System Architecture in a Heavy Duty Vehicle Platooning System using xPC Target™

CARL-JOHAN ELM

KTH Electrical Engineering

Master's Degree Project
Stockholm, Sweden 2013

XR-EE-RT 2013:016

Abstract

The ongoing environmental debate emphasizes the need for reducing CO_2 emissions in order to counteract rapid climate change. The transportation industry is growing and is already today a significant contributor to green house gas emissions. Information and Communication Technology (ICT) can be utilized to create intelligent transportation systems which have proven to be able to reduce fuel consumption significantly. The concept of vehicle platooning involves vehicles traveling in a longitudinal convoy with a short intermediate distance. By introducing ICT to a vehicle platoon the intermediate distance can be even shorter without compromising traffic safety. The shorter distance also allows for reduced drag, thus reduced fuel consumption.

A research group from KTH Royal Institute of Technology and Scania CV AB competed successfully with an autonomous platooning system in the Grand Cooperative Driving Challenge in 2011. In 2012 the cooperation continued and a new platooning system including lateral maneuvers, i.e. lane changing, was implemented. This master thesis presents the design and implementation of a new hardware and software architecture for the platooning system.

For flexibility and control prototyping issues a new real-time platform supporting the system was tested. The choice was to evaluate and test the xPC Target™ system developed by MathWorks. The xPC Target platform supports real-time code generation from Simulink® and also provides an extensive I/O library needed for communication with external physical systems. In order to adapt to the xPC Target platform and lateral platooning scenarios a new system architectural framework was suggested. The architecture emphasized the need for separated components each with clearly delimited tasks.

Another important requirement was to provide a system with a short learning curve for new master students that will work with the system in the future.

A new system architecture approach was presented and implemented in two real-time computer systems. The first computer executed a real-time software created in C++ which primarily handled wireless transmission. The second unit utilized a real-time software created in Simulink® and executed on the xPC Target platform. Successful vehicle platooning experiments, including lateral maneuvers, were performed with the new system in November 2012.

Acknowledgements

Acknowledgments to my supervisor Jonas Mårtensson at the Automatic Control department at the Royal Institute of Technology (KTH) in Stockholm for his help as my supervisor during this Master thesis. I also appreciate the help from Sagar Behere and Dennis Sundman, both PhD students at KTH. I would also like to thank the students and faculty from Chalmers University and The Institute of Technology: Linköping University for their participation and dedication to the CoACT project. Special thanks go to Per Svennebrandt from Linköping University for a successful cooperation during CoACT 2012.

Many thanks to the master student project members in Scoop 2012. A great atmosphere has made my days enjoyable both in times of trouble and in times of success.

Finally I want to express gratitude towards my closest friends and family for their support during my studies.

Contents

Chapter 1

Introduction

This is a Master Thesis at KTH Royal Institute of Technology in Stockholm. The work is made in cooperation with the predevelopment department at R&D Scania CV AB in Södertälje. The ambition is to participate in a final demo of the CoACT project in November 2012.

1.1 Background

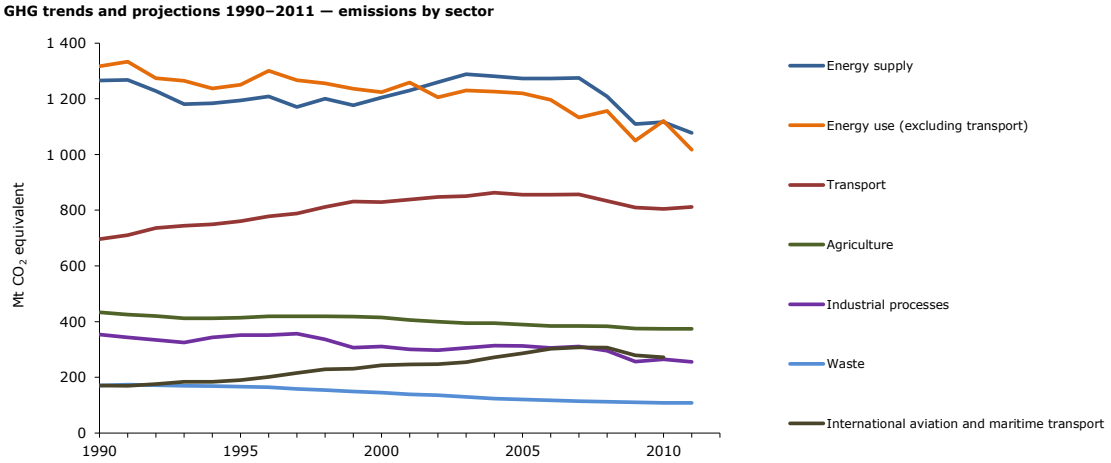
The continuously ongoing environmental debate emphasizes the need for reducing CO_2 emissions in order to counteract climate changes. The *"20-20-20"* targets are objectives constituted by the European Union. The *"20-20-20"* targets aims at reducing green house gas emissions by increasing the amount of renewable energy sources and increasing energy efficiency in the european area by year 2020.

The transportation industry is a key sector in order to accomplish permanent reduction of CO_2 emissions. But the transportation industry is currently facing huge challenges. Statistics from the European Commission shows that almost all sectors in Europe except the transport industry has decreased their emissions of CO_2 between the years 1990 and 2007. During the same period the European road transport sector has increased its total emissions by 24% in the EU-15 [2]. The trends in greenhouse gas emissions is depicted in Figure 1.1a and the emission distribution among different sectors is shown in Figure 1.1b.

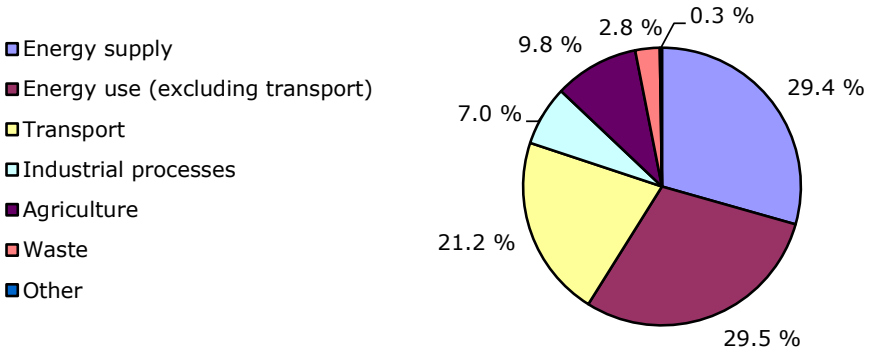
Freight transport by heavy duty vehicles, which plays a big part of the transport sector, has attained specific interest when addressing the reduction of CO_2 emissions. Technological innovations are important in order to counteract the environmental effects and ensure a more sustainable future for the freight transport sector. Suggested areas of research for decreasing the fuel consumption are improvements in fuel and engine efficiency. But also the use of information and communication technology (ITC) to create intelligent transportation systems could potentially tackle these problems.

1.1.1 Vehicle platooning

By introducing ITC to a fleet of vehicles, an intelligent vehicle platooning system can be utilized. The concept of vehicle platooning involves vehicles to travel in a longitudinal convoy with a short intermediate distance. By introducing intelligence to vehicle platooning system the intermediate distance can be even shorter, than during ordinary manual operation, without compromising traffic safety. An automated system for controlling the lateral speed and acceleration is implemented to counteract the effects a closer distance imposes on traffic safety. Empirical evidence indicates that platooning possibly can reduce



Trends in greenhouse gas emissions by sectors in the EU-15 zone between years 1990-2011.[1]



Share of greenhouse gas emissions by main source in the EU-15 zone in 2010.[1]

Figure 1.1: Trends and shares of greenhouse gas emissions in the EU-15 zone. From 1990 only the transport, aviation and maritime transport sector has increased their emissions.

traffic congestion and fuel consumption. An example shows that intelligent platooning experiments utilizing an Adaptive Intelligent Cruise Control has managed to accomplish 7.7% reduction of fuel consumption for two heavy duty vehicles (HDVs) [3].

The intelligence of a platooning system is based on the use of inter-vehicular communication, so called Vehicle-to-Vehicle (V2V) communication. This allows for vehicle control systems to communicate using wireless transmission techniques. By broadcasting vehicle information including position, speed, acceleration and additional information, surrounding vehicles can react instantaneously to unexpected changes in the current traffic scenario.

Vehicles equipped with communication systems can also possibly interact with the surrounding environment. The phenomena is referred to as Vehicle-to-Infrastructure (V2I) communication. Information from smart traffic lights or traffic flow monitors increases the ability to automatically adapt to the road infrastructure resulting in increased traffic throughput.

1.2 GCDC

The Grand Cooperative Driving Challenge (GCDC), is an example of a research project utilizing V2I- and V2V-communication in order to increase traffic throughput. GCDC was an initiative from High Tech Automotive Systems and TNO, that was hosted for the first time in May 2011 in the Netherlands. The main goal of this event was to create a competition were international teams would compete to deliver the most effective vehicle-infrastructure system in predefined traffic scenarios. Information about the scenarios, rules, requirements and technology were provided by the GCDC organization. Complete documentation can be found in [5].

In cooperation with Scania CV AB, the Royal Institute of Technology(KTH) participated in GCDC with a team called Stockholm cooperative driving (Scoop). The team consisting of master and PhD students, researchers and engineers worked together to create a platooning system for a Scania truck [4].

1.3 CoACT 2012

CoACT is a Swedish research project specialized in the field of cooperative driving systems. The project is a collaboration between Chalmers University of Technology, Halmstad University, Linköping University, KTH and the Viktoria Institute. CoACT was involved in the GCDC 2011, by cooperating the preparations for the Swedish teams before the final competition in Netherlands. In 2012 no GCDC challenge was hosted. But in order to motivate continuous development of Swedish cooperative driving systems CoACT took the initiative to host a national challenge called CoACT 2012. During CoACT additional scenarios and platooning functionalities were added to the GCDC setup. Special interest was directed towards lateral maneuvers, i.e. autonomous lane changing and reordering of vehicles within a platoon. CoACT 2012 consisted of two events, a first workshop hosted in Nyköping, Sweden and a final event at Stora Holm test track in Gothenburg, Sweden.

1.3.1 CoACT 2012 Scenarios

A major part of the GCDC was to counteract string effects in vehicle platooning. String effects in terms of platooning refers to a behavior when small changes in velocity of a single vehicle propagates and amplifies throughout the platoon leading to unwanted platooning characteristics. Research on string stability in HDV platoons reveals that string stability is guaranteed if the platoon has a descending order of vehicle weights within the platoon [10].

In the GCDC only longitudinal platooning operations were considered, but in order to guarantee string stability the platoon order might have to be changed. The new scenarios in CoACT 2012 therefore included platoon ordering changes during autonomous driving. The scenarios in CoACT 2012 did not require communication with infrastructure, as was the case in GCDC. During the final event of CoACT four vehicles are intended to perform intelligent platooning, one platoon leader with three followers. The platooning maneuvers in the final event are:

- Intelligent longitudinal platooning
- Leave from middle
- Create gap

- Join from side
- Switch position

The wireless communication protocol developed for the GCDC will still be used in CoACT, with additional changes added to comply with lateral operations. In Table 1.1 abbreviations with corresponding explanations are presented to assist and clarify the explanation of the scenarios in CoACT 2012.

Abbreviation	Explanation
PL(Platoon Leader)	Manually controlled vehicle responsible for coordinating ordering and requests in the platoon.
MRQ(Maneuver ReQuest)	All changes in a platoon that differs from ordinary platooning state must be initialized by a MRQ. MRQ contains information about reference vehicle, destination vehicle, longitudinal offset and lane offset.
MS(Maneuver State)	During an active maneuver the destination vehicle periodically transmits a message declaring that its currently performing a maneuver.
MOA(Maneuver Offset Active)	Upon completion of a maneuver the destination vehicle transmits a MOA message to signal that the current maneuver is now finished.

Table 1.1: A brief description of the platooning glossary and abbreviations used in CoACT 2012.

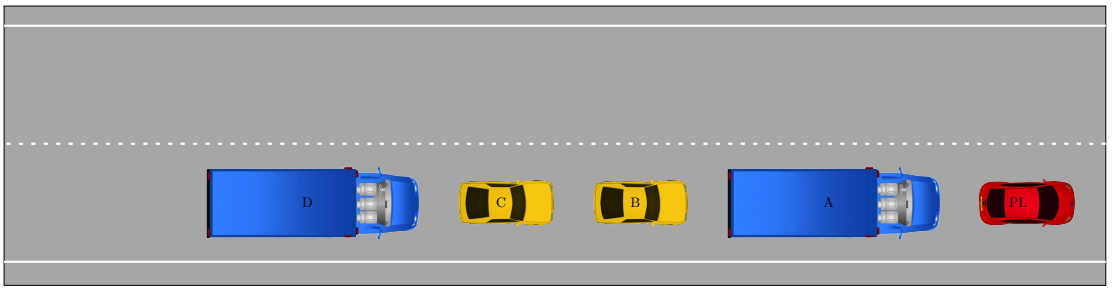
The Switch

In Figure 1.2 a description of the "Switch" maneuver is presented. The scenario test the ability of the participants to change position of vehicles within the platoon. This scenario is constituted of all new maneuvers presented in CoACT, this includes creating gap, leave from middle, join form side and ordinary intelligent vehicle platooning.

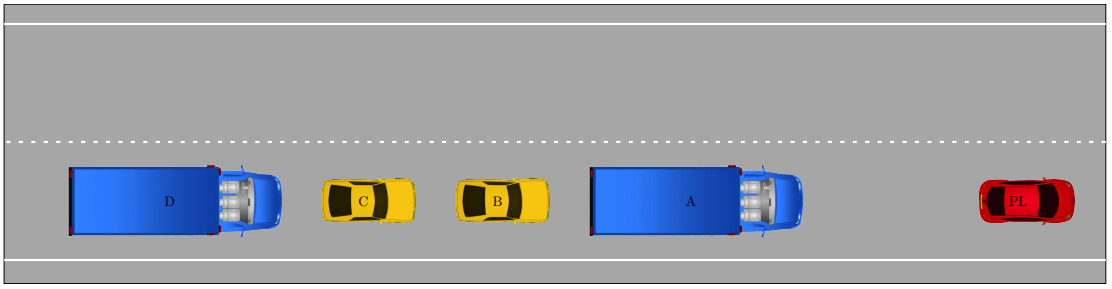
1.3.2 CDS 2012

As of year 2012 a new system for cooperative driving will be developed at KTH in cooperation with Scania. For the GCDC all control strategies were implemented on a Scania specific Electric Control Unit (ECU). In order to have a more independent autonomous platooning system the ECU was to be removed. An important design requirement for CDS 2012 is hence to replace the ECU with a more general real-time computer. The choice by the Automatic Control at KTH department was to use xPC Target developed by the MathWorks company. xPC Target is a software solution for running real-time applications developed in Simulink on Target PCs.

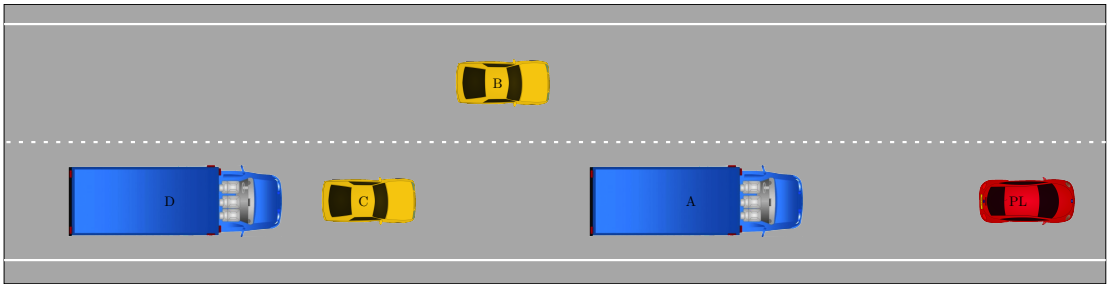
A project group from KTH consisting of seven master students was to develop different parts of a cooperative driving system. Ali Reyhanoğulları worked with the logical operations performed within the platoon of vehicles. Achour Amazouz examined the possibility of creating a model predictive control (MPC) approach for a vehicle platooning. While A. Amazouz was deriving an MPC for control purposes, Ali Madadi were responsible for evaluating different control approaches and choosing a good controller for the scenario at



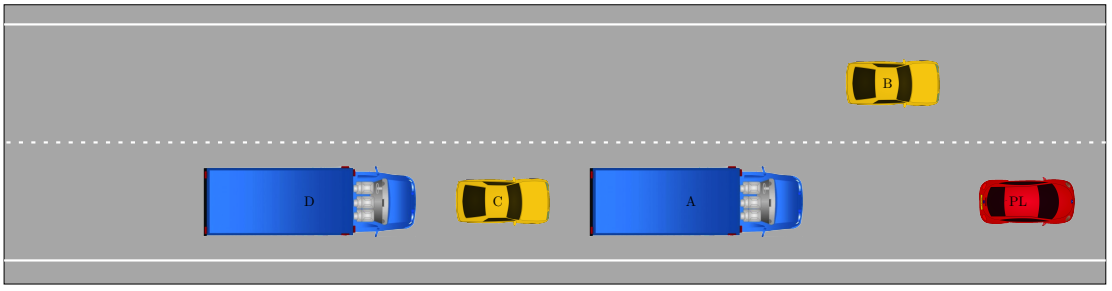
Initial platooning sequence. PL sends a MRQ to vehicle A to increase the inter-vehicular distance to PL.



Vehicle A responds to the MRQ by starting to transmit MS and when finished with the maneuver sending MOA. After receiving MOA, PL sends an MRQ to vehicle B, that it should maneuver in the left lane with PL as reference vehicle.



