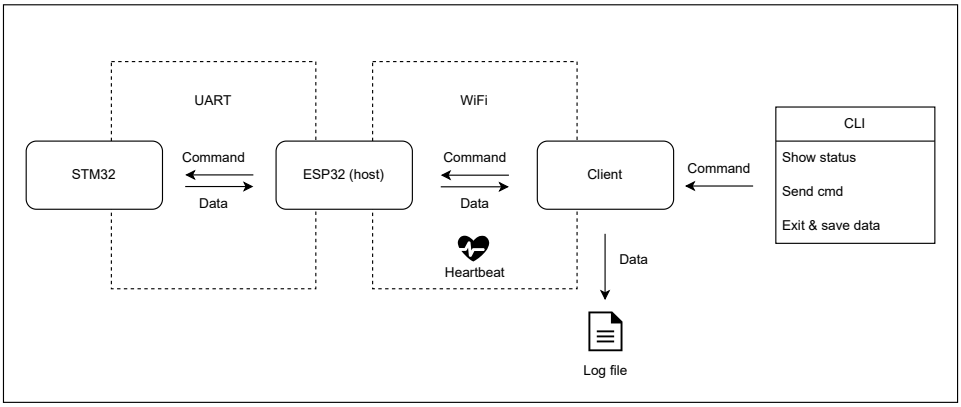


Figure B.7: Communication flow chart.

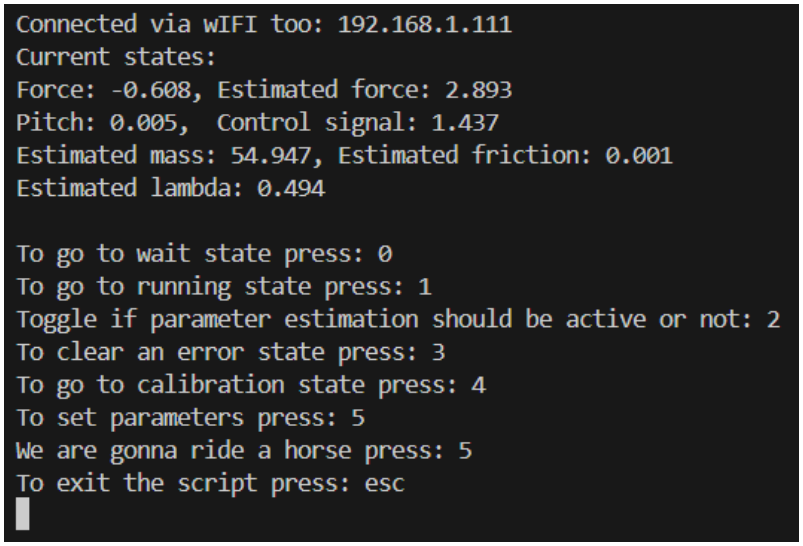
B.3.1.1 UART protocol

The microcontrollers use the interrupt-based Universal Asynchronous Receiver Transmitter (UART) protocol for communication. This is a method of data exchange between devices where data is sent or received in response to hardware interrupts, which are signals that temporarily pause the normal flow of execution to handle specific events.

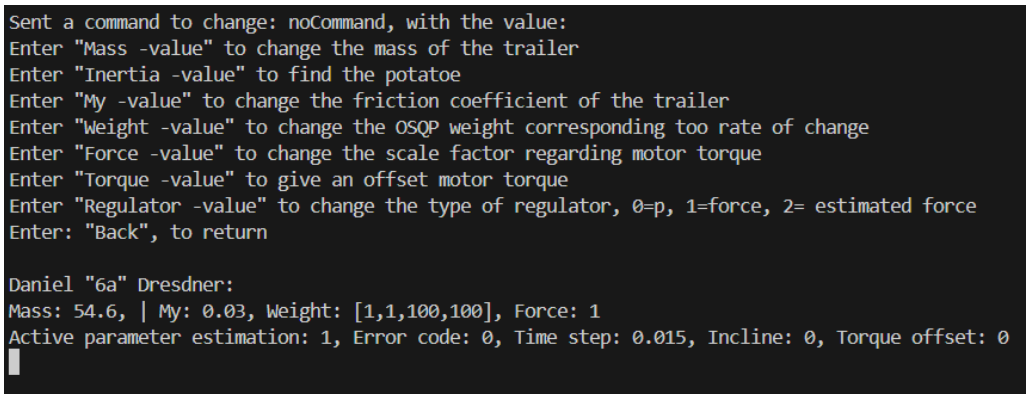
B.3.1.2 User communication and data logging

The ESP32 receives data from the STM32 via an interrupt-based UART. It acts as a local WiFi server and relays the information to a client computer, allowing the

user to send and receive commands, measurements and error codes from a distance. As a safety measure, a heartbeat function has been implemented. It acts as a digital switch and ensures that if at any point the client loses contact with the host, an error message is sent to the STM32 to shut down the system.
The local client uses a python script with a CLI. The CLI allows the local client to interact directly with the main system, providing a tool for real-time control and adjustment of parameters. It displays the last received information and has the possibility to send commands back to the STM using the same same communication system. See Figure B.8 for an example of the CLI. All received data from the current session are saved in a log file, which can then be processed and analyzed afterwards. The communication structure is shown in Figure B.7.



