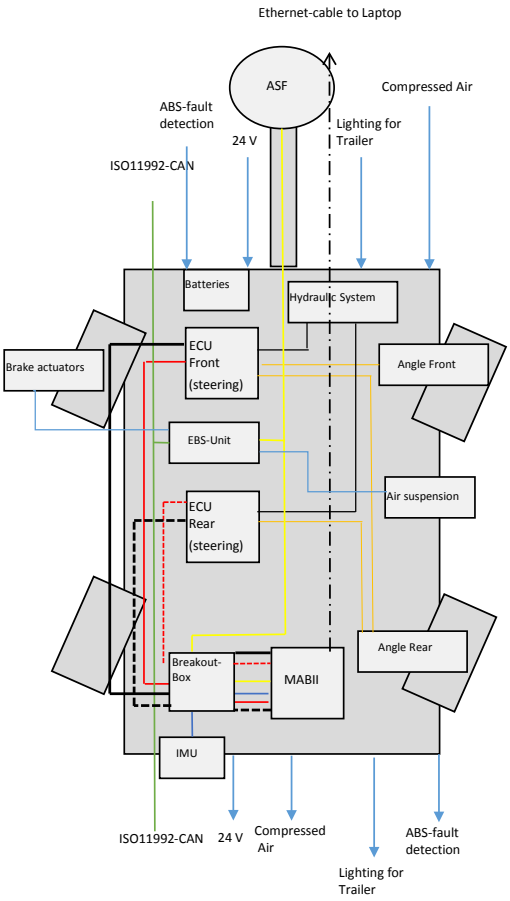
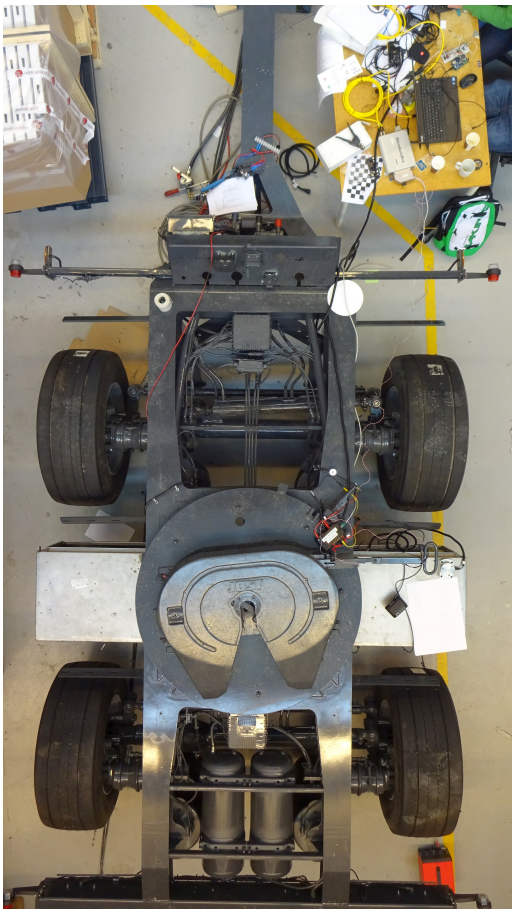
CHALMERS





Real-Time Control Interface for a Steered and Braked Converter Dolly for High Capacity Transport Vehicles

Master's thesis in Automotive Engineering

SEBASTIAN FRANZ
MICHAEL HOFMANN

Department of Applied Mechanics
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2015

MASTER'S THESIS IN AUTOMOTIVE ENGINEERING

Real-Time Control Interface for a Steered and Braked Converter Dolly for High Capacity Transport Vehicles

SEBASTIAN FRANZ
MICHAEL HOFMANN

Department of Applied Mechanics
Division of Vehicle Engineering and Autonomous Systems
Vehicle Dynamics Group
CHALMERS UNIVERSITY OF TECHNOLOGY

Göteborg, Sweden 2015

Real-Time Control Interface for a Steered and Braked Converter Dolly for High Capacity Transport Vehicles
SEBASTIAN FRANZ
MICHAEL HOFMANN

© SEBASTIAN FRANZ, MICHAEL HOFMANN, 2015

Master's thesis 2015:62
ISSN 1652-8557
Department of Applied Mechanics
Division of Vehicle Engineering and Autonomous Systems
Vehicle Dynamics Group
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone: +46 (0)31-772 1000

Cover:
The cover picture shows a block diagram of all the subsystems of the converter-dolly used in this project

Chalmers Reproservice
Göteborg, Sweden 2015

Real-Time Control Interface for a Steered and Braked Converter Dolly for High Capacity Transport Vehicles
Master's thesis in Automotive Engineering
SEBASTIAN FRANZ
MICHAEL HOFMANN
Department of Applied Mechanics
Division of Vehicle Engineering and Autonomous Systems
Vehicle Dynamics Group
Chalmers University of Technology

Abstract

To improve the dynamic behaviour and maneuverability of high capacity transport vehicles, the converter dolly can be equipped with steered and braked axles. By improving the actuation and steering strategy, the combinations performance can be increased drastically.

This thesis modified an existing steered dolly to accommodate a rapid prototpying system, so that previously developed steering algorithms can be executed in real-time and evaluated in a hardware-in-the-loop simulation as well as on-track testing further on. The main-outcomes are a ready-to-use interface for algorithm development in MATLAB/Simulink that allows to request steering angles individually for each axle and supplies feedback for all relevant parameters (e.g. articulation angle between different units of combination, steering angle of tractor) for their computation in real-time. On top of that a concept for requesting braking torques from the dolly's braking system for future projects was developed. To ensure safety of the whole combination, especially during track-tests, a supervisor system was implemented to limit the request based on hardware limitations. Additionally an inertial measurement unit setup was developed and implemented for use on the combination to gather sufficient data for each of the combination's independently moving units dynamic behavior during testing to calibrate and parametrize the tested algorithms.

In the scope of this thesis the dolly was modified to conduct hardware-in-the-loop simulations. All necessary systems to achieve this were built as prototypes for a workshop test-environment limiting the use-case to a standing-system. Reliable track-proof solutions were developed and commissioned for manufacturing, so that track-testing is possible soon after.

Keywords: High capacity transport vehicles, steered converter dolly, rapid prototyping, dSpace MicroAutobBoxII, Arduino platform, Truck dynamics

Acknowledgements

This thesis was carried out at the Vehicle Engineering and Autonomous Systems division at Chalmers. We are very thankful for the friendly and welcoming environment that we found here. Our sincere thanks especially go to Manjurul Islam, PhD - our supervisor - and Prof. Bengt Jacobson who co-supervised us. Immense trust was put in our work in advance and we were provided with the possibility to work independently early on with very advanced technology and had the possibility to develop our concepts with great support and funding within the Active Dolly Build-Up research project. Thank you very much for a very interesting and educational time.

As this practical thesis ties in with many suppliers and different parties, we also want to express our gratitude towards the many contributors and interesting discussion partners. First and foremost Leo Laine, PhD (Volvo Trucks AB) has to be mentioned, who advised on our thesis and gave some vital input and impulses for our work and linked us with many contacts within Volvo Trucks. Oskar Gellerbrant and Martin Härberg from Fenco AB, who gave great help and input on dSpace-related issues. Richard Dahlsjö from VM-trailers for providing us with some practical advice and supplies for early prototypes. VSEs Rik de Zaaijer provided with very detailed information and good input, which greatly accelerated work in the early stages. The local WABCO subsidiary (Joakim Jonsson, Erik Helldin) provided us with nice contacts, diagnosis equipment, great advice and technical help concerning the brake systems on the dolly. The Chalmers workshop provided us with an uncomplicated way to build our lab-prototypes. For further equipment manufacturing we had the nice chance to discuss and cooperate with Segula AB (Jesper Larsson, Arpit Karsolia).

Contents

Acronyms

.dbc CAN database.

ABS Anti-lock braking system.

ASF Angle Sensor Front.

CAN Controler Area Network.

CoG Center of Gravity.

DC Descrete Current.

EBS Electronic Brake System.

ECU Electronic Control Unit.

ESC Electronic Stability Control.

ETS Electronically-controlled hydraulic Trailer Steering.

FMEA Failure Mode and Effect Analysis.

GPS Global Positioning System.

GUI Graphical User Interface.

HCT High Capacity Transport.

HiL Hardware-in-the-Loop.

HMI Human Machine Interface.

I^{2}C Inter-Integrated Circuit.

IDE Integrated Development Environment.

IMU Inertial Measurment Unit.

IP Internet Protocol.

LEMO Swiss company specialized in industrial connector manufacturing.

MABII dSpace MicroAutoBoxII.

MSD Motion Support Device.

OEM Original Equipment Manufacturer.

RCP Rapid Control Prototyping.

RPN Risk Priority Number.

RTI Real-Time Interface.

SPI Serial Peripheral Interface.

UART Universal Asynchronous Receiver Transmitter.

UDP User Datagram Protocol.

UML Unified Markup Language.

VDS Volvo Dynamic Steering.

VMM Vehicle Motion Management.

VSE V.S.E. Vehicle Systems Engineering B.V..

VTM Virtual Truck Model.

ZIF Zero Insertion Force.

List of Figures

List of Tables

1 Introduction

1.1 Purpose

Heavy goods-transport on the road has constantly increased over the last decades. Coupled with the stricter environmental regulations concerning CO_2-emissions and pollution, the desire for more economical transport solutions has led to the wider introduction of High Capacity Transport (HCT) vehicles. HCT vehicles, also known as long combination vehicles, are vehicle combinations that are longer and heavier than standard combinations. Those truck-trailer combinations have a longer history in geographical areas with low population density, mining and transport within factory sites where rail-road transport is not a viable option but transportation of large volumes and tonnages is necessary. The prospects of saving costs on driver's salaries, reduced fuel consumption and decreased costs suggests the introduction of those combinations in other environments as well. Introduction of a new vehicle class leads to many challenges in safety, research and development, and legislation.

The driving behaviour of HCT vehicles is in many ways different to that of single unit trucks and needs to be researched in great detail to gain an understanding of the vehicle's dynamic properties, that is equally detailed as it is for other vehicle classes. This will lead to development of better safety and assistance systems and thus reduce threat potential, accidents and fatalities involving this emerging mode of transportation. Different usage patterns of HCT vehicles have to be considered as well, when developing functions for HCT vehicles. For example inner-city use is no prevalent use-case for HCT vehicles, whereas highway safety features and handling properties at higher speeds are prime goals due to high percentage of highway-driving for HCT vehicles. Nevertheless maneuverability for docking is also a development goal.

The research project in which this thesis is embedded aims to develop an active dolly, meaning that steering will be autonomously conducted by the dolly based on the driving situation at hand and various vehicle parameters (e.g. speed, steering wheel angle). Further on braking capabilities are to be implemented to act in a similar fashion as an Electronic Stability Control (ESC) by creating a yaw-moment countering undesired vehicle movements. This counter-steering will be achieved through wheel-individual brake-application.

This high-level control algorithm will be executed on a rapid-prototyping system which is linked to and controls the dolly. To supply this connection between the hardware and control-algorithm implemented in the modeling-environment Simulink is the main-task of this thesis.

1.2 Objectives

The main-goals that are supposed to be achieved within this thesis' scope of work are:

- Develop a software interface for the high-level control algorithm implemented in Simulink to be run on a rapid-prototyping system to control the steering system on an active dolly.

- Develop the physical hardware interface with the dolly; establish a suitable environment connection for the rapid-prototyping system on-board of the dolly

- Develop a measuring solution to determine the processing delays in the sensing system as well as the delays introduced by computation and actuator reaction times. Determine and try to minimize these occuring delays.
- Verification of designed systems through different stages following the V-model
- Develop a safety system that continuously monitors the active steering system, prevents malicious inputs and triggers necessary warnings.
- Prepare on-track testing (design road-ready measuring equipment, start-up procedure, repeatable pre-recorded maneuver decription for automated testing)

1.3 Limitations

To implement the actual high-level algorithm to compute the desired angle for the dolly's steerable axles is not in the scope of this thesis. It was developed in the research project in which this thesis is embedded. Nevertheless to establish an easier insight into the interfaces' parameters, an overview of the structure, in- and outputs of the underlying computational steering model is needed and shall be presented in chapter 3.

The hardware- and low-level control-system of the hydraulic actuators is in place already and thus will not be part of this thesis. It is supplied as turn-key software by the manufacturer and readily available on the dolly's Electronic Control Unit (ECU). Substantial modifications are necessary to achieve the desired goals, though. The ECU's software version will be available fully calibrated and parametrized for the dolly at hand and thus provide a reliable working base to build upon. This legacy system is outlined in chapter 4.

Braking the dolly's axles individually will not be part of the testing and development scope of this work. Nevertheless the conceptual solution for including this functionalities furtheron will be outlined and a foundation for sending actuation request to these systems will be included in the rapid-protoyping interface block resulting from this thesis.

1.4 Structure of this thesis

In the first two sections of this thesis a brief overview of the legal situation concerning HCT vehicles for different countries, the current state of the art and ongoing research in the field of HCT vehicles shall be presented (chapter 2). Furthermore an introduction to the steering controller model, that will be run on the rapid-prototyping system will be given (chapter 3). Those two chapters are meant to give an introduction into the matter and are mainly based on literature review.

In the succeeding chapters the conducted development work will be described in detail. As a loose guiding structure for this work the development process after the V-model (see 2.3) will be referred to. Starting with a description of the utilizied hardware-systems and their interconnections in chapter 4, followed by detailing the different software-tools and environments running on those hardware-platforms in chapter 5. After the development work was described the created (sub-)systems will subsequently be validated and peu-à-peu put together and following the V-model evaluated at different stages of integration (chapter 6). As at the planned high speeds and great inertia for testing safety is a major concern, safety

functions will be implemented and systematically evaluated. This will be outlined and discussed in chapter 7.

The work closes with a discussion of the results collected during testing (chapter 8) and a conclusion in chapter 9 where the authors will try to give recommendation for practical implementation and outline future research work in the field.

2 Overview

2.1 Research on High Capacity Transport vehicles

There is a lot of research going on regarding HCT vehicles. The main topics are the safety of HCT, their impact on road infrastructure and the environment as well as economical aspects. There are many concerns about the safety of HCT in public opinion, so many studies investigated this issue. One concern is the extended length of HCT, which causes an increase of time needed to overtake a truck combination. Studies showed that accident risk for overtaking a long combination is not statistically higher than for standard combinations[1]. For overall safety, the majority of studies show that the use of HCT increases traffic safety due to the reduced number of vehicles on the roads, which is induced by the higher capacity of HCT vehicles. However, HCT might have a higher accident rate for individual vehicles [1]. In another study crash data from the Swedish Traffic Accident Data Acquisition was analyzed. The results showed that there is no higher rate of fatal or severe crashes per traveled vehicle km for HCT vehicles compared to standard vehicle combinations[3].

To prevent any negative effects on safety, which might occur with the use of HCT, different safety systems are being developed.

One aim of safety systems for HCT is to reduce off-tracking of the last trailer for low speed cornering[1]. This issue can be addressed by using a steerable dolly. In section 3.1 a controller to improve the path-following is described.
Another problem of HCT is the rearward amplification[2] of yaw rate and lateral acceleration. A possibility to achieve that is also to use a steerable dolly. In section 3.2 a controller to reduce rearward amplification for high speed is described.

When it comes to the impact of road transport on road infrastructure, road wear is the most important effect. The main factor of road wear is the axle load of the vehicles. The following equation for a load equivalency factor shows how different axle loads impact road wear.

$$\frac{N_{ref}}{N_x} = \left(\frac{W_x}{W_{ref}}\right)^4 \tag{2.1}$$

Where W_x and W_{ref} are axle loads and N_x and N_{ref} are the corresponding numbers of load applications[13]. Equation (2.1) shows that for example doubling the axle load leads to a road wear that is 16 times higher.

Assuming an equal distribution of the load over the different axles, the axle load for a vehicle is calculated as follows:

$$W_x = \frac{m_{tot}}{n_{axl}} \tag{2.2}$$

Where m_{tot} is the total mass of the vehicle and n_{axl} is the number of axles. A standard tractor-semitrailer combination has five or six axles and a maximum mass of 40 tonnes. With

[1] Off-tracking means that the trailer doesn't follow the path of the tractor but moves on a radius that is, when using a passive converter dolly, smaller then the tractor's.

[2] In truck combinations there is the effect that lateral acceleration and yaw rate the different units experience increases the longer the distance between those unit and the tractor gets. This effect is called rearward amplification.

equation (2.2) the axle load, if fully loaded, for that combination is 8 tonnes/axle and 6.67 tonnes/axle, respectively. An A-double-combination, which has ten or eleven axles and a maximum mass of 80 tonnes, also reaches a maximum axle load of 8 tonnes/axle and 7.27 tonnes/axle, respectively. For a 25.25 m combination, which has nine or ten axles and a maximum mass of 60 tonnes, the maximum axle load is even lower than for the standard combination. The maximum axle load for this case is 6.67 tonnes/axle and 6 tonnes/axle, respectively.
Based on that it can be concluded that high capacity transport vehicles don't lead to higher road wear than standard combinations.

Since one of the reasons for using of HCT vehicles is the expected positive influence on the environment compared to standard combinations, several studies that address this topic can be found. Overall these studies showed that with the use of HCT vehicles the environmental impact of goods transportation can be reduced significantly for high loading factors. The fuel consumption per transported unit of cargo is about 15% lower compared to the standard vehicle combinations used in the EU. In the number of trips a reduction of 32% can be seen. This reduction is achieved because two HCT vehicles can carry the same amount of goods as three standard vehicle combinations[2]. However, the environmental impact of HCT vehicles can become negative if due to a high penetration of HCT vehicles goods transportation is shifted from rail to the road[7].

The introduction of new systems is usually driven by economical aspects, especially in the transportation industry, hence positive effects of HCT vehicles are expected. First of all, the transport costs decrease by 1.8-3.4%. Together with the environmental costs, that are lower compared to standard combinations, an overall benefit of using HCT vehicles can be seen[1].

2.2 Market overview for existing solutions

There are some solutions for dollys with steerable axles that exist today. For those solutions however, the steering angles of the axles is only based on the articulation angle between the dolly and the trailer or truck it is attached to.
One of these solutions is the *active articulated dolly* manufactured by *Fahrzeugwerk Bernard Krone*. The front axle of this dolly is steerable. In order to achieve that, the drawbar of the dolly is mechanically connected to the frontaxle and is rotateable. So if the drawbar is turned the wheels of the front axle steer. The transmission of the steering can be varied by changing the length of the adjustable drawbar. For safety reasons the steering is mechanically locked automatically depending on the velocity. The main reason for implementing this steering capability is to reduce off-tracking. This dolly is equipped with air suspension, an electronic breaking system as well as a stability system[12].
Other manufacturers that offer steerable dollys, that work the same way, are *Schmitz Cargobull* and *Kögel Trailer*[18][29].
In order to use the dolly to improve the vehicle dynamics of a combination in a more advanced way, a solution that allows independently steering is preferable. The dolly used in this project will be capable of steering both front and rear axles individually and independent of the articulation angle between the dolly and the trailer in front of it after it is equipped with the systems outlined in this thesis.

2.3 Rapid Control Prototyping

In automotive software development standardized processes are applied to assure structured work. The V-model which derives its name from the characteristic shape constitutes one graphic representation of the sequence of steps towards the finished overall system. It also gives an approximate impression about the level of detail of each of the steps on the vertical axle, as well as chronological progress on the horizontal. A simplified version of this V-model can be seen in figure 2.1. First the user or customer needs are analyzed and put into a schematic formulation of the system showing the logic dependencies and underlying physical working principles. This is usually done in a semi-standardized fashion utilizing Unified Markup Language (UML) or similar diagrams, preventing non-specific prose to ensure less room for mis-communication and better readability. Technical decisions and limitations are not yet considered. After the logical system and dependencies have been established they will be analyzed and broken down to actual technical system descriptions defining also which subsystem will take over which function, what parts are to be realized in software or hardware (e.g. filtering, safety functions). The different parts of the technical system are then specified in detail, defining interfaces between different software sub-system, operational states and distribution of functions over sub-systems. The fourth step will define the specifications for the software in details for example: data types, control sequences, decision structures, real-time behavior. Finally the actual design step will implement these specification with regards to the hardware limitations (RAM/ROM, processing power), computation methods, data handling (detailed format, variable types, utilization of parameters or variables). To compile the final system from the component-level sub-system a similar hierarchy is used in inverse. The corresponding steps of the project definition and specification phase (left side of the schematic) are supposed to provide test-procedures and goals. If necessary after insufficient test outcomes, another iteration of the previous step will be performed (vertical iterations). A similar sequence of development steps is applied to develop the hardware platform in parallel[26].

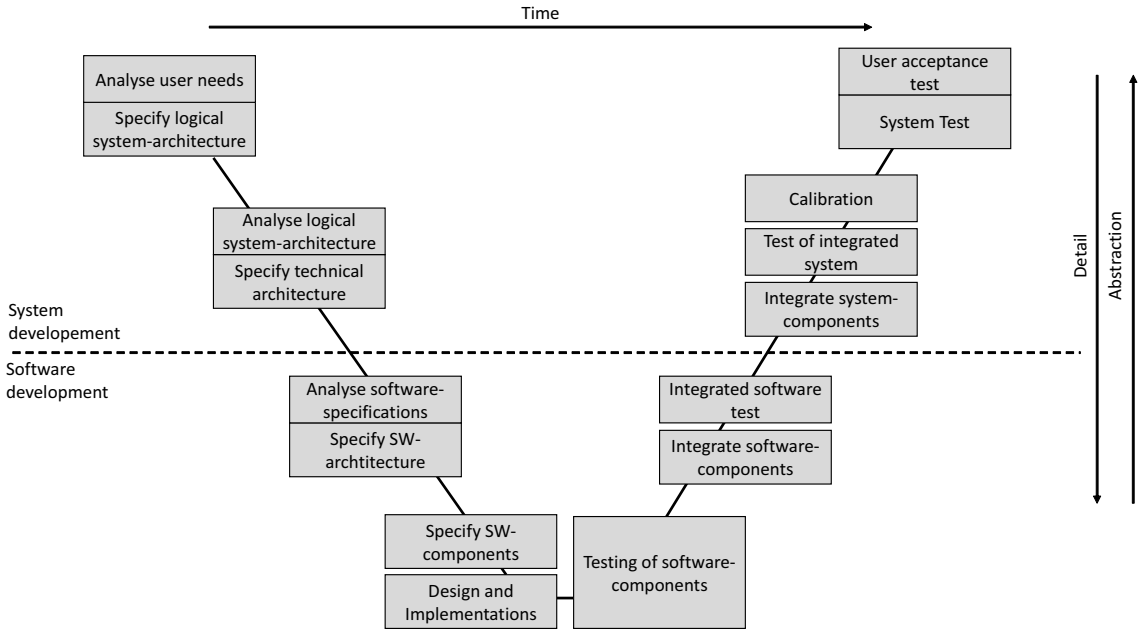


Figure 2.1: Overview for a process in automotive software development after the V-model

The increasing complexity of mechatronic systems and shorter product life-cycles lead to new development methods that made it possible to take 'short-cuts' in the established V-model process. This methods are summarized under the term rapid prototyping - in the case of software functions for mechatronic systems the term Rapid Control Prototyping (RCP) was coined. It is possible to conduct testing and verification with early or abstract soft- and hardware versions which are still under development by having the remaining system and environmental influences simulated by the RCP-platform. In utilizing these methods such as rapid control prototyping and software/hardware-in-the-loop testing, a tremendous decrease in development time can be achieved. It is also possible to validate specifications early in the process, eliminating possible cost-intensive changes in later stages[25]. For the research project in which this theses is embedded it even opens up the possibility of on-road testing, as it eliminates the low-level development steps of the actual implementation and some of the detailed design work. The implementation of the steering algorithm (see section 3) on a stand-alone ECU, hydraulic actuator control, hardware layout, etc. would by far exceed the dimensions of a research project. Applying rapid control prototyping is the only feasible way to establish a functioning on-road prototype vessel.

2.4 Functionality architecture

To describe the different logical subsystems, that constitute a vehicle, the driver and the environment, a functionality architecture developed at Volvo GTT was used. The different parts of this architecture are shown in Figure 2.2.