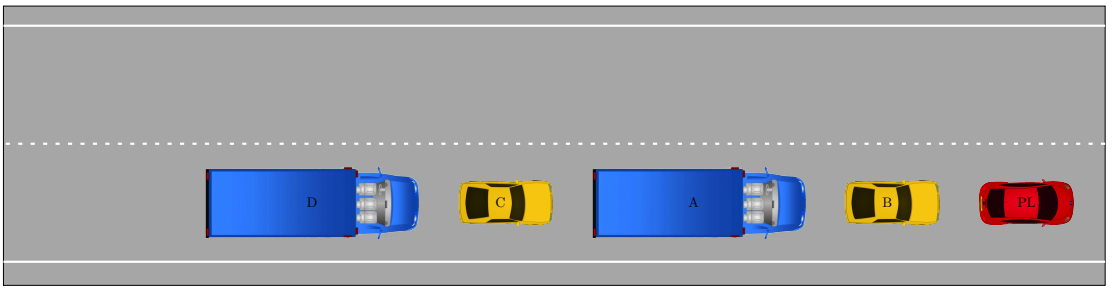
Vehicle B starts to perform the maneuver whilst emitting an MS message.



Vehicle B has now completed the requested maneuver and signals this by transmitting an MOA.

Figure 1.2: The switch

hand. The state estimator and sensor fusion were developed by José Luis González-Conde. Magnus Almroth was appointed the task to develop a simulation environment suitable for testing platooning strategies in PreScan software.



Vehicle B has finished the maneuver and emits an MOA. The platoon is now in normal platooning state again but vehicle A and B has swapped positions within the platoon.

Figure 1.2: The switch

1.4 Problem definition

The goal of this sub-project is to deliver an adaptive system architecture for a Scania truck which is suppose to interact in a cooperative driving environment. An adaptive architecture in this scenario refers to a system were all components are clearly separated, so that local changes does not imply major changes to other parts of the system. The platform for the system architecture is to be developed in Simulink. In the Simulink model all components should be clearly represented and separated, so that the model easily could work as a framework for future platooning projects for Scoop.

A major change compared to the Scoop 2011 system is to implement a new ethernet based communication link for the internal platooning system communication. This communication link creates demands for a new communication protocol, creating and implementing this protocol is within the scope of this masters thesis. All implementations of architecture and communication will be created in an Simulink environment and later executed on a real-time xPC Target kernel.

In order to simplify the representation of the project objectives the main goals will be presented in a dotted list below.

- Implement the GCDC 2011 system on the new real time computer.
- Present a new adaptive system architecture as a framework for implementing controllers, logic and estimators.
- Create and implement required I/O communication links between the xPC Target computer and other platooning system components.

1.5 Outline

A short summary of the thesis is presented in the outline. All chapters are presented below accompanied by a short sentence describing the chapter content.

- **Chapter 2**
  This chapter gives a short introduction to the xPC Target software.
- **Chapter 3**
  The architecture of the Scoop 2011 system is presented along with prerequisites

and requirements for the Scoop 2012 system. Later the system architecture with components are presented shortly.

- **Chapter 4**
  Implementation of new system components are presented in this chapter.
- **Chapter 5**
  Results of simulations and experiments of the Scoop 2012 system are presented.
- **Chapter 6**
  Analysis and discussion around the results of this master thesis. Suggestions to future improvements and add-on functionalities are also presented.

Chapter 2

xPC Target

In order to test and execute Simulink models created for real-time operations, MathWorks has developed a software environment called xPC Target. During execution the xPC Target environment consists of two separable physical platforms; a Host PC and a Target PC. The Host PC is used for model development and for controlling applications during runtime. On the Target PC real-time applications created in Simulink are uploaded and executed. The environment provides a library of drivers for I/O devices and a real-time kernel for code execution. The features combined enables interaction between a real-time application on the Target PC and a physical plant.

This chapter starts with a brief introduction to Simulink code generation using Simulink Coder, with certain attention directed towards code generation for the xPC Target. This is followed by an introduction to the xPC Target environment. The last part of this section is dedicated to discuss issues of implementation using code generation in xPC Target.

2.1 xPC Target Software

xPC Target is a MathWorks product that can be executed on any nominal PC. But xPC Target is not a standalone software product, it requires additional MathWorks products. A list of the required software along with their respective role in the xPC Target software interaction is presented below.

- MATLAB® - Configuration and interaction with the xPC Target software.
- Simulink® - Modeling of dynamical systems and controller blocks.
- Simulink Coder™ - Compile Simulink model files into C code.
- C-Compiler - A Simulink supported third-party compiler, used for building target executable applications.

In addition to the software listed above the xPC Target system requires the following hardware.

- Host PC - Application development and runtime interfacing.
- Target PC - Real-time execution and interfacing with physical systems.

2.1.1 Real-Time Code Generation

The basis for any real-time Simulink application is to generate target specific C/C++ code from a Simulink model file. Nominally this is performed using Simulink Coder. The workflow for creating real-time executable code from a Simulink model is depicted in Figure 2.1.

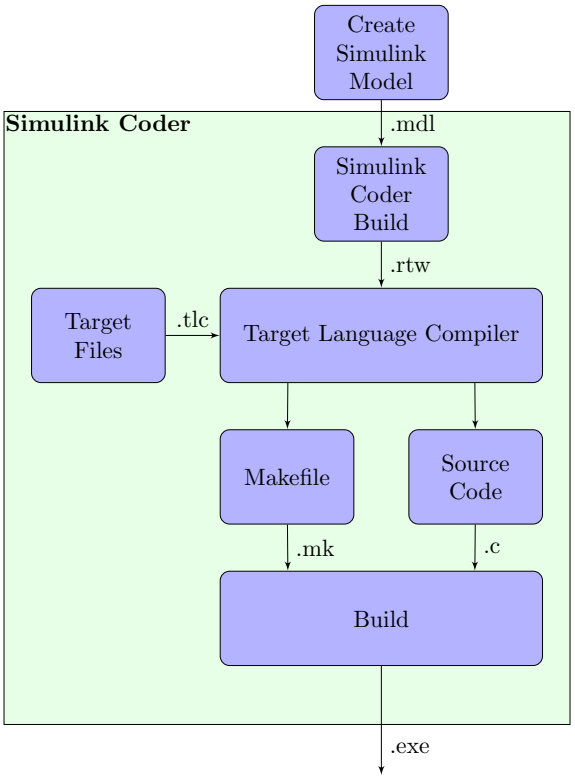


Figure 2.1: Build procedure overview of a real-time executable application from a Simulink model file.

Using the workflow chart in Figure 2.1 as reference the procedure of code generation in Simulink will be described step by step.

- The first step of the development procedure is to create a Simulink model file (`.mdl`) dedicated for real-time execution. If the model is intended to interact with a physical system specific Simulink library blocks with I/O drivers must be included during development. Some operations and structures are not supported for code generation, discussion concerning these problems will be presented in Section 2.4.

- After creating an `.mdl` file, the initial build step performed by the Simulink Coder is to create a model description file (`.rtw`). The `.rtw` file is a high-level descriptive file of the Simulink model. The content of the `.rtw` is mainly descriptions of blocks, signals, parameters, sampling times and dynamic states within the model. An important note is that the `.rtw` code is not platform specific, instead it works as a compiled version of the `.mdl` file, compatible with any hardware with appropriate I/O cards.

- After model compilation the Simulink Coder generates target specific code. This is performed by the Target Language Compiler (TLC) in combination with a TLC

function library. Together these are designed to create C/C++ source code for supported platforms. The target library (`.tlc`) files contains block specific files used to separately transform each block in the `.rtw` file to platform specific code.

- After the makefile and source code is generated from the TLC, a third-party compiler creates an executable application file. This real-time application can now be downloaded and executed on the platform specified during the build process.

2.1.2 Host-Target Communication

The xPC Target environment requires two different computers, a Host PC and a Target PC. The Host computer is an ordinary PC platform running a Windows operating system. The Target is also an ordinary x86 PC, but running a real-time kernel provided by xPC Target. An overview of the Host-Target communication and architecture is presented in Figure 2.2.

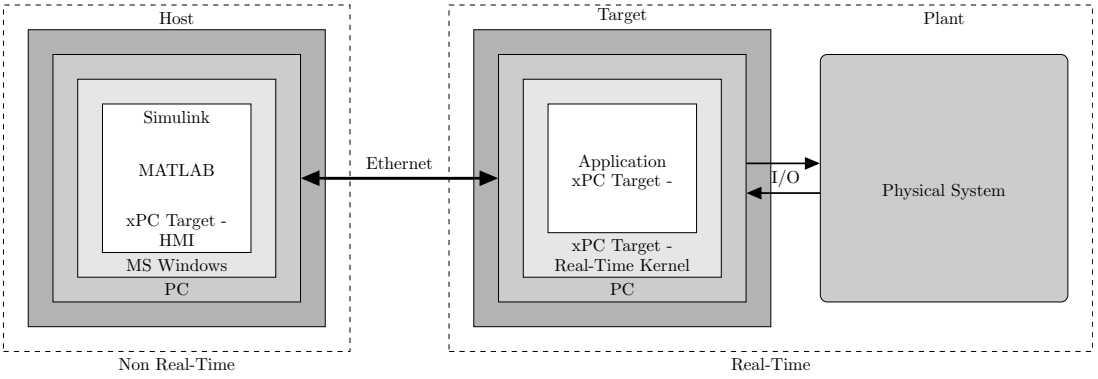


Figure 2.2: Architectural overview of xPC Target environment. Including Host PC, Target PC and Host-Target communication link.

The tasks between the Host and Target are clearly separated. Software development and testing, that does not require communication with external hardware, are performed on the Host computer. All development is typically performed in Simulink and MATLAB using standard libraries and functions supported for code generation. During runtime operations the Host computer is also designed for supporting the user with a Human-Machine Interface (HMI), which can be used for online data visualization, parameter tuning and logging. Further descriptions of the functionalities during runtime are described in Subsection 2.2.

Communication between the Host PC and Target PC, Host-Target link, supports both ethernet and serial communication. There exists several apparent advantages to using a networked ethernet connection as apposed to serial communication. The most apparent being data bandwidth, which is important when transmitting data from the Target PC to the Host PC for logging and online data visualization.

The Target computer executes generated real-time applications on the xPC Target real-time kernel. The kernel provides a real-time operating system and therefore does not require any additional operating system. A bootable kernel, that can be transferred to any portable media, is created using the xPC Target software on the Host PC and installed on the Target PC. The kernel can be customized for multicore support, which is important when executing applications with multiple sampling rates. During startup the kernel activates the Host-Target link and waits for the target application to be downloaded

from the Host PC. Once the target application is downloaded to the Target PC, it can be executed, monitored and controlled from the Host PC.

The xPC Target environment provides an extensive Simulink library of external I/O device drivers. The drivers contains code that executes on the Target computer for interfacing with external hardware. The standard library contains numerous different types of I/O communication, for example Ethernet, Serial, Video and other communication data buses. When developing models, the I/O blocks works similar to standard Simulink blocks with numerous configurations and settings.

2.2 xPC Target - Real-Time execution

Tuning parameters, data logging and real-time visualization of data are important features for control and testing of dynamical systems. To perform these tasks, the xPC Target software provides a command-line interface and an HMI, called xPC Target Explorer.

The basis for all signal monitoring and logging in xPC Target environment is provided by a single specialized scope, called xPC Target Scope. The scope can be configured to different working modes depending on user preferences. In addition to acquiring real-time data from scopes, Stateflow states can also be monitored.

In the following paragraphs the working modes of the xPC Target Scope will briefly be cover to highlight pros-and cons of using the xPC Target signal monitoring system. One common denominator in the xPC Target software environment is that signals exists that are non-observable. This type of signal cannot be monitored or tuned during real-time execution, this is crucial for user interfacing. The typical signals that are non-observable are structures and signals of multiple datatypes.

2.2.1 Signal Monitoring

Signal monitoring as apposed to signal logging provides ability to visualize application data during real-time execution. To choose between different signal logging and monitor operating modes the xPC Target Scope Simulink block has a Scope type selector with three separate scope types. The first operating mode of the xPC Target Scope is called *TARGET*, which emphasizes that all signals are displayed on a monitor connected to the Target PC. Using this scope reduces computational capabilities as the Target PC continuously needs to perform additional I/O interaction with the graphics cards. Other limitations are for example, the signal displaying environment lacks configurable options and that a maximum of eight *TARGET* scopes can be integrated in the Simulink model.

The second signal monitoring mode of the xPC Target Scope is the *HOST* mode. During real-time execution signal data is transferred via the Host-Target link to the Host PC. As opposed to the *TARGET* this scope can include ten scopes. But each scope reduces the computational power as the scopes interacts with I/O cards on the Target PC. An advantage using this *HOST* mode is the ability to customize an HMI using xPC Explorer.

A third approach to signal monitoring is to use the MATLAB command line. All observable signals in an application are stored as variables in a struct when uploading the application to the Target PC. This approach to signal monitoring is problematic as the application becomes more complex. Since all observable signals are stored in a common list, the list becomes extensive and hard to navigate in. On the other hand this provides the ability to create an application specific HMI using GUIDE or other MATLAB compatible

interfacing softwares.

2.2.2 Signal Logging

Storing signal and time data on the Target hard drive is critical for offline analysis and application development. To perform signal logging in a real-time application add an xPC Target Scope and configure it to run as a *FILE* Scope type. A maximum of eight *FILE* scopes are allowed in one application. During runtime data is stored on the disk of the xPC Target. When configuring the *FILE* Scope an important parameter is to configure the maximum amount of samples that are stored on the hard drive. Once the maximum number of samples are filled, the data starts to overwrite the stored buffer.

When an application has been executed and stopped after an experiment, the operator must download the files from the hard drive of the Target PC. The interaction with the hard drive of the Target PC is performed using the command line in MATLAB or via the provided HMI.

2.2.3 Parameter Tuning

The xPC Target software allows the user to change parameters during runtime. This is an efficient tool since this allows the user to tune controller parameters, change offsets and frequencies of sources or other features during online testing on a physical system. The parameter tuning can be performed using an HMI or directly via the MATLAB command line.

2.2.4 Human-Machine Interface

The xPC Explorer is a software provided for interaction with the Target PC via the Host PC. The interface is designed to perform basic operations such as downloading, starting and stopping an application. But also online presentation of graphical data that can be visualized both in numerical and graphical form. In order to handle applications with huge amounts of variables, signal groups can be created. Signals are then added to a group which is stored as a file on the Host PC, so between separate executions the same signals can be fetched.

In addition to using the xPC Explorer, interaction can be performed using the MATLAB command line. In the Scoop 2012 system the command line interfacing was incorporated into an HMI designed by the members of the group. The interface was created using GUIDE, a graphical interface editor provided by MATLAB. The interface allows for manually tuning selected parameters and visualizing signals during runtime. The GUIDE interface was also used for starting, stopping, downloading the executable to the Target PC and downloading logging data from the Target PC hard drive.

2.3 Hardware

During Scoop 2012 two different computers acting as the xPC Target PC has been tested. First using an ordinary desktop computer with additional I/O boards. The I/O boards included standard desktop units; ethernet, RS232 and USB connectors. But also Controller Area Network(CAN) connectors, for vehicle interfacing were installed.

The second solution was using an industrial Mobile real-time target machine provided by the Speedgoat company. The compact design without moving parts and extensive test-

ing makes it more robust than ordinary computers. The computer hardware specifications of the two real-time computer systems are presented in Table 2.1.

One major difference between the Speedgoat computer and any ordinary desktop computer is that the Speedgoat company supplies a unique Simulink library for interfacing with the Speedgoats I/O modules. This results in that the Speedgoat- and desktop computer cannot use the same Simulink model file when performing I/O interaction.

	xPC Target Computer	
Hardware	**Intel Computer**	**SpeedGoat**
CPU	Intel Atom™ 1.86GHz	Intel Core 2 Duo 2.16GHz
Chipset	Intel NM10	Intel 945GM
Memory	4048MB DDR2 RAM	3072MB DDR2 RAM
USB	4×USB 2.0	4×USB 2.0
Ethernet	1×Host-link 10/100/1000Mbps, 1×Ethernet 10/100/1000Mbps,	1×Host-link 10/100/1000Mbps, 1×Ethernet 10/100/1000Mbps, 4×Ethernet 10/100Mbps
Video	1×VGA, 1×DVI-I	1×DVI-I
I/O interfaces	2×CAN, 2×RS232	4×RS232, 2×CAN
Size (D×W×H)	273mm×300mm×65mm	82mm×270mm×162mm
Weight	3.7kg	1.8kg

Table 2.1: Hardware specifications of the two real-time computer system evaluated in the Scoop 2012 system.

In order to achieve real-time performance a powerful processor is required, in order to within one system sampling period perform all necessary computations. If this is not achieved the xPC Target application is terminated. The xPC Target system achieves this by dedicating all real-time execution to the Target PC. The actual time it takes for the xPC Target to run model calculations and post outputs during each sample interval in an application is called Task Execution Time (TET). The TET is important in order to determine the minimum achievable sampling time of the current application. The TET hence becomes a measurement of the performance of the system and the amount of extra computational complexity that can be added before the system crashes.

2.4 xPC Target Limitations

Creating applications in the xPC Target environment has some limitations that needs to be taken into consideration during development. The following section will emphasize on the issues encountered using the xPC Target environment for application development.

The I/O blocks used during development must be compatible with the I/O cards in the physical hardware. The I/O library of the xPC Target is at the moment limited, but is increasingly becoming larger. 1000 Mbps ethernet cards were not supported until MATLAB 2012b. This was a problem as the Intel Target PC only had this type of ethernet card.

As Simulink is a graphical programming language problems occurs when handling variable sized data structures. Simulink handles this by zero-padding non-used elements.

This problem also occurs when populating empty structures, these has to be predefined in size in order for the system to understand what size is requested.