(a) Main menu.



(b) Settings menu.

Figure B.8: Command line interfaces.

B.3.1.3 Motor control with MCU communication

The communication with the motors is done using the interrupt-based UART protocol, where commands are sent to the motor control units (MCUs), which in turn respond with an echo followed by the corresponding status. See Table B.5 for available commands.

Table B.5: Table of MCU commands.

Command	Purpose	Response
"getm\n"	Get motor information	Speed [kRPM], Torque [Nm], Rotations since startup [$x \in \mathbb{N}^+$], Supply voltage[V], Battery current[A]
"run -s %99.0 -t %3.2\n"	Set motor speed and torque	String confirmation of applied 99 kRPM and 3.2 Nm

When writing to an MCU, the speed parameter only determines the direction, where a positive sign means a forward movement. The torque parameter determines the actual power of the motors and is the control signal selected for the controller. As a precautionary measure, the MCUs have been set with a maximum current threshold to prevent the motor torque from becoming too high.

B.3.2 Optimization solver - OSQP

To solve the optimization problems defined to calculate the control signals and estimate the physical parameters, the OSPQ (Operator Splitting Quadratic Program) solver is used.

OSQP is an open-source numerical optimization program designed to solve convex quadratic programs (QPs) efficiently. It was developed by researchers from the University of Oxford and Stanford University and is tailored to address large-scale QP problems that arise in various fields such as control systems and signal processing.

In addition to its efficiency, OSQP has a Matlab interface and an available C library, making it a suitable tool for simulating data offline and comparing logged data calculated online with STM32. [14]

B.3.3 Main program overview

The main program is implemented as a finite state machine (FSM). An FSM is used to model the behavior of a system with a number of states. These states are different modes in which a system can operate exclusively and imply a deterministic behavior. The transitions from one state to another are based on certain conditions. These transitions define how and when a change of state occurs. External or internal events can trigger state transitions. These events can be user inputs or internal conditions such as errors.

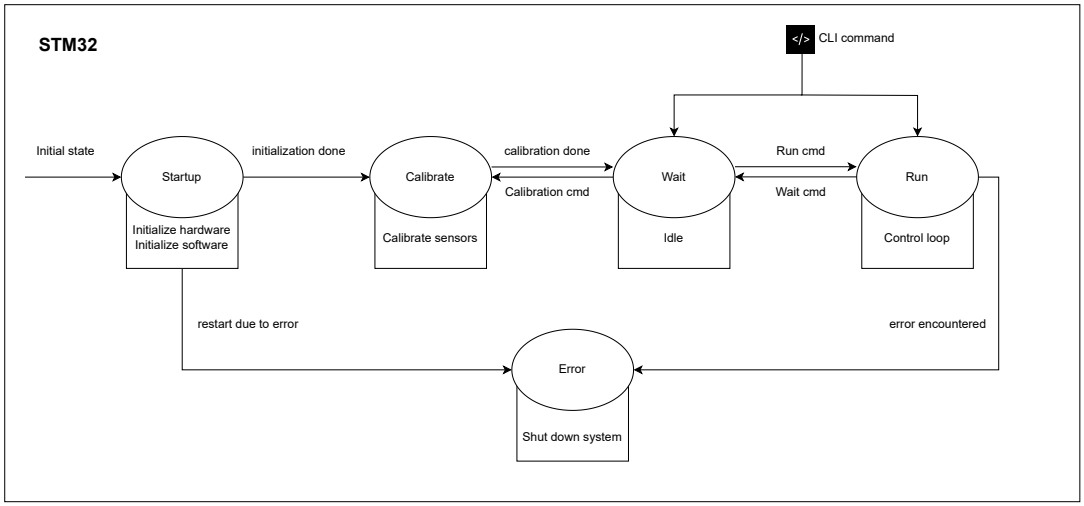


Figure B.9: Main program loop.

When the STM32 is powered on or reset the FSM starts in the start-up mode, where hardware interrupts and variables are initialized.