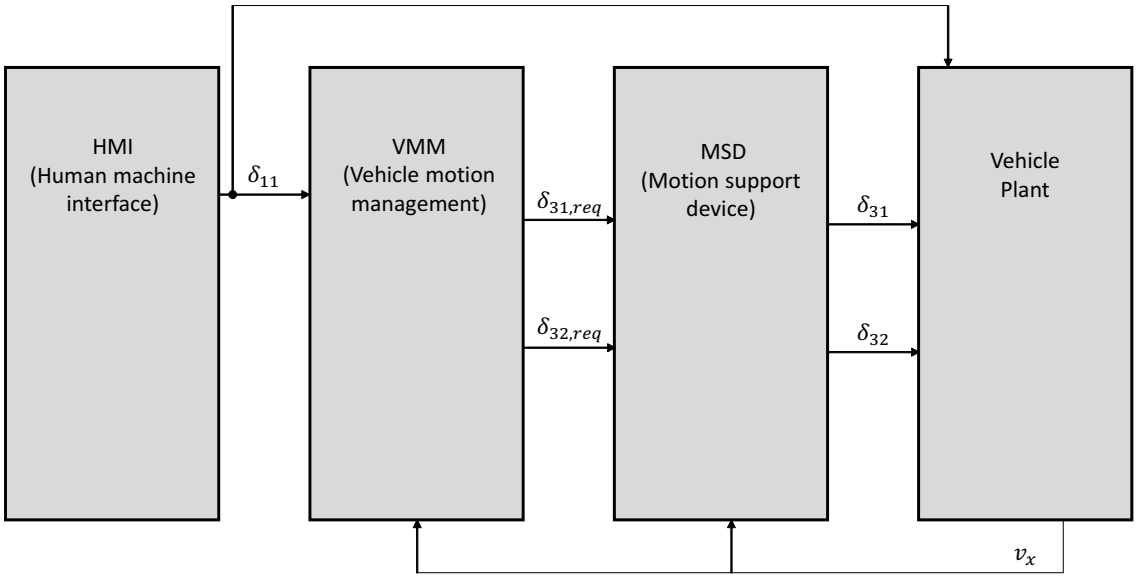


Figure 2.2: Functionality architecture

The functionality architecture consists of four different parts: The Human Machine Interface (HMI), the Vehicle Motion Management (VMM), the Motion Support Device (MSD) and the

vehicle plant.
The HMI is the interface between the driver and the vehicle. The driver can input requests to the vehicle, for example a steering angle via the steering wheel or a torque request via the accelerator pedal. The driver gets feedback from the vehicle about the vehicle's state, for example via the speedometer, displays but also via vibrations, movements and accelerations of the vehicle.

The VMM contains the high-level controllers for the vehicle's motion and stability. The time horizon, in which the VMM acts, is up to one second[20]. As an input it gets the request of the driver from the HMI and transforms these requests into requests for the actuators of the vehicle. These request are then sent to the MSD. From the MSD the VMM receives information about the state of the actuators and the current capabilities.

The MSD contains the actuators and low-level controllers of the vehicle. In there the requests from the VMM are executed. Information about the status of the actuators is sent both to the VMM and the vehicle plant.

In the vehicle plant it is defined how the vehicle reacts on different inputs in terms of vehicle dynamics.
In different kind of tests, the parts of the functionality architecture are either simulated or real hardware is used. The more advanced the development process is, more and more of the functionality is shifted from simulation to actual hardware.

3 Vehicle Model controllers

The vehicle model controllers that were used in this thesis were developed in a research project at Chalmers University of Technology. There are two different controllers used to control the steerable axles of the dolly. One is used for low-speed and one for high-speed maneuvers. The main purpose of the low-speed controller is the reduction of off-tracking. For the high-speed controller the aim is to reduce the rearward amplification.

3.1 Low-Speed controller[15]

The low-speed controller is implemented in path-distance domain η. The path-distance is calculated by integrating the forward speed $v_{x,1}(t)$.
The steering angle of the first axle of the dolly is calculated as follows:

$$\delta_{31}(\eta) = k_1\delta_{11}(\eta - \eta_1) + k_2\Delta\psi_1(\eta - \eta_2) + k_3\Delta\psi_2(\eta - \eta_3) + k_4\Delta\psi_3(\eta - \eta_4) \tag{3.1}$$

where δ_{31} is the steering angle of the dolly's front axle, δ_{11} is the steering angle of the tractor's front axle, $\Delta\psi_1$ the articulation angle between the tractor and the first semitrailer, $\Delta\psi_2$ the articulation angle between the first semitrailer and the dolly, $\Delta\psi_3$ the articulation angle between the dolly and the second semitrailer, η_1 the distance between the tractor's first axle and the dolly's first axle, η_2 the distance between the first articulation joint and the dolly's first axle, η_3 the distance between the second articulation joint and the dolly's first axle, η_4 the distance between the third articulation joint and the dolly's first axle, and k_1, k_2, k_3, k_4 are gains.
The steering angle of the second axle of the dolly is calculated using the steering angle of the first dolly axle as follows:

$$\delta_{32}(\eta) = k_5\delta_{31}(\eta - \eta_5) \tag{3.2}$$

where δ_{32} is the steering angle of the dolly's second axle, η_5 is the distance between the dolly's second and first axle, and k_5 is a gain.
The gains k_1, k_2, k_3, k_4, k_5 were optimized using particle swarm optimization. Table 3.1 shows the improvement achieved with the controller for different maneuvers compared to a simulation without steering of the dolly.

Table 3.1: Performance improvement with low-speed controller

Maneuver	**Improvement [%]**
180-deg turn	+44.61
90-deg turn	+37.75
S-turn	+45.07

3.1.1 Input parameters

The input parameters for the low-speed controller are:

- The articulation angles between the different units: $\Delta\psi_1$, $\Delta\psi_2$, $\Delta\psi_3$
- The steering angle of the tractor's first axle: δ_{11}

- The distances of the tractor's first axle, the articulation joints of the different units and the dolly's second axle to the dolly's first axle: $\eta_1, \eta_2, \eta_3, \eta_4, \eta_5$
- The vehicle speed for the transformation from time to path-distance domain: v_x

They are constants or can be easily acquired via articulation sensors or are present on one of the Controler Area Network (CAN)-buses of the truck.

3.2 High-Speed controller[16]

The high-speed controller is implemented using a nonlinear inverse of a dynamic vehicle model. The aim of the controller is to copy the movement of the first unit to the following ones at a certain point. Therefore the movements of the following units need to be delayed by the time it takes them to reach this point. For the dolly unit, this delay τ_{13} is calculated with the vehicle speed v_x and the distance l_{cg13} between the Center of Gravity (CoG) of the tractor and the dolly as follows:

$$\tau_{13} = \frac{l_{cg13}}{v_x(t)} \tag{3.3}$$

With this delay and the lateral acceleration of the tractor a_{y1} a reference signal for the dolly's lateral acceleration is calculated as follows:

$$a_{y3ref}(t) = a_{y1}(t - \tau_{13}) \tag{3.4}$$

This reference signal, together with the steering angle of the tractor, the forward speed of the vehicle and the actual steering angle of the dolly, is used as inputs to an inverse single track model, that is modeled in the Modelica language. The outputs of the model are the request steering angle δ_{dolreq} and the lateral acceleration a_{y3} of the dolly. Figure 3.1 shows a block diagram of that model.

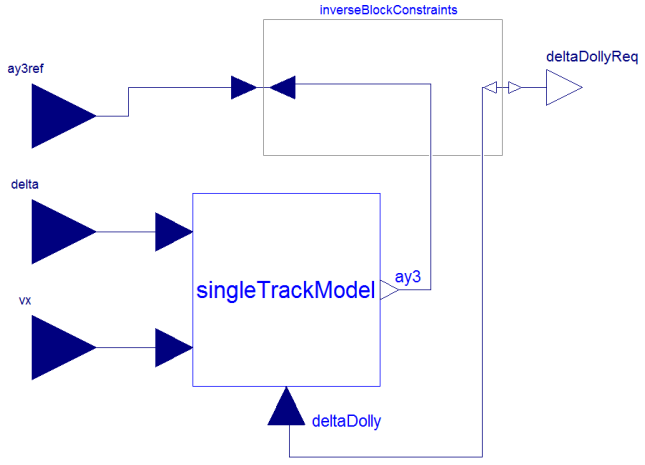


Figure 3.1: Inverse of the single track A-double model [16]

To determine the performance of this controller, simulations of a single lane change at 20 meters per second and a sine wave steering input of 0.4 Hz with an amplitude of 0.026 rad were done. As a result the rearward amplification of the yaw rate of the second trailer is 0.95 with this controller compared to 1.49 for a passive dolly. Furthermore the transient off-tracking is improved by 67%.

3.2.1 Input parameters

Figure 3.2 shows the input and outputs of the inverse model controller.

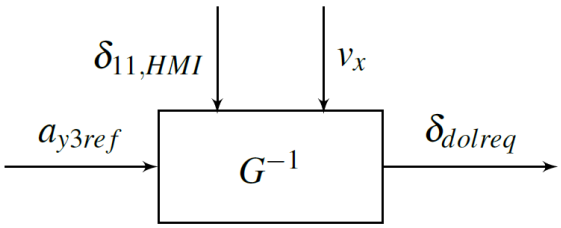


Figure 3.2: Inverse model controller[16]

The input parameters of the high-speed controller are:

- The vehicle speed: v_x
- The steering angle of the tractor's front axle: δ_{11}
- The reference signal of the lateral acceleration of the dolly: a_{y3ref}

4 Hardware architecture

4.1 Hardware overview

As already outlined in chapter 2 the HCT vehicle used in this project consists of a truck with two semi-trailers. To achieve the goals of this thesis, modifications had to be done to the combination as well as equipment mounted in addition to the legacy fixtures. A scheme of the utilized combination can be seen in figure 4.1, the shown unit number will be referenced further-on to give easier insight into the implemented systems' mounting points.

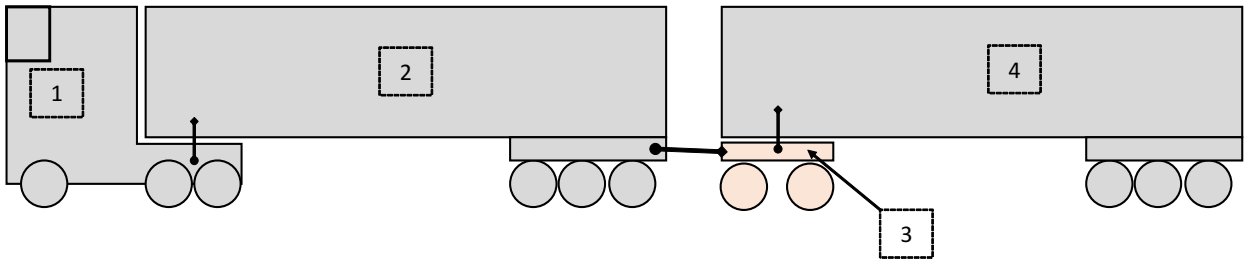


Figure 4.1: Positioning of different sensors on the High Capacity Transport vehicle

The first semi-trailer's (unit 2 in figure 4.1) rear needs to be equipped with a drawbar eye to accommodate the dolly's drawbar. Besides this obvious physical requirement power supply for the dolly and second trailer, pressurized air hoses for brake actuation, e-brake CAN-buses and lighting connectors have to be routed through the first trailer and supplied to the second trailer with a sufficiently robust cable. This solution had to mainly be custom-made, as these appliances usually would not be available in the rear of a semi-trailer as usually no further towed unit is present. These installations would also apply for any passive combination or for the legacy V.S.E. Vehicle Systems Engineering B.V. (VSE) Electronically-controlled hydraulic Trailer Steering (ETS) system. Furthermore CANs have to be patched through the first semi-trailer (see section 4.5). The Real-Time system will be mounted on the dolly directly (position 3) to ensure short signal paths and minimize wiring efforts. However the Host-PC will have to sit in the tractor (position 1) to sufficiently protect it against the track environment, which would be impossible if sitting on the dolly directly and to allow the test-operator to monitor the system during run-time while driving. This leads to an additional Ethernet-cable for the Host-PC connection in the cabin which also needs to be routed through the first semi-trailer. The second semi-trailer does not require these modifications, as it is merely a passively towed unit.

Figure 4.1 also depicts another important aspect. Round bullets on the unit-connecting lines indicate a connection with one rotational degree of freedom (DOF) square bullets mean fixed connection. As can be gathered from the figure there are three DOF in the whole combination: between tractor and first semi-trailer, between first-semitrailer and the dolly, and between the dolly and the second semi-trailer. As the angle between the different units greatly influences the steering performance of combination, the angles have to be accounted for in the simulation and of course also the steering algorithm. It is thus necessary to measure them and provide the measurings via CAN to the rapid-prototyping system which is described in great detail

in chapter 3 for the steering model-side and in chapter 4 for the hardware details. These articulation angle sensors also need to be wired from their mounting point to the dolly, where they are fed into the rapid prototyping environment.

4.2 Utilized dolly system

The utilized dolly by Parator Industri AB (Parator) is equipped with two steerable axles. They are controlled by an after-market solution called ETS supplied by VSE, of which figure 4.2 gives an overview.

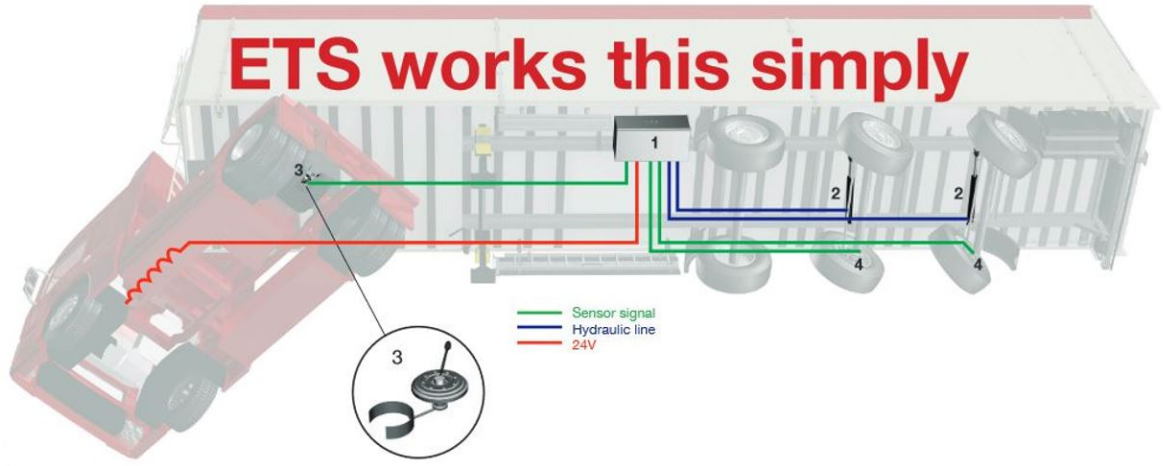


Figure 4.2: Active dolly legacy steering system supplied by VSE [4]

Their product includes sensors, ECUs and hydraulic systems which come in a ready-to-mount housing, which is placed on the trailer/dolly. The box, that includes the ETS-system, can be seen in Figure A.2 mounted on the dolly. This solution is usually sold as a low-speed active steering system for truck-trailer combinations to provide better maneuverability at low speeds in inner-city areas. Besides electrical power and compressed-air (see '2' in the figure) supply there is no connection with the truck in place. This allows for use with many different truck/trailer Original Equipment Manufacturer (OEM)s, as no insight into proprietary CAN-communication is needed. In the original VSE system the two parameters that influence the actuation of the dolly's steering are vehicle speed ('4') and kingpin-deflection ('3'). This deflection is the angle between the truck and trailer, which is measured by an additional kingpin-angle sensor supplied by VSE and mounted in the eye of the kingpin hub. Furthermore every steering-knuckle of the dolly is equipped with an angle sensor to provide appropriate wheel-individual feedback for the VSE control-system. There is a static dead band in the steering system for articulation angles between -2° and 2°. For those angles the ETS doesn't steer.

A diagnosis screen is available in the VSE-unit, which allows for relatively simple set-up, calibration and parametrization to be done[4]. An overview of the implementation of the logical signal paths in the ETS-system in the dolly can be seen in figure 4.3.

VSE's ETS provides assisted steering up to a speed of 25km/h over which the possible

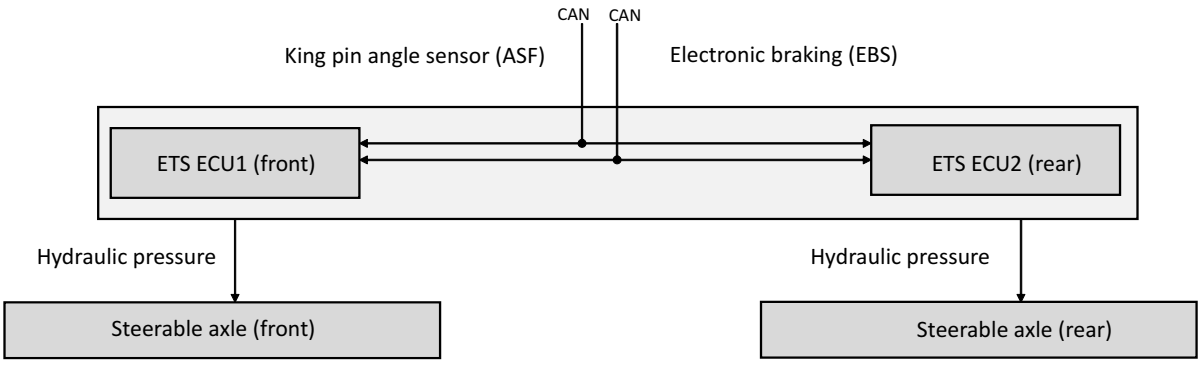


Figure 4.3: System overview for sensors and signal path for Electronically-controlled hydraulic Trailer Steering legacy system by VSE

steering angle is limited until it reaches zero at 55km/h. At this threshold the steerable axles are locked and thus behave like normal passive axles. This according to the manufacturer is to ensure stability at higher speeds[4]. Locking the steering at higher speeds leads to a more predictable behavior for the user and system robustness. However, performance during high speed maneuvers be improved by uniformly steering the dolly as well[17]. The desired demonstration of the algorithm presented in chapter 3 will as outlined in the introduction to this thesis, include steering at higher speeds, thus a work-around had to be established.
Table 4.1 provides an overview of the mechanical properties of the dolly.

Table 4.1: Properties of dolly

Length	6192 mm
Width	2600 mm
Maximum load total	18,000 kg
Maximum load front axle	9,000 kg
Maximum load rear axle	9,000 kg
Maximum load drawbar	500 kg
Maximum load fifth wheel	18,000 kg

4.3 Interfaces and connections with dolly

Figure 4.4 gives an overview of all the in- and outputs of the dolly.

To the first semitrailer the dolly is connected with a standard 7-pin ISO 7638-2 connector.

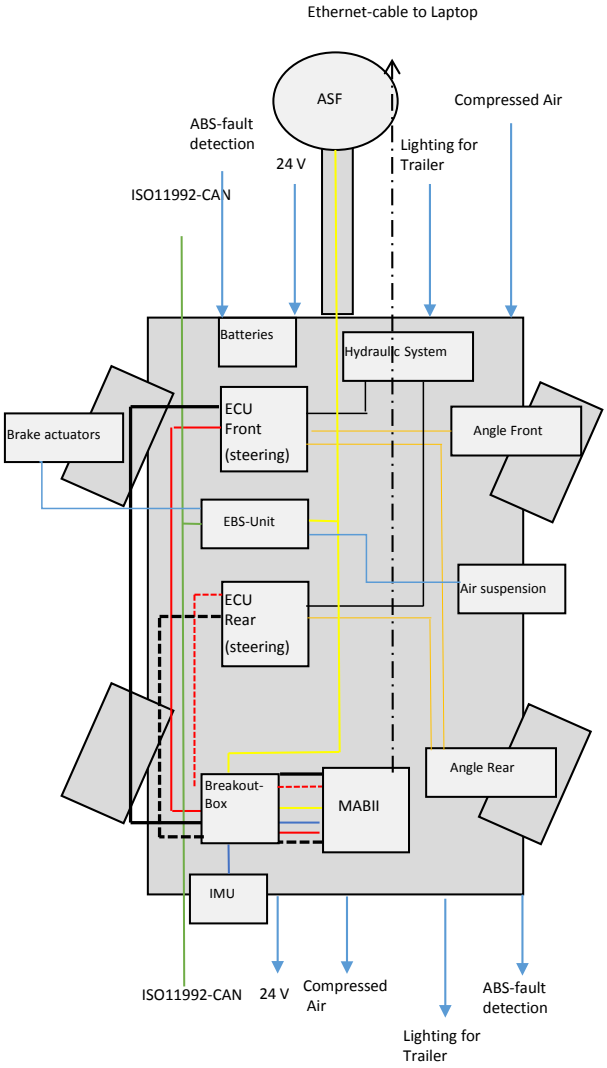


Figure 4.4: Interfaces and communication systems on the dolly

This connector includes 24 V electrical power, a signal for Anti-lock braking system (ABS) fault detection as well as the ISO-11992 CAN (see section 4.4). There is a second connector, which includes different pins for different kinds of lighting on the dolly, like the turn indicators and brake light. The dolly also receives compressed air from the first semitrailer for its braking system and air suspension. For controlling the MABII an Ethernet connection is established between the dolly and the tractor unit, where the Host-PC is located. The interfaces of the dolly with the second semitrailer are the same as those to the first one. To control and measure signals of the dolly a connection to the dolly's ETS- and EBS had to be established. To achieve this, different kinds of CAN-buses had to be used. That was necessary because the various systems of the dolly use different CAN-standards. In total five CAN-buses connect the MABII with the dolly. Two for each of the ETS-ECUs and one for the ASF- and EBS-Signal.

One of the CAN-buses contains the signals from the EBS-ECUs and the ASF- sensor, which measures the angle between the dolly and the first semitrailer. It uses the extended CAN-standard at a baud rate of 250kbit/s . In the original state this CAN is directly routed to the CAN of the two ETS-ECUs (see section 4.2). In order to achieve steering that is both

independently from the articulation angle and the velocity of the vehicle the connection between the ASF/EBS-CAN and the CAN of the ETS-ECU was cut. Instead the ASF/EBS-CAN is connected to the MABII. Figure 4.5 shows an illustration of that.

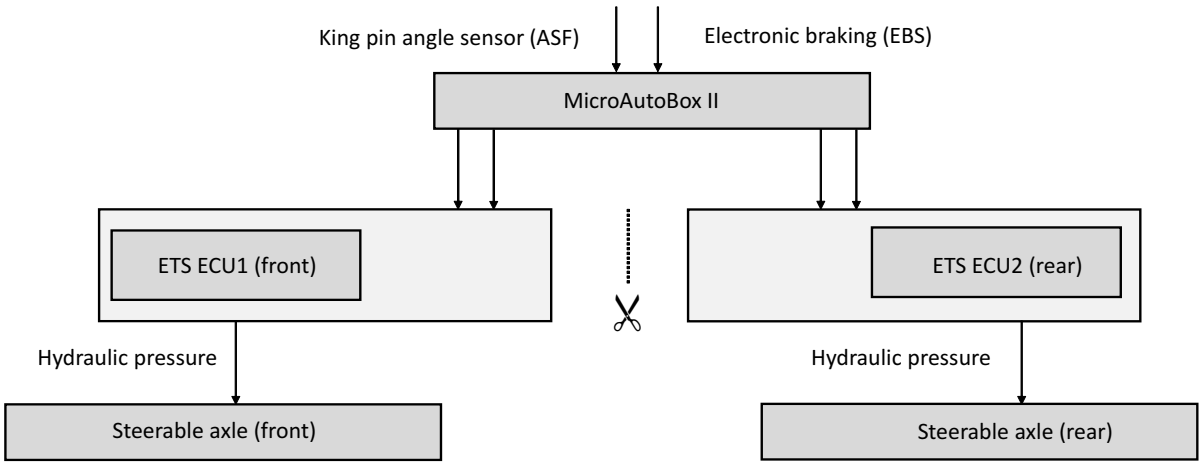


Figure 4.5: Split of ETS-CAN

The MABII is connected to the CAN bus of each ETS-ECU. Because the CAN standards of the ASF/EBS-CAN and the ETS-ECU-CAN, which uses the J1939 protocol, differ, two CAN ports are needed on the MABII for each of the two ETS-ECUs. Those two CANs are merged inside the breakout-box. Figure 4.6 shows the breakout box which is used to as an interface between the different CANs that were used and the MABII. The merging of the two CANs that contain signals for the ETS-ECU can be seen on the right side of the scheme.

4.4 Braking system actuation

The dolly is equipped with an EBS by WABCO. It is a so called 4s2m system, meaning that it as one speed sensor on each wheel ('4s') and two modulators ('2m') to apply brake-pressure to the axles. The EBS is actuated via the ISO-11992[1] standardized CAN-protocol. Physically this CAN is connected via a 7-pin connector coming from the truck (see also figure 4.4). It is possible to request braking-torques electronically via CAN. Legislation requires every brake system to be redundant and have to actuation paths. The EBS therefore also needs a pressure level high enough on the pneumatic request-line coming from the truck, as in older systems,

[1]Standard for communication derived from CAN between truck and towed units and systems mounted on them, such as brakes, status messages, leveling systems, lighting. Both hardware level for communications and protocol details are stated.