Logging and online visualization is limited due to scope limitations. When the incoming data consists of vehicle structures with fifteen elements, the maximum amount of signals quickly exceeds the signal limitation.

In order to have a compact system overview during application development, Simulink structures were used. This is an issue since these are nonobservable in the xPC Target environment, which imposed design workarounds for logging and data monitoring.

Chapter 3

System Design

This chapter focuses on describing the Scoop 2012 system from an architectural viewpoint, starting with a brief introduction of the Scoop 2011 system. Both hardware and software components, from an architectural and functional viewpoint will be presented. Presenting requirements and design decisions is subsequent to the Scoop 2011 system description. A system overview including hardware and software components is then presented.

3.1 Scoop 2011 architecture

The end result of the Scoop 2011 project was a vehicle platooning system named Cooperative Driving System (CDS). The main components of the CDS are the Wireless Sensor Unit (WSU) and the Electronic Control Unit (ECU). In Figure 3.1 a graphical representation of the CDS hardware and component architecture is presented.

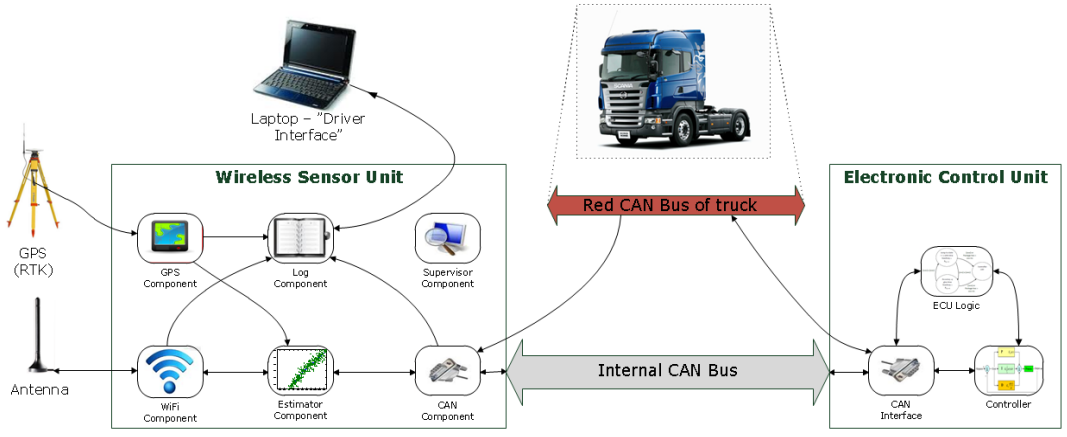


Figure 3.1: Architecture of the CDS. The WSU with components is presented to the left and the ECU with its components on the right-hand side. [9]

The WSU is a generic computer running a real-time software created in C++ using the Orocos-framework. The top-level functionality executed in the WSU are mainly used for information gathering, estimation, supervision and information broadcast. In addition to these function the WSU is also used for logging and real-time visualization.

The ECU is a standard Scania control unit. The main functionalities of the ECU are control strategies and system logic. The software in the ECU is created using auto generated real-time C-code from Simulink. Both the WSU and ECU reads important

sensory data from the vehicles Controller Area Network(CAN) bus and writes relevant actuation signals back to the CAN.

Now follows a short description of the top-level functionalities implemented in the CDS.

3.1.1 Information gathering

Using the GPS, CAN and WiFi components the system gathers information about the ego vehicle and the surrounding environment. The environment refers to other vehicles, road signs, traffic lights and platooning commands from a platoon leader. The information gathering functionality is also responsible for transmitting necessary raw sensor data to other components, so they can perform their tasks efficiently.

3.1.2 Estimation

The estimation function receives raw sensory data from the information gathering function. The estimation includes sensor fusion and estimation. The sensor fusion is necessary due to the fact that both the GPS and CAN provides velocity measurements which combined provides a more accurate measurement.

3.1.3 Control

The control function sends actuator signals to the braking system or cruise controller of the truck. Depending on the current traffic scenario different control strategies are used, these controller configuration decisions are taken by the control function. The control is also responsible for logical decisions regarding platoon composition, i.e. joining or leaving platoons.

3.1.4 Information broadcast

Vehicles participating in platooning scenarios must according to the GCDC broadcast certain information about the ego vehicle. When vehicles are performing maneuvers additional information indicating ongoing maneuvers must also be broadcasted. This is performed by the information broadcast functionality which is singularly responsible for all outgoing data from the CDS.

3.1.5 Supervisor

The supervisor function is responsible for system diagnostics and coordination of information flow in the system. It also performs management between different modes and settings of the system.

3.2 Scoop 2012 Architecture

External requirements posed on the Scoop 2012 system compared to the CDS was the use of a new real-time computer instead of the Scania ECU. The main reason was to have a standalone development platform for platooning projects at KTH. The new platform was also more agile in the sense that new I/O features could easily be added and modified. In addition to the external hardware requirements, new platooning scenarios were added which emphasized a demand for new software functionalities. Especially additional requirements were posed upon platooning logic to handle the new maneuvers.

Due to bandwidth and packet size limitation of the CAN bus in the CDS, some platooning logic were performed in the WSU. Separating functionality over multiple components in this manner, makes changes hard to implement and also prolongs the learning curve for new students. The Scoop project is intended to work as a continuous student project at KTH, a clear distinction of components and functionality is necessary to simplify future development of the Scoop system.

3.2.1 Design Decisions

A first design decision was to implement a new ethernet based connection between the WSU and xPC Target PC, were CAN communication was previously used. This was an important step to have a more agile communication channel between these computers. CAN imposes a packet size limitation that forced splitting of incoming packets before transmission. Using an ethernet based communication link offers a better alternative with larger packet size and bandwidth.

In CDS code generation from Simulink was used successfully for creating controllers and controller logic implemented in the ECU. Simulink is also a widespread tool in algorithm development and simulation in many universities around the world, which makes it a suitable platform for student projects. In addition Simulink also contains numerous libraries for signal processing, control theory and logical flowcharts. The combination of these facts were baseline arguments for also developing and executing estimation and platooning logic in the xPC Target. Moving the components made the old system architecture of the WSU problematic and major changes had to be made. Thus a new system design was presented, a component overview is presented in Figure 3.3. The xPC Target was designed for supporting numerous types of I/O interaction thus the GPS and CAN interaction components were moved to the xPC Target PC.

3.2.2 Hardware Setup

In Figure 3.2 the hardware setup of the Scoop 2012 system is depicted. The system consists of two real-time computers, the WSU and the xPC Target PC. Both computers are controlled by an external laptop. In the xPC environmental setup it is used as a Host PC for data visualization and parameter tuning during runtime. The laptop is also used to deploy the Scoop 2012 system, i.e. starting the software executed on the two platforms. In addition to this an external wireless router, called ALIX, is utilized for wireless transmission. In accordance with Figure 3.2 all hardware components mentioned are connected to a switch. In order to acquire appropriate sensor data the xPC Target PC is connected to external hardware. The GPS is connected to the Target PC via serial connection, the truck via CAN and to the WSU via ethernet.

3.2.3 Software Setup

The distribution of software component is depicted in Figure 3.3. The WSU operates as wireless data distributor, incoming data is sent to the xPC Target PC and outgoing data is transmitted wirelessly to other vehicles. The WSU is still running a real-time software using the Orocos-framework, as was the case in the CDS.

The xPC Target PC contains all platooning functionalities such as controllers, estimators and logic. The Target PC also receives data from WSU, truck and GPS. All

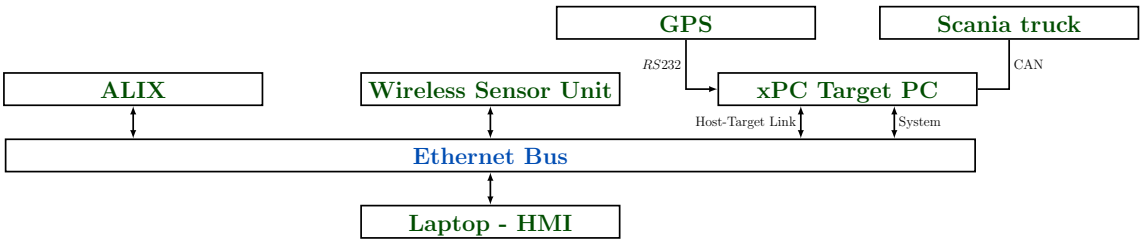


Figure 3.2: Hardware setup of the Scoop 2012 system. An ethernet switch is the main connection point of the system. Both WSU, xPC Target PC and ALIX are directly connected to the switch. Connections to the truck and GPS are utilized using CAN and RS232 respectively.

components in the xPC Target PC are designed and tested in Simulink.

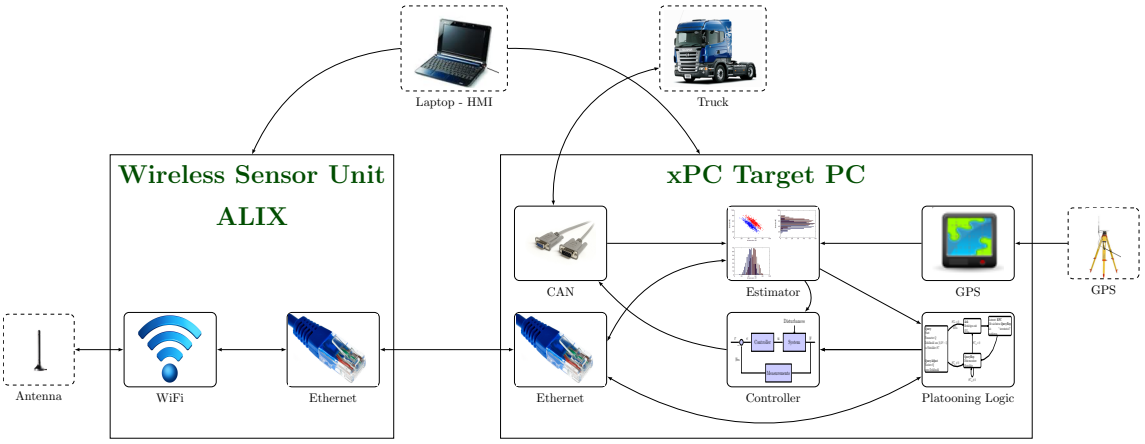


Figure 3.3: Component overview of the Scoop 2012 system. The WSU handles WiFI communication and data forwarding. Platooning components are deployed in the xPC Target PC.

3.3 Scoop 2012 System Components

In Figure 3.4 an overview of the components in the xPC Target PC is presented. The blue boxes represents external communication via I/O ports, yellow estimator blocks, green logical operations and red controllers. Using the component overview in Figure 3.4 all components of the Scoop 2012 system will be described briefly. Certain attention will be targeted towards description of the inputs and outputs of all components, which are highlighted in Figure 3.4 using capital letters *A*-*J*.
On the left-hand side the external communication input blocks are presented. These are the GPS, CAN and Ethernet modules. The information provided by the GPS component, denoted with *A* in Figure 3.4, is presented in Figure 3.5.

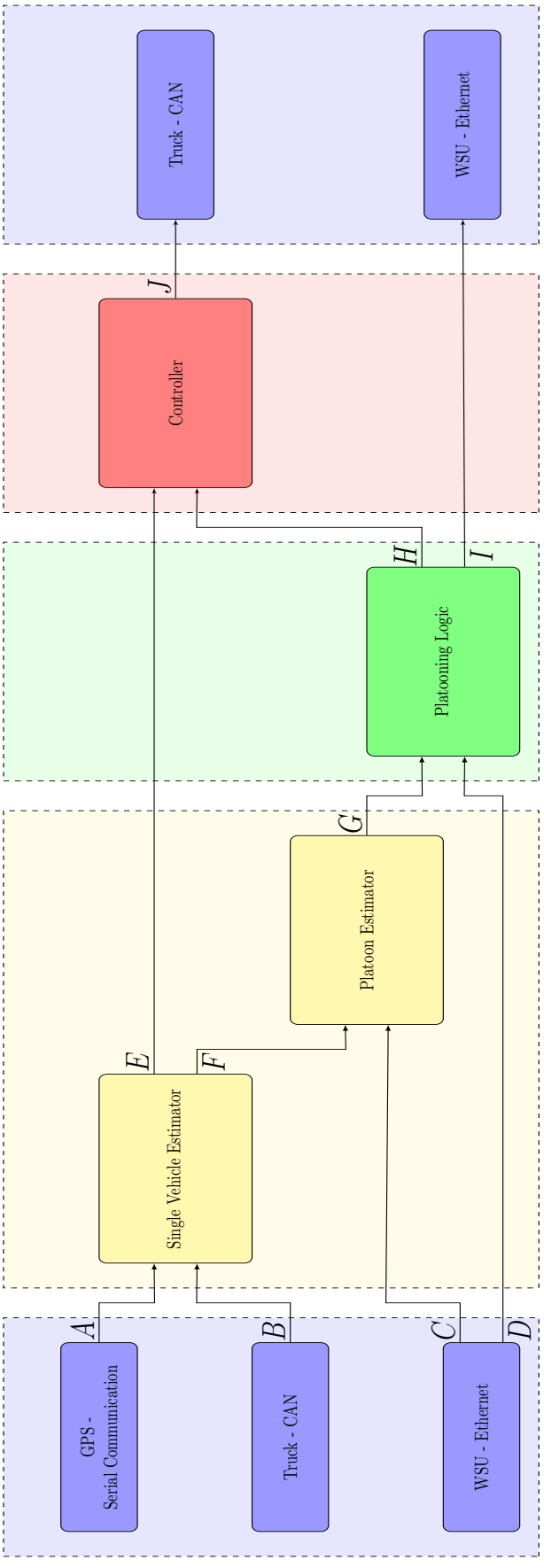


Figure 3.4: Scoop 2012 system xPC Target component overview. The coloring scheme is separating communication, estimation, logic and controller components. The capital letters *A* to *J* are used to highlight the information data flow between Scoop 2012 system components.

GPS Component Data Flow

Message	Write to	Repetion rate(ms)
GPS_Status	Estimator	10
GPS_Sensor	Estimator	10

GPS_Status

Signal	Type	Unit	Range	Description
Status	double		–	65 = data valid or 86 = data not valid

GPS_Sensor

Signal	Type	Unit	Range	Description
Latitude	double	degrees	-90 - 90	Longitude position
Longitude	double	degrees	-180 - 180	Latitude position
Speed	double	m/s	–	Vehicle speed
Heading	double	degrees	0 - 360	Relative to true north

Figure 3.5: An overview of the output signals from the GPS component.

The information from the CAN and Ethernet modules are presented in Figure 3.6 and Figure 3.7 respectively.

CAN Component Data Flow

Message	Write to	Repetion rate(ms)
CAN_Sensor	Single Vehicle Estimator	10
CAN_Radar	Platoon Estimator	100
CAN_PedalPosition	System Logic	100
CAN_SystemSwitches	System Logic	100

CAN_Sensor

Signal	Type	Unit	Range	Description
Speed_data	double	km/h	-	Vehicle speed
Speed_time	double	ms	-	UTC milliseconds since 1970-01-01 00:00
Acceleration_data	double	m/s^2	-	Vehicle acceleration
Acceleration_time	double	ms	-	UTC milliseconds since 1970-01-01 00:00
YawRate_data	double	-	-	Vehicle yaw rate
YawRate_time	double	ms	-	UTC milliseconds since 1970-01-01 00:00

CAN_Radar

Signal	Type	Unit	Range	Description
DistanceVehicle_1	double	m	-	Distance to vehicle ahead
DistanceVehicle_2	double	m	-	Distance to second vehicle ahead
DistanceVehicle_3	double	m	-	Distance to vehicle ahead in left lane
DistanceVehicle_4	double	m	-	Distance to vehicle ahead in right lane
RelativeSpeed_1	double	m/s	-	Relative velocity to vehicle ahead
RelativeSpeed_2	double	m/s	-	Relative velocity to second vehicle ahead
RelativeSpeed_3	double	m/s	-	Relative velocity to vehicle ahead in left lane
RelativeSpeed_4	double	m/s	-	Relative velocity to vehicle ahead in right lane

CAN_PedalPosition

Signal	Type	Unit	Range	Description
AcceleratorPedal	double	%	0-100	
BrakePedal	double	%	0-100	

CAN_SystemSwitches

Signal	Type	Unit	Range	Description
System_On	double	-	0-1	
System_Off	double	-	0-1	
Silent_On/Off	double	-	0-1	0=System silent, 1=System active

Figure 3.6: An overview of the output signals from the CAN component.

Ethernet Component Data Flow

Message	Write to	Repetion rate(ms)
DynamicalVehicleInformation	Platoon Estimator	0.833
StaticVehicleInformation	Platoon Estimator	0.833
StaticRSUInformation	Platoon Estimator	0.833
TrafficLightInformation	Platoon Estimator	0.833
SpeedSignInformation	Platoon Estimator	0.833
ManeuverRequest	Platooning Logic	100

DynamicalVehicleInformation

Signal	Type	Unit	Range	Description
NodeID_lsb	uint32	-	-	
NodeID_hsb	int32	-	-	
VehiclePositionLon	uint32	deg	-90 - 90	
VehiclePositionLat	uint32	deg	-180 - 180	
VehiclePositionTim-estamp_sec	uint32	s	-	
VehiclePositionTim-estamp_msec	uint32	ms	0 - 999	
VehiclePositionAccuracy	double	m	-327.68 - 327.67	
VehicleVelocity	double	m/s	-327.68 - 327.67	
VehicleAcceleration	double	m/sˆ2	-20 - 20	
VehicleHeading	double	deg	0 - 360	
VehicleYawRate	double	deg/s	-327.68 - 327.67	
PlatoonLeaderID_lsb	uint32	-	-	
PlatoonLeaderID_hsb	uint32	-	-	
PlatoonState	int32	0 -	0 - 1	0 - Stable, 1 - Transition

StaticVehicleInformation

Signal	Type	Unit	Range	Description
NodeID_lsb	uint32	-	-	
NodeID_hsb	int32	-	-	
VehicleWidth	double	m	0 - 10.23	
VehicleLength	double	m	0 - 163.83	

StaticRSUInformation

Signal	Type	Unit	Range	Description
NodeID_lsb	uint32	-	-	
NodeID_hsb	int32	-	-	
RSUPositionLon	double	deg	-90 - 90	
RSUPositionLat	double	deg	-180 - 180	

TrafficLightInformation

Signal	Type	Unit	Range	Description
NodeID_lsb	uint32	-	-	
NodeID_hsb	int32	-	-	
TrafficLightColor1	int32	-	0 - 2	0 - Green, 1 - Yellow, 2 - Red
TrafficLightTime1_sec	uint32	s	-	
TrafficLightTime1_msec	int32	ms	0 - 999	
TrafficLightColor2	int32	-	0 - 2	0 - Green, 1 - Yellow, 2 - Red
TrafficLightTime2_sec	uint32	s	-	
TrafficLightTime2_msec	int32	ms	0 - 999	
TrafficLightColor3	int32	-	0 - 2	0 - Green, 1 - Yellow, 2 - Red
TrafficLightTime3_sec	uint32	s	-	
TrafficLightTime3_msec	int32	ms	0 - 999	
TrafficLightColor4	int32	-	0 - 2	0 - Green, 1 - Yellow, 2 - Red
TrafficLightTime4_sec	uint32	s	-	
TrafficLightTime4_msec	int32	ms	0 - 999	

SpeedSignInformation

Signal	Type	Unit	Range	Description
NodeID_lsb	uint32	-	-	
NodeID_hsb	int32	-	-	
SpeedLocation1Lon	double	deg	-90 - 90	
SpeedLocation1Lat	double	deg	-180 - 180	
MaximumSpeed1	double	m/s	-327.68 - 327.67	
Heading1	double	deg	0 - 360	
SpeedLocation2Lon	double	deg	-90 - 90	
SpeedLocation2Lat	double	deg	-180 - 180	
MaximumSpeed2	double	m/s	-327.68 - 327.67	
Heading2	double	deg	0 - 360	
SpeedLocation3Lon	double	deg	-90 - 90	
SpeedLocation3Lat	double	deg	-180 - 180	
MaximumSpeed3	double	m/s	-327.68 - 327.67	
Heading3	double	deg	0 - 360	

ManeuverRequest

Signal	Type	Unit	Range	Description
ManeuverID_sec	uint32	s	-	
ManeuverID_msec	int32	ms	0 - 999	
DestinationVehicleID_lsb	uint32	-	-	
DestinationVehicleID_hsb	uint32	-	-	
ReferenceVehicleID_lsb	uint32	-	-	
ReferenceVehicleID_hsb	uint32	-	-	

Figure 3.7: An overview of the output signals from the Ethernet component.

The output information from the estimator components in Figure 3.4 are presented in Figure 3.8 and Figure 3.9.

Single Vehicle Estimator Component Data Flow

Message	Write to	Repetion rate(ms)
Estimator_Initialized	System Logic	10
Estimator_VehicleInformation	Platoon Estimator	10
	Ethernet Sender	10
	Controller	10

Estimator_VehicleInformation

Signal	Type	Unit	Range	Description
Latitude	double	deg	-90 - 90	
Longitude	double	deg	-180 - 180	
Velocity	double	m/s	-	
Acceleration	double	m/s^2	-	
Heading	double	deg	0 - 360	Relative to true north
YawRate	double	rad/s	-	
PositionAccuracy	double	m	-	
Timestamp_sec	uint	s	-	
Timestamp_msec	uint	ms	0 - 999	

Estimator_Initialized

Signal	Type	Unit	Range	Description
Initialized	double	-	0-1	0 - Non active, 1 - Active

Figure 3.8: An overview of the output signals from the Single Vehicle Estimator component.

The output from the Platooning Logic component in Figure 3.4 are denoted with H and I. This data is presented in Figure 3.9.

Platoon Estimator Component Data Flow

Message	Write to	Repetion rate(ms)
PlatEstimator_PlatoonVector	Platooning Logic	10

PlatEstimator_PlatoonVector

Signal	Type	Unit	Range	Description
RelativeDistance	double	m	-	Relative distance to all vehicles in platoon
Velocity	double	m/s	-	Velocity of all vehicles in platoon
Acceleration	double	m/s^2	-	Acceleration of all vehicles in platoon
LaneInformation	double	-	0-2	0 - left lane, 1 - same lane, 2 - right lane

Figure 3.9: An overview of the output signals from the Platoon Estimator component.

Platooning Logic Component Data Flow

Message	Write to	Repetion rate(ms)
ControllerFeed	Controller	10
SenderMode	WSU - Ethernet	10
ManeuverState	WSU - Ethernet	10
ManeuverOffsetActive	WSU - Ethernet	10

ControllerFeed

Signal	Type	Unit	Range	Description
DistanceReference	double	m	-	Reference distances to vehicles ahead
RelativeDistance	double	m	-	Relative distances to vehicles in platoon
Velocity	double	m/s	-	
Acceleration	double	m/s^2	-	
LaneInformation	double	-	0-2	0 - left lane, 1 - same lane, 2 - right lane
NodeID	uint32	-	-	

SenderMode

Signal	Type	Unit	Range	Description
Mode	double	-	1-5	Choose WSU sender mode

ManeuverState

Signal	Type	Unit	Range	Description
ManeuverID_sec	uint32	s	-	
ManeuverID_msec	int32	ms	0 - 999	
ManeuverState	int32	-	-	

ManeuverOffsetActive

Signal	Type	Unit	Range	Description
ManeuverID_sec	uint32	s	-	
ManeuverID_msec	int32	ms	0 - 999	
DestinationVehicleID_lsb	uint32	-	-	
DestinationVehicleID_hsb	uint32	-	-	
ReferenceVehicleID_lsb	uint32	-	-	
ReferenceVehicleID_hsb	uint32	-	-	
LongitudinalOffset	int32	m	-	
LaneOffset	int32	0-2	-	0 - left lane, 1 - same lane, 2 - right lane

Figure 3.10: An overview of the output signals from the Platooning Logic component.

The data represented with a *J* in Figure 3.4 is completely presented in Figure 3.11.

Controller Component Data Flow

Message	Write to	Repetion rate(ms)
Controller_References	Truck - CAN	16.67

Controller_References

Signal	Type	Unit	Range	Description
UBrake	double	m/sˆ2	-	Vehicle reference retardation
USpeed	double	km/h	-	Vehicle reference velocity

Figure 3.11: An overview of the output signals from the Controller component.

Detailed descriptions of the components in Figure 3.4 are presented in Section 3.3. The component description will start from the left-hand side with I/O inputs and end at the right most with I/O outputs.

3.3.1 Wireless Communication

The wireless interaction protocol, [6], implemented in the GCDC is also utilized in CoACT 2012, but with additional messages to handle new maneuvering messages. Thus the WiFi component developed for CDS remains, only slightly modified. Description of the workflow and functionality of the WiFi component can be found in [4].

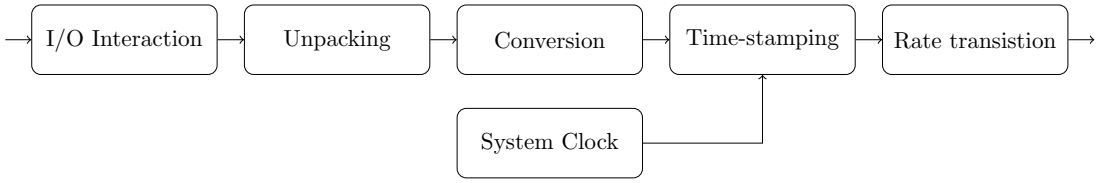
Communication

All components in the Scoop 2012 system that receives data from external sources via I/O ports are structured in a similar manner. The systematic approach is to create a consistent pattern for new users when interacting with the system Simulink model. The workflow process of the CAN and GPS communication components are depicted in Figure 3.12a, the Ethernet component workflow is presented in Figure 3.12b.

1. The first stage is I/O interaction, where the xPC Target Simulink library provides the user with the appropriate blocks for I/O card interaction. The blocks are configured according to the operating scenario, depending on communication component different sampling times are chosen to coordinate with the sensory information rate. The Simulink environment requires statically sized connection between blocks. Messages are hence zero-padded in order to achieve static output size from the receiver, even when incoming messages differ in size.

2. The unpacking block reads the protocol header of received messages, then the payload is extracted according to predetermined protocol patterns. The incoming messages are of different length and the zero padding is removed during unpacking.

3. The conversion block performs both unit conversion and data type conversion. Unit conversion is necessary since sensors provide measurements of physical quantities in different units. Data type conversion is also performed since all inputs from the WiFi are integer types and the system needs to handle floating point variables.



GPS and CAN communication component system structure.

Ethernet communication component system structure.

Figure 3.12: Overview of the workflow structure of the Scoop 2012 I/O communication components. The CAN and GPS components add timestamps to measurements in order for the estimator to work appropriately. The buffering block is situated in the Platooning Estimator and is used to handle a dynamically changing amount of vehicles in the platoon.

4a. Adding timestamps to sensor measurements is essential for single vehicle estimation. Every time instance the current time of the system clock is appended to the sensor measurements. This in order to enable state estimation, increase the weight on recent samples and to discard old sensor values when performing vehicle estimation.

4b. In order to create dynamical storage of several vehicles, traffic lights, etc. circular buffers are used to temporarily store incoming data. More detailed information on functionality of the buffers is presented in Chapter 4.

5 The root frequency of the Scoop 2012 system is 100Hz but communication components must operate faster. The reason is that for instance, the WiFi component is receiving multiple vehicle messages at 10 Hz. Thus the working frequency must be sufficiently larger than the receiving frequency in order to process all incoming data, else the receiving buffers will overflow. The rate-transition blocks are therefore added to down-sample the information output from the buffers and sensors to achieve a common frequency, that is sufficiently low for the rest of the system.

3.3.2 GPS Component

The xPC Target PC communicates with the Trimble GPS using an RS232 serial connection. The Trimble GPS is configured to transmit messages using the NMEA-0183 standard. The main functionality of the GPS component is to support the system with positional coordinates in longitude, latitude pairs. But the GPS also provides additional measurements, a list of measurements is presented in Figure 3.5.

Time-Synchronization

In addition to providing positional information, the Trimble GPS also provides accurate time measurements. These measurement are used to perform precise time synchronization. This is crucial in a platooning system, since in order to handle estimation and sensor fusion, synchronized data streams are of great importance. From a platooning overview perspective, synchronized vehicles improve the ability to compensate for wireless data losses in platooning estimators.

The time-synchronization is part of the system startup process and is performed before the system can be activated. During runtime the system continuously reads the system clock comparing it to the UTC time provided by the Trimble. This operation is performed in order to counteract system clock drift. If the drift exceeds 150 milliseconds the system clock is re-calibrated by the time-synchronization component.

3.3.3 CAN Receiver Component

The industrial standard for internal vehicle communications is Controller Area Network bus (CAN) [7]. The CAN is hence used in all Scania trucks to provide communication between all control units and sensors within the vehicle. In order to allow for communication with the Scania truck from the Scoop 2012 system a CAN gateway was created. The gateway is important from safety perspective since it only allows for specific messages to enter the CAN bus of the truck. Otherwise messages could disrupt or interfere with other important functionalities in the truck.

In order to properly decode the CAN gateway messages a CAN database file (`.cdb`) has been developed. The specifications and message contents are presented in the Appendix. The following sensory information used in the Scoop 2012 system is provided by the gateway:

CAN Receiver output
Velocity
Longitudinal acceleration
Yaw rate
Radar information
Pedal positions
System switches

Table 3.1: CAN Receiver component output.

- The velocity, acceleration and yaw rate measurements are forwarded to the vehicle estimator.
- The radar information includes measurements of relative position and speed. These are forwarded to the platooning estimator.
- The pedal positions and system switches are driver actuators. This enables the driver to start, stop and change modes of the system during autonomous drive. These are important features for the system to ensure safety and allows the driver to manually control the vehicle at any time. The switches are implemented to turn on and off the system, but also to configure WiFi broadcasting settings.

3.3.4 Ethernet Component Receiver

The Ethernet component receives data via ethernet from the WSU using the UDP protocol. The information received in the WSU via wireless transmission contains all information needed to handle the CoACT scenarios.

The Ethernet component creates a server and a client working in the xPC Target according to a client-server architecture [8]. A client running in the WSU transmits messages via UDP to the receiver client in the xPC Target. The ethernet unit is configured

with an IP-address, a subnet mask and a port number to have an appropriate method for addressing over the local network.

In Simulink the network receivers are not able to handle dynamically changing packet sizes, so a maximum packet size is specified. Shorter packages are then zero-padded in order to fill the remaining bytes. The message size was chosen to be equal to the maximum packet sent via the UDP protocol, which was 90 bytes. The Ethernet component then works according to the workflow depicted in Figure 3.12b. The unpacking block first reads incoming messages and unpacks them appropriately and also removes the zero-padding which was added in the I/O module. The resulting output form the Ethernet component is ten different platooning messages that is forwarded to the system. These include; static and dynamic vehicle information, maneuvering related messages, road side unit data including static and dynamic information and other messages. Messages handling vehicles and road side units needs to be buffered in order to handle multiple instances of similar objects. The buffering of incoming data is handled in the Platooning Estimator.

During CoACT, data from road side units was not considered but for future work this feature was added. The workflow for road side unit unpacking and buffering is similar to that of platooning vehicles.

3.3.5 Vehicle Estimator Component

The Vehicle estimator component utilizes information acquired via CAN and GPS in order to provide internal Scoop 2012 system components with more accurate measurements of velocity, acceleration, position, heading and yaw rate. The information from the estimator is also broadcasted via WiFi to provide platoon members with dynamical vehicle information.

An estimator is needed to handle imperfections of sensors, typically noise, drift and bias. The Vehicle estimator also provides sensor fusion capabilities. Since several sensors measure the same physical quantity a combined measurement provides a more accurate estimate.

The filter consists of two different parts an initialization module and one Kalman filter module. In order to initialize the system properly and provide the Kalman filter with proper initial conditions the Vehicle estimator has a short startup period. The initialization procedure sets an origin position and also provides the filter with initial measurements and covariance. The initialization module therefore provides an output indicating when the filter is initialized. This boolean is used in the System Logic preventing autonomous driving without the filter working properly.

The Kalman filter module implements the extended Kalman filter algorithm. In order to handle sensor measurements the filter includes a buffer to handle messages arriving in non-sequential order. Old messages not fitting the buffer timeframe are discarded. A combination of different sensor inputs is used to compute estimated states of system variables. The outputs from the filter consists of an eight-tuple of dynamical vehicle data. The elements of the tuple is included in Table 3.4. The PositionTimestamp element is not provided by the filter. Instead the time is generated by the system clock at each update. More information on the implementation and execution cycle of the Vehicle estimator component is described in [11].

Kalman filter output
Longitude position
Latitude position
Velocity
Acceleration
Heading
PositionAccuracy
YawRate
PositionTimestamp

Table 3.2: Vehicle estimator component output.

3.3.6 Platoon Estimator Component

The Platoon Estimator is responsible for buffering, sorting and estimation of vehicle and road side unit data received from the Ethernet and Vehicle Estimator component.

The buffering is the initial phase of the Platoon Estimator, which represents the fourth step shown in Figure 3.12b. Data transmitted via WiFi is buffered in a fixed sized FIFO buffer in order for the system to handle a dynamic amount of vehicles and road side units. In addition to temporarily storing incoming data, the buffers also combine incoming static and dynamic data into a common vehicle structure. The same operation is performed on traffic light and speed sign data. This is necessary for performing platoon estimation, for example calculating relative distances. More information about he buffers will be presented in Chapter 4. The outputs from the buffers are unsorted lists of vehicles and road side units.

The next phase of the Platoon Estimator is a sorting algorithm, where incoming data is the unsorted lists mentioned above combined with ego vehicle information from the Vehicle Estimator. Three different sorting blocks are being executed simultaneously the sorting of traffic lights, speed signs and platooning vehicles. Depending on sorting block, different outputs are calculated and forwarded to the Platooning Logic component. The traffic light sort calculates, from the incoming buffer, which traffic light is closest ahead of the ego vehicle. In most scenarios only the closest traffic light ahead can impose changes to the platooning behavior.

As opposed to the traffic lights, speed sign sorting instead generates two outputs: the maximum allowed speed of the current road segment and speed sign information form the closest sign ahead. The current maximum allowed speed is necessary to stop the truck from exceeding the given speed limit. The speed limit ahead, is taken into consideration when approaching the sign with a faster velocity than allowed. In this scenario the truck is forced to decelerate to achieve the allowed velocity before passing the sign.

The third sorting algorithm calculates the distance to all vehicles currently populating the vehicle buffer. The algorithm starts by calculating the absolute distance from the ego vehicle to all vehicles using an arc approximation to compensate for road curvatures.

The final stage is to use the radar measurements. If the radar detects a vehicle, the detected distance is compared to the distance provided by WiFi data. If the distances is within a threshold, the most conservative distance is appended to the vehicle. If the radar distance cannot be mapped to an existing vehicle, this is considered to be a vehicle which is not part of the current platoon. But in order not to collide with the unknown vehicle, this measurement is appended to the output vector from the vehicle sort. The output generated by the vehicle sorting is a sorted vector of up to ten vehicles with the following

five tuple: The vehicles are sorted in descending relative distance, i.e., the vehicle with the

Platooning Estimator output
Relative Distance
Velocity
Acceleration
Lane Information
Node ID

Table 3.3: Platooning Estimator output

longest relative distance is the first element in the vector. The velocity and acceleration are appended for control purposes. The lane information is currently not being used, but lane detection is a necessary element in future platooning projects, hence this was added. The Node ID is added for logic purposes during platooning maneuvers.

3.3.7 Platooning Logic Component

The Platooning Logic component is responsible for numerous tasks, including: handling maneuvers initiated within the platoon, providing the Controller component with appropriate references, populating and broadcasting the appropriate platooning maneuver messages.

From the Platoon Estimator, a sorted list of vehicles currently populating the vehicle buffer is forwarded to the Platooning Logic. The Platooning Logic takes the sorted incoming vector of vehicles and appends reference distances to all of them. During active maneuvers the Platooning Logic reads active maneuver messages and is responsible for intelligently appending the appropriate distance references if gaps are created, vehicles change lanes or vehicles join, etc.

In addition to feeding the appropriate distance references to the controller, the Platooning Logic is responsible for controlling the broadcasting of messages from the outgoing ethernet component. Nominally, the Scoop 2012 system is transmitting static and dynamic vehicle information at 1 Hz and 10 Hz, respectively. But while the ego vehicle is performing maneuvers, additional maneuvering messages are broadcasted.

3.3.8 Controller Component

The Controller component is responsible for calculating appropriate velocity and retardation references to the internal control system of the truck. The controller is fed information from the Vehicle Estimator and Platooning logic. The Vehicle Estimator provides the controller with filtered acceleration and velocity measurements. The input from the Platooning Logic is a sorted list of other vehicles in the platoon; this list will be referred to as the *platoon vector*.

The *platoon vector* contains information from up to ten different vehicles. The vehicles are sorted in descending order, i.e. the platoon leader has the highest index in the list. The contents of the *platoon vector* for each vehicle are: distance reference, actual distance, velocity, acceleration, lane information and nodeID. The lane information is important, since the system is designed not to control on vehicles performing maneuvers in the leftmost lane.

The Controller component has two ways of interacting and controlling the velocity of

the vehicle. In order to increase or to withhold a velocity, speed references are sent to the vehicle cruise controller. Speed decrease can be achieved through engine braking or by applying mechanical brakes. Since retardation using engine braking is limited, controller logic has been implemented in order to make decisions wether to apply mechanical braking or not. The logical components of the controller is depicted in Figure 3.13.

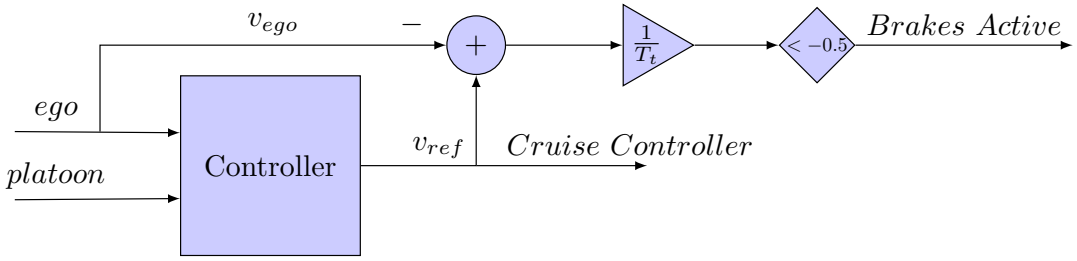


Figure 3.13: Flow chart representation of the Scoop 2012 system controller logic.

Another feature for the controller is the ability to choose between different controller modes depending on the current platooning scenario. A list of different control strategies are presented in Table 3.4. The Idle mode is active at system startup and can be entered at any time by entering *idle* state in the System logic. Further information on the System Logic is presented in Chapter 4. A vehicle operating as platoon leader cannot utilize sensors to follow vehicles ahead. Instead platoon leaders are commanded to drive at the maximum allowed speed limit of the road. This is referred to as the Leader mode.

Abbreviation	**Explanation**
Idle	Manual driving.
Leader	Follow current speed limit of the road segment.
PI - *two vehicles*	Only controlling on the vehicle ahead using a PI controller.
PI - *multiple vehicles*	Controlling on up to three vehicles ahead using a PI controller.
LQ - *two vehicles*	Only controlling on the vehicle ahead using an LQ controller.
LQ - *multiple vehicles*	Controlling on all vehicles ahead using an LQ controller.

Table 3.4: Controller configurations of the Scoop 2012 system.

The last four modes presented in Table 3.4 are used for follower vehicles. The control approaches implemented in the Scoop 2012 system were PI and LQ. The reason for implementing numerous controllers of PI and LQ type was to evaluate eventual performance differences between the controllers.

3.3.9 CAN Sender Component

The CAN sender periodically transmits an actuator message from the Scoop 2012 system to the gateway via a CAN connection. The actuator commands used for controlling the vehicle are velocity references sent via the Engine Management System to the cruise controller and retardation references to the Electronic Braking System. The gateway is for safety reasons implemented such that if a request to the cruise controller and the braking system is sent simultaneously, the brake system overrides the cruise controller.

In addition to sending references to the truck, the CAN sender transmits indications to a dashboard, that notifies the driver which mode the Scoop 2012 system currently is operating in.

Chapter 4

System Component Implementation

In order to successfully perform the maneuvers present in the CoACT 2012 challenge, several new system components had to be developed. The single vehicle estimator, platooning logic and controller strategies were designed and implemented by other members of the Scoop 2012 team. Functionality and further descriptions can be found in [11], [12] and [13]. In addition to the components listed above, ethernet communication and system logic components were necessary to accomplish the CoACT objectives. Describing the implementation and functionality of the these components will be within the scope of this chapter.

The chapter starts with a description of the new ethernet communication link and network protocols, followed by a description of the network buffers. The latter parts of the chapter will describe in detail the functionality of the system logic component.

4.1 Ethernet Communication Protocol

Internal I/O communication between the WSU and the xPC Target PC is utilized using ethernet based communication. The new communication link was implemented using the standard User Datagram Protocol (UDP) at the transport layer level. The reason for implementing the UDP protocol was that this offered the best support from xPC Target. On the application layer level a protocol based on the CoACT 2012 WiFi Interaction Protocol was used. The CoACT 2012 Interaction Protocol was similar to the communication protocol used in GCDC, but including new messages to handle lateral platooning maneuvers.

New ethernet components were created both in the WSU and in the xPC Target PC. The focus of this section will be on describing the implementation of the internal interaction protocol as details of the workflow of the ethernet components are described thoroughly in Chapter 3.

As stated in the introduction of this section a new application layer protocol was created. The internal protocol packet content consists of header and payload data. The header consists of seven fields, each containing data used for identification and diagnostics. The packet structure is presented in Figure 4.1 with appended explanations described in Table 4.1.

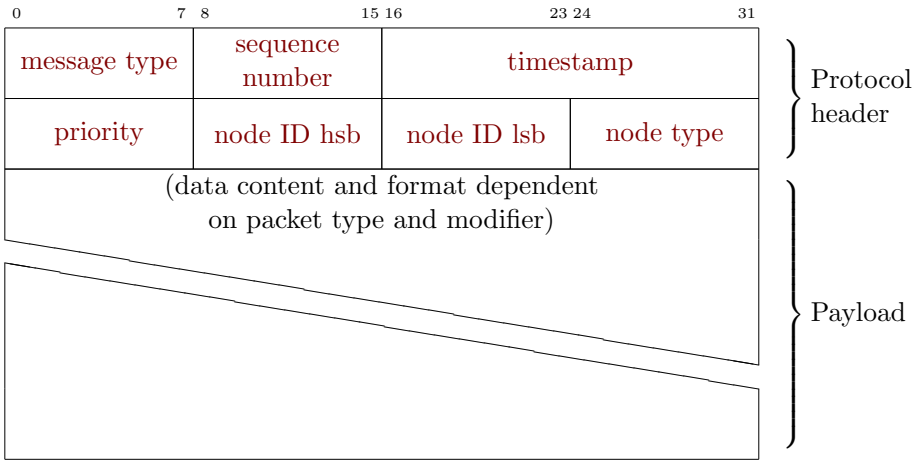


Figure 4.1: The internal communication protocol header.

Message Field	Explanation
message type	Message identifier used for separating different types of messages.
sequence number	Sequence numbers are added in order to implement detection of missed messages.
timestamp	The time instance at which the current message was sent. Important to neglect old messages as information from these are not of interest.
priority	Priority is not currently in use, but is intended to highlight priority to important messages.
node ID lsb/hsb	Each vehicle and road side unit are identified by a unique 64-bit node ID. Simulink cannot handle 64-bit integers hence the node ID is split into two 32-bit integers, high and low bits.
node type	Node types are added to distinguish vehicles from road side units.

Table 4.1: Description of the fields in the Scoop 2012 ethernet application layer protocol header.

The header is used in the xPC Target PC in order to read and extract the incoming messages in the appropriate manner. All messages are statically sized, the information of the lengths of all messages are hence known in advance. The *message type* field is used to separate, decode and encode wireless messages. The node ID fields are also important since all vehicles and RSUs are each associated with a unique node ID. This field is hence used to differentiate and associate the incoming data with the correct unit in the system buffers and platooning logic.

4.1.1 Circular Buffers

The final output from the communication block presented in the system overview in Figure 3.3 are different messages from vehicles, traffic lights, speed signs or platooning maneuvers. In order to provide the system with the ability to handle incoming data from several vehicles and road side units simultaneously, an internal component dynamically storing data is needed. In Scoop 2012 this is solved by using three circular First-In-First-Out (FIFO) buffers. But in addition to store incoming data during runtime the buffers also combine incoming data packets. The information broadcasted from every single vehicle

within the platoon is of two types: static and dynamic. The static information provided is length and width. The dynamic information contains information from a variety of sensors. Examples of dynamic vehicle data are velocity, heading, acceleration and platoon leader ID etc. But in order to perform platooning operations, dynamic and static vehicle information must be combined into a common vehicle data structure. The similar holds for traffic light and speed sign information.

A circular buffer is a data structure that uses a fixed size buffer that works as if it were end-to-end connected. Every time an element is added to the buffer the buffer pointer is incremented and once it reaches the end it starts from the beginning. If the pointer points to a position in the buffer already containing information this is overwritten once new information is received.

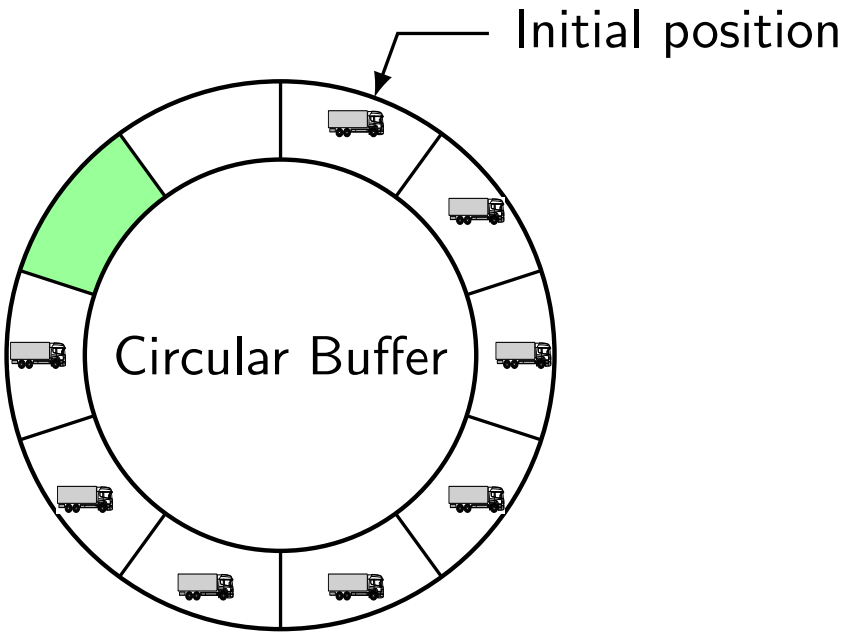


Figure 4.2: Example of a circular vehicle buffer populated by eight vehicles. The current position of buffer pointer is indicated by the green field.

In the Scoop 2012 system the circular buffers were used as storage for all vehicles and road side units transmitting appropriate wireless information. This enabled the Scoop 2012 system to handle information from multiple vehicles and road side units. The algorithm implemented in the circular buffer will now be described thoroughly. The description will emphasize on describing the algorithm of the vehicle buffer but the road side unit buffers works in a similar manner.

At system startup the buffer is initialized as a 10 element circular buffer containing empty vehicle structures. The vehicle structure is a combination of dynamic and static vehicle information. When the system is running, the buffers have two inputs: static and dynamic vehicle data. The buffer algorithm will now be described.

1. Search the buffer for an existing vehicle with the same Node ID as the incoming

dynamic vehicle structure.

(a) If Node ID exists in the circular buffer, point to that buffer element and update dynamic vehicle information.

(b) If Node ID does not exist, clear the element the buffer pointer is currently pointing to; this in order to prevent non-coherent static and dynamic data to be written to the same element. Then populate the element with dynamic vehicle data before incrementing the buffer pointer.

2. Search the buffer for an existing vehicle with the same Node ID as the incoming static vehicle structure.

(a) If Node ID exists in the circular buffer, point to that buffer element and update static vehicle information.

(b) If Node ID does not exist, clear the element the buffer pointer is currently pointing to. This in order to prevent non-coherent static and dynamic data to be written to the same element. Then populate the element with static vehicle data before incrementing the buffer pointer.

3. Write buffer content to block outputs.

4.2 System Logic

At system startup, a sequence of components must be initialized before the automatic control system can operate the vehicle. A component handling these startup conditions is created in Simulink using Stateflow. A flowchart of the system logic is depicted in Figure 4.3. Besides controlling the autonomous operation of the vehicle, the system logic also handles the transmission state of wireless messages while the system is active. This feature is added in order to be able to perform system testing without transmitting wireless data to surrounding platooning vehicles. In Table 4.2 short explanations are added to describe the transition requirements between states in Figure 4.3.

Transition	Explanation
A	Estimator initialized, time synchronization performed.
B	Wireless-switch active, Estimator initialized, On-switch active, GPS-status OK.
C	Brake pedal active, Gas pedal active, Off-switch active.
D	Wireless-switch active
E	Wireless-switch non-active
F	Wireless-switch non-active, Estimator initialized, On-switch active, GPS-status OK.
G	Brake pedal active, Gas pedal active, Off-switch active.
H	Wireless-switch active
I	Wireless-switch non-active

Table 4.2: Description of the state transitions in the system logic.

The Scania ECU is removed and replaced by a more general real-time computer, so a new CAN gateway unit for external interaction with the truck was developed by Scania. In addition to communicating with the truck, the gateway is also connected to a panel

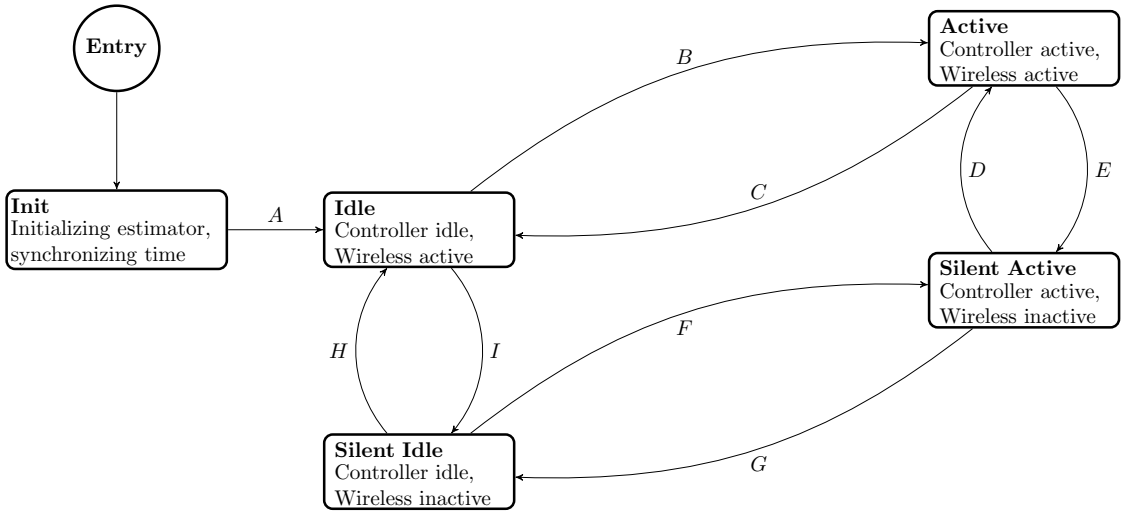


Figure 4.3: The startup and safety logic for the Scoop 2012 system. Each state is represented by a box with a system state description. States are also added in order to handle the state of the wireless transmission.

used for online driver interfacing. The panel is equipped with five lamps used as indicators and four switches used for sending commands via the CAN gateway. All states depicted in Figure 4.3 are associated with a particular combination of lamps. The truck panel is shown in Figure 4.4.

Figure 4.4: Panel mounted in the truck to allow driver interaction with the Scoop 2012 system. The lamps are used to indicate current mode of the system logic. Lamp coding, Off: red, Idle: yellow, Active: green. The blue lamp is used as wireless transmission indicator.

Chapter 5

Testing of the Framework

In order to evaluate the Scoop 2012 system simulations were performed at KTH using a simulation environment developed by M. Almroth, for more information see [14]. The main objective of the CoACT 2012 project was to perform successful platooning including lateral maneuvering in Stora Holm, Gothenburg in November 2012. The results of simulations and experiments are presented in this chapter, starting with simulation experiments.

5.1 Simulation experiments

The Scoop 2012 system was for testing purposes implemented to a large extent in a simulation and software development program called PreScan [15].During the following simulations a four vehicle platoon was deployed in PreScan all utilizing the same control system and logic components. The first experiment is ordinary platooning starting from standstill. The second experiment contains the new platooning maneuvers performed in CoACT 2012. All the maneuvers needed to perform the switch scenario were tested in the simulation environment.

5.1.1 Intelligent platooning

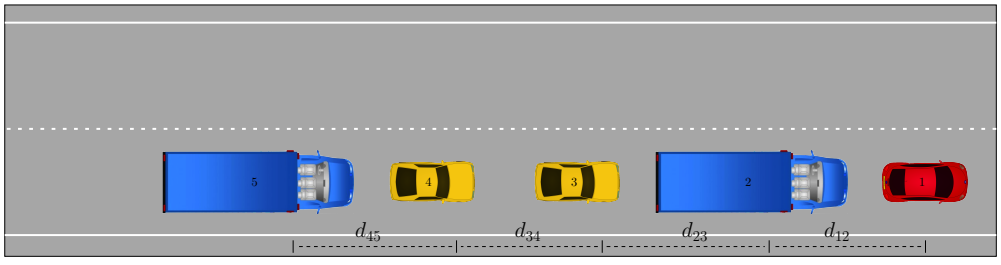


Figure 5.1: A graphical presentation of the vehicles during an intelligent platooning scenario. The intermediate distances are denoted d_{ij} were i and j denote vehicle numbers.

A graphical presentation of the vehicle setup during intelligent platooning is presented in Figure 5.1. All vehicles starts from standstill at an inter-vehicular distance of 23 meters. In Figure 5.2 the speed and relative distance profile of the platoon is plotted during a plain platooning scenario. Vehicle 1 starts to accelerate thus the increase in relative distance in relation to Vehicle 2 occurs during the first seconds of the simulation. The same phenomenon occurs later when Vehicle 4 starts to accelerate but at this time the relative distance does not increase as much as in the first case. The reason is that the

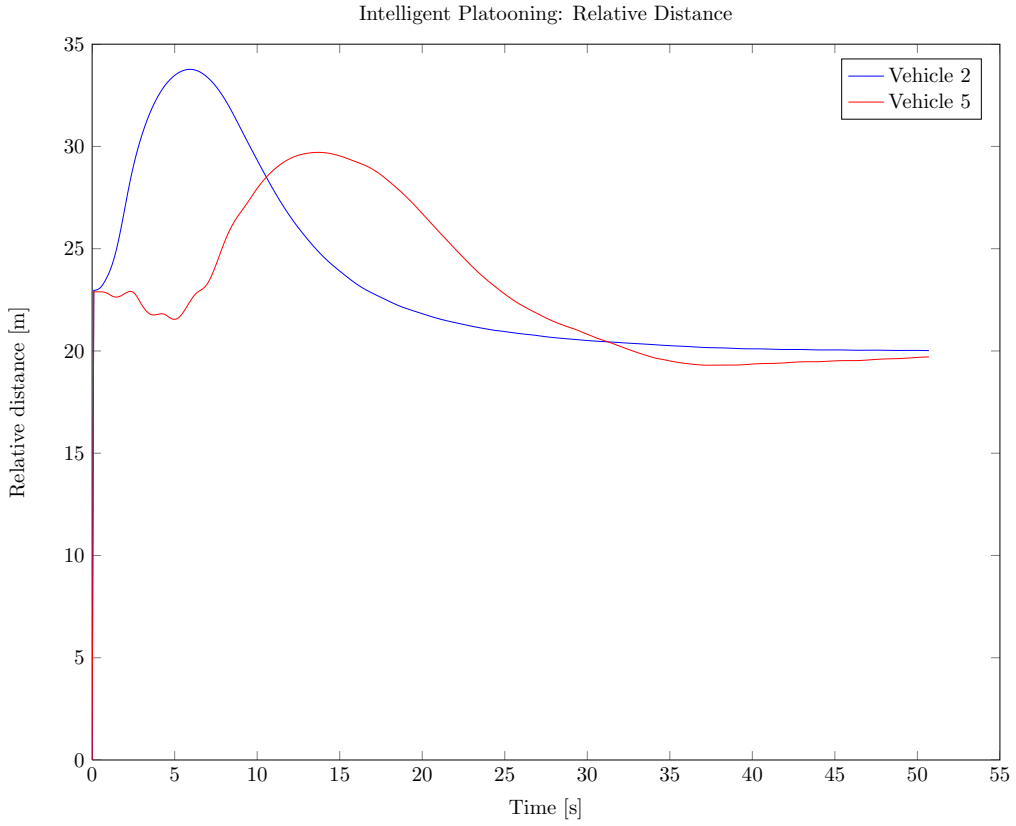
utilized controller is using the velocity of several vehicles ahead to calculate appropriate velocity references. In Figure 5.2b the velocity profiles of the vehicles are depicted. Here its obvious that Vehicle 5 faster achieves a higher maximum velocity and thus reduces the relative distance faster.

5.1.2 Platooning with overtaking

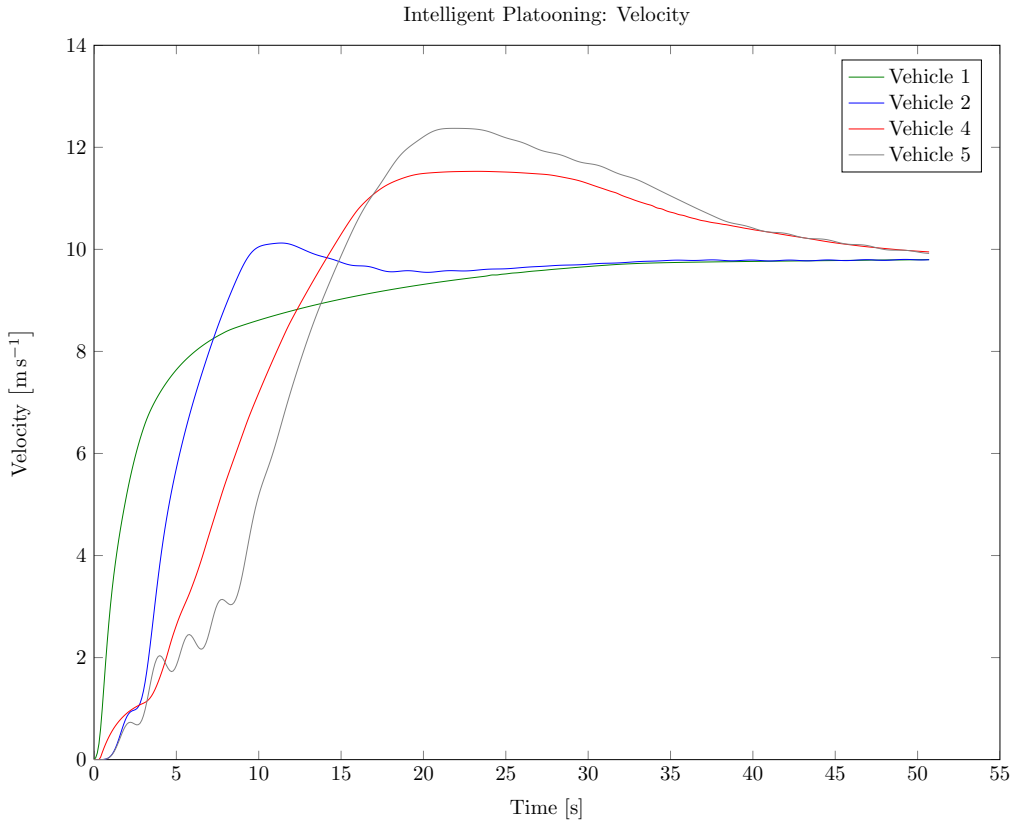
The vehicle behavior during an overtaking maneuver is depicted in Figure 5.3. In Figure 5.4 graphics illustrating different parts of the platooning scenario are depicted. Initially all vehicles are standing still with an intermediate distance of 20 meters. The simulation is initialized and the trucks start to move. When reaching stable platooning, at the first black vertical line, an increase gap request is sent commanding Vehicle 2 to increase the distance to Vehicle 1 to 45 meters. The vehicle starts to decelerate and the intermediate distance to the platoon leader increases up to 48 meters.

When MRQ 2 is broadcasted vehicle 3 is commanded to change lane and the reference vehicle is the platoon leader instead, thus the jump in relative distance after 52 seconds. Vehicle 3 now wants to achieve a relative distance to the platoon leader equal to 20 meters. Vehicle 3 therefore accelerates and after approximately 80 seconds of simulation time Vehicle 3 is in the left most lane 20 meters behind the platoon leader. During this period Vehicle 2 has kept a constant distance to the platoon leader.

MRQ 3 commands Vehicle 3 to return to the right lane, but in front of Vehicle 2. Vehicle 3 which while driving in the left lane was not taken into consideration, is now added to the control vector. This is indicated by a jump in relative distance for Vehicle 2 after 80 seconds. The switch maneuver is now completed and the platoon returns to ordinary intelligent platooning.

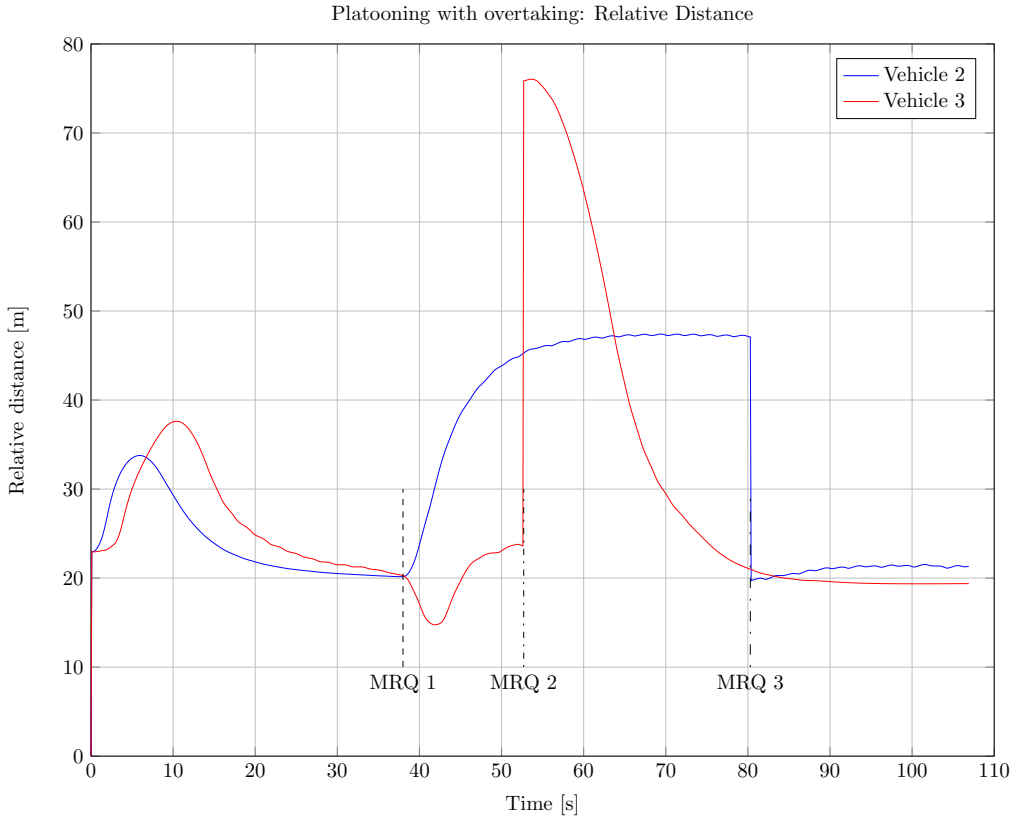


Relative distance data from Vehicle 2 and Vehicle 5 during intelligent platooning.

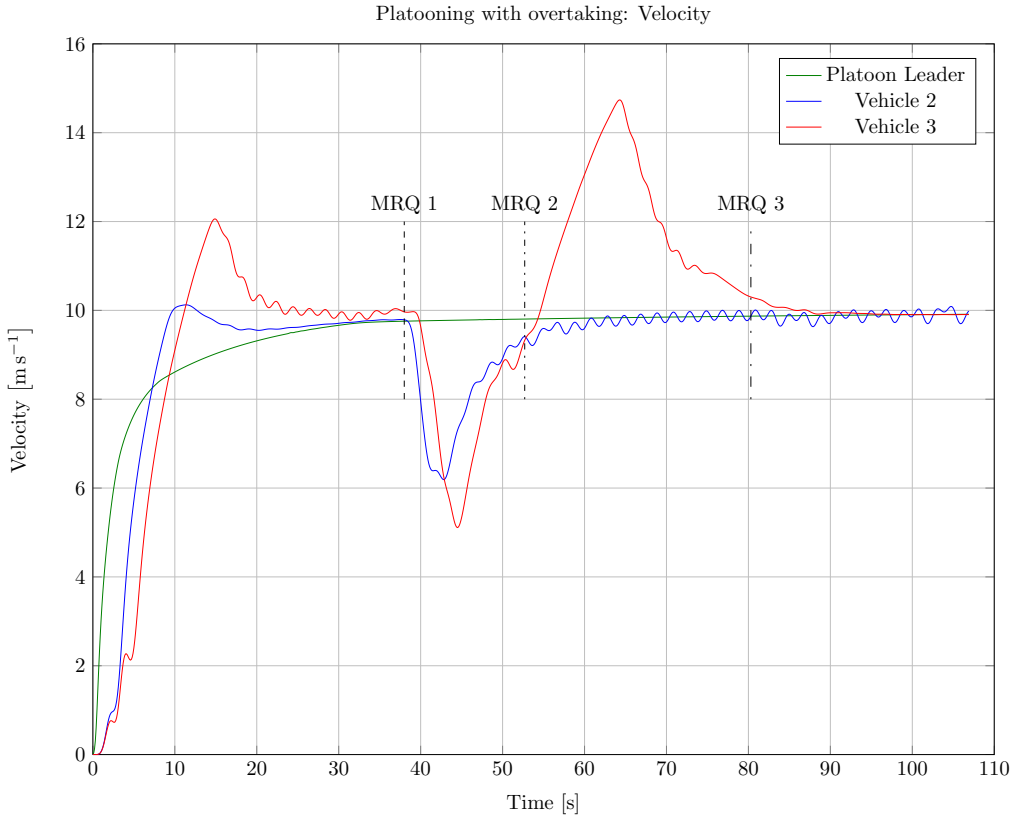


Velocity data from Platoon leader, Vehicle 2, Vehicle 4 and Vehicle 5 during intelligent platooning.

Figure 5.2: Relative distance and velocity data from 4 vehicles during the intelligent platooning.

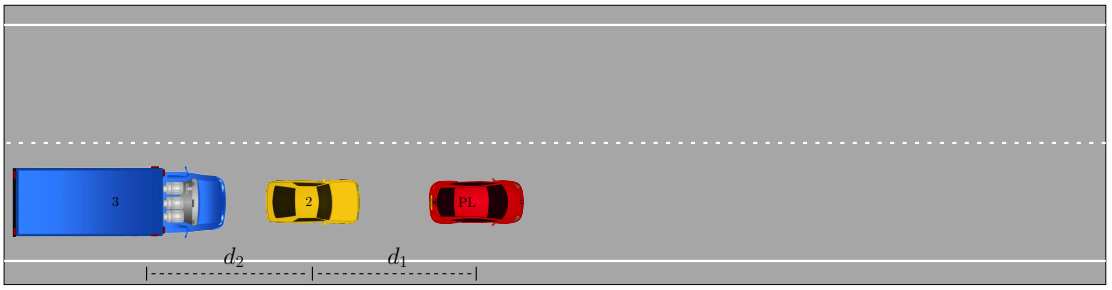


Relative distance data from Vehicle 2 and Vehicle 3 during a switch maneuver.

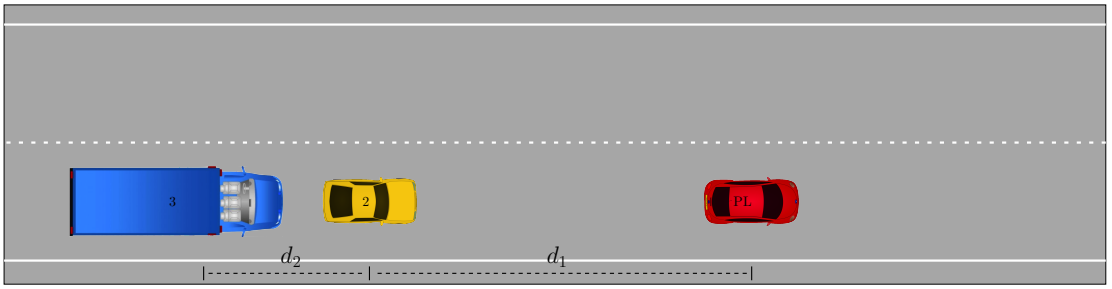


Velocity data from Platoon leader, Vehicle 2 and Vehicle 3 during a switch maneuver.

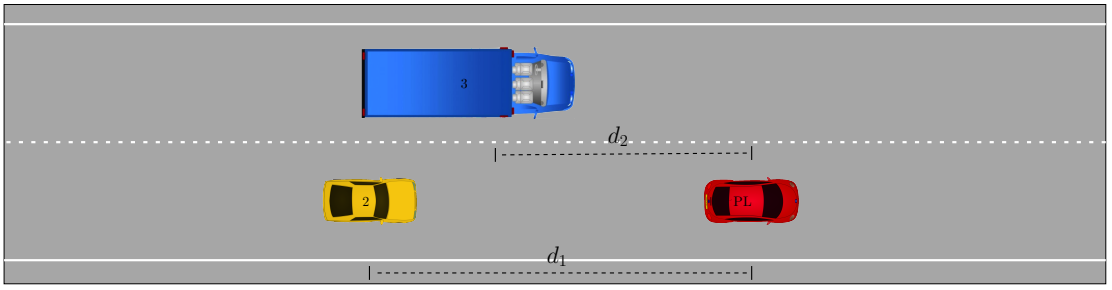
Figure 5.3: Relative distance and velocity data from 3 vehicles during the switch maneuver. The vertical black line indicate the time instance at which maneuvering requests were broadcasted to the platoon.



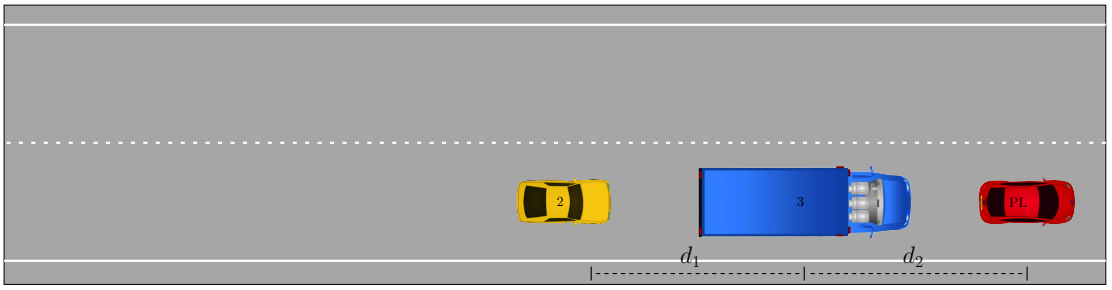
Initial platooning sequence, before MRQ 1 is sent.



After 35 seconds Vehicle 2 responds to MRQ 1 by increasing distance to PL.



MRQ 2 tells Vehicle 3 to change lane and achieve an offset distance of 20 meters to PL.



MRQ 3 has been successfully completed and all vehicles are traveling with an intermediate distance of 20 meters.

Figure 5.4: The switch maneuver during simulations using PreScan. d_1 is the relative distance from Vehicle 2 to the vehicle ahead. The same holds d_2 when it comes to Vehicle 3.

5.2 Stora Holm experiments

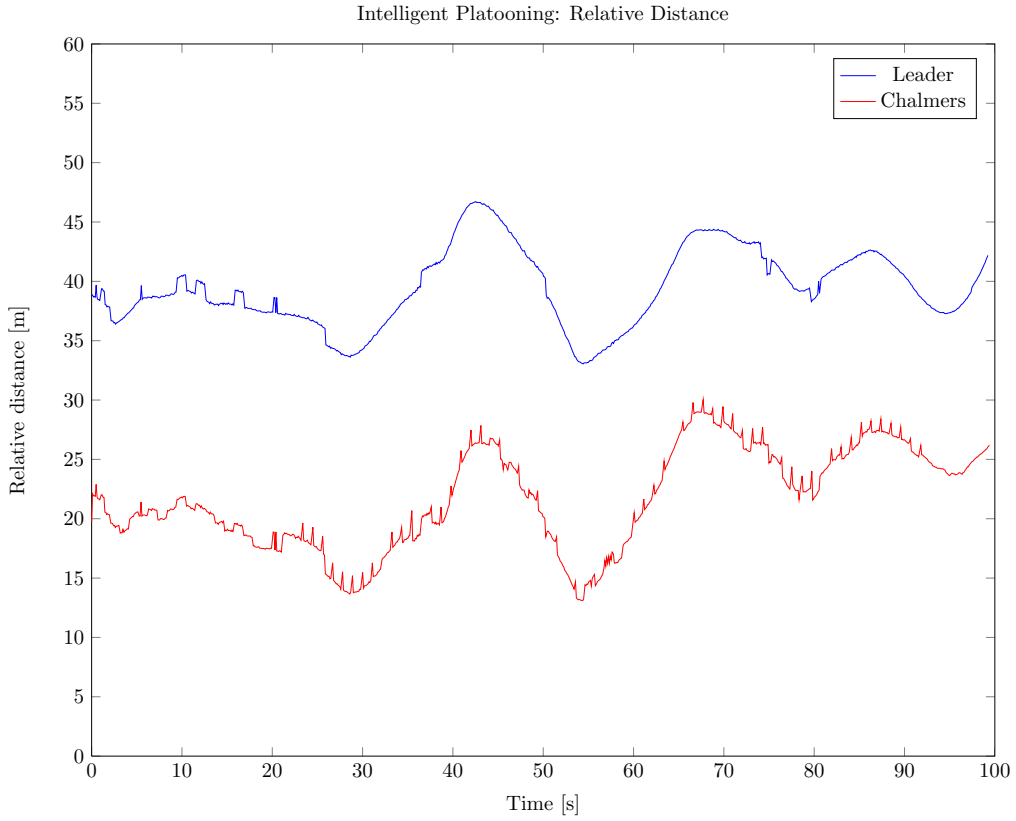
In cooperation with Chalmers and Linköping the final CoACT event was hosted at Stora Holm test track outside of Gothenburg in November 2012. Data is collected from two experiments, the first including ordinary platooning with the KTH vehicle starting in third position with Platoon leader and Chalmers Volvo car in front. The second experiment contains a switch maneuver were the KTH vehicle starts behind the Platoon leader and in front of the Chalmers vehicle.

In Figure 5.5 is velocity and relative distance data depicted from three vehicles while performing intelligent vehicle platooning. In Figure 5.5a the relative distance to the Platoon leader and Chalmers car is depicted. The spikes are results from downsampling a noise signal, hence this behavior. The reason why the reference distance is oscillating, is that in order to have a smoother behavior the distance reference included not only a constant distance but also a constant times the current velocity.

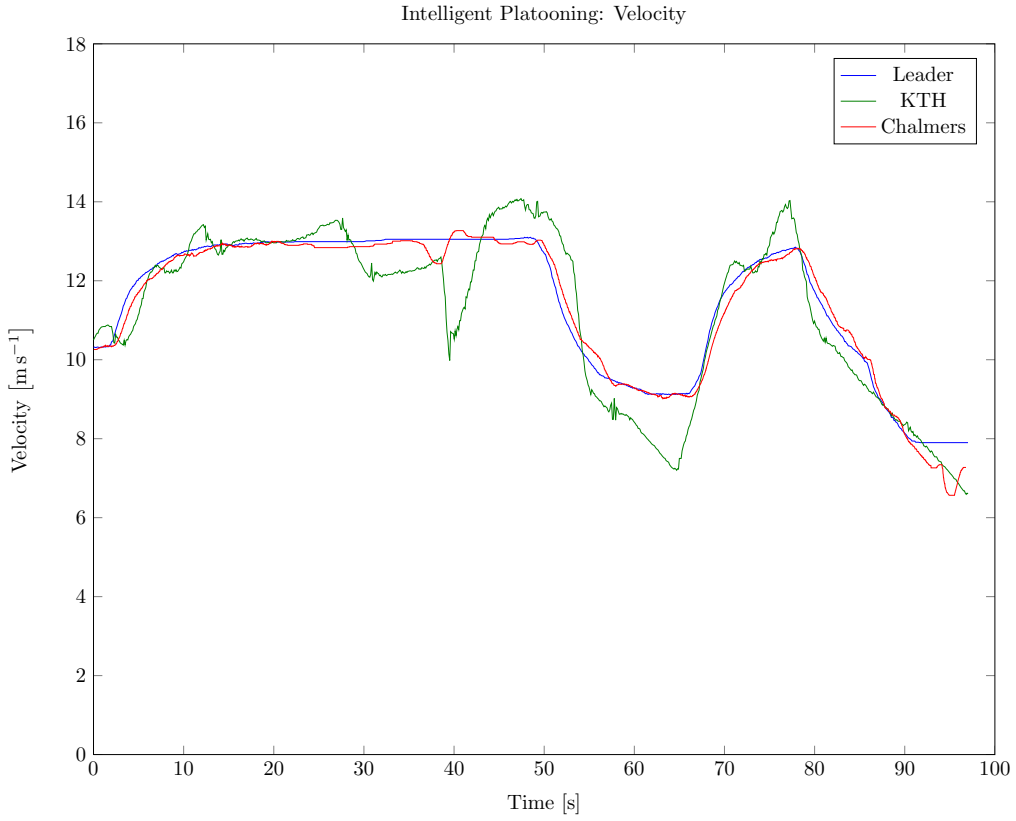
In Figure 5.6 data from the switch maneuver is depicted. As opposed to the simulation data four requests are sent instead of three. The reason is that in the simulation MRQ 1 and MRQ 2 in the experiment was combined into one single maneuver. The experiment stars with the vehicles standing still with an intermediate distance of 8-15 meters. The Platoon leader starts to accelerate and when all vehicles are moving, MRQ 1 tells the Chalmers car to change lane with the KTH vehicle at a reference distance of 15 meters.

Once the Chalmers car is in position a new request is sent setting the Platoon leader as reference vehicle with a distance of 15 meters. The reference distance of the Chalmers vehicle increases instantly to 45 meters and the car starts to accelerate to compensate for the distance offset. During this period the KTH truck still keeps a relative distance of 30 meters to the Platoon leader. Chalmers is ready after 38 seconds and MRQ 3 is sent, requesting KTH to open a gap of 50 meters. At this time KTH is far to close and starts to decelerate, in Figure 5.6b the velocity decreases rapidly down to 3 m/s.

After approximately 50 seconds the gap between the Platoon leader and KTH vehicle is large enough for the Chalmers vehicle to enter. MRQ 4 is sent after 64 seconds telling the Chalmers car to join the platoon again in front of KTH. At this instance the Chalmers vehicle becomes a part of the feedback vector to the controller and therefore a jump in relative distance occurs when the MRQ 4 is broadcasted. The Chalmers vehicle starts to change lane and after a few seconds the switch maneuver is completed.

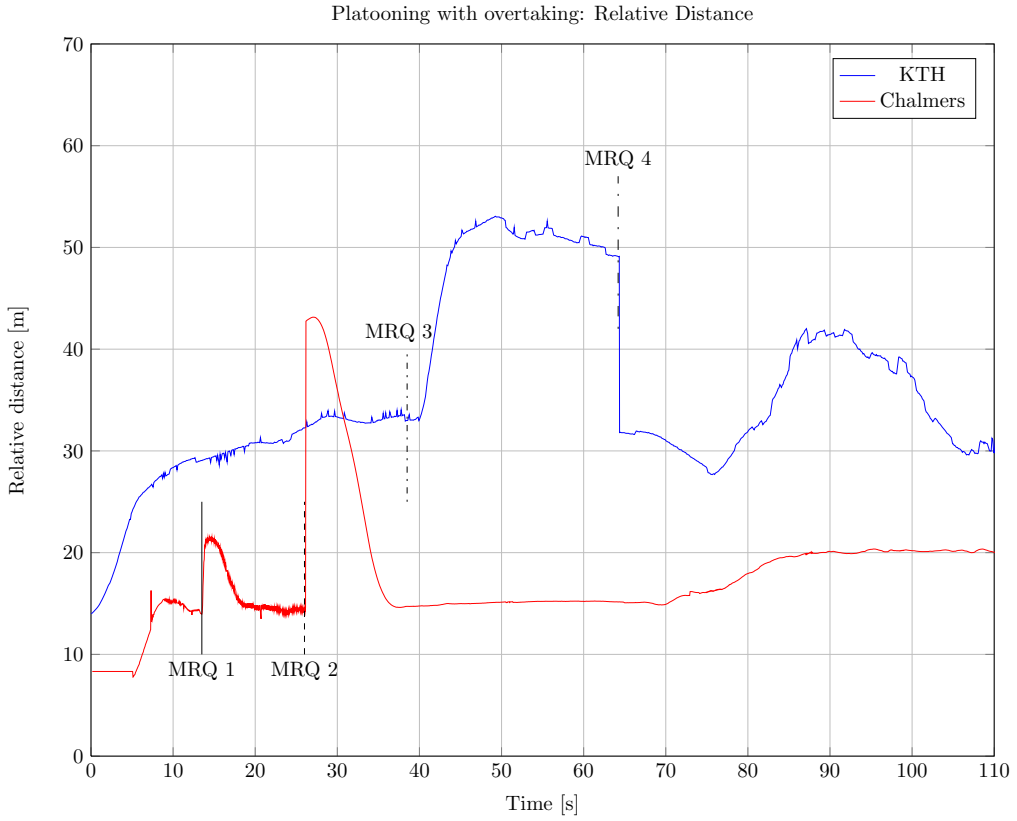


Relative distance data from KTH vehicle to Chalmers vehicle and Platoon leader.

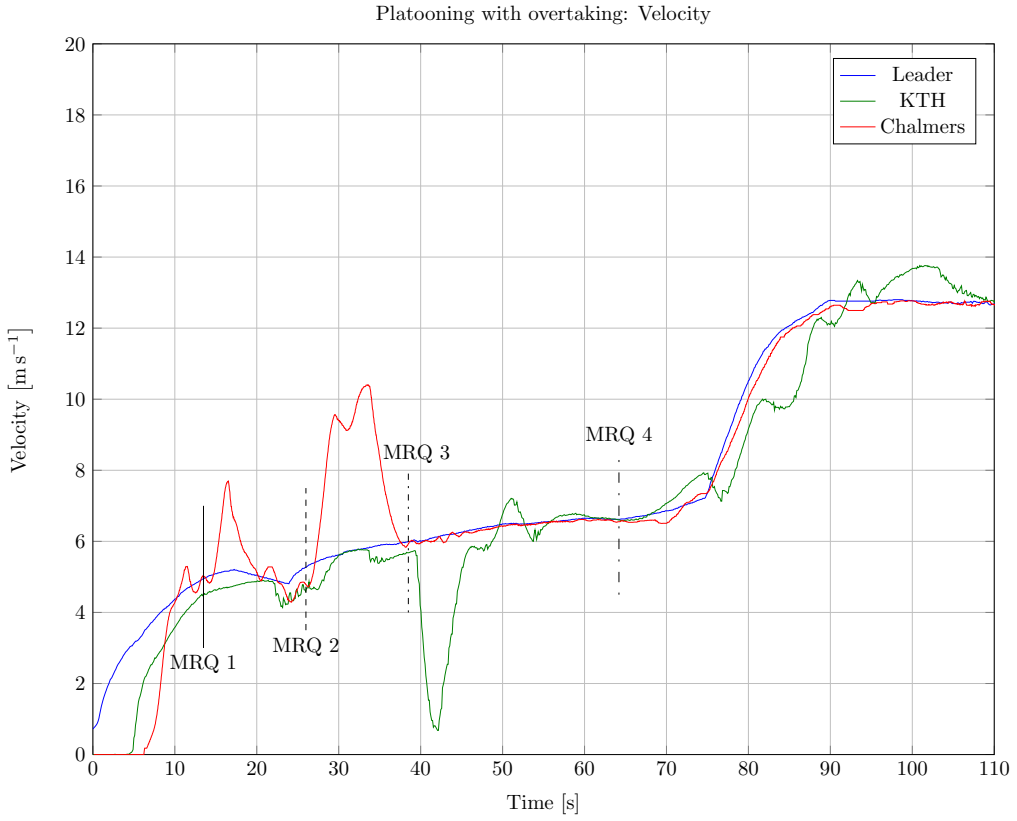


Velocity data from Platoon leader, KTH and Chalmers vehicle during intelligent platooning.

Figure 5.5: Relative distance and velocity data from 3 vehicles during the intelligent platooning in the final event of CoACT 2012.

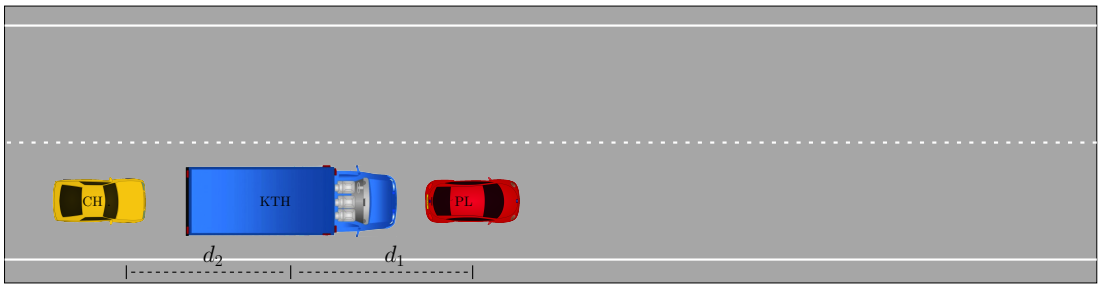


Relative distance data from KTH and Chalmers vehicles during a switch maneuver.

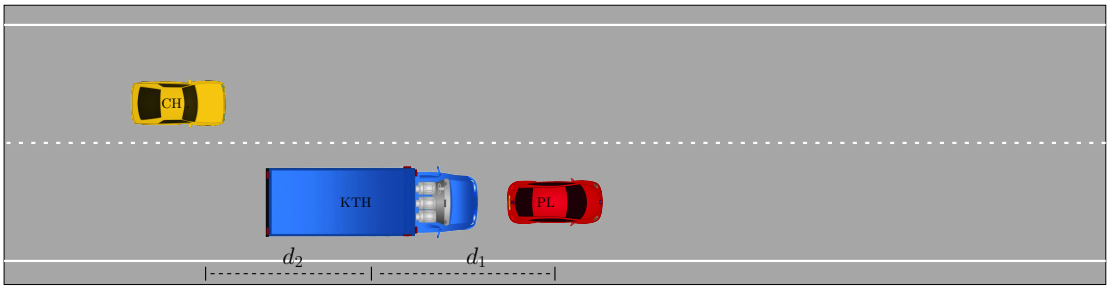


Velocity data from Platoon leader, KTH and Chalmers vehicle during a switch maneuver.

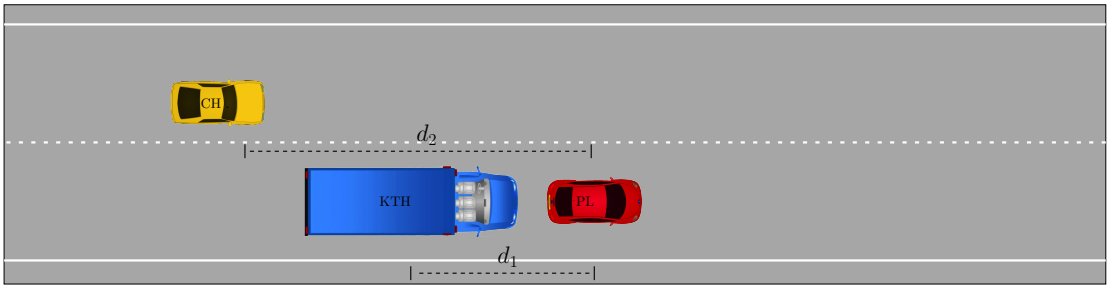
Figure 5.6: Relative distance and velocity data from 3 vehicles during the switch maneuver. The vertical black line indicate the time instance at which maneuvering requests were broadcasted to the platoon. Data was collected at the final event of CoACT 2012.



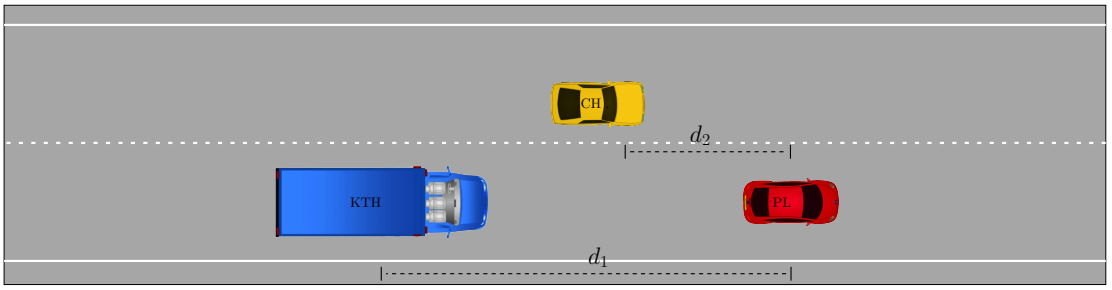
All vehicles starts and standstill.



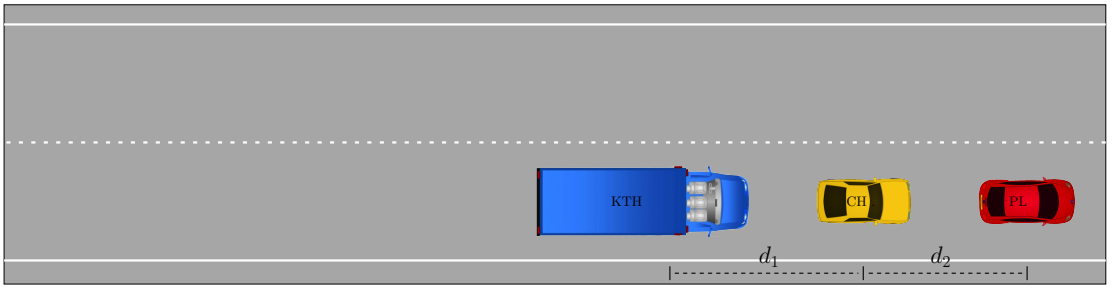
PL sends MRQ 1 after 15 seconds telling CH to change lane but to keep reference distance d_1 constant.



After 25 seconds MRQ 2 is sent requesting CH to change reference distance d_2 to PL instead of KTH.



MRQ 3 is sent after 38 seconds commanding KTH truck to increase inter vehicular distance to PL to 50 meters.



The final command, MRQ 4 is sent after 65 seconds. CH now returns to the initial lane keeping PL as reference vehicle. KTH changes reference vehicle to CH, thus completing the swap maneuver.

Figure 5.7: The switch maneuver during Stora Holm experiments.

Chapter 6

Conclusions and Further Work

6.1 Conclusions

Implementing a platooning system using xPC Target has both advantages and disadvantages compared to using a generic real-time linux system. The use of MATLAB in the academic sector is widespread and all students at technical universities are accustomed to the MATLAB computing language. This reduces the learning curve for new students when introducing control algorithms, estimation and logic components in a platooning system. The development procedure for specific platooning components is also shorter and more efficient than using a hand-coding approach.

However developing a framework in xPC Target has proven to be tedious and sometime inefficient. The software does not support standard protocols for parsing, instead manual protocols and unpacking procedures has been handwritten in S-functions in MATLAB.

Problems has also occurred when I/O cards that are not supported by the xPC Target environment are installed in the provided hardware. But the I/O library is growing with each MATLAB release and now covers most conventional I/O cards.

During application development in the Scoop 2012 project, several problems with the xPC Target software has been experienced. The software just recently in MATLAB R2012b started to support 64-bit operations which was used in this project. Software crashes during application development has occurred frequently, which imposes that consistency issues are still problematic. The software crashes are part of the software and will not be handled until new MATLAB releases.

A limitation in the area of signal monitoring is the number of xPC Target Scope allowed in a single application. The maximum number of allowed xPC Target Scopes are ten *TARGET*, ten *HOST* and eight *FILE* scopes. But as each scope reduces the computational power the number of allowed scopes is smaller. This can prove to be a problem in complex real-time applications when computational power is of essence. A tradeoff between signal logging, necessary for application development, and runtime performance is certainly not a requested behavior in any real-time application.

The main difference in implementation between Scoop 2011 and Scoop 2012 systems is the use of the WSU. The WSU in CDS is the main unit performing all I/O interaction and is also the central processing unit for estimation and logging. These functionalities are in the Scoop 2012 system located in the xPC Target PC. Moving Platooning functionality has proven to be successful as the support for estimation in Simulink is extensive. This makes development of such components easy using Simulink. The major problem of Scoop 2012 compared to CDS has been signal logging and displaying. The provided HMI,

xPC Explorer, works great when building applications with a small number of signals and parameters. However when the amount of signals grows orienting in the HMI becomes tedious and hard. This makes the signal logging and monitoring capabilities limited which is problematic when performing live experiments. Developing a customized HMI has therefore been mandatory but using the GUIDE software provided by MATLAB has also been problematic when dealing with huge amounts of data streamed via ethernet. The HMI becomes slow and the use of an external program controlling the monitoring and logging would be suggested for the future.

If I were to make changes to the design decisions made during the project, I would not have chosen to move the GPS and all CAN communication to the xPC Target PC. Since logging and I/O data extraction have proven to be hard to achieve using xPC Target. The decision would instead be to move all CAN interaction to the WSU and keep the GPS, logging and supervisor components. Then create one ethernet communication link transmitting not only WiFi data but also CAN and GPS data in the WSU. Then create two identical ethernet components in the xPC Target PC using different interaction protocols. Thus utilizing the benefits of the hand coded structure from CDS, with I/O libraries, existing logging tool and supervisor. In combination with the Simulink compatibility and computational capabilities of xPC Target.

6.2 Further Work

Several different possible directions to developing the Scoop 2012 system exists. Two different types of changes will be presented in the following discussion. The first being suggestions to enhance and secure the performance of the Scoop 2012 system in the scenarios of CoACT 2012. The second being additions that can incorporate additional features to the system. The discussion will start at the first type of suggestions.

The Scoop 2012 system would benefit from having a supervisor component. This would work as a diagnostic tool for other components sending information about their current status of execution. This would be beneficial for error handling purposes for future expansion and additions of components to the system. In addition to the supervisor I would suggest to build a customized HMI for monitoring and data logging, as the HMI provided by the xPC Environment is not sufficient when dealing with complex applications.

In the Scoop 2012 system the Trimble GPS is connected to the Target PC using a serial port connection. Especially the hardware setup could benefit from replacing this with an ethernet connection connected to the system switch. This would reduce the number of cables connected to the Target PC so that all external communication, except truck interaction, is performed using ethernet.

During the CoACT the first step towards lateral changes of position within a platoon were taken. However all lateral maneuvers were performed by a truck driver. With additional sensors and actuators in the truck the lateral maneuvers could be performed by the system instead. But to securely perform lateral maneuvers a system for lane detection is required. A possible additional feature for the Scoop system would therefore be to incorporate a map-based or an image based system to support lane detection. From a system design perspective the most reasonable approach to this type of component would be to develop it in the WSU. This due to the amount of data storage a map based system requires and the limited video sensor support in xPC Target. A map-based system requires

a huge database stored on a local hard drive. Read and write operations to a local disc has limitations in the xPC Target.

Bibliography

[1] European Union: European Environment Agency(EEA) (2012) *Greenhouse gas emission trends and projections in Europe 2012*. Copenhagen: EEA, Report no. 6, pp 92-93.

[2] European Union: European Environment Agency(EEA) (2009) *Greenhouse gas emission trends and projections in Europe 2009*. Copenhagen: EEA, Report no. 9, pp 43.

[3] Alam, A. (2011) *Fuel-Efficient Distributed Control for Heavy Duty Vehicle Platooning*. Licentiate thesis, Royal Institute of Technology.

[4] Mårtensson, J. et al. (2012) *The Development of a Cooperative Heavy-Duty Vehicle for the GCDC 2011: Team Scoop*. In: *IEEE Transactions on Intelligent Transportation Systems, Vol. 13, No. 3, September 2012*. IEEE Intelligent Transportation Society, pp. 1033-1049.

[5] GCDC organization (2011) *Rules & Technology Document Ver. 2.0.* [WWW] Helmond: TNO. Available from: http://www.gcdc.net/mainmenu/Home/technology/Rules_and_Technology/ [Accessed 2012-12-11].

[6] Baijer, O. et al. (2011) *GCDC Interaction Protocol* [WWW] Helmond: TNO. Available from: http://www.gcdc.net/mainmenu/Home/downloads/Technology/GCDC_-_Interaction_Protocol_2.4 [Accessed 2012-13-18].

[7] Johansson, K. H., Torngren, M. and Nielsen, L. *Vehicle Applications of Controller Area Networks*. in Hristu-Varsakelis, D. Levine, W. S. *Handbook of Networked and Embedded Control Systems*. Invited paper, Available from: http://www.s3.kth.se/ kallej/papers/can_necs_handbook05.pdf [Accessed 2013-02-18].

[8] Kurose, J., Ross, K. (2010) *Compter Networking: A top Dow Approach*. 5th ed. United States of America: Pearson Education, Inc.

[9] Alam, A. et al. (2011) *Cooperative Driving According to Scoop*. [WWW] Helmond: TNO. Available from: http://www.gcdc.net/mainmenu/Home/downloads/Papers [Accessed 2012-12-11].

[10] Liang, K. (2011) *Linear Quadratic Control for Heavy Duty Vehicle Platooning*. Master Thesis, Royal Institute of Technology.

[11] González-Conde Pérez, J.L. (2012) *Single vehicle estimator* Internal report, Royal Institute of Technology.

[12] Amazouz, A. (2013) *Optimal Control in Decentralized Vehicle Platooning.* Master Thesis, Royal Institute of Technology.

[13] Reyhanoğulları, A. (2013) *Algorithms and Models for Lateral Merging of Vehicle Platoons.* Master Thesis, Royal Institute of Technology.

[14] Almroth, M. (2013) *Simulation, Validation and Design of Cooperative Driving Systems using PreScan.* Master Thesis, Royal Institute of Technology.

[15] PreScan product information (2012)
TASS. Available from:
http://www.tass-safe.com/en/products/prescan

Appendix

CAN Specifications

GCDC00

Identifier	Length (bytes)	Repetion rate(ms)
419387392	8	10

Explanation	Start bit	Length (bits)	Resolution	Offset	Min	Max	Unit	Note
Version	0	8	1	0	0	250	-	
Status	8	8						
VehicleControllable	10	2	1	0	0	1	-	1 = No
EmergencyStop	12	2	1	0	0	1	-	1 = Active
Switch1	16	8						
On	16	2	1	0	0	1	-	1 = Active
Off	18	2	1	0	0	1	-	1 = Active
Up	20	2	1	0	0	1	-	1 = Active
Down	22	2	1	0	0	1	-	1 = Active
Switch2	24	8						
1	24	2	1	0	0	1	-	1 = Active
2	26	2	1	0	0	1	-	1 = Active
3	28	2	1	0	0	1	-	1 = Active
4	30	2	1	0	0	1	-	1 = Active
Odometer	32	32	1/10	0	0	42110812	m	

GCDC01

Identifier	Length (bytes)	Repetion rate(ms)
419387393	8	10

Explanation	Start bit	Length (bits)	Resolution	Offset	Min	Max	Unit	Note
VehicleSpeed	0	16	1/256	0	0	250.996	kph	
EngineSpeed	16	16	1/8	0	0	8031.875	rpm	
ActualEnginePercen	32	8	1	-125	-125	125	%	
DriverDemand	40	8	1	-125	-125	125	%	
SourceAddressOfCo	48	8	1	-125	-125	125	%	
Mode	56	8						

GCDC02

Identifier	Length (bytes)	Repetion rate(ms)
419387394	8	10

Explanation	Start bit	Length (bits)	Resolution	Offset	Min	Max	Unit	Note
DistanceToVehicle1	0	16	1/100	0	0	642.55	m	
DistanceToVehicle2	16	16	1/100	0	0	642.55	m	
DistanceToVehicle3	32	16	1/100	0	0	642.55	m	
DistanceToVehicle4	48	16	1/100	0	0	642.55	m	

GCDC03

Identifier	Length (bytes)	Repetion rate(ms)
419387395	8	10

Explanation	Start bit	Length (bits)	Resolution	Offset	Min	Max	Unit	Note
RelativeSpeedOfVehicle1	0	16	1/128	-250	-250	251.992	kph	
RelativeSpeedOfVehicle1	16	16	1/128	-250	-250	251.992	kph	
RelativeSpeedOfVehicle1	32	16	1/128	-250	-250	251.992	kph	
RelativeSpeedOfVehicle1	48	16	1/128	-250	-250	251.992	kph	

GCDC04

Identifier	Length (bytes)	Repetion rate(ms)
419387396	8	50

Explanation	Start bit	Length (bits)	Resolution	Offset	Min	Max	Unit	Note
MeanFronAxleSpeed	0	16	1/256	0	0	251	kph	
RelativeSpeedFrontAxleLeft	16	8	1/16	-7.8125	-7.8125	7.8125	kph	
RelativeSpeedFrontAxleRight	24	8	1/16	-7.8125	-7.8125	7.8125	kph	
RelativeSpeedDriveAxleLeft	32	8	1/16	-7.8125	-7.8125	7.8125	kph	
RelativeSpeedDriveAxleRight	40	8	1/16	-7.8125	-7.8125	7.8125	kph	
RelativeSpeedThirdAxleLeft	48	8	1/16	-7.8125	-7.8125	7.8125	kph	
RelativeSpeedThirdAxleRight	56	8	1/16	-7.8125	-7.8125	7.8125	kph	

GCDC05

Identifier	Length (bytes)	Repetion rate(ms)
419387397	8	50

Explanation	Start bit	Length (bits)	Resolution	Offset	Min	Max	Unit	Note
SteeringWheelAngle	0	16	1/1024	-31.374	-31.374	31.374	rad	
SteeringWheel	16	8						
TurnCounter	16	6	1	-10	-10	10	turns	
SensorType	22	2	1	0	0	3	-	0=Relative 1=ABS 2=Reserve 3=NA
YawRate	24	16	1/8192	-3.92	-3.92	3.92	rad/s	
LateralAcceleration	40	16	1/2048	-15.687	-15.687	15.687	m/s^2	
LongitudinalAcceler	56	8	1/10	-12.5	-12.5	12.5	m/s^2	

GCDC06

Identifier	Length (bytes)	Repetion rate(ms)
419387398	8	10

Explanation	Start bit	Length (bits)	Resolution	Offset	Min	Max	Unit	Note
AcceleratorPedal	0	8	2/5	0	0	100	%	
BrakePedal	8	8	2/5	0	0	100	%	
Status1	16	8						
IdleSwitch	16	2	-	-	-	-	-	
Kickdown	18	2	-	-	-	-	-	
EngineStart	20	2	-	-	-	-	-	
NotAvailable	22	2	-	-	-	-	-	
VehicleSpeed	32	16	1/256	0	0	250.996	kph	
NotAvailable	48	16	-	-	-	-	-	

GCDC07

Identifier	Length (bytes)	Repetion rate(ms)
419387399	8	10

Explanation	Start bit	Length (bits)	Resolution	Offset	Min	Max	Unit	Note
ActualGearRatio	0	16	1/1000	0	0	64.255	-	
CurrentGear	16	8	1	-125	-125	125	-	
RetarderLever	24	8	1	0	0	13	-	
ClutchPedalPosition	32	8	2/5	0	0	100	%	
NotAvailable	40	24	-	-	-	-	-	

GCDC20

Identifier	Length (bytes)	Repetion rate(ms)
419387424	8	1000

Explanation	Start bit	Length (bits)	Resolution	Offset	Min	Max	Unit	Note
Lamp1	0	8						
Red	0	2	-	-	-	-	-	1 = Active
Yellow	2	2	-	-	-	-	-	1 = Active
Green	4	2	-	-	-	-	-	1 = Active
Blue	6	2	-	-	-	-	-	1 = Active
Lamp2	8	8						
White	8	2	-	-	-	-	-	1 = Active
NotAvailable	10	2	-	-	-	-	-	
RedOut	12	2	-	-	-	-	-	1 = Active
GreenOut	14	2	-	-	-	-	-	1 = Active
VehicleSpeedReques	16	16	1/256	0	0	250.992	kph	
ExternalAccelerationr	32	16	1/2048	-15.687	-15.687	15.687	m/s^2	
NotAvailable	48	16	-	-	-	-	-	