After start-up, the FSM enters the calibration mode, where the accelerometer and force sensor are calibrated. This mainly consists of adjustments in relation to the initial position of the trailer and is crucial for accurate estimates. The calibration mode can be chosen by the user from wait-mode, which is convenient without having to restart the system.

The wait state is an idle mode where the FSM waits for a command to be received. At start-up, it is advisable to adjust parameters and settings or re-calibrate in the idle-mode, before sending a run command. Otherwise there is a risk of using incorrect measurements. The idle-state also serves as a safety measure to prevent unexpected behavior of the trailer at start-up, which could arise from using settings from a previous session (although this is unlikely).

The run state is where the control loop is executed. It is described more in Section B.3.3.1.

If at any point an error is encountered, the FSM will go into the error state and shut down the system. This means in practice that 0 torque is sent to the motors and to stop calculating new control signals. The errors are described in Section B.3.3.2.

B.3.3.1 Control loop overview

This section describes the main component of the FSM, the control loop. In short, the program processes sensor data, filters this data, performs calculations and controls the motors.

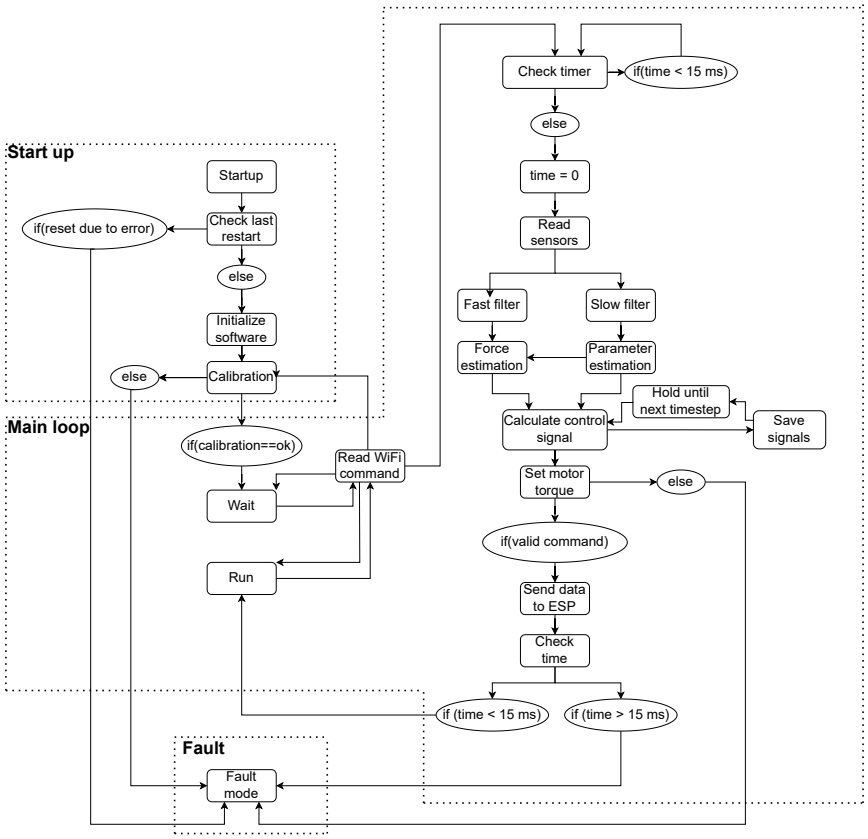


Figure B.10: Finite state machine.

The control loop starts by sending a command to the MCUs to find out the current state of the engines. The communication with the MCUs is not instantaneous, which means that the control loop is processed while waiting for the MCUs response. Low-pass filtering is performed on the motor speeds obtained from the previous iteration, which will be used as data for the current loop. In addition, data is read from accelerometer is read and the force sensor is read and filtered.

Following this, the OSQP solver updates the optimization problem using the filtered signals and estimated force. The solver then calculates the optimal control inputs, which are applied as torques to the motors through another MCU command. Concurrently, parameters are estimated with OSQP based on the slow filtered data.

Once all the data has been received and processed, the system sends the user's data of choice to the ESP32. It then waits until the motor speed buffers have processed a full message, from which the speed is saved, ready for use in the next iteration. Finally, before exiting the loop, the system checks for any timeout conditions, which could indicate potential problems, and in such a case, throws an error.

B.3.3.2 Fault handling

Fault handling is crucial to ensuring system reliability and security. Different types of errors are covered, from hardware malfunctions to software failures.