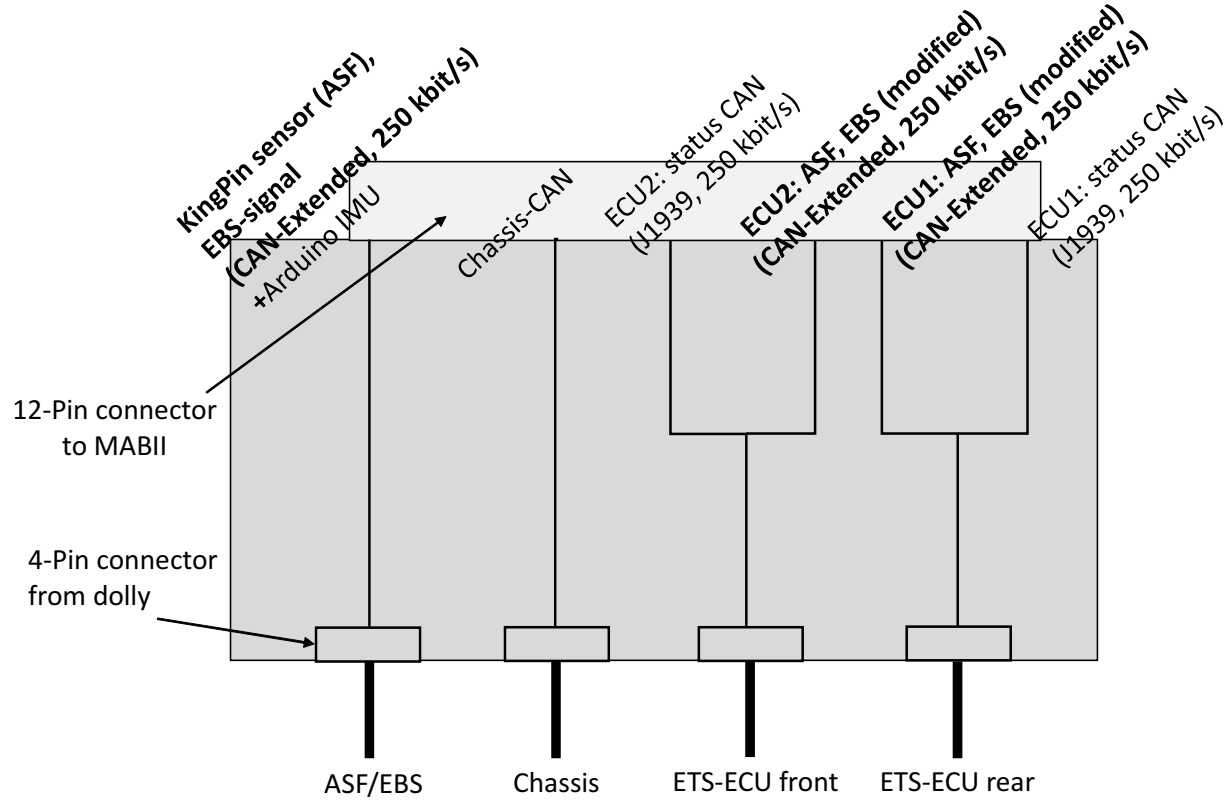


Figure 4.6: Breakout box scheme

where this is the only method of actuation for combinations. So it is not possible to request brake torques solely electronically via CAN. Installing an electronic valve and a mechanisms as depicted in figure 4.7.

It would allow to command one electronic valve directly from the electronic outputs of the MABII, putting pressure from the dolly's own reservoir directly on the brake-request line and therefore enabling braking request via ISO-11992 CAN-messages. Furthermore the inclusion of the original brake request pneumatic line coming from the truck guarantees an intact system where the driver still can stop the vehicle via normal brake-actuation without any differences to the legacy system. The check valves in the depicted solution ensure, that both actuation paths do not interfere with each other.

4.5 Interfaces with truck

The vehicle model controllers depend on inputs from the tractor unit (see section 3). One of the inputs is the steering angle of the tractors front axle, another the current vehicle speed. For track-testing the MABII is connected to the chassis CAN of the tractor, in order to receive those signals. Since the chassis CAN of the tractor is not usually available on trailers, it needs to be accessed somewhere in the tractor and routed all the way to the dolly. For the Hardware-in-the-Loop (HiL) -testing the tractor unit is simulated using Virtual Truck Model

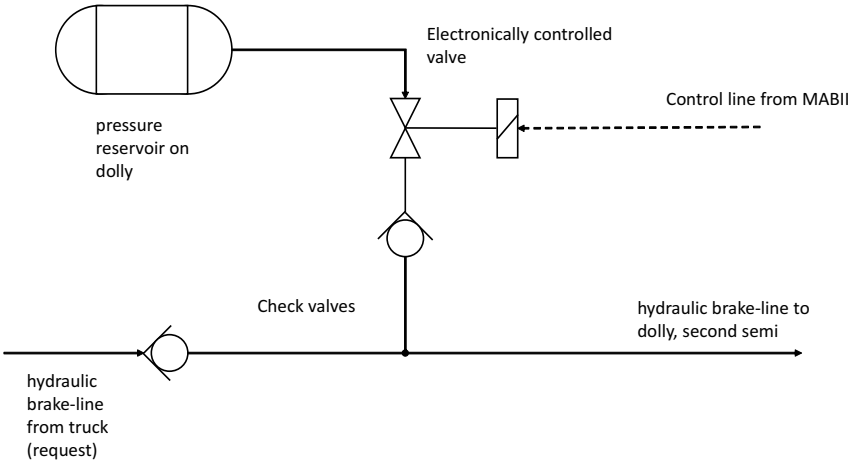


Figure 4.7: Pneumatic system for EBS

(VTM) (see section 5.1.2). As already mentioned in section 4.3 the dolly gets power supply and compressed air indirectly from the tractor over the first semitrailer.

4.6 Arduino Due

The Arduino platform series was developed by SmartProjects company in Italy and consists of a micro-controller board with several in- and output pins, and a set of programming tools to program said micro-controller (see also section 5.4). Hardware layouts as well as software of this solution are released under an open-source licence, so changes at any level of detail can be made and many other suppliers are now offering solution that tie in with the Arduino. Furthermore this lead to a broad user base and availability of vast supporting resources. For a number of tasks in this thesis, it was decided to rely on this cost-efficient and flexible microcontroller platform. Namely those purposes are:

- data collection from Inertial Measurment Unit (IMU) and GPS and transmission over CAN
- CAN-to-UDP gateway
- CAN-to-Serial gateway for HiL -testing
- benchtesting sensor outputs and quick CAN-debugging

SmartProjects offers various Arduino boards mainly differing in number of communication pins, memory size, clock frequency and physical size. It was decided to utilize the to date most powerful Arduino Due, offering 84 MHz of clock-speed on a ARM Cortex-M3 processor, 512kB flash memory for program-code and a 96kB working SRAM. It is the first Arduino with 32-bit architecture. This was done in order to allow for future use of this platform in other projects and having enough space for different sub-programs on the controller as well as enough processing power to deal with basic signal filtering and parsing at appropriate speeds.

The Due has a native Inter-Integrated Circuit (I^2C) interface, which was used to gather the measurings from the IMU. It also has a Serial Peripheral Interface (SPI) which is needed to access the ethernet controller necessary for the CAN-User Datagram Protocol (UDP)-gateway (refer to section 4.7.1), serial communication is also handled on hardware level. The availability of these ports in hardware form allow for robust systems and eliminate the need to implement those protocols on software level (bit-banging), which frees up memory resources. In addition to this digital communication possibilities the Due offers an abundance of 54 digital and 12 analog freely configurable I/O-pins. The Due is also the first Arduino to host an on-board CAN-controller, which in this thesis will be used for measurement transfer to the logging system (MABII), eliminating the need for additional hardware for CAN-bus interfacing on the Arduino side.

Besides the power-supply and the IMU-chip a MCP2551 CAN-transceiver by Microchip Technology Inc. was incorporated to take care of the pyhsical layer of CAN-bus communication by converting the digital signals from the Due's CAN-controller to the standardized voltage levels of the CAN. The MCP2551 is capable of different CAN standards and fully ISO-11898 compatible, which makes future use in different environments or as unit for in-vehicle CAN-interfacing feasible.

The Arduino Due was used to determine the delays in the system (see section 6.3.2) as well as a light-weight solution to quickly read CAN-outputs of the various systems during this thesis' work. This proofed to be an easy debugging solution.

4.7 Real-Time Environment

To run the utilized steering controller outlined in chapter 3, a MABII by dSPACE company is utilized, an embedded computer running on a 900MHz PowerPC CPU with 32MB RAM. The MABII has very compact dimensions of only 200mm x 225mm x 50mm making it possible to fit it inside the housing locker of the VSE ETS (position 3 in figure 4.1). It accepts a wide variety of input currents for power supply, is shock resistant to a high degree and equipped with sturdy LEMO connectors; the unit is cast into very robust aluminium housing and has mounting points, so it can be utilized in tougher conditions found in automotive applications. It boots up almost at the speed of a standard ECU and can run headless ensuring safety, even without the Host-PC to control it. Besides normal digital and analog in- and output pins, CAN, Ethernet and many other standard interfaces are readily available, making for a practical use of these protocols on a high level.

The connection to the various CAN-buses for this thesis' purposes is established via the dSpace Zero Insertion Force (ZIF)-connector. This is a specially made connector that allows to access all of the 64 in- and output-pins of the MABII and quickly plug and unplug the MABII. Ethernet connections with reliable LEMO connectors are used to communicate with the Host-PC which runs logging and start-up sequences during testing. Power-supply is realized from the buffer batteries of the hydraulic steering system of the dolly during testing and a standard laptop charger during bench- and HiL-testing.

The hardware platform is part of an integrated chain of software tools to develop functions and models in the sense of rapid control prototyping. It integrates with MATLAB's Simulink and compiles and uploads executable code of the abstract Simulink-models directly into the MABII's program memory. Furthermore a tool (ControlDesk 4.2) to log and manipulate the executed simulation running on the MABII is used on the connected Host-PC. This software

tools will be detailed in section 5.1 and 5.3[11].

4.7.1 CAN-bus extension

The MABII comes with a preset number of in- and output ports. The maximum number of CAN-buses, that can be connected to the MABII's native controllers is limited to six. As there is no off-the-shelf solution by dSpace for extending the available bus-connections, it was necessary to come up with a gateway solution that allows to patch the needed amount of additional CAN-buses through to the ethernet connection which is also available on the MABII. It should be mentioned that this issue with lacking CAN-buses mainly arises due to the limitation, that buses have to be kept separate as shown in figure 4.5. Without this limitation it would of course be possible to merge the different physical CAN-bus lines to one line.

A gateway from CAN-protocol to UDP used for communication on Ethernet infrastructure was implemented to achieve this. It is a very light-weight protocol, that is straight-forward to implement and runs well, even with limited processing power on a microcontroller. For future purposes the broadcasting capabilities of UDP might also proof useful, as many nodes could be connected to this CAN-to-UDP gateway all receiving the broadcasted datagrams for example for visualization on different computers or additional logging outside of the MABII environment. One limitation that was decided on, is to have receiving capabilities only for the MABII to eliminate the need for extensive computation and CAN-matrix handling on the Arduino.

Excursus: The CAN-protocol is the most widespread protocol to allow for communication between different ECUs in the automotive field. Development began at the Robert Bosch GmbH, but is now standardized and enhanced and adjusted for special purposes internationally. Messages are broadcasted by the bus-participants and stamped with their unique identifier, which also doubles as an arbitration token to handle message prioritization. Each bus-participant can listen to all available messages and "only picks from the bus what he needs". The standard message has a size of eight bytes à eight bit, transmitted after the identifier and followed by an ending sequence. Hardware-wise CAN-communication relies on only a pair of twisted wires, where a very robust voltage difference signal is transmitted[5].

By utilizing UDP to acquire data into the simulation, the MABII environment's very convenient possibility to incorporate CAN database (.dbc)-files[2], which is the usual way to exchange information and instructions for CAN-networks is no longer available. To decompose the UDP packets into the original signals it was necessary to implement the function of a .dbc-file in the underlying Simulink-model.
All messages that the CAN-gateway is supposed to handle and forward are put into one UDP frame in succession with respective CAN-identifiers and additional spacer bytes between signals. The UDP frame is broadcasted every time a new CAN-message reaches the gateway on one of its CAN-buses. Messages that were not updated with the incoming CAN-message will be kept at their previous value. This is called for, as UDP is a connection-less protocol without

[2]Correlates physical human readable/understable signals with units to the actual distribution over the different bytes of a CAN-frame. It allows to "decrypt" the information which is available on a CAN-bus and to code and send messages in the respective format that the other bus-participants (ECUs, sensors) expect.

any loss-prevention mechanisms like acknowledgment-handling or retransmission of messages. Holding the values if not available ensures, that at least some value is available on the bus. UDP also doesn't have native timestamping, which is why a Global Positioning System (GPS)-shield was introduced to receive the high-precision time to allow for easy synchronizing of logging over different platforms (MABII-environment, IMUs, external CAN) by adding the global time to each sent packet.

To allow for easier scalability to add more CAN-buses in the future it was decided to not use the Due's two native CAN-controllers but rely on the external MCP2515 in combination with the already mentioned transceiver (MCP2551). As shown in figure 4.8 it is controlled via SPI. More bus-participants can be added conveniently by incorporating additional instances of these two microchips and attaching them to a device selection pin (depicted as SS) each and the remaining SPI-bus lines. The depicted ENC28J60 microcontroller is an ethernet-controller which needs very little periphery to run and can be commanded via SPI. It has a send and receive buffer, taking away this task from the Arduino.

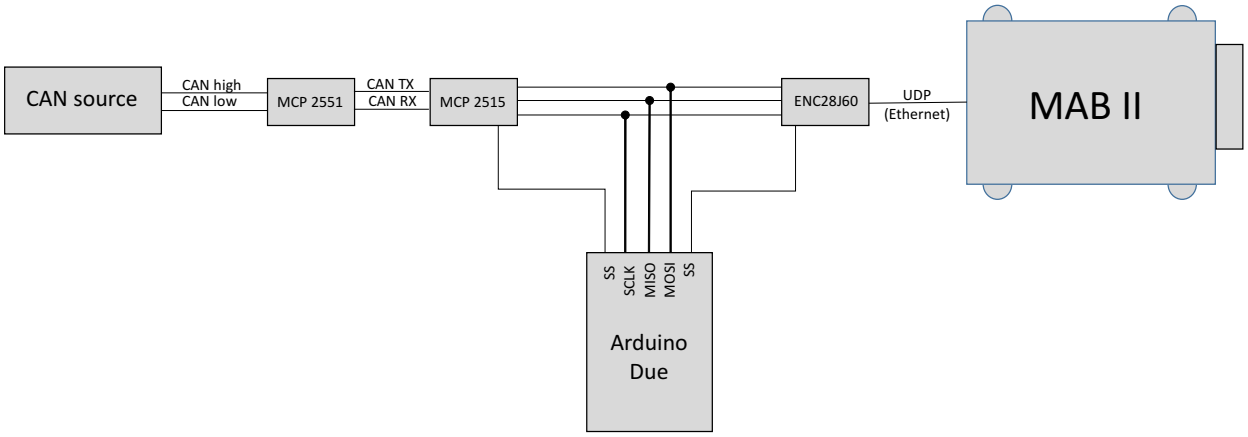


Figure 4.8: Extension of MABII CAN-buses (hardware overview)

4.8 Measurment Setup

4.8.1 On-board sensors

Table 4.2 gives an overview for the sensors that are in place on the legacy VSE ETS.

Sensor	Measurings	Type	Mounting point
Draw bar sensor	Articulation angle Temperature	External	Draw bar
EBS	wheel speed	External	wheel brakes
Steering angle sensor	steering angle (first, second axle)	External	steering knuckles

Table 4.2: Available sensors of VSE's ETS

4.8.2 Inertial measurement unit

To determine the processing delays in the control chain (refer to chapter 6.3) as well as logging implementation for verification and analyses purposes a number of inertial measurement units (IMU) were utilized throughout this thesis' work. The system at hand combined a gyroscope (L3GD20H), and an accelero- and magnetometer (LSM303D) into an IMU put on one circuit board[22]. This one-chip solution allowed for a convenient access to the sensor measurings, as the sensor outputs could be received via Inter-Integrated Circuit-protocol (I^2C) which eliminates the need for a transducer. Furthermore a high-pass filter is integrated into the IMU's accelerometer, which leads to simpler compensation of the immanent drift of the magnetometer. These units supply the measurings for three axes each at a maximum frequency of 1600Hz for the accelerometer and 757.6Hz for the gyroscope.[28][27]

The IMU-chip was put in a suitable plastic housing, to fixate the IMU securely and prevent movement relative to the trailer-frame on which it will be mounted as well as ensuring water-proofing.

Figure 4.9 shows, how the IMU is connected to the Arduino Due utilizing the I^2C-interface on the Due. Not pictured are the power supplies (both 5V) for the MCP2515 CAN-transceiver and the IMU itself. To minimize cables that have to be rooted through the combination and avoid problems with cable length for the power cables, it was decided to equip each of these assemblies with a rechargeable battery-pack. This also allows to measure in remote location and collect data locally on the Arduino Due during testing, eliminating the need for CAN-communication completely if needed in the future.

A maximum of two IMUs can run on one I^2C line as it is possible to override the last bit of the otherwise hard-coded I^2C address via a hardware jumper. If both I^2C lines of the Due are used, it would be possible to connect up to four IMUs to one Arduino. Though eight IMUs are desired for the whole HCT vehicle, this was outruled as a possibility for this project, due to the limitations of the physical bus-length of the I^2C lines of only a few meters (as the name already suggests I^2C is not meant for long distance communication). As can also be seen in the overview figure of the setup on the truck in figure 4.1 this distance would be exceeded with the distance between the different mounting points (15-20m).

In the final measuring setup the assembly shown in figure 4.9 will be present four times in the whole combination, to gather motion data from each independently moving unit in the combination.

Figure 4.1 shows those placements in an overview sketch. The correct placement will be as close as possible to the CoG of the respective unit to have as little influence of the vehicle's rotary motions as possible, which drastically reduces work in data analysis further on. CoG locations can be gathered from Volvo's internal database and are available at the test site. Putting two IMUs in each spot allows for averaging and ensure redundancy should one IMU chip fail or produce corrupt outputs, which occured from time to time during the bench-testing of the assembly if the voltage of the preliminary power-supply dropped due to other loads.

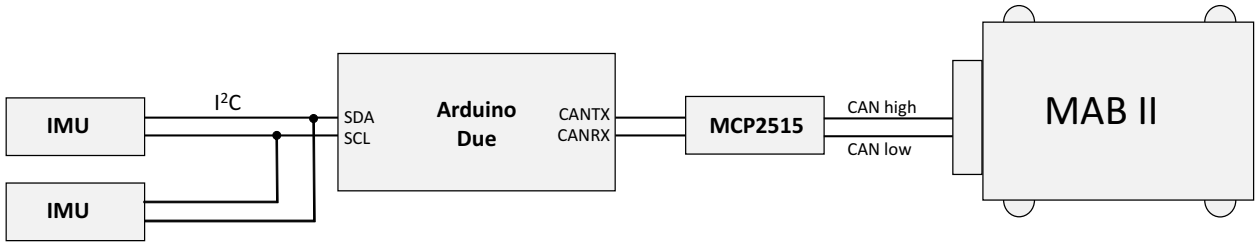


Figure 4.9: Connection of IMU with Arduino Due to MABII

5 Software Architecture

5.1 Matlab/Simulink environment

This section is fairly detailed to document important details for future work.

5.1.1 dSpace RTI-blockset

The dSpace Real-Time Interface (RTI)-Blockset is a plugin for MATLAB/Simulink that makes it possible to connect a Simulink-model to the different inputs and outputs of the MABII There are RTI-blocks for CAN, Ethernet, and the analog and digital outputs of the MABII. In this project the RTI CAN MultiMessage and the RTI Ethernet (UDP) Blocksets were used.

RTI CAN MultiMessage Blockset

The RTI CAN MultiMessage blocks establish an interface between the physical CAN-Buses of the MABII and the Simulink-model running on the MABII. There are four different blocks in this blockset that were used in this project:

- *GeneralSetup*
- *ControllerSetup*
- *Main*

The *GeneralSetup* block is used for setting the paths of the model root and the destination folder for generated files.

The *ControllerSetup*-block is used for setting up a CAN-controller. Figure 5.1 shows the parameters that need to be set in it.

First the name of the controller has to be set. After that the physical CAN-Bus, that should be used, is set by choosing a module number and a controller number. How the module- and controller numbers have to be set for the different CAN-buses of the MABII is shown in table 5.1. The table also shows which pins are used for the different CANs on the ZIF-connector (see figure A.1), which is used to input them to the MABII.

Table 5.1: CAN-layout MABII

CAN	Module number	Controller number	ZIF-Pin CAN-High	ZIF-Pin CAN-Low
CAN1	1	1	c2	c3
CAN2	1	2	b2	b3
CAN3	2	1	B2	B3
CAN4	2	2	A2	A3
CAN5	3	1	P2	P3
CAN6	3	2	N2	N3

After setting the module- and controller number the identifier format has to be set to either standard or extended format, the transceiver type must be chosen between ISO11898-2 and

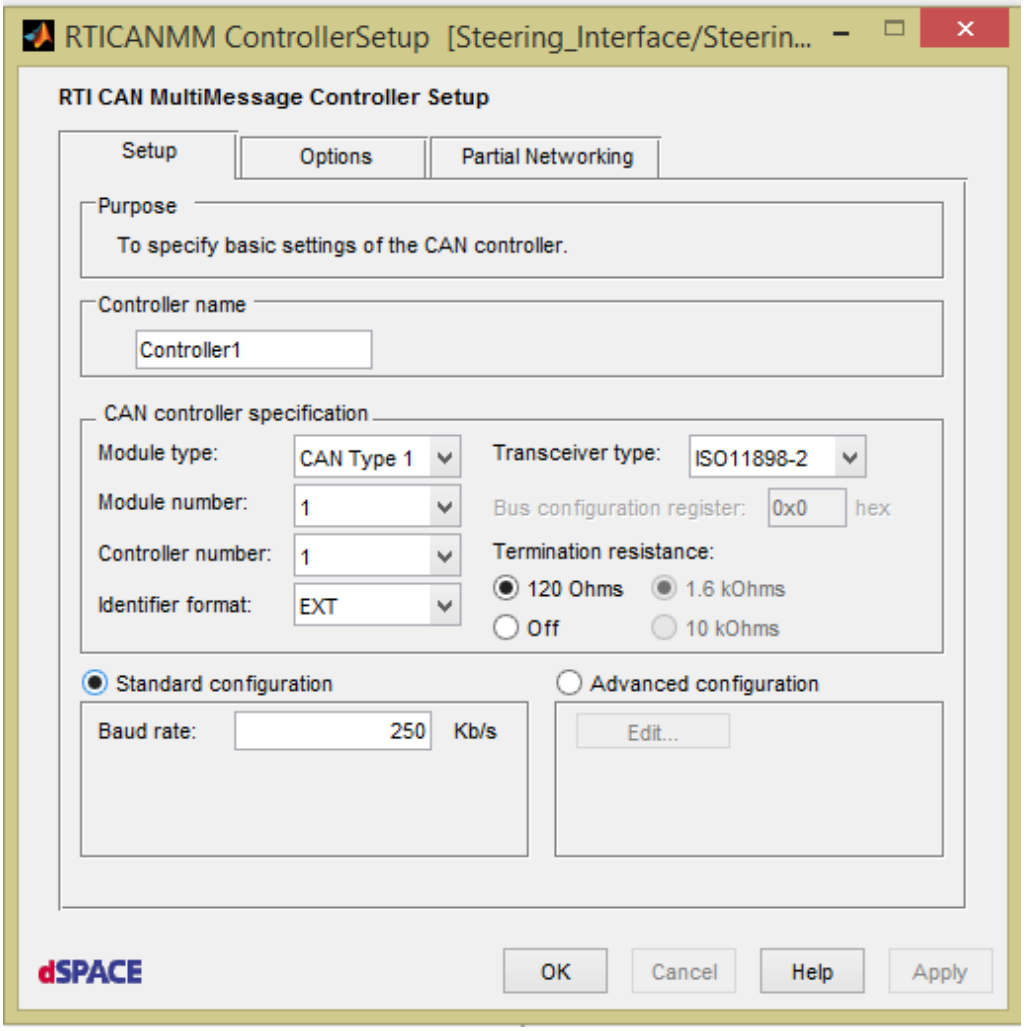


Figure 5.1: RTI CAN MultiMessage ControllerSetup-block

ISO11898-6. ISO11898-2 is used for a high-speed medium access unit and ISO11989-6 for the selective wake-up functionality of a high-speed medium access unit. If needed, a termination resistance of 120 Ohms can be set in the block as well. As a last step the Baud rate of the CAN-bus has to be defined.

In the *Main*-Block a .dbc-file is connected to one of the controller blocks, that were created before. Figure 5.2 shows the Graphical User Interface (GUI) of this *Main*-Block.
A *ControllerSetup* block can only be connected to one *Main Block* at a time. When the .dbc-file is loaded in the *Main Block*, the different messages and signals that are specified in the .dbc-file can be chosen as inputs and/or outputs of the *Main block*. Figure 5.3 shows an example of the signals that can be chosen for the .dbc-file of the ASF-sensor.
For signals that should be transmitted it has to be defined when they should be transmitted. There are different ways to specify this. One way is to set a cycle time in the Simulink model, that determines with which frequency the signal is sent out. Another way is to use triggers that trigger the transmission every time a specific event happens. This can be set in the tab *Messages* under *Triggering*.

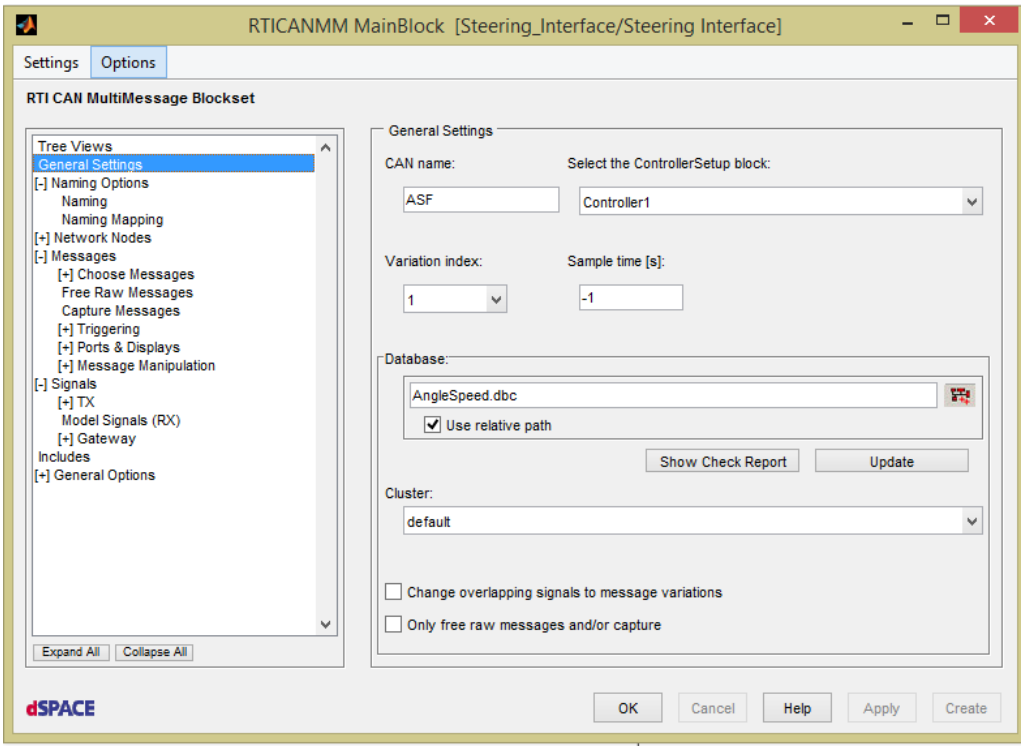


Figure 5.2: RTI CAN MultiMessage Main-block General settings

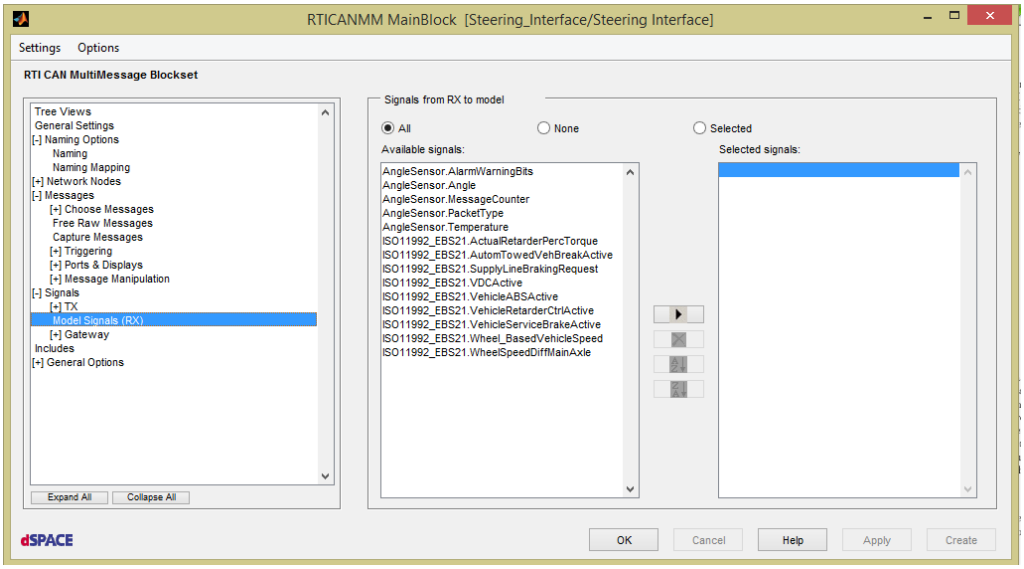


Figure 5.3: RTI CAN MultiMessage Main-block Rx-signals example

RTI Ethernet (UDP) Blockset

The RTI Ethernet (UDP) Blockset was used for two purposes: On the one hand for the HiL -testing (see section 6.5), on the other hand for the CAN-bus extension of the MABII. The RTI Ethernet (UDP) Blockset contains four different blocks: A setup-, RX-, TX-, and an interrupt-block.
The *Ethernet_UDP_SETUP* block consists of two pages. In the *Unit* page the main settings for the Ethernet are made. First the interface name has to be chosen. Then the board type

has to be selected. There are two types of boards: *ETH Type 1* and *ECU Type 1 ETH*. If the Ethernet cable is connected to the Ethernet port of the MABII, *ETH Type 1* has to be chosen. The module number, which has to be set in the next step, must be set to 1 since the used MABII only has one module. After that, the local Internet Protocol (IP) address has to be set. On the options page of the setup block up to four different sockets can be enabled. For each socket a local port number, the remote ip address and the remote port number have to be set.

The *Ethernet_UDP_RX* block is used to receive data over Ethernet. To set up this block the board type, module number and socket number need to be set corresponding to the settings that were made in the "Ethernet UDP Setup" block. Furthermore the maximum message size can be defined. Outputs of this block are the received data, the message size and a status. The datatype of the received data is uint32. If the data wasn't transmitted as uint32 it needs to be decoded. In the demo for the Ethernet blocks a block called *DSDecode32* can be found. With this block the decoding can be done, by choosing the desired output data format. To get the single signals the output of that block just needs to be demuxed. If the data type of the transmitted data was already uint32 the signal can be demuxed right away.

The *Ethernet_UDP_TX* block is used to send data over Ethernet. This can be done in a similar way as for the receive block. The only difference is that in the transmit block additionally a send timeout can be set. The maximum message size must match the port width of the data that is send out. It can be calculated by:

$$data\ port\ width = (MaxMessageSize + 3)/4 \tag{5.1}$$

The inputs to the transmit block are the data to transmit, the message size and a trigger to enable the transmission. The data format of the data to transmit needs to be uint32. Similar to the receive block a so called *DSEncode* block can be found in the demo for this blockset. This block encodes multiplexed signals of different types in a way, that they can be transmitted with the transmit block. In the *DSEncode* block the data type of the different signals, that are put in the block, need to be specified.

The *Ethernet_UDP_INTERRUPT* can be used to make the hardware interrupts available as trigger sources.

5.1.2 Virtual Truck Modelling Library

The VTM is a computational framework used within Volvo Trucks to simulate the dynamic behaviour of trucks and combinations. As a library it extends Simulink, where maneuvers, track layout and the trucks kinematic and dynamic properties are linked together and then computed. This toolbox was used as a base for the simulation of HCT vehicles including the dolly for this project. It is possible to run simulation offline with a predefined maneuver and a given environment as well online where these parameters are fed into the calculation live or as measurings from the real-world environment. For online use all relevant parameters and states can be accessed in Simulink or in case of execution on the MABII through ControlDesk as their respective representation of the Simulink variable.

5.2 Steering interface

The steering interface block constitutes the interface between the vehicle model controller and the different CANs of the dolly. It is a Simulink model that runs on the MABII Figure 5.4 shows the steering interface block with its in- and outputs.

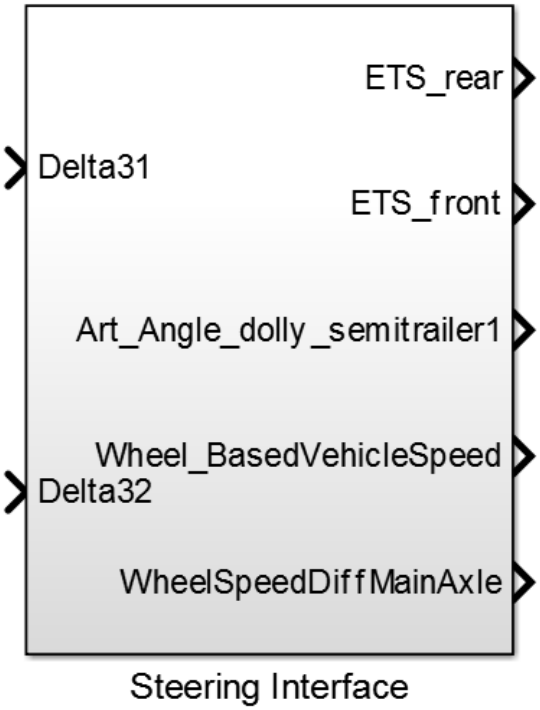


Figure 5.4: Steering interface block

The inputs to the block are:

- Delta31: Steering angle requests for front axle of the dolly sent from the steering controller model
- Delta32: Steering angle requests for rear axle of the dolly sent from the steering controller model

The outputs of the block are:

- ETS_rear: All signals on the ETS-CAN rear axle
- ETS_front: All signals on the ETS-CAN for front axle
- Art_Angle_dolly_semitrailer1: The actual articulation angle between the dolly and the first semitrailer
- Wheel_BasedVehicleSpeed: The vehicle speed signal from the dolly's EBS
- WheelSpeedDiffMainAxle: The wheel speed difference from the dolly's EBS

The outputs are used to monitor the dolly's state and for the calculation of the current capabilities (see section 7.2). Figure 5.5 shows the inside of the steering interface block. It contains RTI CAN MultiMessage blocks (see section 5.1) for communication with the dolly.

The software CAN-layout corresponds to the hardware CAN-layout described in section 4.3. There are five different CAN-controllers. Controller1 is used to receive the signals of the ASF-Sensor and the dolly's EBS system. These signals, except for the *Angle*, *Wheel_BasedVehicleSpeed* and the *WheelSpeedDiffMainAxle* signals are forwarded to Controller2 and Controller3. Controller2 corresponds to the ETS-CAN of the front axle and Controller3 to the one of the rear axle. The inputs *Delta31* and *Delta32* are sent to the front ETS-CAN (Controller2) and rear ETS-CAN (Controller3), respectively. For the signals *Wheel_BasedVehicleSpeed* and *WheelSpeedDiffMainAxle* constants with the value zero are forwarded both to Controller2 and Controller3. Controller4 and Controller5 also correspond to the front ETS-CAN and rear ETS-CAN, respectively. As mentioned in section 4.3 this is necessary because different CAN-protocols are used. The signals received from those Controllers, together with the signals from Controller1, that weren't forwarded, are outputs of the steering interface block. In figure 5.6 a scheme of the modification of these signals is shown.

5.3 ControlDesk monitoring environment

The software used to monitor the Simulink model running on the MABII and for logging of data is called dSpace ControlDesk. It also provides the possibility to control the Simulink model, for example by changing parameters like constants in the model during run-time. Therefore it was used for bench-testing the dolly system (see section 6.2) to directly send steering angle request to the ETS-CAN via the steering interface (section 5.2). In ControlDesk the CANs that the MABII is connected to can be monitored as well. This proofed very useful for checking if all systems work correctly during early stages of development.

5.3.1 Maneuver control

Though as mentioned in section 4.7 the MABII is capable of running headless and it would be possible to execute the maneuvers fully independently on start-up, it was decided to rely on manual maneuver triggering to have an extra level of safety. For controlled testing environments with clearly delimited maneuvers, the structure shown in figure 5.7 was introduced and allows to either command a steering request of zero to the dolly or patch through the calculated steering angle that is computed by the steering controller and thus starting the maneuver.

Sending a steering request of zero results in the dolly behaving like a passive un-steered dolly, which is also the fall-back level implemented in hardware in the legacy ETS system. The constant block in the depicted system can be changed manually in ControlDesk to use this as a maneuver start-switch, but of course can also be accessed by the model itself. The supervisor-block can thus shutdown the steering completely on software level, preventing the triggering of the hardware-level error-mode. The hardware implementation moves the hydraulic piston of the steering system back to the middle position, if pressure is not present anymore. In doing so it enters error-mode, which needs to be reset by turning the truck's ignition off and on again in standstill. Software shutdown within the environment developed in this thesis, allows reverting back from locked mode to steering mode.

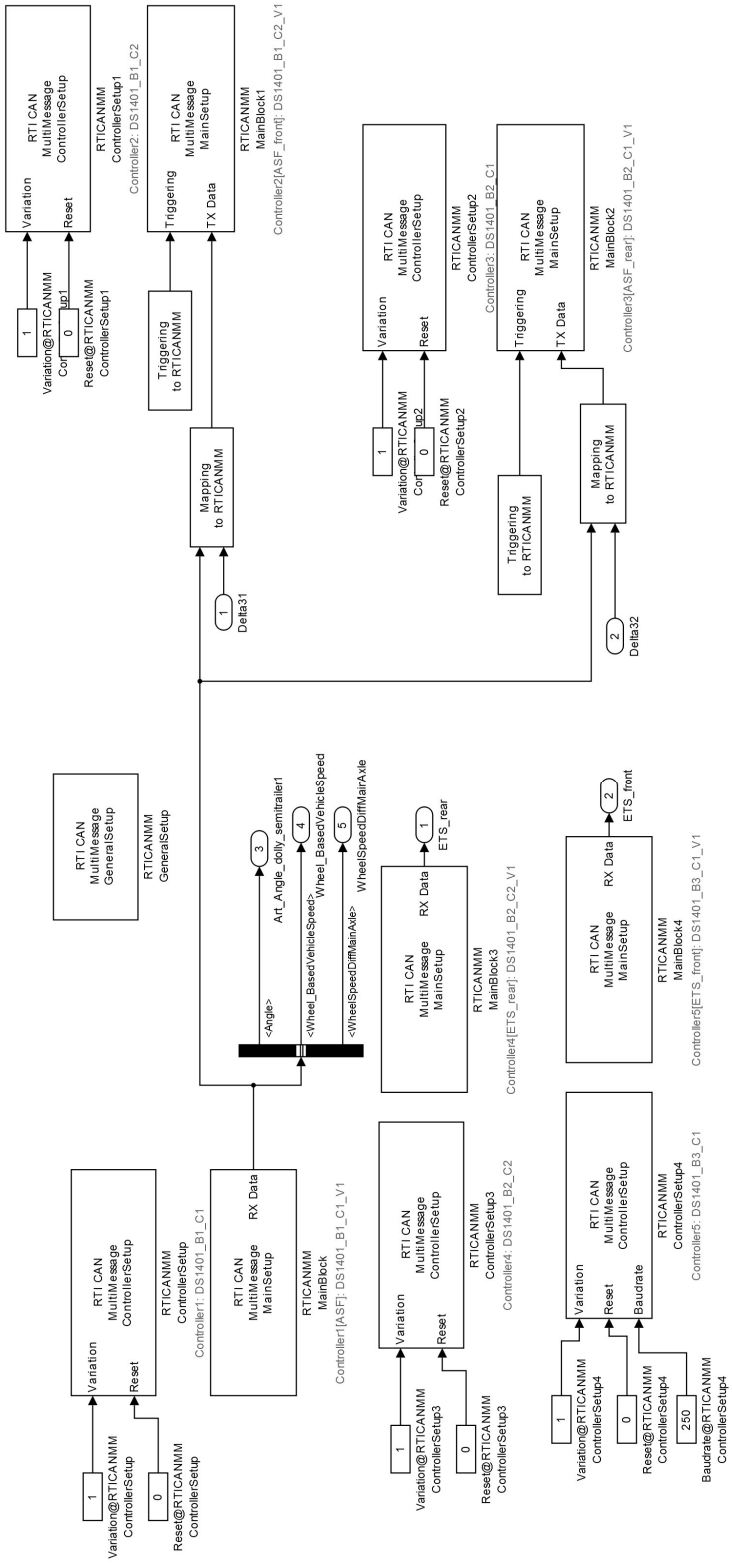


Figure 5.5: Steering interface block CAN layout

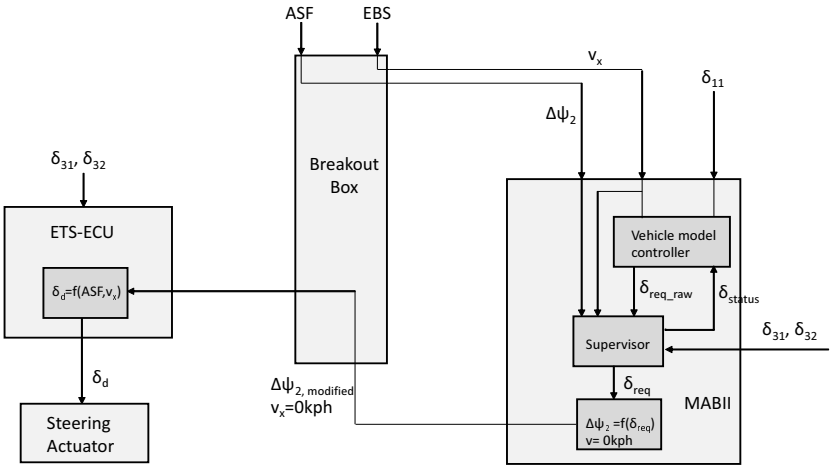


Figure 5.6: Modification of ASF- and EBS-signals in MABII

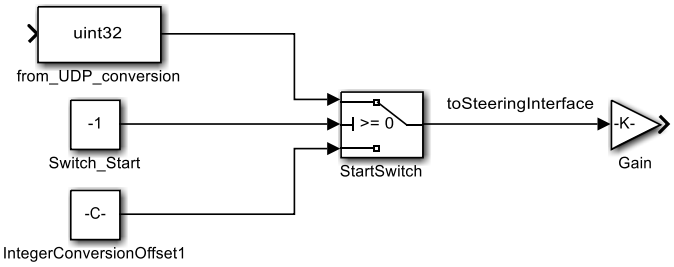


Figure 5.7: Trigger to start maneuver or fall back to safe-mode (e.i commanding a steering angle of zero degree on the dolly); this structure was used in HiL-testing

5.3.2 Monitoring and logging

All relevant information from the dolly, the installed IMUs as well as additional information from the truck (see section 4.5) are available on the CAN-buses in the MABII. Using the MABII to log data thus provides a straight-forward way to keep log-files consistent and synchronized over all interconnected systems. ControlDesk lets the user select representations of all parameters and variables that are included in the underlying Simulink-model to be logged. Here these selection will be directly timestamped and exported to .mat-files for further analysis. The logging frequency corresponds to the step-time of the simulation running on the MABII, 0.001s in this case. Values that did not change in this time span, which is true for a lot of measurings and signals broadcasted on the CAN, are held at their previous values.

5.4 Arduino IDE and applications

The Arduino system provides an Integrated Development Environment (IDE) written in Java, providing cross-platform support. It is used to develop the code as well as compiling the code and subsequently uploading it into the microcontroller via the computers serial interface. Within the IDE it is also possible to load some of the officially supported libraries directly. It conveniently is possible to monitor the computer's serial interface as well, which is the most practical way to monitor and debug code that is executed on the Arduino.

Library name	**Purpose**	**Comment**
LSM303	read magnetometer on IMU via I2C	[24]
L3G	read gyro & accelerometer on IMU via I2C	[23]
UIPethernet	control ENC28J60 via SPI	[21]
TinyGPSPlus	acquire and parse GPS signal from EM-506 via serial	v0.94b[19]
mcp_can	implement CAN via MCP2515 and MCP2551 via SPI	[9]
due_can	implement CAN via MCP2515 using Due's CAN-transceiver	[8]

Table 5.2: List of utilized libraries on the Arduino platform

6 Verification and validation of steering

6.1 Overview

In different states of the project different kind of tests were performed. As a first step bench tests were done to verify the developed software (section 6.2). Following that, several tests on the actual dolly, with the dolly in standstill and no trailers connected, were made (sections 6.3 and 6.4). The final step achieved in this thesis was HiL-testing (section 6.5). For the future, tests with the dolly as part of a A-double combination on a test track are planed (section 6.6).

6.2 Bench-Testing

6.2.1 VTM maneuver verification

In order to get data to compare the results of the tests that would be conducted, several simulations using VTM and the vehicle model controllers described in chapter 3 were performed in Simulink. Therefore all parts of the functionality architecture introduced in section 2.4 are simulated. For the HMI lookup tables were used for the steering angle and vehicle speed input, so instead of actual driver input a pre-defined maneuver was executed. The VMM consists of the vehicle model controller. The steering actuators in the MSD domain were simulated by implementing a first order filter and a time delay, that represents the delay caused by the hydraulic system. For the vehicle plant the VTM (see section 5.1.2) was used. Figure 6.1 shows how the parts of the functionality architecture are divided to different systems for the simulation of the vehicle model controllers.

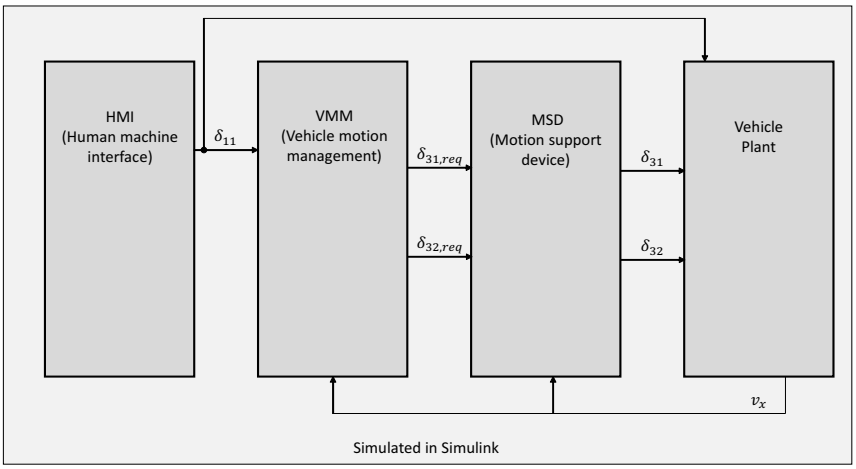


Figure 6.1: Functionality architecture in simulation

6.2.2 CAN verification

To test the functionality of the ASF-Sensor as a first step it was just connected to power supply and an Arduino with a attached CAN-shield. After the CAN-message was received

successfully with the Arduino, as a next step the sensor's CAN pins were connected to the MABII. On the MABII a simple model, that only contained one CAN-Controller with the corresponding .dbc-file, was loaded. The bus-monitor tool of ControlDesk, that shows all incoming and outgoing signals on the CAN for the corresponding .dbc-file, was used to check the sensor signals.

As outlined in section 4.3 the signals generated for each ETS-ECU need to differ to achieve independent steering. The CAN-wiring of the legacy system thus had to be interrupted and split. To ensure that only the correct set of messages was available after the splitting point, Arduino Dues were spliced into the system at all relevant junctions. This was a necessary procedure as gateway-systems between all different CAN-subnets on the dolly are in place and needed to be circumvented, which proofed to be a challenging task operating under the maxim of easy revertability to the original state.

6.3 Processing time evaluation

6.3.1 Background

The desired solution is supposed to operate at any speed. For high speeds a quick processing and transmission time is required to ensure prompt and realtime intervention of the control system based on the measured input signals. If the delays induced by the different components in the complete system are known or can be estimated, they can be compensated for in the steering-algorithm running on the rapid prototyping system.

As outlined in section 4.8 the dolly is equipped with a system to determine the deflection angle of the draw-bar and both the kingpin angles of the HCT combination. The sensor's raw signal is then parsed and filtered in an integrated low-level system which feeds the filtered signals to a private CAN-bus, where it is picked up by the MABII. The filtering operation takes a certain time and thus induces a delay. As the model which is executed on the MABII runs with a fixed step size of 0.001s evaluation of the computation time of this subsystem was not necessary. The steering mechanisms on the dolly are also a delay-inducer due to the inertia in the hydro-mechanic system. This is of course unavoidable in any physical system, but when measured can as well to a certain degree be compensated for in the calculations.

6.3.2 Delays within ETS-system

The response time of the dolly's steering system was determined in workshop environment in stand-still without any additional axle loads. Tests were conducted for both axles individually to ensure that maximum power was available preventing further delays through insufficient electrical power or hydraulic pressure. A repeated steering angle in the form of a step input was used to determine the reaction time of the system. The step amplitude was increased over different tests to account for possible dependencies on articulation of the steering system. The tests each started and ended with a steering angle of 0°. An example for the test sequence can be seen in figure 6.2. Clearly the difference between the requested angle and the actual articulation of the wheel can be seen. Further discussion of the results can be found in section 8.1.

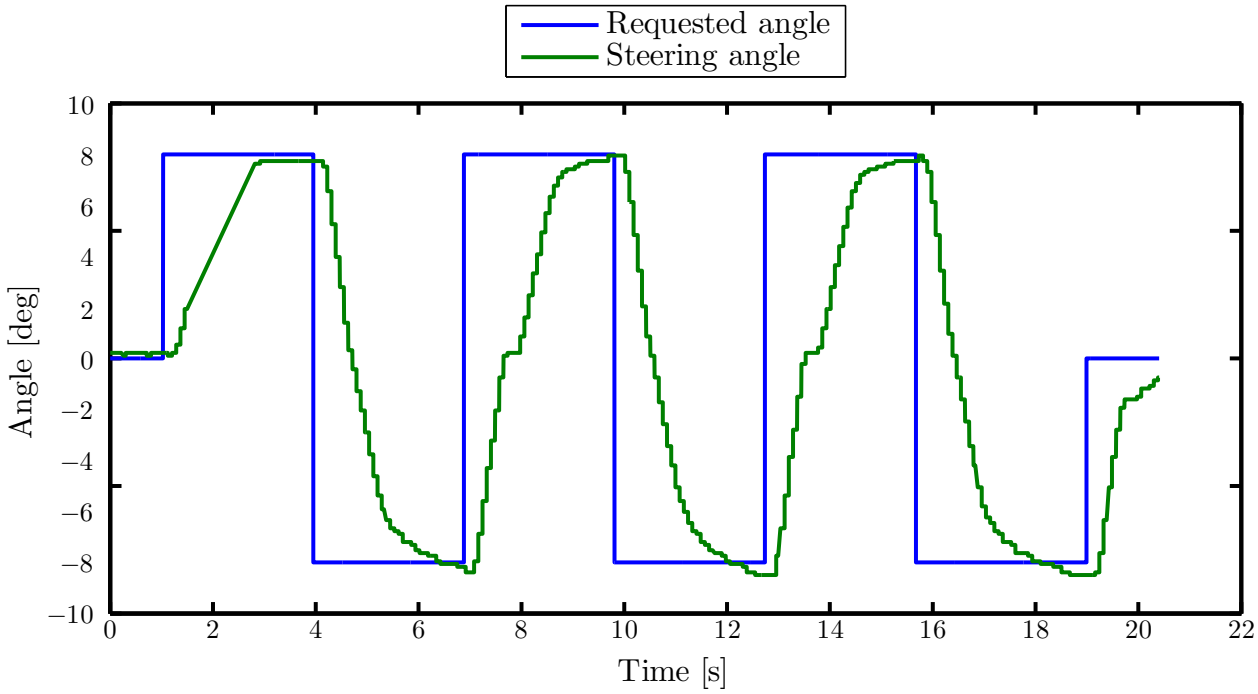


Figure 6.2: Example of step input for delay determination, front axle lifted, 8° amplitude, six repetitions.

To eliminate tire friction the investigated axle was suspended in the air in further series of testing by slightly craning up the dolly's front/rear (see figure A.2) to allow for free movement of the tires. An example for the utilized testing routine can be gathered from table 6.1. Between the different steps (double amplitude) a sufficient wait time of 3500ms was used to allow the system to reach the requested steering angle. This waiting time was kept constant throughout all tests.

As also described in section 8.1 additional response tests were conducted utilizing a ramp input. Here a maximum steering amplitude of 15° was used and the slope of the ramp was varied. As the request was not in the form of a step input, no error was triggered like in test #9 and #18 (see table 6.1). The test matrix for this setup is outlined in table 6.2. Besides the changed input-pattern the test- and logging-setup was left unchanged

All testing was automated in ControlDesk using Python scripts to loop through the different tests. The utilized GUI can be seen in figure A.4. Logging within ControlDesk was started and stopped manually after each test and automatically exported to MATLAB for parsing and analysis. All data was acquired from the steering knuckle sensors originally present in the ETS. This sensor's signal is available on the ETS-CAN (also see section 4.2).

It was decided to neglect the delays induced by the CAN communication. The reason for this is, that both the request as well as the measuring are send over the CAN. Which means that the two transmission times are counted with different signs and thus almost exactly cancel out only leaving double the worst case transmission (maximum stuffing bits, worst case arbitration) time as a maximum error of 232μs. This is sufficiently low regarding the fact, that

[1]If a too big of a step is requested from the steering system, VSE's ECU will go into emergency mode. Steering is completely locked, turning the dolly completely passive.

#	Amplitude [°]	Axle	Comment
1	1	front	lift front axle, ETS dead band
2	2	front	
3	3	front	
4	4	front	
5	5	front	
6	6	front	
7	7	front	
8	8	front	
9	9	front	error mode, step too big[1]
10	1	rear	lift rear axle, ETS dead band
11	2	rear	
12	3	rear	
13	4	rear	
14	5	rear	
15	6	rear	
16	7	rear	
17	8	rear	
18	9	rear	error mode, step too big[1]

Table 6.1: Test matrix for delay measuring with lifted axle (analogous matrix for non-lifted testing (#19-#36))

#	Amplitude [°]	Rate	Sample time [s]	Comment
1	15	0.01	0.001	
2	15	0.001	0.001	
3	15	0.1	0.001	
4	15	0.3	0.001	
5	15	0.4	0.001	error mode, slope too steep

Table 6.2: Test matrix for delay measurings with ramp input

the simulation on the MABII is executed every 0.001s.

6.3.3 Delays in logging measuring chain

To have a complete picture of the occuring delays in the measuring chain all relevant bigger execution blocks of the code running cyclically on the Arduino Due to broadcast the IMU data to the CAN-bus were evaluated for execution time. Figure 6.3 gives an overview, into which functions the code was grouped as well as the respectively used protocol for communication. The output via the serial port (Universal Asynchronous Receiver Transmitter (UART)) for visualization purposes on the computer while logging takes very long compared to the execution times of the other depicted code blocks.[1]It therefore is disabled while actually logging.

The delays were evaluated by inserting time-stamped variables into the code before and after

[1]The maximum transmission rate on the UART for the Due is 115200bit/s. Transmitting the payload (3x3 sensor axes, signed integer) only, excluding all control bits accumulates to 3 sensors x 3 axes x 16bit = 144bit or 1.25ms

each examined function's call. Using the Due's *micros*() function a resolution in microseonds CAN be achieved[2] with an accuracy of the inverse of the processors clock frequency (i.e. 11.9ns for the Due). *micros*()' call/execution time was neglected as it is in the magnitude of a couple of dozen nanoseconds.

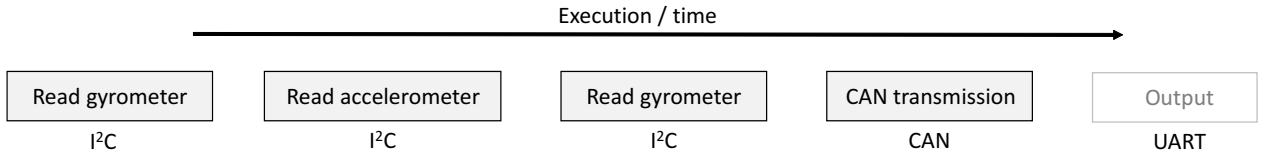


Figure 6.3: Execution blocks for IMU data acquisition on the Arduino Due (overview)

6.4 Vehicle testing

6.4.1 System calibration

The two ECUs on the standard ETS were calibrated to account for the hardware characteristics before handing-over the legacy system to the active converter dolly project. The parameter set included zeroing of hydraulic pressure levels of the steering system and the integration of the EBS-CAN signals as well as the ASF-CAN, as it would theoretically be possible to use other sources for the speed-signal needed in the legacy ETS. This calibration ensured a correctly and fully working base-system, which in case of failure could be reverted back to. This parameter sets will not be changed to prevent inconsistent behaviour.

6.4.2 Actuator tests

Actuator tests to verify working hydraulic system for the steering system and correctly pressurized brake-lines as well as normally running ECUs receiving correct analog sensor-information (voltage levels) within the legacy ETS and CAN-signals are possible with VSE's diagnosis screen, that usually sits in the ETS locker but was patched out for the HiL-tests and during setup in the workshop. As this diagnosis utility ties in directly with the core-functionality, it provides a very robust and reliable way to track errors during testing. An example of the home-screen output of the diagnosis display can be seen in figure 6.4. Note that it is in "Alarm Mode", which is the fall-back safety level, where all steering cylinders are locked in middle position (i.e. wheels straight). The ETS reverts to this state, if sensor information are corrupted or deemed inconsistent. This particular example was triggered by the tests described in section 6.3.2.

6.4.3 Sensor testing

The three angle-sensors to determine the articulation between the different independently moving units of the combinations differ in form-factor as they are mounted either on the

[2]On earlier/smaller Arduinos *micros*() resolution is 4μs!

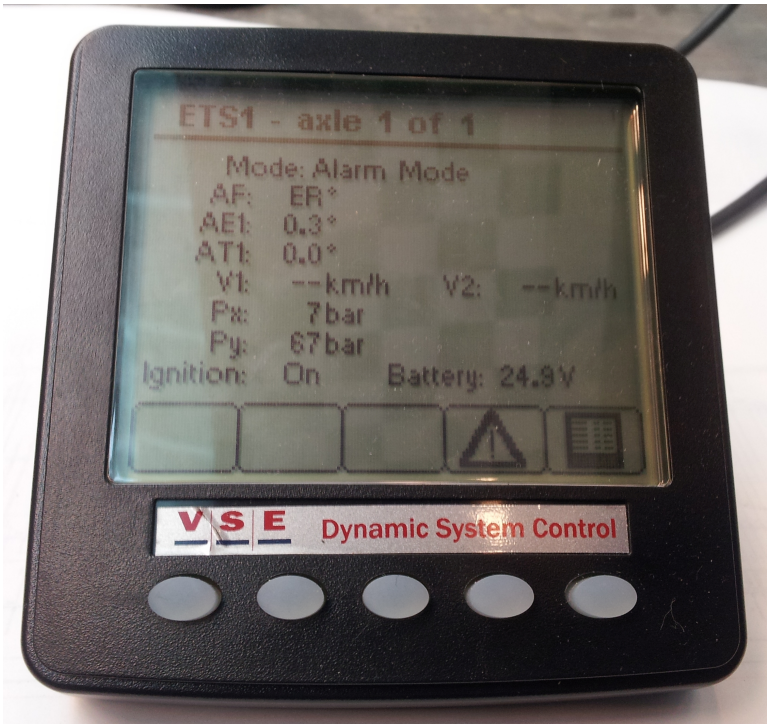


Figure 6.4: ETS' diagnosis sceen patched out of the original ETS-locker to allow easy access during workshop testing. The display is showing the homescreen.

fifth-wheel turntable or the king-pin respectively, nonetheless all rely on the same measuring principle and broadcast their gathered data in the same fashion on a private CAN-bus (refer to figure 4.4).

As can be gathered from figure 5.6 some CAN-messages needed to be patched through to the receiving ECU and some had to be changed. It was necessary to run the sensors in bench-tests first to ensure this communication was understood and controllable before implementing it into the final system. The VSE sensors need to be connected to 24V Descrete Current (DC) and then will independently setup and broadcast the angle of deflection of their turning-plate. The CAN-signal was picked up with a slim Arduino CAN-setup at first to verify the data. In further tests the CAN-reception was realized in the MABII where subsequently the describes by-pass with modification of selected signals was developed. Further calibration or testing was not necessary for this sensors, as they are very robust and reliable.

6.5 Hardware-in-the-loop testing

Track-testing takes a lot of time and preparation and thus is very expensive and oftentimes also dangerous. As only the complete system can be tested, it is only very late in the development process that it becomes possible to conduct such tests. To front-load the work in and distribute the verification process more evenly over the whole development cycle, testing of sub-assemblies of the final hardware system with simulation of the remaining parts in real-time allows to validate respective components along with their development-process. To do so a simulation environment has to be established with connection to the hardware-subsystem that is under evaluation. In this chapter the connection of the actual dolly, running the prototypes of the

final track-setup, will be tested with the controllers to verify proper interaction[25].

VTM includes a lot of processing-heavy sub-models (tire models, vehicle parameter sets) which lead to processing power not sufficing to allow for VTM's execution in the dSpace environment on the MABII. To perform HiL-testing it was thus necessary to split the computational load and accomplish real-time data exchange between the hardware controlling-system on the MABII and the rest of the simulation which will be run parallely in Simulink in real-time on a standard PC. Though there are dedicated real-time platforms available to achieve real-time execution it was decided to rely on a standard PC to minimize costs and have a lean work-process without extra steps of code conversion to different platforms. Figure 6.5 illustrates the distribution of the HiL-setup's different components according to the functionality model over two computers, the MABII and the actual hardware.

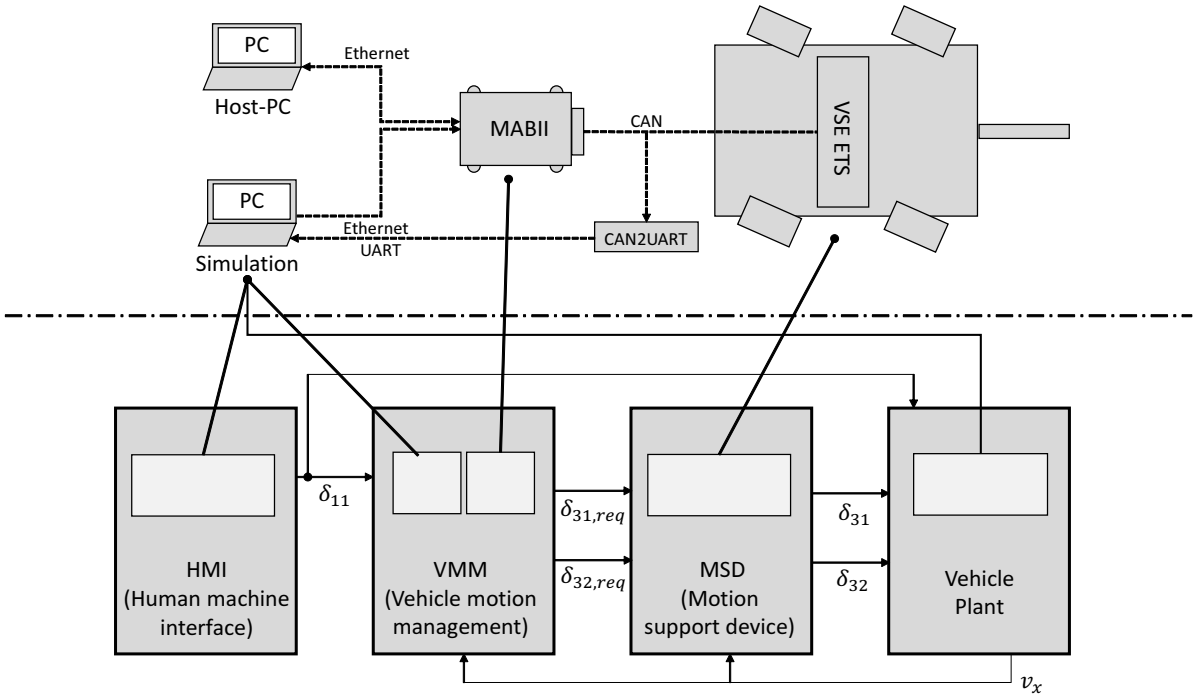


Figure 6.5: Overview of HiL-simulation, distribution of sub-functions over different physical platforms (top) and correlation to Volvo functionality architecture (bottom)

The Host-PC fulfills the usual task to start-up the dolly and simulation during HiL testing and log the behaviour of the dolly system and the IMUs. This is done through the ControlDesk GUI which was described in section 5.3 and of which a screenshot can be found in the appendix (figure A.4). It already runs the final version of the control and capturing environment and does not require changes for track testing. It is connected to the MABII via the Host-PC port.

On the Simulation-PC the simulation of the complete rest of the hardware-system is computed in real-time. Mainly truck, trailers, environment and steering behaviour of the truck are part of this model. All these sub-systems are modeled in VTM which is executed on this computer in Simulink.

Usually Simulink simulations are computed as fast as possible and then afterwards analysis of the results and system's behavior is conducted offline. This is an unsuitable approach for this thesis' purposes, as it is important to have real-time execution for this sub-system to provide

the actual hardware of the HiL-test with the needed parameters and simulate the environment at the correct instance also considering the hardware's feedback. I.e. the hardware runs in real-time, so the rest of its environment has to as well. Simulink's Desktop Real-Time toolbox provides the functionality to execute models in real-time on a desktop PC as well as support for a variety of hardware systems for data-exchange. This toolbox was used to connect the MABII's Ethernet port to the simulation-PC's Ethernet port using the UDP protocol. Also part of this toolbox is a sub-packet to access the PC's serial port within the simulation-environment. As it was not possible to receive reliable real-time data via the Ethernet-interface which lead to the introduction of the serial interface for the feedback-loop. A similar setup as described in section 4.6 was used to realize the conversion of the CAN-messages directly taken from the EBS-CAN to serial output on a Arduino Due. As depicted in figure 4.5 it is essential to to keep the CAN-buses of the two axles separated to ensure individual steering. Thus two MCP2551 CAN-transceivers had to be used for the feedback-loop in the HiL-setup as well.

The logging during the HiL-testing has to be split into two sub-logging systems. One is present on the Host-PC and works just the same as during future track-testing (outlined in section 5.3). However to capture the sub-systems, which are not implemented in hardware, but simulated in VTM, need to be separately captured on the Simulation-PC. The relevant values can directly be exported to a .mat-file using Simulink's *toFile*-block. To get correctly synchronized data between these two independent system an additional time-stamp was introduced and send over the serial interface between the two environments.

Limitations: HiL-testing took place in workshop environment and thus the dolly was not rolling compromising the tire friction compared with a rolling combination. As outlined in section 8.1 this does not increase the reaction time of the system, though. Furthermore the additional axle load was zero for both axles, as it was not possible to increase the load without a trailer in the lab environment at hand. This is also unlikely to change the results, as is suggested in 8.1 the axle-load is no deciding factor in the wheel articulation's performance.

6.5.1 Low-speed controller

The maneuver that was used for testing the Low-speed controller was a 180° turn at constant speed of 2m/s. The steering angle of the tractor δ_{11} can be seen in Figure 6.6.

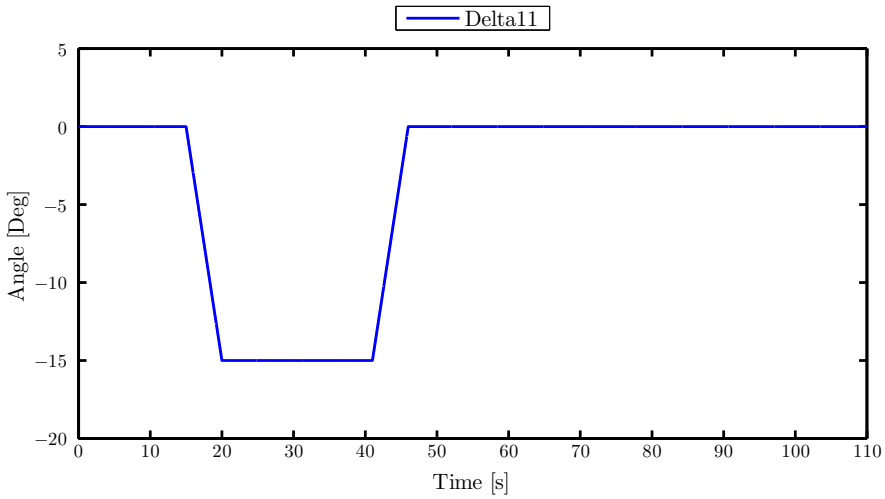


Figure 6.6: steering angle of the tractor for the low-speed Hardware-in-the-Loop-test

As a first step a simple controller was used for controlling the steering of the dolly. The controller used the articulation angle between the first semitrailer and the dolly as an input. This controller was develop in a research project at Chalmers University of Technology[17]. Equations (6.1) and (6.2) show how the steering angle for the dolly's front and rear axle respectively correlates with the articulation angle between the dolly and the first semitrailer.

$$\delta_{31} = -0.8 * \Delta\psi_2 \tag{6.1}$$

$$\delta_{32} = -0.8 * \Delta\psi_2 \tag{6.2}$$

The results of the HiL-testing are shown in section 8.2.

6.6 Track testing

For testing on a test track the different parts of the functionality architecture (see section 2.4) will be handled as follows: Since a complete vehicle combination will be used for track testing, this corresponds to the vehicle plant of the functionality architecture. The HMI domain is represented by the driver of the vehicle. The VMM is running on the MABII, which works as an interface between the VMM, the dolly and the vehicle. Figure 6.7 shows how the parts are divided into different systems.

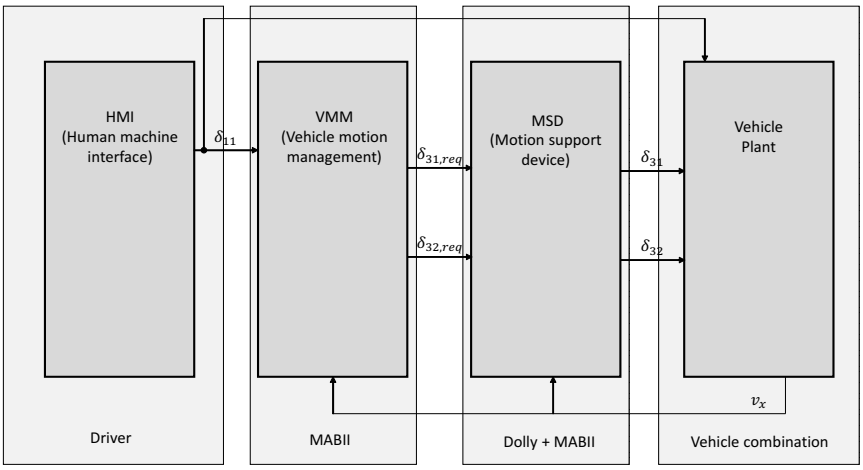


Figure 6.7: Functionality architecture on-track testing

6.6.1 Testenvironment AstaZero

The test with this dolly platform will be carried out at the AstaZero test-site close to Gothenborg, a cooperative automotive test-track funded by SP Technical Research Institute of Sweden and Chalmers University of Technology. It provides a state of the art closed off environment, which guarrantess safety and repeatable traffic conditions as well as enough space to carry out dynamic maneuvers at high speeds with a HCT combination. Especially the so called high-speed area will be especially useful, as the combination has enough distance to gather the desired speed before maneuvering.

6.6.2 Testmaneuvers

As outlined in the respective publications, the test-maneuvers for the low-speed controller will be a 180° turn and a S-turn[15]. The high-speed controller will be tested with a single lane-change representing an over-taking maneuver in traffic or the avoiding of an obstacle[16]. To input this driving scenarios consistently with a high repeatability into the combination, Volvo's Volvo Dynamic Steering (VDS) steering system will be utilized. Originally it is developed to support the driver by mechanically decoupling the steering column from the steering-wheel but introducing electric-motors to give feedback and torque sensors to monitor driver inputs. As this system can fully control the truck's steering, it is possible to tap into the respective ECU. The necessary steering pattern was generated for the work of the mentioned papers already and is send out to the truck from the MABII to the CAN coming from the truck (see also section 4.5).

7 Fault detection and system ability

7.1 Failure Mode and Effect Analysis

Failure Mode and Effect Analysis (FMEA) was developed by the US military in the late 1940s and later also used by NASA in the Apollo project in the 1960s. Failure Mode and Effect Analysis (FMEA) are used to detect possible failures before they appear. Therefore the FMEA is done in a early stage of a project in order to be able to take the results of the FMEA into consideration when developing a system. In the process every possible failure of a system are taken into consideration.

To detect the safety critical parts of the whole dolly system and improve the safety a FMEA was conducted for this project. In a first step a block diagram of the system with all of its inputs, outputs and subsytems was created to gain a complete understanding of the system. Every subsystem was broken down to the lowest level and for those subsystems block diagrams with all the components, inputs and outputs were created as well.

Starting from the block diagrams all potential failure modes were determined for every component of the system respectively subsystem. As a next step the potential effects of these failure modes were determined and the severity of these effects were evaluated on a scale from one to ten, with 1 being the lowest severity and ten being the highest. For the evaluation a table that shows how different severities correlate with the numbers was used (see Table A.1). Following this, potential causes for every failure mode were defined. Then the probability of occurrence for each cause was evaluated, also using a scale from one to ten (Table A.2). After that the current control mechanisms, that detect the failure when it should appear, were listed for each failure mechanism and the detectability of the failure mechanism was evaluated using a scale from one to ten (Table A.3). After the severity, probability and detectability were evaluated, a Risk Priority Number (RPN) is calculated as follows:

$$RPN = severity * probability * detectability$$

This RPN is used to identify the failure mechanisms that need to be addressed. After the RPN for every causes is calculated, the RPN are ranked from highest to lowest using a Pareto diagram (see figure A.5). The causes with the highest RPN need to be addressed first. Special attention needs to be paid to cases where a high severity number, which means 9 or 10, occurred, even if the RPN isn't as high as for other cases. The two lines that can be seen in figure A.5 were defined with a RPN of 40 for the green and a RPN of 100 for the red line. Where the green line is the limit for the RPNs under which no action needs to be taken and the red line is the limit for the RPNs over which the causes needs to be addressed in any case. For RPNs between the green and the red line it was decided, after evaluating each of the causes individually, if actions needs to be taken.

In order to lower the RPN for those cases actions need to be decided, that either lower the severity, probability or detectability of them or more than one of those factors. After the performance of these actions the severity, probability and detectability are evaluated again and a new RPN is calculated. If the RPN still is to high a iteration of the process needs to be done [6]. Figure 7.1 shows a screen shot of the FMEA-Excel-worksheet for the first iteration. The consequences of the FMEA conducted for this thesis were:

Item / Function	Potential Failure Mode(s)	Potential Effect(s) of Failure	Sev	Potential Cause(s)/ Mechanism(s) of Failure	Prob	Current Design Controls	Det	RPN	Recommended Action(s)	Responsibility & Target Completion Date	Actions Taken	Nev Sev	Nev Occ	Nev Det	Nev RPN
CAN-Connection															
	No Connection														
		No/Wrong Steering Angle	10	MABII unplugged	2	Check of Connection in ControlDesk	1	20							
			10	Failure of breakout-box	6		5	300							
			10	ECU unplugged	1		1	10							
Signal Angle_front															
	No Signal														
		Wrong Steering Angle	10	Connection problem	1	Error Code in Steering-ECU	1	10							
			10	Failure of Sensor	2	Error Code in Steering-ECU	1	20							
			10	ECU unplugged	1		1	10							
	Wrong Signal														
		Wrong Steering Angle	10	Failure of Sensor	2	Comparison of front/rear sensors	1	20							
	Different														

Figure 7.1: Example of FMEA-worksheet for the first step of the FMEA

- Special attention needs to be paid to the mounting of the MABII in the Dolly's ETS-box
- The design of the breakout box was revised
- Bench- and HiL-testing is necessary to validate the vehicle model controllers and to exclude dangerous behavior, that might be caused by delays in the system.
- A start-up checklist for the track testing needs to be created

7.2 Maximum capabilities of the system

The capabilities of the system can be divided into two parts. The first part includes the capabilities that are independent from influences. These are the maximum steering angle and maximum steering rate, which is limited through the ETS-ECU, because of the physical limitation of the hydraulic system. The other part includes the capabilities that are influenced by the vehicle speed, load of the dolly, state of the hydraulic and electrical system of the dolly, and environmental influences. Depending on those parameters the capabilities are lowered. The maximum angle for stand-still is 24° in both direction for front and rear axle. The requested angle can only given as a step input of maximum 7°. For higher step inputs, the ETS-system switches to alarm mode. This can be seen in figure 7.2 for a step input of the request angle of 8°. Because the system is in alarm mode the request angle transformed to a target angle and hence there is no steering. The maximum steering rate was determined with the tests described in section 6.3.2 to 5°/s.

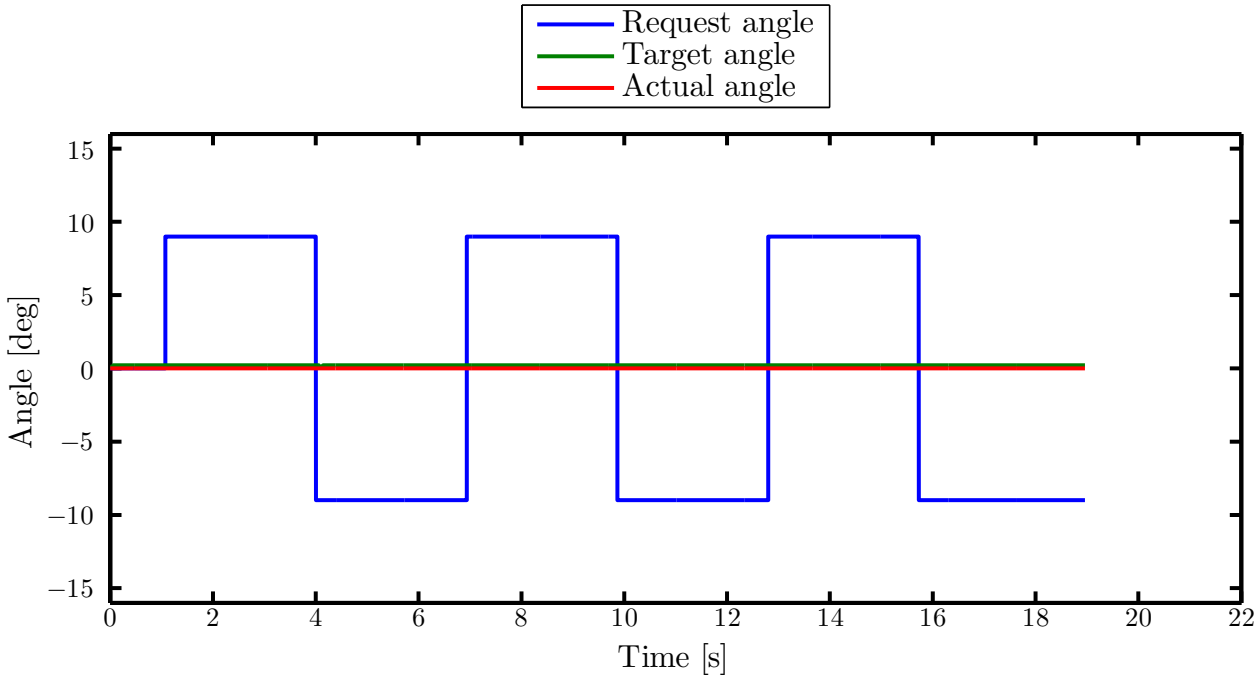


Figure 7.2: Steering angle request step input 8°

7.3 Warning and state-info system

In the original state, the ETS-ECU monitors all safety critical components of the dolly. In case of a malfunction, an error code is sent via the CAN-Bus and the error message is visible on the diagnose displays of the dolly. Depending on the severity of the error the steering is deactivated and the axes steer back to a steering angle of 0°. Due to the required modification of the ASF- and EBS-signal, the inbuilt supervision of the ETS-ECU is no longer fully functioning. Therefore an additional supervision of the system has to be implemented in the Simulink-model. This supervisor considers the current capabilities of the dolly and reduces the requested steering angle and steering rate, if needed.

8 Discussion

8.1 Results from processing time evaluation

Figure 8.1 shows the requested angle, target angle[1] and actual angle of the dolly's front axle over time. As described in section 6.3.2 the requested angle is a step input, in this case of 5°. The target angle's slope is constant and lower than the slope of the request angle. The actual

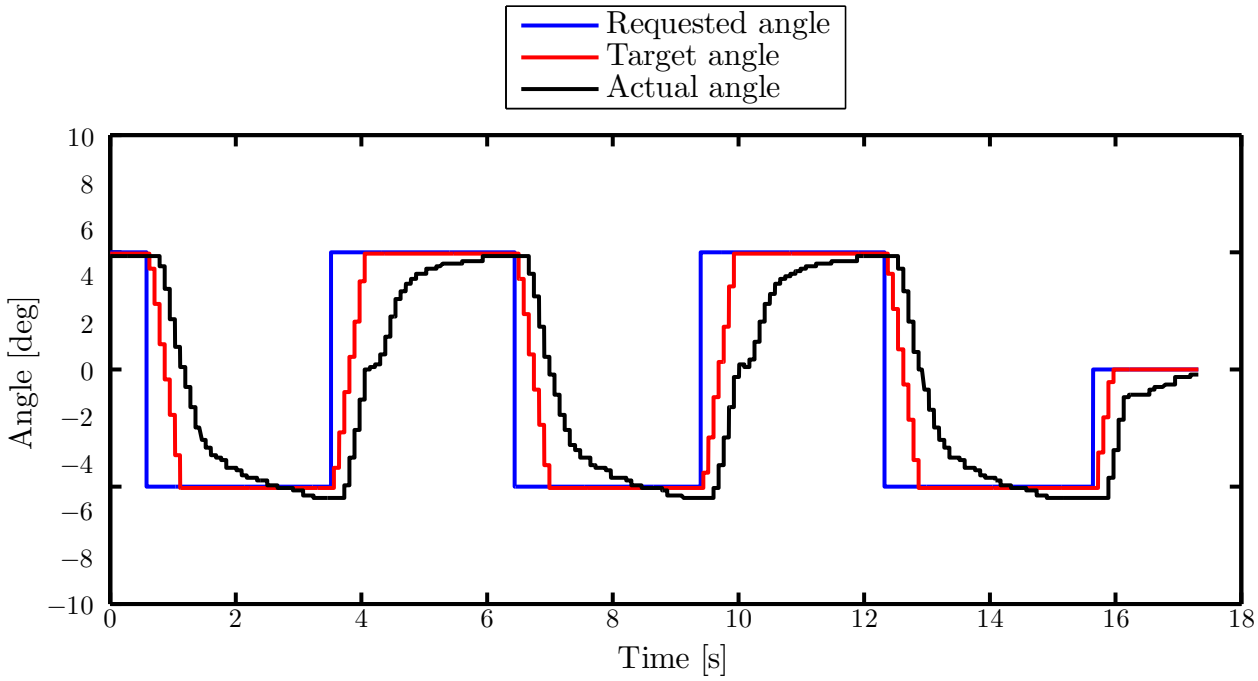


Figure 8.1: Steering angle request step input 5°

angle has approximately the same slope as the target angle, but it is delayed. The actual angle tends to overshoot the requested angle for falling slopes. A reason for this could not be determined. Furthermore the slope of the actual angle is reduced when the actual angle almost reached the request. There is also a saddle point around an angle of 0° for rising slopes of the actual angle. The target angle's slope is constant and lower than the slope of the request angle. The actual angle has approximately the same slope as the target angle, but it is delayed. The actual angle tends to overshoot the requested angle for falling slopes. A reason for this could not be determined. Furthermore the slope of the actual angle is reduced when the actual angle almost reached the request. There is also a saddle point around an angle of 0° for rising slopes of the actual angle. The delays over the different request angle step inputs were plotted and a linear regression was made to deduct the actual delay for small changes of the request angle. This delay is the relevant delay, since in real driving situations the requested angle is never a step input but rather changes only slowly. This delay over the step inputs together with the linear regression is shown in figure 8.2.

[1]The target angle is an internal parameter within the legacy VSE ETS. It corresponds to the steering angle that the ETS-ECU commands to the hydraulic steering. It therefore includes some of the internal limiters and safety systems of the ETS and thus moves slower that the requested angle send from this thesis' interface.

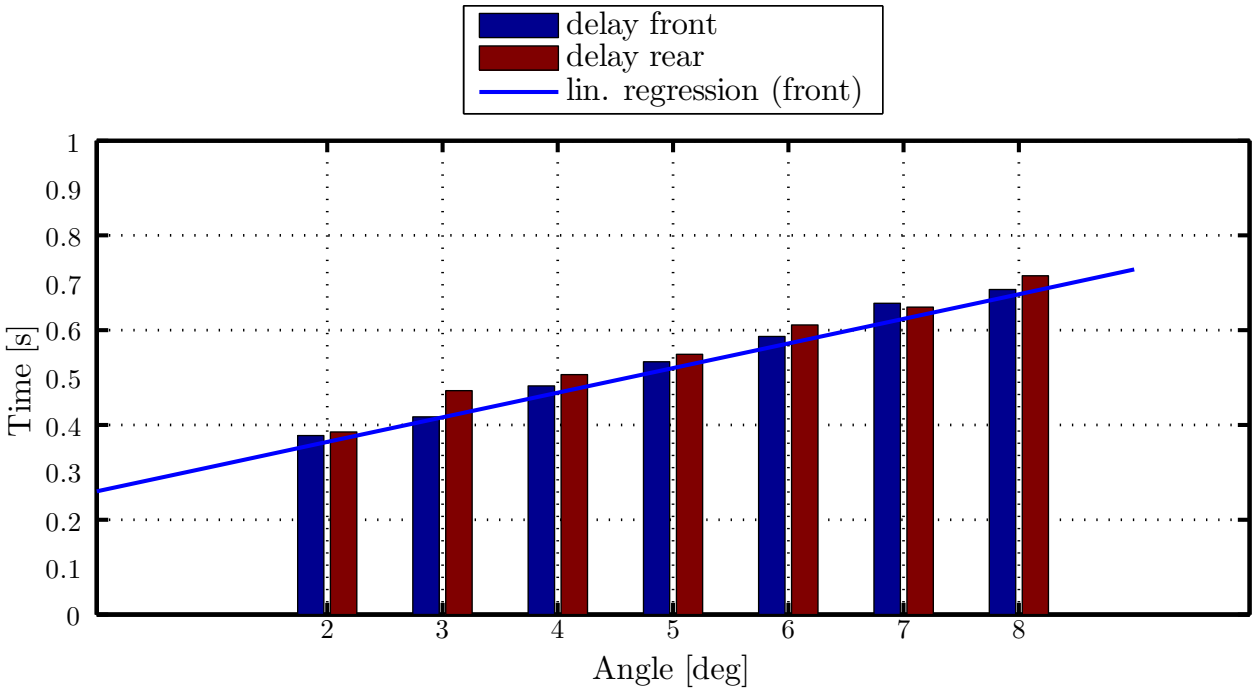


Figure 8.2: Rise time from send out request to achieve a certain steering angle on the dolly.

The deducted delay is 0.26s for the front axle and 0.3s for the rear axle.
During plot analysis of preliminary results it was discussed, whether it would be possible to by-pass the occuring delay by feeding a ramp-input instead of a step-input into the ETS. It was assumed that there is a low-pass filter in place which would not affect ramp inputs. Subsequent tests with ramp inputs with varying slopes and amplitudes showed that it was indeed possible to eliminate the low-pass filter leading to no delay between the target angle and the request angle. Examples of the ramp inputs with different rates and the subsequent reaction of the ETS system can be seen in the figures 8.3, 8.4 and 8.5.

With a rate of 10°/s the request angle and target angle align. They also align at a rate of 1°/s but for that case their slope is much lower, so it takes an insufficiently long time to reach the indented angle. For a rate of 100°/s the signals of the request angle and the target angle do not align anymore. For rates higher than 300°/s, the ETS-system switches to alarm mode, because the request angle changes to fast, so the signal is implausible for the system. This corresponds to the behavior which was caused by sending step inputs (also see section 6.3.2).

Comparing figure 8.3 and figure 8.5 indicates that it is not possible to get any faster change with a ramp input than with the request of a step input. It thus can be concluded that the maximum steering rate of 5°/s can not be circumvented.

The processing time evaluation was gathered over 2000 loop cycles. The minimal and maximal execution time can be gathered from table 8.1. As can be seen, the deviation in execution time is very small (+- 1μs). These results present the values gathered after streamlining the original libraries and eliminating many *delay*()-calls which were present to make sure the CAN-transmission did not get jammed. After extensive testing it was possible to cut them down to a minimum.

So the overall execution time of the looped function on the IMU-system is 10.1ms, which

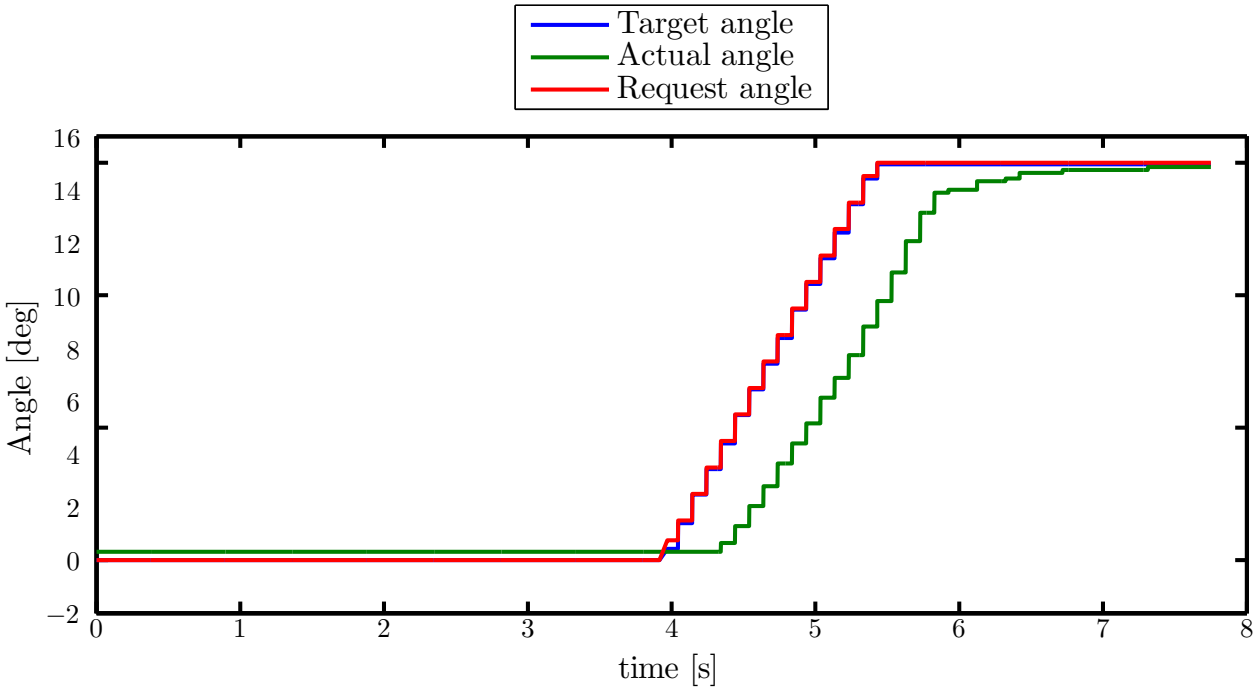


Figure 8.3: Steering angle request ramp input with a rate of 10°/s

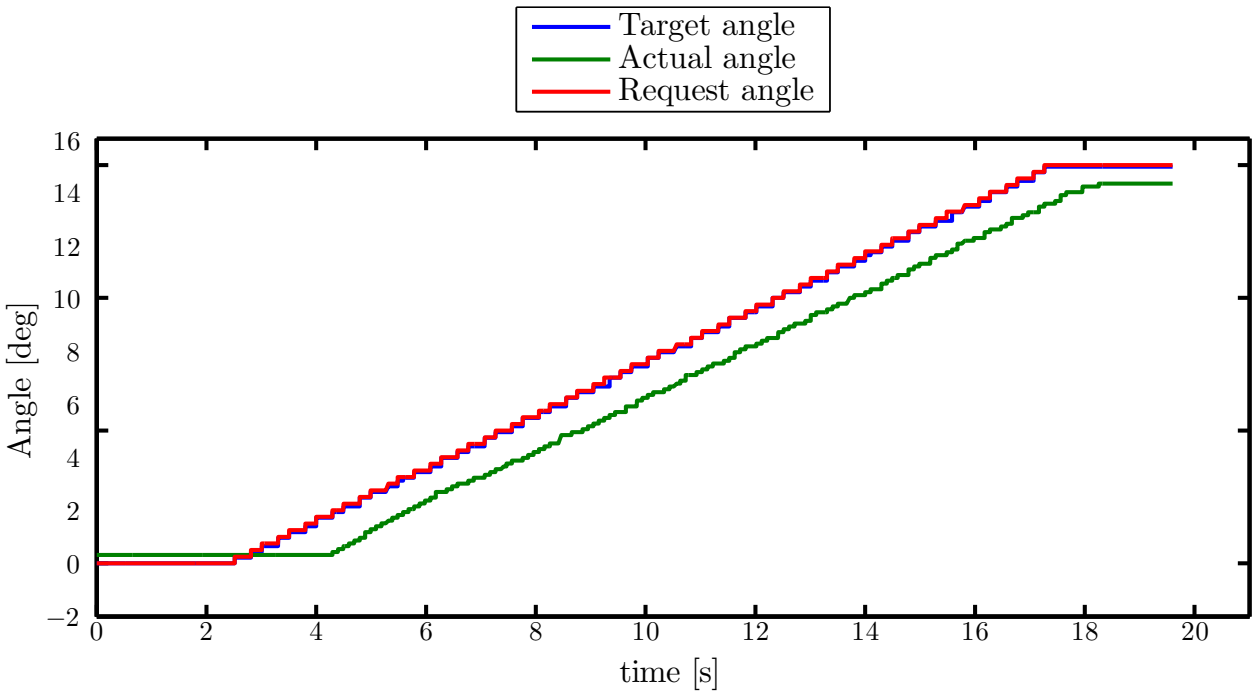


Figure 8.4: Steering angle request ramp input with a rate of 1°/s

was sufficiently low to introduce a GPS-unit to time-stamp the measuring data. The reading of the GPS-unit is also conducted via UART (and thus rather slow). The reading of the GPS-unit can be reliably achieved in 3ms; this timespan was also minimized in an iterative fashion. The additional delay introduced by the broadcast of additional CAN-messages can be neglected. So the the overall sum for all readings and the subsequent transmission of the data via CAN

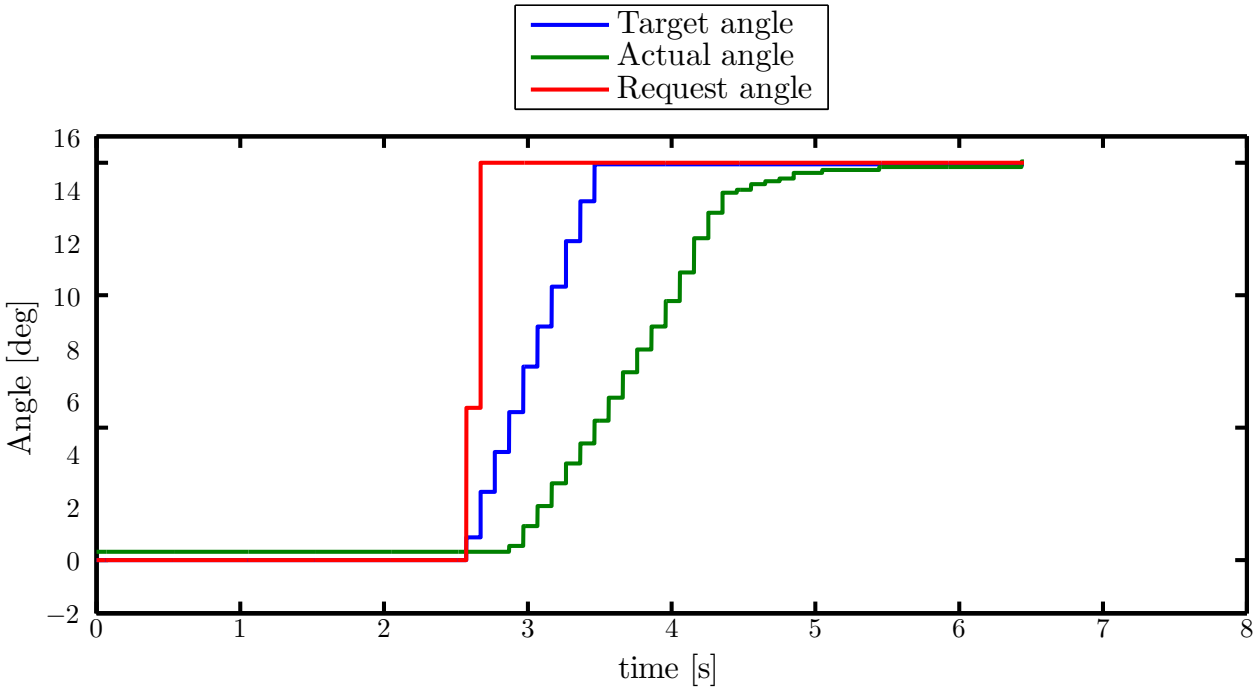


Figure 8.5: Steering angle request ramp input with a rate of 100°/s

Table 8.1: Overview of processing time of different execution blocks in the IMU measuring setup

	read_gyro	**read_accel**	**read_compass**	**can_send**	**serial_send**	**sum**
min. (μs)	2717	864	860	900	4804	10146
max. (μs)	2717	865	861	901	4804	10147

takes 13.1ms. It was decided to introduce a sampling frequency of 50Hz for the measurings to have a round and common number as a sampling frequency. The original concern, that the different systems would not run sufficiently fast, which lead to the investigation carried out for this section, can now be dismissed.

8.2 Results from hardware-in-the-loop testing

Figure 8.6 shows the measurements of the ETS-CAN of the front axle of the dolly for the low-speed maneuver described in section 6.5.

The request angle is the angle that was requested from the controller and then sent from the Simulation-PC to the MABII. The target angle is the angle that was calculated by the ETS-ECU and sent as a request to the actuators of the dolly. The actual angle is the angle that the sensors on the dolly's steering knuckles measured and which represent the articulation of the dolly's wheel on the ground. Figure 8.7 shows the same measurements for the rear axle. As can be seen in both figure 8.6 and 8.7, for a constant request angle, the actual angle is slowly decreasing after a certain time. This is probably caused by the steering system shutting off the hydraulic pump after a few seconds, when there is no change in the requested angle.

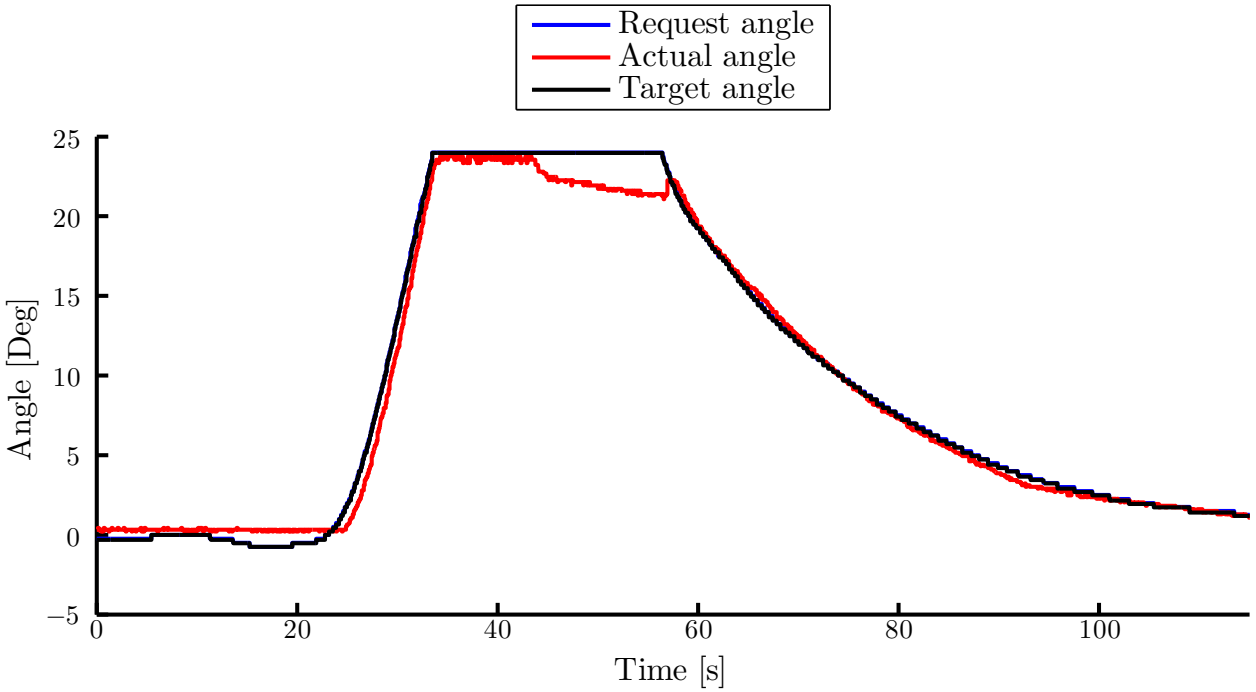


Figure 8.6: Hardware-in-the-Loop-test low-speed front axle

This leads to a decrease of the actual angle, because of the tension between the wheels and the ground, which creates a torque, that forces the wheels to turn back towards their original position.

Since the requested angle for front and rear is the same, below only the front axle is pictured.

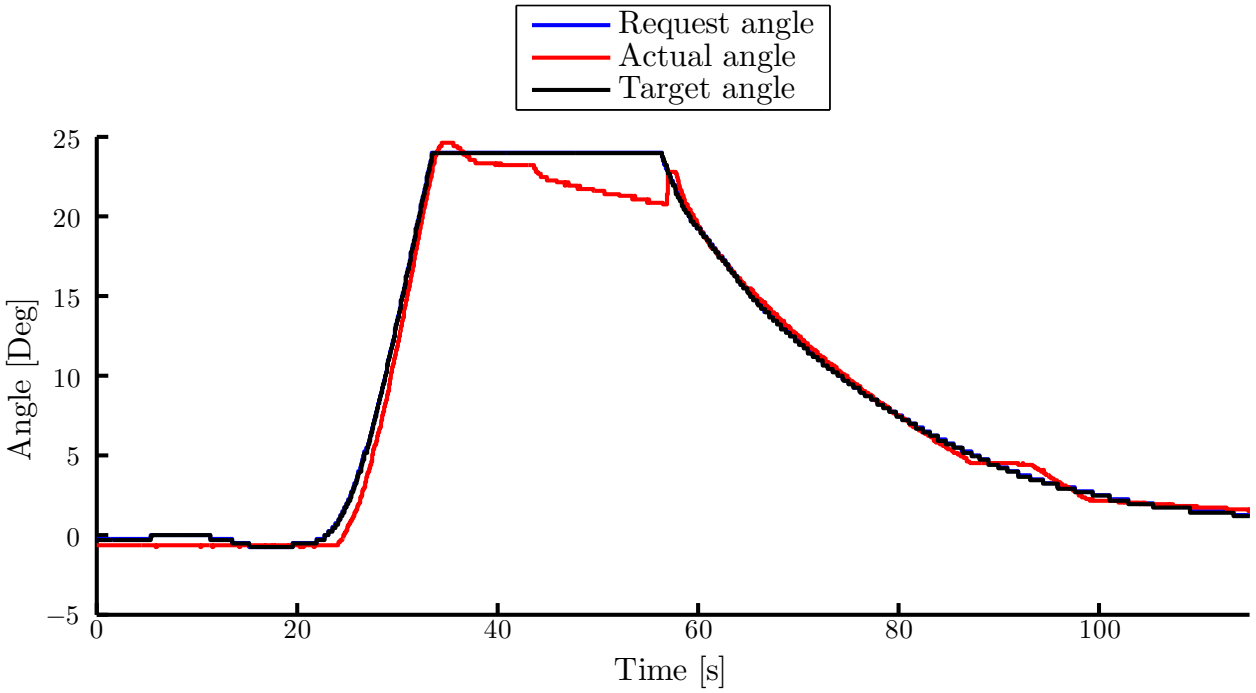


Figure 8.7: Hardware-in-the-Loop-test low-speed rear axle

Both axles show a very similar behavior and thus only considering one is justifiable. Figure 8.8 shows an extraction of the measurements of the front axle.

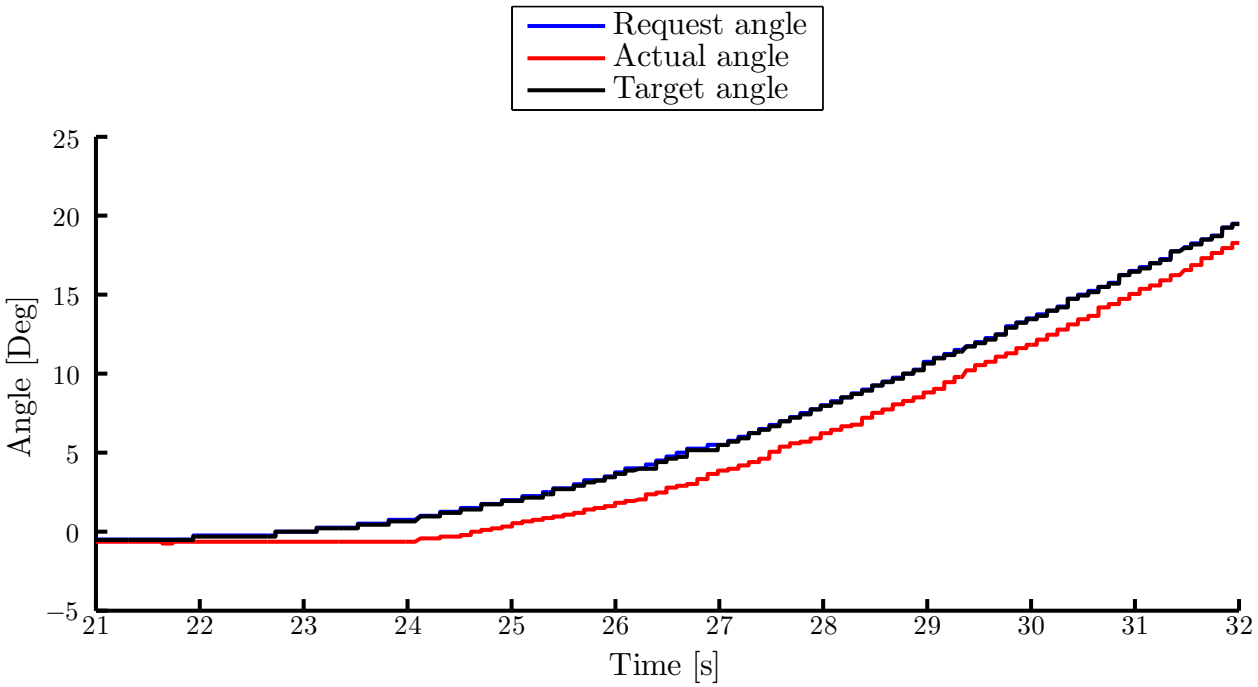


Figure 8.8: Details of Hardware-in-the-Loop-test low-speed front axle

As already mentioned in section 6.3.2 there is a delay between the angle requested of the ETS-ECU and the actual angle of the wheels. This can very distinctly be seen in this plot. In comparison to this, the delay between the angle request that is sent from the MABII to the CAN and the target angle that is sent from the ETS-ECU to the actuators is very small and can therefore be neglected.

To get an impression of the overall results of the HiL-test, figure 8.9 shows the measurements conducted on the simulation-PC and the measurements done in ControllDesk in the same plot.

For one thing the plot shows the limiting of the angle that is requested from the controller by the supervisor block described in section 7.3. For another thing, it shows that there is a relatively high delay between the signal of the actual angle of the dolly from the ETS-CAN and the signal that is fed-back to the Simulation-PC. This is most likely caused by the use of the serial interface to transmit the signal from the CAN to the simulation-PC.
To get a more detailed view of this delay as well as the limitation of the request angle figure 8.10 shows a extraction of figure 8.9.

The tested controller was evaluated in HiL testing as well as in pure simulation in VTM. The two resulting curves of the vehicles path can be seen in figures A.6 for the pure VTM-simulation and A.7 for the HiL-testing of the same maneuver respectively. For easier comparison figure 8.11 depicts the driven paths for both environments. The steering reacts a bit slower for the HiL-tests, but overall the paths are very similar. The actual performance of the algorithm and its evaluation is not part of this thesis; HiL-testing was done to validate the matching between simulation and VTM environment and also to proof the functioning of the developed interface.

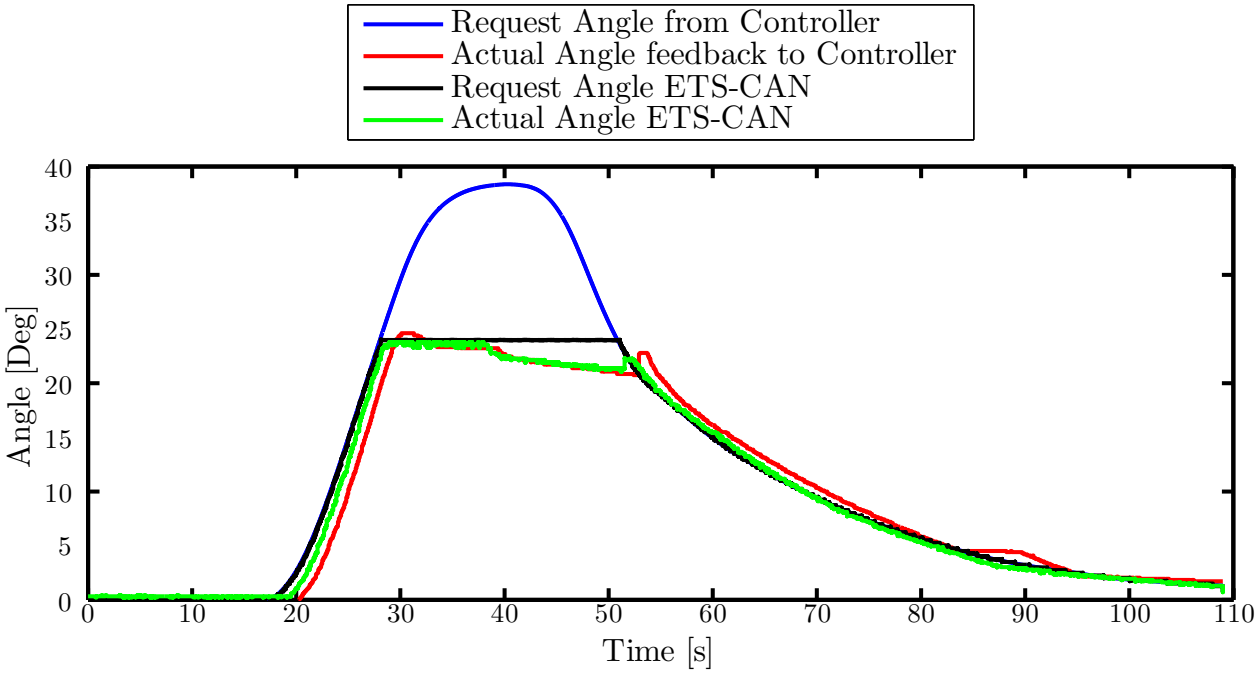


Figure 8.9: Hardware-in-the-Loop-test front axle measurements from VTM and ETS-CAN

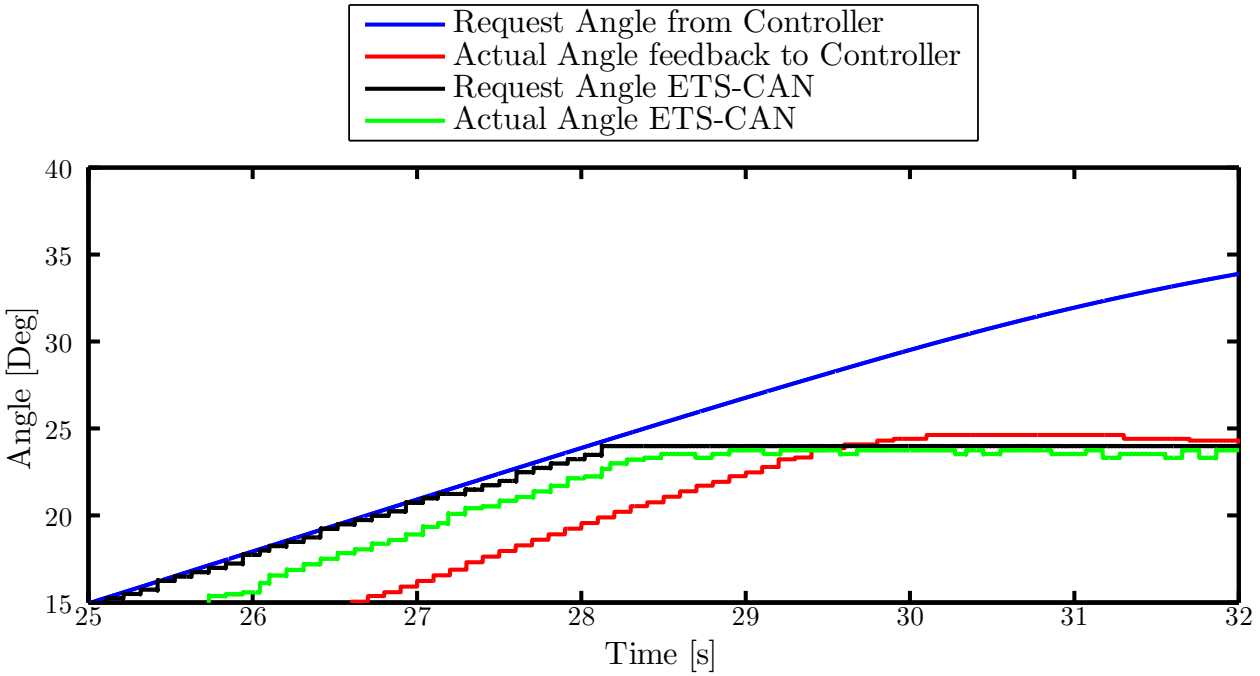


Figure 8.10: Details of Hardware-in-the-Loop-test front axle measurements from VTM and ETS-CAN

Figure 8.12 shows the difference of the path between the HiL-test and the test, that was performed with the same controller in VTM. As can be gathered from figure 8.12, the error between the two environments accumulates over time. This is a consequence of slightly different paths they choose due to the dolly not really allowing steering of very small angles (-2 to 2°). This is caused by hardware limitations and a dead-band filter which is present in the ETS-ECU.

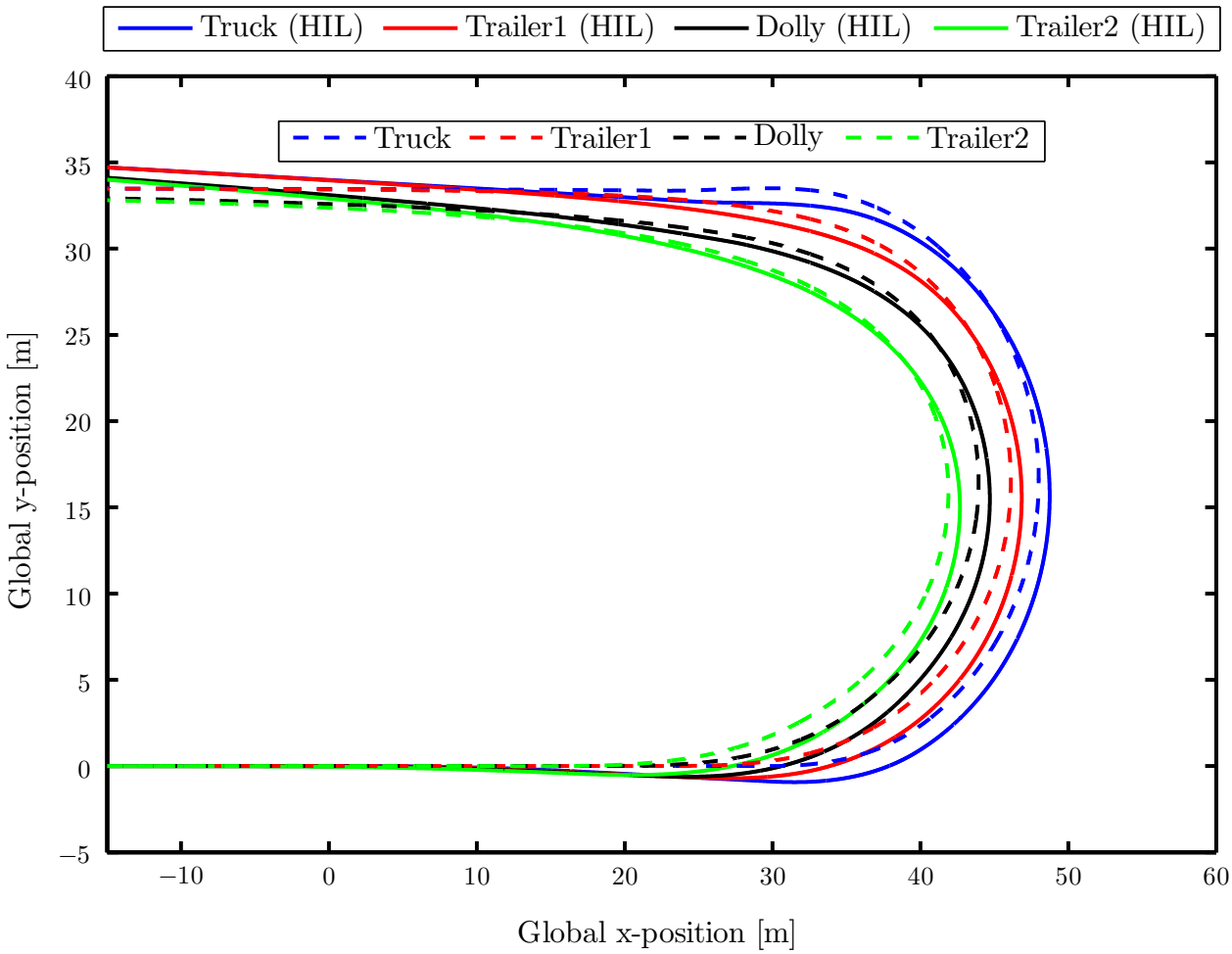


Figure 8.11: Comparison between swept paths of different units in the HCT combination during Hardware-in-the-Loop-testing and VTM-simulation

The end of the measuring can practically be ignored from approximately 100m of distance traveled, as this is mostly caused by different angles at which the combination exits the turn. This can be compensated for by fine-tuning the steering inputs, which was omitted for this thesis, as it was only the aim to demonstrate the general function of the controller. Fine-tuning will only be conducted once the final controller is ready and correctly implemented.

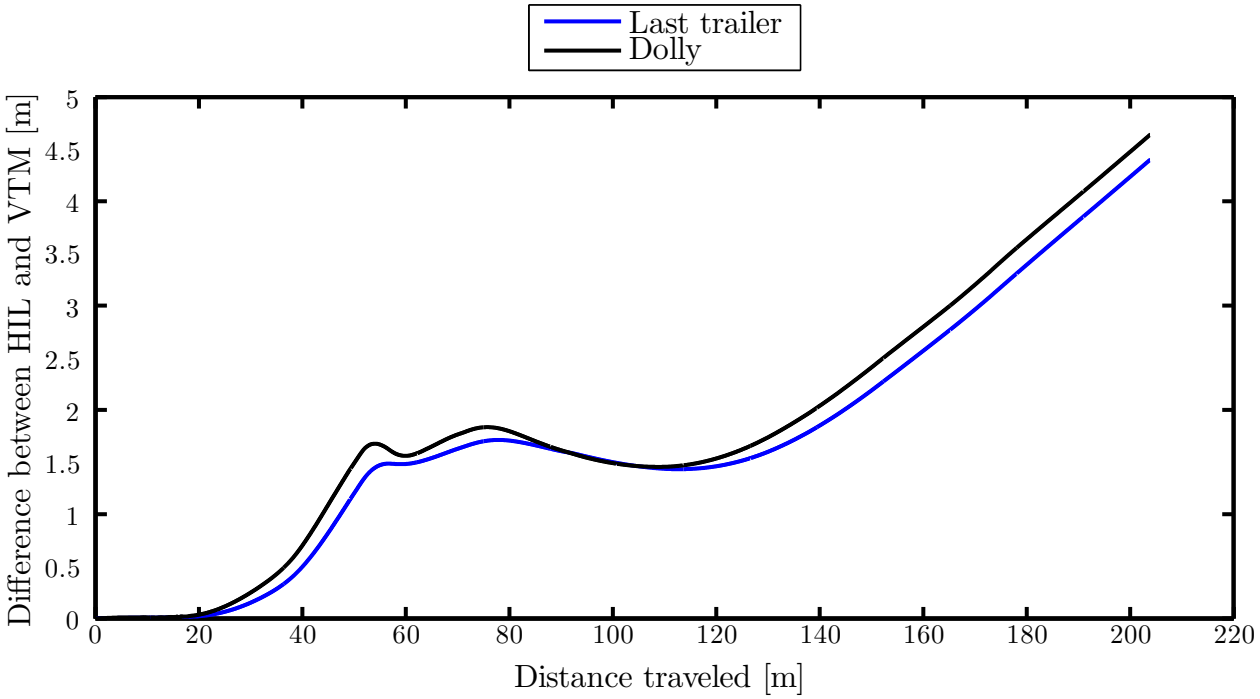


Figure 8.12: Deviation between Hardware-in-the-Loop-testing and VTM-simulation plotted over distance traveled during 180°-turn

9 Conclusion

The goals which were set in the introduction section 1.2 were reached:

- A software interface was established an now runs in the MABII-environment.
- The dolly's CAN-buses were modified in such a way, that the developed solution was able to deploy the necessary signals for independent steering for both axles.
- The delays in the actuation of the ETS system could me measured and are now to be included in the steering-algorithms further on. A quick measuring in the outlined fashion should be conducted, once the semi-trailer is docked on the dolly and fully loaded, to assure equal values of the system's behavior.
- Testing of all systems took place in parallel with the actual development of the prototypes and the programming of the code.
- A supervisor-block was included to limit the steering output of the system. However the envisioned solution to include more parameters (e.g. yaw-rate, simulation results) was not achieved, as time-constraints did not permit simulation of all relevant situations to provide useful additions in safety. The block can easily be expanded though, as the format and structure now is defined.
- Track testing was not possible as the scope of a Master thesis is too short to, equip a whole combination from scratch. The equipment is however ready to be taken to the track after manufacturing the hardware professionally which was planed out by the authors. Relevant sketches and technical drawings were submitted to the manufacturing company.

9.1 Recommendation

The mounted systems only give information about the physical state of the combination but leave no possibility to take environmental circumstances into consideration. Sensing for the surroundings, such as radar/LIDAR or cameras can greatly simplify and enhance the results in post-processing for the data logged during testing.

The supervisor-functions have to be extended as well to incorporate a broader use of the combinatation's dynamic data (yaw, acceleration).

For simplified delay-free HiL testing a dedicated real-time system could be utilized, which would also provide the possibility of including environment to a greater detail (e.g. randomly generated road situations) to achieve a very robust verification in safe lab-environment before integrating the dolly in an A-double combination on track.

Direct command of the ECUs with designated request messages would greatly simplify the system, eliminating many work-around solutions that were developed in this thesis. Also directly controlling the hydraulic actuation on a lower control level could benefit the investigated delays by circumventing the added control-layers.

Utilizing electric motors for the steering instead of hydraulic systems could decrease the delays induced by the mechanical mechanism greatly.

9.2 Future Work

The developed solutions can fairly easily be adopted to accommodate wireless network support for logging as well as direct controls, because all main-signals at some are transmitted via IP. The controller is executed robustly on the MABII and runs independently as soon as HiL testing is left behind for track-testing; this ensures safety, as all the supervisor systems are in place before the immanently fragile wireless transmission of the data. Wireless networking support makes the implemented systems on the dolly more accessible for example for convenient and quick debugging, visualization or even app-controlled steering of the dolly in shunting or couplings situations where the driver could maneuver his combination outside of the tractor cabine, greatly enhancing his view-point.

The braking solution that was outlined in this thesis has to be implemented in detail and evaluated with great precautions before taking it to the track. The safest solution would be to request braking torques/brake pressures via the brake ECU's diagnosis interface, as this would leave the original braking CAN-communication intact and prevent a solution similar to the one implemented for the interception and modification of the ETS sensor signals.

An air-suspended leveling system is available on the dolly and CAN be controlled fairly easily over the standardized ISO-11992 CAN, physically available on the 7-pin trailer connector which carries the ABS/EBS signals. It has to be evaluated whether this functionality is available while the combination is moving and of course whether it contributes anything to the enhanced dynamic behaviour of the whole combination.

Wheel individual propulsion via hub-mounted electric/hydraulic motors is already available for cars and leads to a safety system similiar to the working principal of ESC but without the disadvantage of slowing down the vehicle. These implementations are usually called torque vectoring or active yaw and greatly increase the performance in highly dynamic situations. Of course it is not possible to drive the second semi-trailer independently for extended periods of time as this exceed the output of the trucks alternator and would thus require impossibly large energy-storage in the second semi-trailer. Besides the great advantages that can be expected in dynamic situation, a electronically propelled dolly would allow independent shunting of the decoupled second semi-trailer unit in constricted spaces.

A Appendix

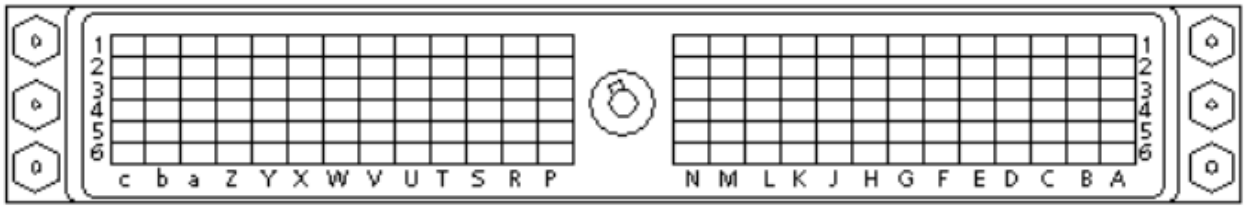


Figure A.1: ZIF-connector layout [10]

Figure A.2: Dolly craned up slightly to determine delay of steering actuation

Figure A.3: dSpace MicroAutoBoxII viewed from the top including cables for ZIF-connector, Host-PC, Simulation-PC and power supply (clockwise, starting on the right)

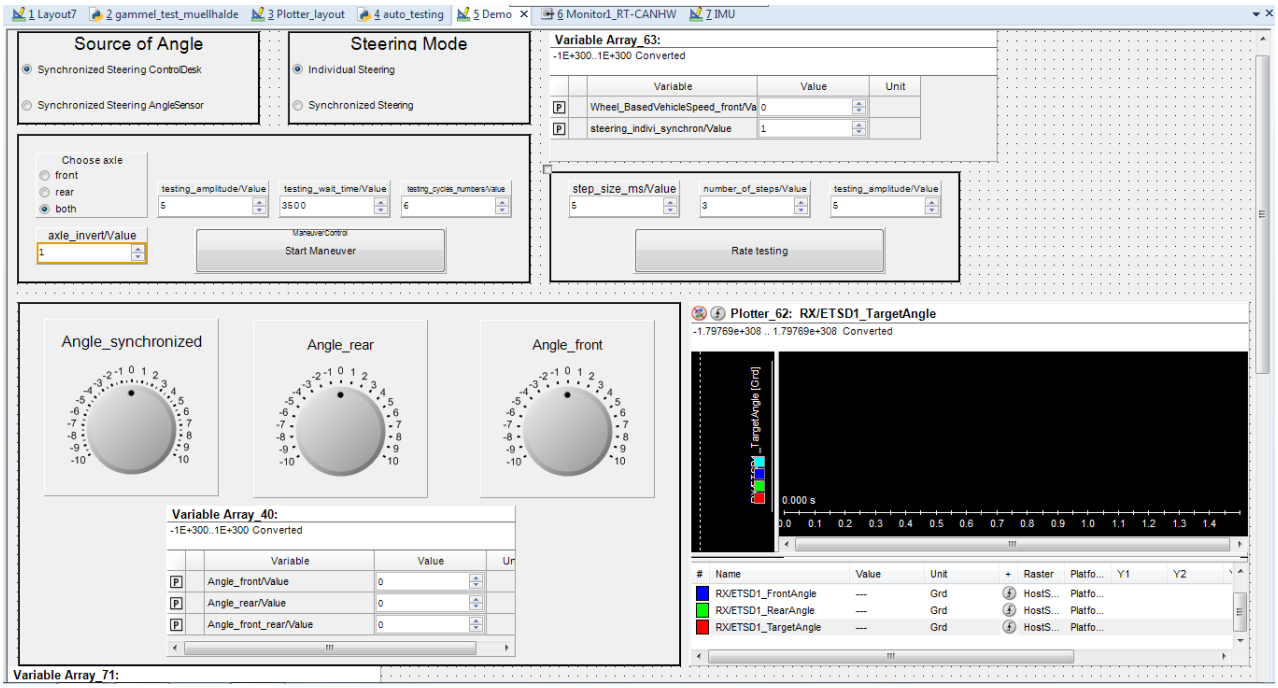


Figure A.4: ControlDesk GUI for basic testing and start-up of the dolly

Table A.1: Severity table for FMEA [14]

Severity	**Description**	**Ranking**
Hazardous without warning	Very high severity ranking when a potential failure mode effects safe system operation without warning	10
Hazardous with warning	Very high severity ranking when a potential failure mode affects safe system operation with warning	9
Very High	System inoperable with destructive failure without compromising safety	8
High	System inoperable with equipment damage	7
Moderate	System inoperable with minor damage	6
Low	System inoperable without damage	5
Very Low	System operable with significant degradation of performance	4
Minor	System operable with some degradation of performance	3
Very Minor	System operable with minimal interference	2
None	No effect	1

Table A.2: Probability table for FMEA [14]

Probability	**Description**	**Ranking**
Very High: Failure is almost inevitable	>1 in 2	10
	1 in 3	9
High: Repeated failures	1 in 8	8
	1 in 20	7
Moderate: Occasional failures	1 in 80	6
	1 in 400	5
	1 in 2,000	4
Low: Relatively few failures	1 in 15,000	3
	1 in 150,000	2
Remote: Failure is unlikely	<1 in 1,500,000	1

Table A.3: Detectability table for FMEA [14]

Detectability	**Description**	**Ranking**
Absolute Uncertainty	Design control cannot detect potential cause/mechanism and subsequent failure mode	10
Very Remote	Very remote chance the design control will detect potential cause/mechanism and subsequent failure mode	9
Remote	Remote chance the design control will detect potential cause/mechanism and subsequent failure mode	8
Very Low	Very low chance the design control will detect potential cause/mechanism and subsequent failure mode	7
Low	Low chance the design control will detect potential cause/mechanism and subsequent failure mode	6
Moderate	Moderate chance the design control will detect potential cause/mechanism and subsequent failure mode	5
Moderately High	Moderately High chance the design control will detect potential cause/mechanism and subsequent failure mode	4
High	High chance the design control will detect potential cause/mechanism and subsequent failure mode	3
Very High	Very high chance the design control will detect potential cause/mechanism and subsequent failure mode	2
Almost Certain	Design control will detect potential cause/mechanism and subsequent failure mode	1

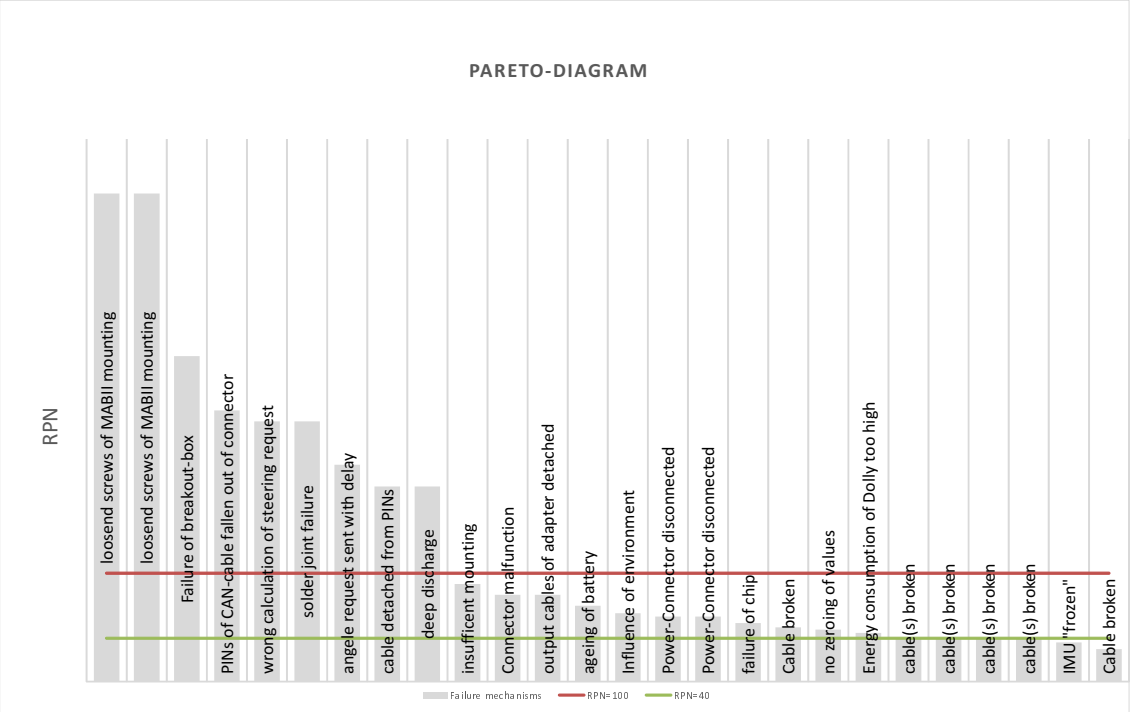


Figure A.5: Pareto diagram for RPNs of FMEA

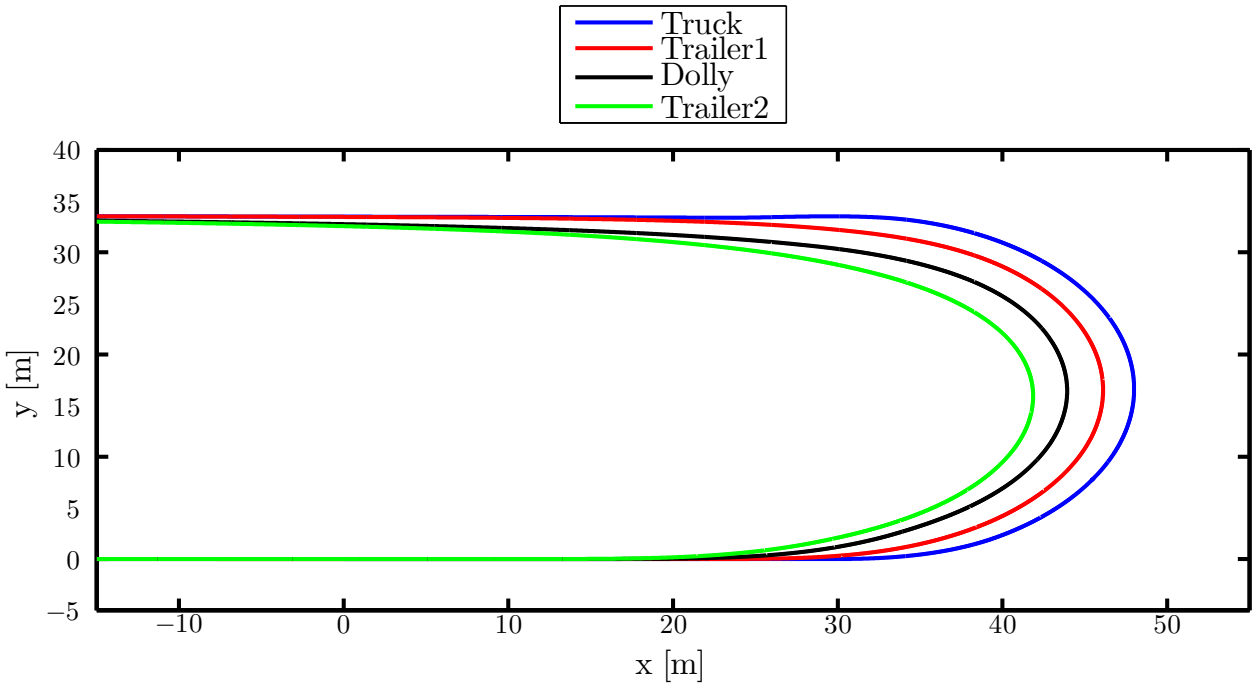


Figure A.6: Path representation of different units in the HCT combination simulated in VTM

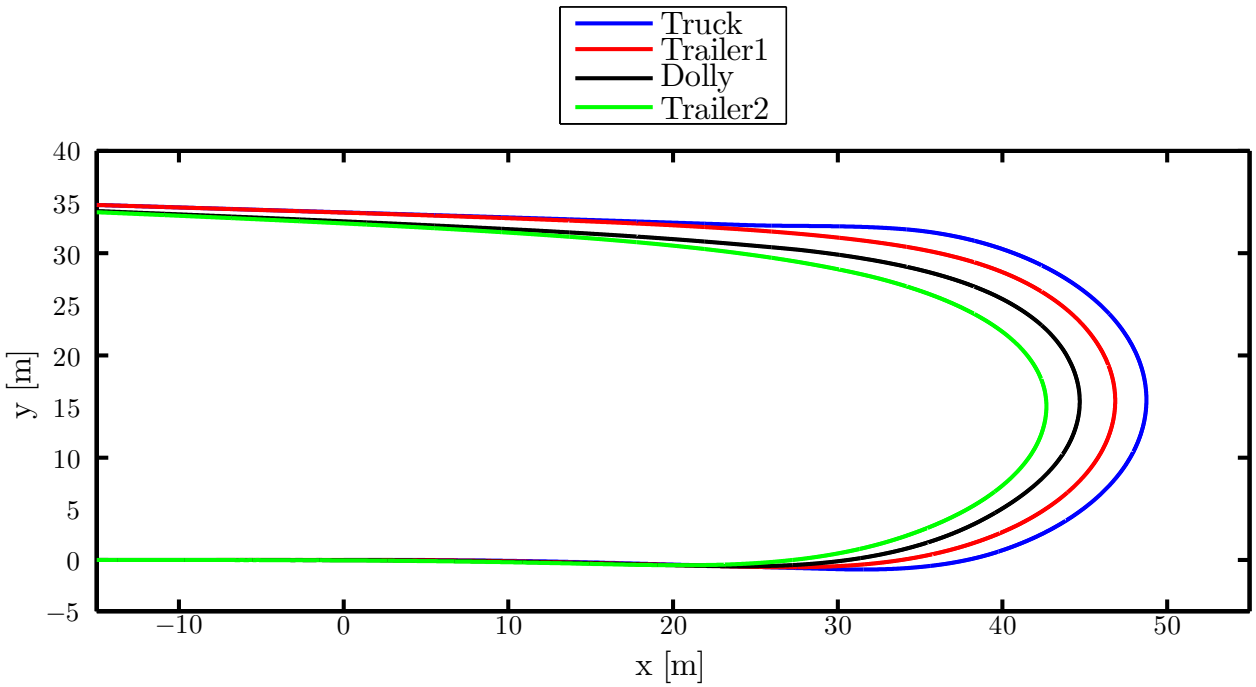


Figure A.7: Path representation of different units in the HCT combination during Hardware-in-the-Loop-testing

References

[1] I. Åkerman and R. Jonsson. European Modular System for road freight transport–experiences and possibilities. *TFK report* (2007).

[2] H. Backman and R. Nordström. Improved performance of European long haulage transport. *Transport Research Institute (TfK), Stockholm* (2002).

[3] A. Bálint et al. "Correlation between truck combination length and injury risk". *Australasian College of Road Safety Conference, 2013, Adelaide, South Australia, Australia*. 2013.

[4] V. V. S. E. B.V. *Product information ETS for trailers*. 2014.

[5] S. Corrigan. Introduction to the controller area network (CAN). *Texas Instrument, Application Report* (2008).

[6] E. DIN. 60812: 2006-11 (2006) Analysetechniken für die Funktionsfähigkeit von Systemen–Verfahren für die Fehlzustandsart-und-auswirkungsanalyse (FMEA)(IEC 60812: 2006). *Deutsche Fassung EN* **60812** (2006).

[7] C. Doll et al. Long-Term Climate Impacts of the Introduction of Mega-Trucks, Study to the Community of European Railways and Infrastructure Companies (CER). *Fraunhofer ISI (Study Co-Ordinator, Karlsruhe) TRT (Milan), NESTEAR (Gentilly), Fraunhofer-ATL (Nuremberg), Fraunhofer-IML (Dortmund), Karlsruhe* (2009).

[8] GitHub. *collin80/due_can*. 2015. URL: `https://github.com/collin80/due_can`.

[9] GitHub. *Seeed-Studio/CAN_BUS_Shield*. 2015. URL: `https://github.com/Seeed-Studio/CAN_BUS_Shield/`.

[10] dSpace GmbH. *ControlDesk Next Generation 5.2 - HelpDesk*. 2014.

[11] dSpace GmbH. *MicroAutoBox II - Product Brochure*. 2013. URL: `https://www.dspace.com/shared/data/pdf/2013/ProductBrochure_MicroAutoBox-HW_E_ebook.pdf`.

[12] F. B. K. GmbH. *Excellent driving characteristics outstanding manoeuvrability - The active articulated dolly*. Sept. 2012. URL: `http://www.krone-trailer.com/english/neuheiten/excellent-driving-characteristics-outstanding-manoeuvrability/` (visited on 05/12/2015).

[13] M. Hjort, M. Haraldsson, and J. M. Jansen. *Road Wear from Heavy Vehicles: An Overview*. NVF Committee Vehicles and Transports, Nordiska vägtekniska förbundet, 2008.

[14] U. of Idaho. *FMEA*. URL: `http://www.ece.uidaho.edu/ee/classes/EE481S03/FMEA.PDF` (visited on 05/25/2015).

[15] M. M. Islam, L. Laine, and B. Jacobson. "Improve Safety by Optimal Steering Control of a Converter Dolly using Particle Swarm Optimization for Low-Speed Maneuvers". *2015 IEEE 18th International Conference on Intelligent Transportation Systems, Las Palmas de Gran Canaria, Spain*. 2015.

[16] M. M. Islam, L. Laine, and B. Jacobson. "Inverse Model Control Including Actuator Dynamics for Active Dolly Steering in High Capacity Transport Vehicle". *2015 IEEE Intelligent Vehicles Symposium, Seoul, Korea*. 2015.

[17] M. S. Kati et al. "Performance Improvement for A-double Combination by introducing a Smart Dolly". *13th International Heavy Vehicle Transport Technology Symposium, San Luis, Argentina*. 2014.

[18] K. T. G. . C. KG. *Kögel Lang-Lkw: Drei Dollys stehen zur Wahl*. Apr. 11, 2012. URL: `http://www.koegel.com/de/news/artikel/datum/2012/04/11/koegel-lang-lkw-drei-dollys-stehen-zur-wahl/` (visited on 05/12/2015).

[19] *mikalhart/TinyGPSPlus*. 2013. URL: https://github.com/mikalhart/TinyGPSPlus/.
[20] P. Nilsson. On Traffic Situation Predictions for Automated Driving of Long Vehicle Combinations (2015).
[21] *ntruchsess/arduino$_u$ip*. 2014. URL: https://github.com/ntruchsess/arduino_uip.
[22] PololuCorporation. *AltIMU-10 v4 Gyro, Accelerometer, Compass, and Altimeter (L3GD20H, LSM303D, and LPS25H Carrier)*. 2015 (accessed February 12, 2015). URL: https://www.pololu.com/product/2470/.
[23] *pololu/l3g-arduino*. 2015. URL: https://github.com/pololu/l3g-arduino.
[24] *pololu/lsm303-arduino*. 2015. URL: https://github.com/pololu/lsm303-arduino.
[25] "Rapid Control Prototyping". German. *Rapid Control Prototyping*. Springer Berlin Heidelberg, 2006, pp. 295–318. ISBN: 978-3-540-29524-2. DOI: 10.1007/3-540-29525-9_7. URL: http://dx.doi.org/10.1007/3-540-29525-9_7.
[26] J. Schäuffele and T. Zurawka. *Automotive software engineering: Grundlagen, Prozesse, Methoden und Werkzeuge effizient einsetzen*. Springer-Verlag, 2012.
[27] STMicroelectronics. *L3GD20H - MEMS motion sensor: three-axis digital output gyroscope: Datasheet - production data*. 2013.
[28] STMicroelectronics. *LSM303D - Ultra compact high performance e-Compass 3D accelerometer and 3D magnetometer module: Datasheet — preliminary data*. 2012.
[29] G Wangrin, B Stürmer, and M Wöhrmann. Technische Erprobung von Fahrzeugkombinationen mit einer Gesamtlänge bis 25, 25m („GigaLiner "). *Abschlussbericht NRW Modellversuch. TÜV Rheinland Kraftfahrt GmbH, Köln* (2009).