- **Timeout**: In order to meet the requirement that the control loop should with a maximum sample time of 15 ms, a timer is dedicated to the control loop. If an iteration has exceeded the time limit, an error is triggered.
- **Watchdog interrupt**: A watchdog has been implemented which handles the entire FSM loop. It requires a callback function to be called at certain intervals, otherwise it will produce an error. This is to prevent the program from unexpectedly freezing due to possible undiscovered bugs.
- **WiFi timeout**: Similarly to the watchdog, a heartbeat function is implemented to make sure that the WiFi connection is not lost between the server and client. It accepts a pulse every second, otherwise it produces an error. A lost connection would mean that no commands could be received, which can lead to unsafe scenarios.
- **Calibration error**: A calibration error is defined as an outlier among several repeated calibrations. Should the calibration be inaccurate it could lead to failed estimates and unexpected behaviour from the trailer.
- **Reset error**: There are some hardware errors that inherently make the STM32 restart and, therefore, at startup a check is needed to make sure that no such error triggered the reset. These could be due to hardware issues, such as high voltage, or software related, e.g. a watchdog reset.

B.3.4 Sensor readings and calibration

This section gives a brief explanation of the software used to interact with the different sensors used in the project. Both the force sensor and the accelerometer are implemented with their own offline calibration method used as the baseline, as well as an online temporary calibrating method at startup to make minor adjustments.

B.3.4.1 Force sensor

The force sensor readings are mostly linear and an offline calibration method where the sensor readings were described as an affine function $y = kx + m$ is used. Using a known exact weight, the slope k, and displacement m, were calculated by putting the known weight on the sensor and noting the raw values. Using the straight line and the calculated values, the raw values were converted to a force. At start-up, the sensor is re-calibrated to account for small variations in the position of the mounted sensor, resulting in sensor readings close to zero at start-up.

B.3.4.2 Accelerometer

The three axes depend on the mounting position of the sensor, which does not correspond to the local coordinate frame in Figure 4.2. The accelerometer is aligned to the local coordinate frame by multiplication with the rotation matrix $R_{xy}(\theta)$

(defined in Appendix A.1):

$$\begin{bmatrix} A_x \\ A_y \\ A_z \end{bmatrix} = R_{xy}(\theta) \begin{bmatrix} \bar{A}_x \\ \bar{A}_y \\ \bar{A}_z \end{bmatrix}, \tag{B.1}$$

where θ is the rotation in radians and $\bar{A}_x$, $\bar{A}_y$ and $\bar{A}_z$ are sensor readings. The accelerometer was calibrated offline by moving and rotating the sensor while taking its maximum and minimum raw values. The sensor is calibrated online at start-up by taking the average of the readings over 500 ms and using it as an offset.

B.3.5 Velocity and acceleration

Velocity can be measured using the MCU's output information on angular speed for each wheel. By converting the RPM to linear velocity and differentiating, the acceleration is obtained.

B.4 Discussion of system requirements

In this section the previous stated system requirements and design goals will be discussed and evaluated.

B.4.1 Safety requirements

When building the system safety, aspects were part of the main focus. The hardware safety functions worked as planned with loose parts and cables kept to a minimum, a dead man's switch used by the operator in case of emergency, as well as added fuses in multiple steps around the power management system. The system software architecture allows for a robust deterministic behaviour while the logic complexity of the system grows larger. The addition off a communication heartbeat, remote start/stop commands and the software fault handler results in a system that can be expanded upon while keeping its safety features. A hard limit of 15ms period time was set for the system, which can be maintained if the number of data fields logged is kept to the recommended amount of six fields per package. The recommended field limit of six can be surpassed without affecting the safety of the system but will affect the integrity of the logged data and an increase in corrupted messages will be noticed.

B.4.2 Mechanical parts for isolating sensor readings

To mount the force sensor in horizontal position (see Figure B.4), a mechanical bracket was designed to remove unwanted rotational forces acting on the sensor as well as shear and bending forces. After testing the force sensor using the mechanical mount, it can be shown that a significant part of the shear and bending forces are removed. See Appendix A (Figures A.2 and A.1), where a motion test is performed with and without the mechanical mounting.

B.4.3 Communication protocols and CLI interaction

The implemented CLI allows remote control of the system, including emergency shutdown, re-calibration and data logging. The CLI makes the interaction with the system user-friendly and easy to use even when problems occur.

B.5 Future work

The current hardware of the system is oversized and overdimensioned, both in terms of physical size and computational capabilities, and the trailer does not currently have space for actual loads. The next step is to develop a more streamlined version using less expensive hardware that maintains the same level of performance. For instance, smaller batteries could be installed under the trailer's undercarriage instead of within the main volume, providing more storage space. Additionally, replacing the existing force sensor with e.g. a force sensitive resistor strip could reduce costs, save space, and minimize maintenance needs.

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY