Graz University of Technology

Implementation and Verification of a Standards-Compliant Car-2-X Demonstrator

Master's Thesis

submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Electrical Engineering

by

Bernhard Grosswindhager
Registration Number 0831038

to the Faculty of Electrical and Information Engineering
at the Graz University of Technology

Advisor: Univ.-Doz. Dipl.-Ing. Dr. techn. Daniel Watzenig
Assistance: Dipl.-Ing. Joachim Hillebrand

Graz, 28.10.2014

(Signature of Author) (Signature of Advisor)

Statutory Declaration

Bernhard Grosswindhager
Grazbachgasse 25/3, 8010 Graz

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Graz, 28.10.2014

(Place, Date)

(Signature of Author)

Acknowledgements

First of all I want to thank my family for all the support they gave me in the last years. It would have been impossible to finish my studies and this master's thesis without their financial help but also the encouragement in stressful and difficult times. I will never take this as granted and will always be thankful for the opportunities they offered me.

Moreover, I want to thank all my friends and girlfriend for their patience and showing understanding when I had to concentrate on my exams and thesis. I try my best to catch up the time I have missed with you.
Everytime when I was driving back home to Upper Austria I had a smile on my face because I knew that I will have a nice weekend together with my family and friends. It distracted me from my everyday university life which I was very thankful for. The same holds true for my friends here in Graz and I hope we will have more wonderful years together in Styria.

This master's thesis is in cooperation with the industrial partners MAGNA STEYR Engineering AG & Co KG and Virtual Vehicle Research Center, both located in Graz, Austria. It would not have been possible to finish this work without the support of them. My special thanks go to my supervisors at Virtual Vehicle, Dr.techn. Daniel Watzenig and Joachim Hillebrand. As well as my colleagues for the funny and amusing time and Dr.techn. Allan Tengg for the support with the eQuad and several other questions.
I also want to emphasize the good cooperation with MAGNA STEYR Engineering AG & Co KG and I want to thank Christian Payerl and Kurt Tschabuschnig.

Abstract

The transport policy of the European Union aims at fostering clean, safe and efficient travel throughout Europe. An integral part of the strategy is the Car-2-X technology. It enables vehicles to exchange information between each other (Car-2-Car) or with the road infrastructure (Car-2-Infrastructure) via a wireless link (IEEE 802.11p). The driver is informed about an upcoming dangerous situation and hazard at an early stage. Potentials are improved road safety, reduced traffic congestion and more environmentally friendly driving.

At the 6th ETSI ITS workshop 2014 in Berlin, Germany a first version of a Europe-wide specification for Car-2-X was successfully adopted and published. In this master's thesis the applicability of this basic set of standards is investigated by implementing a standards-compliant Car-2-X demonstration system. The focus is on the integration of the Car-2-X platform into series vehicles.

The Car-2-X system is based on Cohda Wireless hardware platforms. A selected list of six Car-2-X use cases are implemented on the embedded platform. It includes *emergency electronic brake lights*, *stationary vehicle warning*, *road works warning*, *approaching emergency vehicle warning*, *motorcycle warning* and *in-vehicle signage*. These use cases are in alignment with the suggestions of the Amsterdam Group for day one deployment.

Visualization of the Car-2-X relevant information is realized on an Android tablet PC. For this purpose an Android App is developed in Java. A Bluetooth interface is set up for the communication between the Car-2-X hardware and the tablet. ASN.1 encoding rules are used for a standardized and efficient data transfer.

Finally the functionality and usability of the developed Car-2-X demonstration system as well as the integration into the vehicle was successfully tested in field tests at the TU Graz campus.

This system is a valuable base for further research activities but also demonstrated that just a few steps are missing until the Car-2-X technology is ready for series production.

Kurzfassung

Die Verkehrspolitik der Europäischen Union zielt auf einen 'sauberen, sicheren und effizienten Verkehr in ganz Europa' ab. Ein wesentlicher Bestandteil dieser Strategie ist die Car-2-X Technologie. Sie ermöglicht Fahrzeugen untereinander (Car-2-Car) oder mit der Straßeninfrastruktur (Car-2-Infrastructure) Informationen, mittels einer drahtlosen Verbindung (802.11p), auszutauschen. Dadurch kann der Fahrer schon frühzeitig vor gefährlichen Situationen gewarnt werden. Das Potential dieser Technologie reicht von einer erhöhten Verkehrssicherheit, einer Reduzierung von Staus bis zu einem umweltfreundlicheren Transportwesen.

Beim 6. ETSI ITS Workshop 2014 in Berlin wurde eine erste Version einer europaweiten Standardisierung von Car-2-X präsentiert. In dieser Masterarbeit wird die Anwendbarkeit dieser Standards, mittels einer Implementierung eines Car-2-X Systems, untersucht. Dabei liegt der Schwerpunkt in der Integration der Plattform in ein Serienfahrzeug.

Als Basis für das System dienen Car-2-X Module von Cohda Wireless. In Summe wurden sechs Car-2-X Use Cases auf der Plattform implementiert. Diese sind: *elektronisches Bremslicht bei Notbremsung*, *Warnung vor einem stehengebliebenen Fahrzeug*, *Baustellenwarnung*, *Warnung vor einem annähernden Einsatzfahrzeug*, *Motorradwarnung* und *Beschilderung im Fahrzeug*. Sie sind konsistent mit den von der Amsterdam Group vorgeschlagenen Use Cases für die Einführung von Car-2-X.

Die Visualisierung der Car-2-X relevanten Daten erfolgt auf einem Android Tablet PC. Zu diesem Zweck wurde eine vollständige Android App in Java entwickelt. Als Interface zwischen der Car-2-X Hardware und dem Tablet dient Bluetooth und um einen standardisierten und effizienten Datentransfer zu ermöglichen, wurde ASN.1 mit den entsprechenden Encoding Rules verwendet.

Die Funktionalität und Benutzerfreundlichkeit des entwickelten Car-2-X Demonstrators sowie die Integration ins Fahrzeug wurde bei Feldtests am TU Graz Gelände erfolgreich getestet.

Das System ist eine wertvolle Basis für weitere Forschungsaktivitäten im Bereich Car-2-X und konnte auch zeigen, dass nur noch wenige Schritte erforderlich sind, bevor Car-2-X bereit für die Serienproduktion ist.

Contents

List of Figures

List of Tables

CHAPTER 1

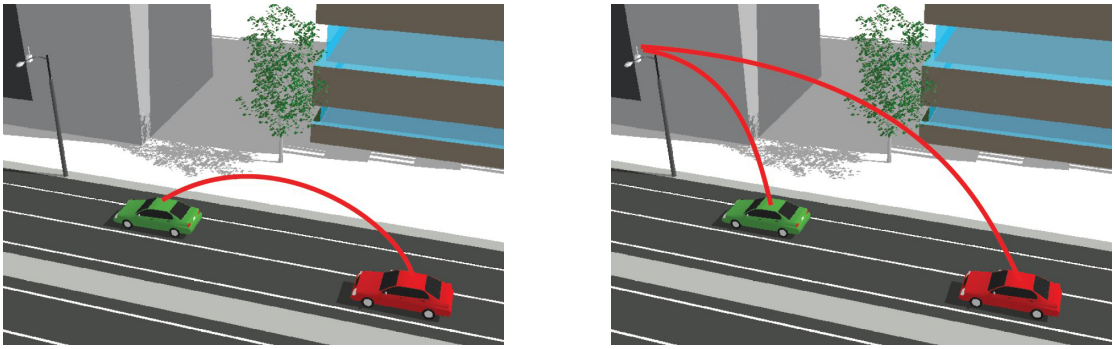
Introduction

In 2012 about 250,000 people were seriously injured in accidents happened on the roads of the European Union and 28,000 people died [1]. According to the World Health Organization (WHO) road traffic injuries are the ninth leading cause of death worldwide [2] in 2012 and the leading cause of death among young people, aged 15-29 years [3] [4].

But apart from that, road transport also has significant economical, ecological and health impacts [5] [6] [7]:

- Traffic congestion was estimated to cost the European economy the equivalent of around 1% of the Gross Domestic Product (GDP).
- Today transport accounts for around one-quarter of EU CO_2 emissions and over 30% of final energy consumption.
- Also the fuel consumption will increase because of congestion, approximately 30% under heavy congestion.
- If there is no significant policy change, prognoses of the European Union predict an increase of the total passenger transport activity and the freight transport activity until 2050 by 51% and 82%, respectively.
- A study of the Centre for Economics and Business Research (Cebr) in 2013 has shown that in Germany the average annual number of wasted hours in congestion is 37.8 hours. The total costs of traffic jams for households in the 22 largest German cities are 7.5 billion € in total and 509 € per household, respectively [8].
- On current trends, by 2020, road traffic injuries are predicted to become the third largest contributor to the Global Burden of Disease (GBD).
- About 90% of road accidents are caused at least in part by human error [9].

These alarming numbers are also perceived by the European Union, that is why the transport policies aim at fostering clean, safe and efficient travel throughout Europe [6]. An integral part of this policy is the deployment of Cooperative Intelligent Transport Systems (C-ITS) [6]. C-ITS uses **Car-2-Car (C2C)** and **Car-2-Infrastructure (C2I)**[1] communication (see Figure 1.1) to improve traffic safety, reduce traffic congestion and ensure more environmentally friendly driving [10].



(a) (b)

Figure 1.1: a) Car-2-Car, b) Car-2-Infrastructure [11]

In this context cooperative means that vehicles have the ability to communicate with each other via a wireless link, so that the intentions and positions of other vehicles are known to each other [12]. Smart vehicles can exchange information between each other (C2C) or between the vehicles and road infrastructure (C2I) like intersection lights or road signs. In general road applications these two types are closely linked, that is why they are usually grouped in the term **Car-2-X (C2X)**.
With this technology the car knows about an upcoming dangerous situation, like a car accident, obstacles on the street, road works or slippery road conditions before the driver could notice that. The Car-2-X system informs him of the danger with an visual or acoustical warning or automatically intervenes in the vehicle (e.g. activate the braking system and reduce speed).

A C2X architecture consists of On Board Units (OBUs), Road Side Units (RSUs) and ITS Central Stations (ICSs). The OBU is integrated in the vehicle and is the interface between driver, vehicle and environment. That means it has to be capable of informing the driver about a certain road situation (visually or acoustically). Moreover it has to send time-critical information to other vehicles and Road Side Units. The European standardization suggests the usage of ITS-G5 (based on IEEE 802.11p) as the wireless communication protocol (see chapter 2.2.2).
The RSUs are the interface between the vehicles on the road and the traffic control center. Communication to the vehicles is also based on ITS-G5 and for the communication

[1] In english literature also referred to as Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communication.

to the control center typically 3G and 4G cellular communications are used. The Road Side Units are mounted on the road infrastructure, for example a traffic light to inform the road users about the duration of the green phase.
Typically, the ITS Central Station is integrated in the traffic control center of a city or region. It analyzes the messages received from the RSUs and determine the traffic situation based on this information. The OBU could request information about the current route from the ITS Central Station. For example if a traffic jam comes up on that road an alternative route can be suggested and the driver reaches the target in the most comfortable and efficient way.
The biggest advantage of the communication and cooperation between the vehicles on the road is the improved driving safety. The vehicular communication offers a full and reliable situation awareness in every direction. The coverage area is significantly bigger than with onboard vehicle sensors because also hazards and obstacles behind the curve and thus not within sight can be detected. Existing technology like radar based sensors could not achieve this [12].

Not just road users will benefit from a deployment of Car-2-X technology but also other stakeholders, like the automotive industry, road operators and the government [13]. The OEMs are able to offer an improved product safety and higher value of the vehicles as well as a better sharing of information with the driver. Reducing the number of accidents and fatalities on the roads should be the main motivation to introduce this promising technlogy, but also economical and ecological reasons. A more efficient transportation reduces costs but also CO_2 emissions significantly. The road operators will also benefit from an optimized road capacity and the commercial value of the collected data, although the legal issue is still not solved.

1.1 Objectives of this thesis

Infotainment wireless systems like GPS or Bluetooth are already state-of-the-art in vehicles, but Car-2-X is the first safety-critical system. That is why a high reliability has to be ensured and a strict standardization is necessary before a successful market launch is possible. In Europe ETSI[2] and several other standardization organizations developed and prepared a set of standards for deployment of C-ITS (see chapter 2.2). Finally in February 2014 at the 6th ETSI ITS workshop in Berlin, Germany the results have been presented and a first version of a Europe-wide specification was successfully published. In this master's thesis these brand-new standards, specifications and guidelines are analyzed, applied and tested by the implementation of a **standards-compliant Car-2-X demonstration system**. The thesis is in cooperation with MAGNA STEYR Engineering AG & Co KG and Virtual Vehicle Research Center, both located in Graz, Austria.

[2]**European Telecommunications Standards Institute (ETSI)** is an independent standardization organization in the Information and Communications Technology industry. Its standards are globally-applicable, probably the most successful one is GSM. `http://www.etsi.org`.

The main parts, novelties and focus areas of this thesis are:

- Analysis of the current Car-2-X standardizations and specifications.
- Integration of an embedded Car-2-X platform into a series vehicle via CAN bus.
- Graphical visualization of Car-2-X relevant information on an Android tablet PC.
- Detection and visualization of all vehicles within communication range equipped with Car-2-X technology.
- Interoperable and expandable implementation by using standards-compliant message types and interfaces.
- Implementation of six by the Amsterdam Group for 'day-one' deployment suggested Car-2-X use cases on the embedded platform.
- Compatibility with the electrically driven technology demonstrator *eQuad* of the industrial partner Virtual Vehicle.
- Verification of the functionality and usability of the Car-2-X demonstration system in field tests.

1.2 Outline

The first chapter covers the state-of-the-art analysis of Car-2-X. Results of the 6^{th} ETSI ITS workshop 2014 in Berlin, Germany are discussed and an overview of important C-ITS consortia, associations and stakeholders is presented. The standardization focuses mainly on topics which are related to the practical work, like the message types and access layer specifications. In one paragraph also the capability of LTE for vehicular communications is analyzed and an algorithm to overcome the limited bandwidth, Decentralized Congestion Control (DCC), is introduced.
Chapter 3 has a focus on the newest Cohda Wireless [11] Car-2-X platform MK4 and the first market-ready chipset for Car-2-X communication, developed by Cohda Wireless and NXP Semiconductors. The interfaces of the platform are explained in more detail and also the software tools necessary to develop full ITS applications are presented.
As described in the introduction the Car-2-X technology has benefits in different kind of applications. That is why chapter 4 gives an overview of use cases for a 'day-one' deployment of C-ITS presented by the Amsterdam Group and the European Union.
All the practical work done within this thesis and the challenges to implement the Car-2-X demonstration system are covered in chapter 5. It starts with the used technologies and interfaces and concludes with a detailed description of the embedded ITS application and the Car-2-X Android App.
Finally the developed Car-2-X demonstrator is investigated in field tests. The results are presented in chapter 6 and a summary as well as an outlook for future works is given in chapter 7.

CHAPTER 2

State-of-the-art Car-2-X

The Car-2-X technology and its specification is still under development and an ongoing process. That is why the current state of the technology, standards and hardware has to be investigated at regular intervals. Another master's thesis at the Virtual Vehicle Research Center [14] already covers some parts, like the difference between European and US standardization and the different protocol layers in detail. Thus this state-of-the-art analysis focuses mainly on other topics, which are important for the practical work and the implementation of the Car-2-X demonstration system.

Several stakeholders and interested parties are involved in the deployment of the Car-2-X technology. The same holds true for the standardization process, currently the C-ITS standards consist of much more than 100 documents, still rising. Many working groups from different standardization organizations contribute to it. Because of that the first pages of this chapter focus on the introduction of important consortia and associations as well as working groups for the standardization. Afterwards, the ITS reference architecture, the specifications of the ETSI ITS-G5 (802.11p) technology as well as the Decentralized Congestion Control (DCC) algorithm, to overcome the limited bandwidth, will be covered.
It is essential that the transmitted messages are standard-compliant. That is why the most important message sets Cooperative Awareness Message (CAM) and Decentralized Environmental Notification Message (DENM), but also the infrastructural message sets are described.

2.1 Consortia and Associations

In the deployment of vehicular communications many different interest groups, like governments, road operators, car and also Car-2-X equipment manufacturers are involved and have to be considered. This is why coordinated efforts for standardization and formation of consortia is essential. Especially at the beginning it is difficult to keep track of

all of them and in particular which interest group they are representing. That is why this chapter gives an overview of the most important European consortia and commissions, including a short explanation:

- **Amsterdam Group** [15] [16]
  The Amsterdam Group was formed in 2011 as a strategic alliance and voluntary cooperation platform of European ITS stakeholders. The objective is to facilitate joint deployment of Cooperative ITS in Europe by cooperation between road authorities and operators, city authorities, industry and the European Commission. It includes the umbrella organizations CEDR, ASECAP, POLIS and C2C-CC.

- **CAR 2 CAR Communication Consortium (C2C-CC)** [17]
  C2C-CC is a non-profit organization initiated in the summer of 2002 by the European vehicle manufacturers to represent automotive OEM's (e.g. Audi, BMW, Daimler,...), equipment suppliers (Cohda Wireless, Denso,...) and research organizations (e.g. Fraunhofer Institute, TNO,...). The C2C-CC works in close cooperation with the European (ETSI TC ITS) and international (ISO/TC 204) standardization organizations to develop an open European standard for C-ITS. The industry cooperation partner of this thesis MAGNA STEYR Engineering AG & Co KG has a basic C2C-CC membership to receive exclusively information about Car-2-X communication.

- **European Association of Tolled Road Infrastructures Concessionaires (ASECAP)**
  ASECAP is an umbrella association of the toll road operators. It was established by national associations, companies and other bodies responsible for the financing, construction, maintenance and operation of road infrastructure. The revenues of these companies are generated in part or in full through collection of fees from end road users [18]. An important issue and activity, beside an electronic toll collection, is to increase traffic safety and environment protection on European roads. Austria is represented by ASFINAG.

- **Conference of European Directors of Roads (CEDR)** [19]
  CEDR was created on September 18th, 2003 in Vienna, Austria as an organization of European public road operators; Austria is represented by the Federal Ministry for Transport, Innovation and Technology and ASFINAG.

- **POLIS**
  POLIS is a network of European cities and regions working together to develop innovative technologies and policies for local transport [20]. This association consists of 58 cities and regions.

- **ERTICO - ITS Europe** [21]
  ERTICO represents the interests and expertise of currently 103 partners involved in developing and deploying ITS solutions, such as public authorities, research institutes, infrastructure operators, suppliers, users and vehicle manufacturers. A

main issue is to inform and advise policy makers on appropriate policy frameworks to remove potential barriers for ITS. The ITS World Congresses are also organized, together with ITS America and ITS Asia-Pacific, by ERTICO. Currently the Compass4D pilot project is coordinated by ERTICO, in which three C-ITS services (Forward Collision Warning, Red Light Violation Warning and Energy Efficient Intersection Service) are deployed in seven European cities (Bordeaux, Copenhagen, Helmond, Newcastle, Thessaloniki, Verona, Vigo). These services will be tested over one year (although the full duration of the project is from 1st January 2013 - 31st December 2015) in buses, emergency vehicles, trucks and taxi drivers, as well as private drivers [22].

- **European Commission**
  Since the successful deployment of Car-2-X stands and falls with the strategy defined by the policy makers, the European Commission plays an important role; or more precisely the Directorate General for Communications Networks, Content and Technology (DG CONNECT), which is responsible for the digital agenda of the European Union [23] and the Directorate General for Mobility and Transport (DG MOVE) ITS unit.

2.2 Standardization

Harmonization and interoperability of different ITS-Systems are key requirements to reach a minimum Car-2-X penetration on the roads. Because it is necessary that vehicles from different manufacturers can communicate with each other and with the road infrastructure. Furthermore, the service has to be provided without interruption also if the consumer crosses national borders. Hence, to take full advantage of the benefits that C-ITS can bring to the mobility sector a global or at least a European standardization is absolutely essential. This is why the European Commission in 2009 addressed the **standardization mandate M/453** to the three European Standardization Organizations (ESOs) ETSI, CEN[1] and CENELEC[2]. In which they are prompted to prepare a coherent set of standards, specifications and guidelines to support the implementation and deployment of Cooperative Intelligent Transport Systems in Europe [24]. CEN and ETSI formally accepted the mandate. CENELEC did not accept it and does not take part in standards development under this particular mandate. Additionally, an international

[1] **European Committee for Standardization (CEN)** is developing European standards in various sectors. The members are the national standards bodies of 33 European countries including all EU member states, three of the EFTA members (Iceland, Norway and Switzerland), Turkey and Macedonia. Standards approved by CEN are accepted in all of these countries. `http://www.cen.eu`.

[2] **European Committee for Electrotechnical Standardization (CENELEC)** is responsible for European standardization in the area of electrical engineering. Its member states are the same as of CEN. `http://www.cenelec.org`.

cooperation is taking place with ISO[3], IEEE[4] and SAE[5] to achieve global harmonization of C-ITS standards.
There are several working groups and technical committees from each of the organizations involved in the standardization. The most important ones for Intelligent Transport Systems are:

- ETSI TC ITS
- CEN/TC 278
- ISO/TC 204
- IEEE 802.11 and 1609 WG

At the 6th ETSI ITS workshop in Berlin, Germany the ESOs have confirmed that the first version of a Europe-wide specification was successfully adopted and published [25] [26]. But there are still some open issues (Security, Congestion Control, Applications,...), so the standardization is an ongoing process. This is why the standards covered in this thesis are the status at the present point in time and can be changed in further releases of the C-ITS standards and should not be considered as fixed.

2.2.1 Reference Architecture

A reference architecture of communications in ITS is defined in the ETSI EN 302 665 standard [27]. It is using an open system approach instead of a proprietary one to enable different implementation architectures. Furthermore it can be seen as a tool-box, where you just implement the functionality you actually need. The ITS-Station (ITS-S) reference architecture is shown in 2.1, it follows the OSI model [28] in the following way:

- **Access** is representing OSI layers 1 and 2 (Physical- and Data link layer)
- **Networking & Transport** is representing OSI layers 3 and 4 (Network- and Transport layer)
- **Facilities** is representing OSI layers 5,6 and 7 (Session-, Presentation- and Application layer)

Three more blocks are involved in the reference architecture, first of whom is the **Applications** block. It presents the ITS-S applications, which make use of the ITS-Station

[3] **International Organization for Standardization (ISO)** was founded in 1947 and is now the largest developer of voluntary international standards. It is composed of representatives from national standards bodies and has currently members from 161 countries. `http://www.iso.org`.

[4] **Institute of Electrical and Electronics Engineers (IEEE)** is the world's largest professional association dedicated to advancing technological innovation and excellence with about 425,000 members in about 160 countries. IEEE performs its standardization work via the IEEE Standards Association (IEEE-SA). `http://www.ieee.org`.

[5] **Society of Automotive Engineers (SAE)** is a U.S.-based, global association of more than 138,000 engineers. It also develops standards, mainly in the transport industries such as automotive, aerospace and commercial vehicles. `https://www.sae.org`.

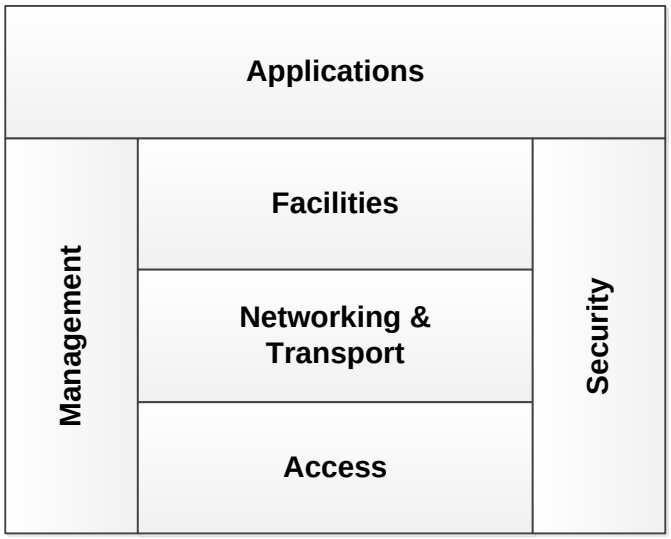


Figure 2.1: ITS-Station reference architecture

services to connect to one or more other ITS-S applications. A connection of two or more complementary ITS-Station applications (e.g. server application in one ITS-Station and the client application in another one) forms an ITS application.
The managing of communications in the ITS-Station is done in the **Management** block and the **Security** layer provides security services to the communication protocol stack. Additionally to the isolated layers there is further functionality related to several of these layers, the so called cross-layer functionality, for more details please refer to [29].

The thesis in [14] already gives a very good overview of the different layers, also in comparison to the United States. That is why this thesis will just focus on selected topics regarding the ITS architecture, which are relevant for the further process of this thesis.

2.2.2 Access Layer Specifications

The two lowest layers of the ISO/OSI model, physical and data link layer, are termed *Access layer* in the ITS reference architecture [27]. The European access technology, defined in the European Standard ETSI EN 302 663 [30], is called **ITS-G5**. It is based on IEEE 802.11 or to be more precise on IEEE 802.11p (incorporated in IEEE 802.11:2012) [31]; a modification of 802.11a[6] (see Table 2.1 for comparison).
It also uses orthogonal frequency-division multiplexing (OFDM) modulation but with double symbol duration (4μs vs. 8μs) and guard period (0.8μs vs. 1.6μs). This reduces the intersymbol interference caused by multipath propagation. Further improvements were achieved by reducing the throughput (27Mbps max. vs. 54Mbps max.). The

[6] 802.11a was defined for the US market. The European harmonization of this specification is called 802.11h.

high multipath delay spread, resulting in a smaller coherence bandwidth and a more frequency-selective fading, is mitigated by halving the bandwidth (10MHz vs. 20MHz).

Originally 802.11p was designed for the US protocol stack Wireless Access in Vehicular Environment (WAVE) but is now also used in the European ITS stack (ITS-G5). There are slight changes in the frequency band (5.850-5.925GHz in USA and **5.855-5.925GHz in Europe**) and the channel allocation (see 2.2.2.2). Moreover, the ITS-G5 standard also adds a Decentralized Congestion Control (DCC) method to control the network load (see 2.2.3).

The modulation scheme depends on the data rate, as you can see in Table 2.1 several transfer rates are allowed in IEEE 802.11p. But just three of them are mandatory: 3 Mbit/s (BPSK), 6 Mbit/s (QPSK) and 12 Mbit/s (16-QAM), especially 6 Mbit/s is very common in ETSI ITS-G5 as it is shown later in Table 2.2.

	IEEE 802.11a	**IEEE 802.11p**
Data Rate	6, 9, 12, 18, 24, 36, 48, 54 Mbps	3, 4.5, 6, 9, 12, 18, 24, 27 Mbps
Modulation	BPSK OFDM QPSK OFDM 16-QAM OFDM 64-QAM OFDM	same
Error Correction Coding	Convolutional Coding with K=7	same
Coding Rate	1/2, 2/3, 3/4	same
OFDM Symbol Duration	4 μs	8 μs
Guard Period	0.8 μs	1.6 μs
Bandwidth	20 MHz	10 MHz
Frequency Range	5.180 GHz - 5.825 GHz	5.850 - 5.925 GHz

Table 2.1: Comparison 802.11a vs. 802.11p [32] [33]

In a vehicular communication system there are several technical challenges which are not encountered in other wireless networks. The most obvious one is that in vehicular environments there are not just stationary objects, like a WLAN router in an apartment, but fast moving vehicles and thus a wide range of relative speeds between nodes. This results in a much faster fading and more Doppler spread [33] which causes inter-carrier interference. These problems have to be addressed in the receiver design.

Furthermore, ITS systems are typically time-, safety- and mixed-critical systems and a failure could cause the loss of life, seriously injure people or damage property. That is why, in contrast to cellular environments, the vehicles have to communicate directly with each other and should not need a base station or an access point (Vehicular Ad-Hoc Network (VANET)). Because that would need time-consuming authentication or

association procedures [34] and the tight latency requirements of most of the road safety applications could not be fulfilled. The 802.11 standard defines the use of access points, in more general Basic Service Sets (BSSs), which is a single access point together with all associated stations. To perform the vehicular communication outside the context of a BSS the 802.11p standard uses a so called wildcard BSSID[7], this is an identifier filled with just binary 1s in the header of the 802.11p frame. It indicates the receiver that there is no need to wait for completion of the authentication and association process. These simple modifications enable a peer-to-peer communication in the 802.11p standard and reduce the latency significantly. This is a prerequisite to overcome the real-time nature of safety applications.

2.2.2.1 LTE vs. 802.11p

In [33] 802.11p is mentioned as the only wireless technology that can potentially meet the short latency requirements of below 100 ms for road safety messaging (see chapter 4). But in fact also the fourth generation (4G) of mobile networks, Long-Term Evolution (LTE) could be capable of it. It is developed by 3GPP and is based on GSM/EDGE and UMTS/HSPA. The main advantages and disadvantages of this new access technology and the potential to support Car-2-X communication are discussed in this chapter.

LTE is providing high data rates of at least 100Mbit/s in downlink and 50Mbit/s in uplink, but also low latency and access delays [35]. According to 3GPP the LTE standard is also able to handle very high speeds of the terminals, which is a prerequisite for vehicular communications. Speeds of up to 350 km/h (or even 500 km/h depending on the frequency band) are supported, even though the performance degrades for high speeds [36].

Several studies have tried to evaluate the applicability of LTE for Intelligent Transport Systems. A master's thesis at the University of Twente in 2011 [36] claims that LTE could meet the ITS requirements in most of the scenarios and in some cases even outperforms the 802.11p standard. For example LTE offers larger capacity (700 ITS users vs. 400 ITS users for 802.11p) and larger communication range (2000 meters vs. 700 meters for 802.11p). But 802.11p has a much lower latency, which is the most important property for road safety applications. Furthermore, if the background load increases significantly, LTE could not meet the strict ITS requirements anymore. Thus a prioritization of ITS traffic over the background traffic is essential.
The conclusion of the work is that 802.11p is still the best solution for the time-critical road safety application, but LTE could cover other ITS applications (see chapter 4) with a latency requirement bigger than 100ms. In such a hybrid solution LTE could reduce the load of the 802.11p channel.

[7] Basic Service Set Identification (BSSID) are unique identifiers for each BSS.

In [37] it is mentioned that the abilities of LTE to support cooperative vehicular safety applications are poor. Because the network easily becomes overloaded even under the idealistic assumptions of the LTE model used in this paper. The bottom line is that 802.11p/WAVE looks more promising for vehicular safety.
The paper in [38] also compared the mobile cellular system UMTS to LTE and 802.11p but just in an intersection scenario. UMTS could not fulfill the C-ITS requirements at all and LTE performed reasonably well. But all together also LTE will likely not be able to provide the same latency and performance as 802.11p communications.
An interesting study is also [39]. It offers a good overview of the main candidate wireless technologies for vehicular communications. The conclusion is similar to the other studies: LTE will not be able to replace 802.11p but it can be a valuable support, especially in the initial deployment phase of vehicular communications.

The literature shows the potential of LTE but at the bottom line it will not be in competition with 802.11p. The strict requirements of road safety applications can not be guaranteed, because of the limited capacity in the case of high background load and the high latency.

Another point of view, which was not even covered in the mentioned studies, is the limited coverage area of LTE. According to a report of the European Commission as of October 2013 59% of all households in the European Union have access to LTE, but the rural coverage is significantly lower (~14%) [40]. Although no reliable numbers are available about the coverage of LTE in percentage of the size of Europe, it is expected that it is even less. But a coverage of nearly 100% is inevitable, because a customer is in the confidence that for example the technology will also work on rural roads.

All together LTE will note replace 802.11p but a solution could be a hybrid system of both. Depending on the application and infrastructure availability one of the technologies can be chosen. LTE will be valuable for internet service use cases (e.g. Transparent car leasing and rental, Map download and update,...) and for the communication between the Road Side Units and the ITS Central Stations (ICS) [35].

A technology which has to be kept in mind is LTE Direct. This technology is intended to enable a device-to-device technology via LTE. It is currently under investigation by the chip manufacturer Qualcomm and several other member of the 3GPP consortium. It is expected that LTE Direct is part of the next standardization release of LTE. Qualcomm published an article in August 2014 [41] describing the main properties. A stated range of about 500 m could make it interesting for vehicular communications but further details of the specification are still missing.

2.2.2.2 Channel Allocation

The preserved ITS-G5 frequency bands in Europe are [30] [42]:

- **ITS-G5A:** *Frequency range: 5.875 - 5.905 GHz.* Dedicated to ITS road safety related applications.
- **ITS-G5B:** *Frequency range: 5.855 - 5.875 GHz.* Dedicated to ITS non-safety road traffic applications (e.g. traffic efficiency). The ITS-G5B band is not allocated European wide, thus local usage restrictions might apply [43].
- **ITS-G5D:** *Frequency range: 5.905 - 5.925 GHz.* Reserved for future ITS applications.
- **ITS-G5C:** *Frequency range: 5.470 - 5.725 GHz.* Usage for Radio Local Area Network (RLAN), Broadband Radio Access Network (BRAN) and Wireless Local Area Network (WLAN). For operation in this frequency band several functionalities (e.g. Dynamic Frequency Selection (DFS),...) are necessary which are not supported when using wildcard BSSIDs, that is why a communication outside the context of a Basic Service Set is not possible in this frequency band.

In the ITS-G5 frequency bands, except ITS-G5C, no Dynamic Frequency Selection is possible that means that scanning for and selecting the least-congested channel is not enabled. Because of that the wireless standard 802.11p requires predetermined frequency channels which are one Control Channel (G5-CCH) and several Service Channels (G5-SCH) (see Table 2.2). The control channel G5-CCH and the service channels G5-SCH1 and G5-SCH2 belong to ITS-G5A (see usage order in 2.2.3), so they are dedicated to road safety. On the other hand G5-SCH3 - G5-SCH5 are used for ITS road efficiency applications.

	Channel type	**Frequency range [GHz]**	**802.11 Ch. No.**	**Default data rate**	**TX power limit**
ITS-G5A	G5-CCH	5.895 - 5.905	180	6 Mbit/s	33dBm EIRP
	G5-SCH1	5.875 - 5.885	176	6 Mbit/s	33dBm EIRP
	G5-SCH2	5.885 - 5.895	178	12 Mbit/s	23dBm EIRP
ITS-G5B	G5-SCH3	5.865 - 5.875	174	6 Mbit/s	23dBm EIRP
	G5-SCH4	5.855 - 5.865	172	6 Mbit/s	0dBm EIRP
ITS-G5D	G5-SCH5	5.905 - 5.915	182	6 Mbit/s	0dBm EIRP
	G5-SCH6	5.915 - 5.925	184	6 Mbit/s	0dBm EIRP
ITS-G5C	G5-SCH7	5.470 - 5.725	94 - 145	several	30dBm EIRP (DFS master) 23dBm EIRP (DFS slave)

Table 2.2: European channel allocation as per ETSI EN 302 663 [30]

Column five and six of Table 2.2 show that the transmitter power limit is between 23dBm (∼200mW) and 33dBm (∼2W) and the default data rate is 6 Mbit/s and 12 Mbit/s (G5-SCH2). For the sake of completeness, there is also a TX power density limit defined in [30]. In practice the more stringent transmission limit (power limit or power density limit) applies.

2.2.3 Decentralized Congestion Control (DCC)

Because of the limited bandwidth, as shown in the last chapter, in traffic situations like a congestion the data load can exceed the wireless channel capacity and a accurate communication can not be guaranteed anymore. That is why a Decentralized Congestion Control as specified in ETSI TS 102 687 [44] is mandatory for ITS-G5 stations operating in ITS-G5A and ITS-G5B frequency bands (ITS-G5C is not under DCC control). It controls the network load and helps to overcome critical situations without a radio range degradation or a failure of the system.
DCC is a cross layer function, requiring components on several layers of the ITS-Station reference architecture (Access, Facilities, Management,...). The main DCC mechanisms are Transmit Power Control (TPC), Transmit Rate Control (TRC), Transmit Datarate Control (TDC), Transmit Access Control (TAC) (ensures fair channel access by being more restrictive to ITS-Stations that transmit many packets) and DCC Sensitivity Control (DSC) (determines whether the transmitter is clear to send or not). These algorithms make use of a consistently sensing and monitoring of the current channel status (e.g. Received Signal Strength Indicator (RSSI)) in the Access layer. Furthermore, DCC relies on a receive model that is used to estimate the communication range for line-of-sight communication, which depends on the transmit power and data rate.

Basically the channels are associated with a certain DCC state RELAXED, ACTIVE or RESTRICTIVE, depending on the current channel load. In the initial state RELAXED the channel is assumed to be mainly free and therefore no restrictions occur. If the channel load increases and exceeds a given threshold for a period of time the channel will go to the DCC state ACTIVE or even RESTRICTIVE (channel is overloaded). The properties of the communication have to be adapted accordingly. As an example a CAM is sent periodically in the ITS-G5A control channel (CCH) with a repetition rate of up to 10Hz, see 2.2.4.1. Depending on the current congestion state (DCC state) of the control channel it could be necessary to reduce the repetition rate and the TX power of the CAM transmission. This reduces the congestion of the channel and the communication can be preserved. Of course this information has to be forwarded to the Facilities layer where the CAM is generated. In the ACTIVE DCC state the repetition rate would be decreased to 5Hz, but if there are messages which have to be transferred with a higher rate the SCH1 channel has to be used. This usage order is also defined in [43], for ITS-G5A the order is (1) CCH (2) SCH1 (3) SCH2 and for ITS-G5B (1) SCH3 (2) SCH4.

2.2.4 Message Sets

2.2.4.1 Cooperative Awareness Message (CAM)

A main characteristic of C-ITS is that the individual road user and road infrastructure knows about the position, status and dynamics of other vehicles; this is the so called Cooperative Awareness (CA). This is necessary to ensure the proper working of ITS applications and services, for example you have to know the exact position of a surrounding car to detect a high collision risk. That is why a regular exchange of information between all the road users (*point-to-multipoint communication*) within direct communication range with periodically sent Cooperative Awareness Messages is necessary. This service is a mandatory facility for all ITS-Stations on the road and in comparison to other message types the received messages will not be forwarded to other ITS-Stations **(single hop)**. The service to send and receive CAMs is provided by the CA basic service, which is an entity in the Facilities layer. An important task of it is to manage the CAM generation frequency, the time interval between two consecutive CAMs. A lower and upper bound is defined in the ETSI standard EN 302 637-2 [45] with **1Hz and 10Hz**, respectively. In between these limits the CA basic service controls the generation frequency depending on the change of the own status, e.g. change of position and speed but especially the channel load as determined by the Decentralized Congestion Control (DCC). Besides the generation frequency it is also necessary that the time required to generate a CAM is below 50ms and to ensure proper interpretation of the message from the different ITS-Stations a time-stamp has to be included. Therefore, a sufficient time synchronization between the different road users is crucial.

The general structure of a Cooperative Awareness Message is shown in 2.2. It consists of the ITS Packet Data Unit (PDU) Header, which includes the protocol version, message type (CAM, DENM,...) and the ITS-S ID of the originator, as well as several containers. They can be used optionally and individually, e.g. a vehicle ITS-Station typically consists of at least a Basic Container and a High Frequency Container.

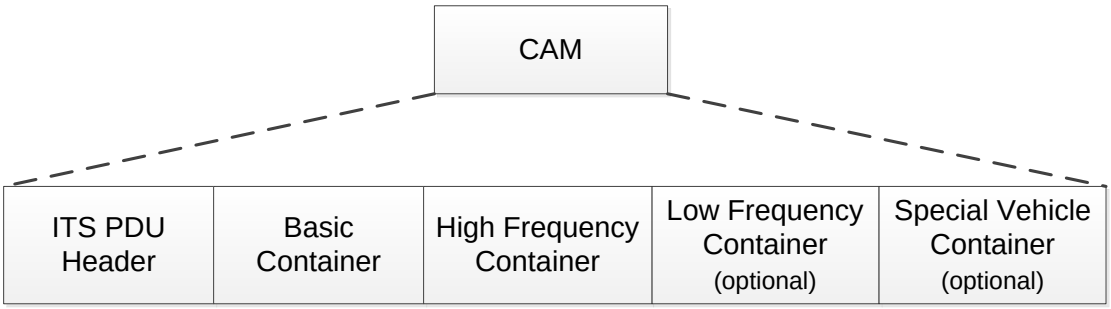


Figure 2.2: General structure of a CAM [45]

Depending on the application and the current traffic situation additional containers are used, where each container consists of several optional or mandatory data elements and data frames (see Annex B in [45]):

- **Basic Container** holds basic information of the sending ITS-Station, like the type of the ITS-Station and the latest geographic position; this container is common for all types of ITS-Stations (vehicle ITS-S, road side ITS-S, personal ITS-S,...).
- **High Frequency Container** includes highly dynamic and fast-changing information of the sending ITS-Station, like speed or heading.
- **Low Frequency Container** contains static and not highly dynamic information of the sending ITS-Station, like the dimension of the vehicle or the status of the headlights.
- **Special Vehicle Container** contains specific information of the role of the vehicle within the traffic, for example an emergency vehicle or if the vehicle is transporting dangerous goods.

The current version of the standard EN 302 637-2 [45] so far just supports vehicle ITS-Stations and as already mentioned for them the Basic Container and the High Frequency Container is mandatory. The Low Frequency Container as well as the Special Vehicle Container are optional. Typically these containers are included in the CAM as soon as more than 500 ms have elapsed since their last transmission.

Like the other message types listed here the standard presents the CAM format in Abstract Syntax Notation One (ASN.1) (see chapter 5.2), for encoding and decoding of CAMs the *Unaligned Packed Encoding Rules (PER)* as defined in [46] are used.

2.2.4.2 Decentralized Environmental Notification Message (DENM)

If a sudden incident happens on the road, the road users have to be informed immediately by the **event-triggered** Decentralized Environmental Notification Message. The DENM protocol is implemented by the DEN basic service in the Facilities layer and contains information about the type of road hazard, its position, a detection time and optional a time duration (see ETSI 302 637-3 standard [47]).
An originator ITS-Station initiates the transmission (New DENM) when an event is detected, but also updates (Update DENM) information of the event and terminates an event (Cancellation DENM). A receiving ITS-Station has to process and analyze an incoming DENM and, if necessary, informs the driver about the dangerous situation. The general data flow of DENM exchange is shown in Figure 2.3.

The main difference to the CAM dissemination is that DENMs are forwarded to other road users. This ensures that also vehicles in a larger distance than the communication range are informed about the event and can react early enough. Actually the forwarding is done in the ITS Networking & Transport layer. As marked with dotted lines in Figure 2.3 in situations, when the originator ITS-Station cannot repeat the DENM message anymore (e.g. out of range), also the DEN basic service is involved in the DENM

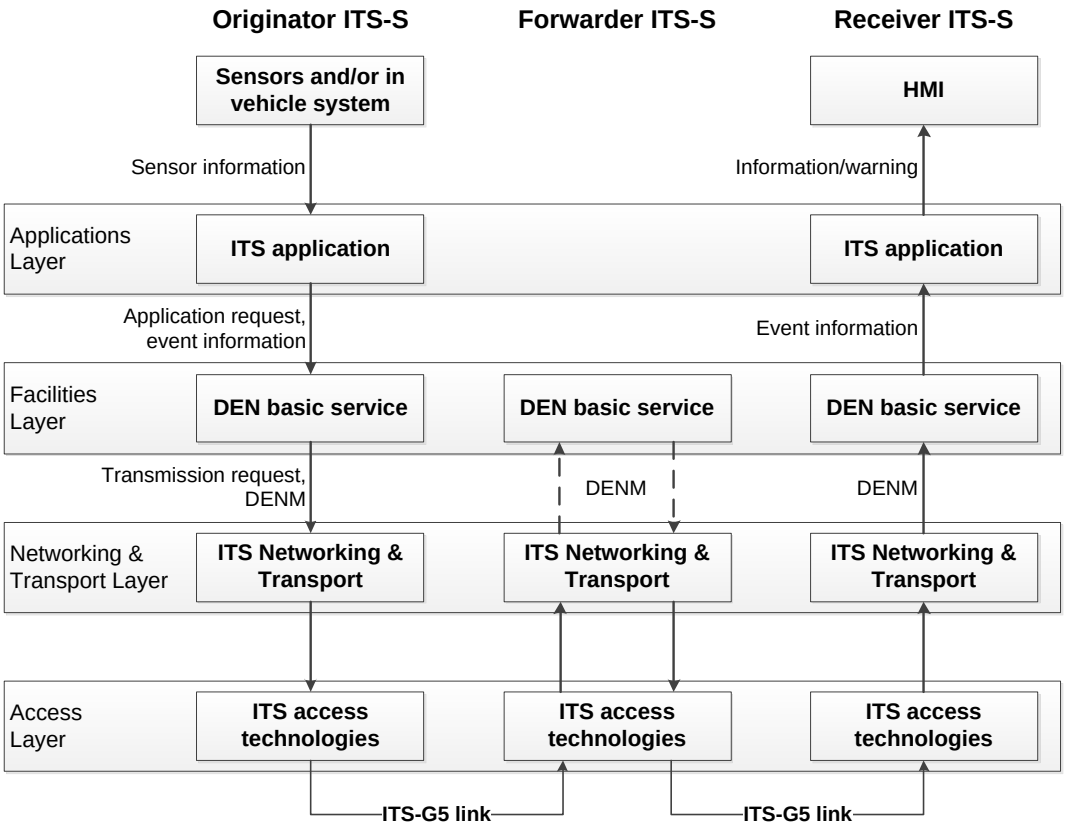


Figure 2.3: General data flow of DENMs [47]

forwarding. Furthermore, there is also interaction with other Facilities layer entities, for example the Local Dynamic Map (LDM). LDM is a database within an ITS-Station containing useful information for active ITS applications, like attributes of surrounding vehicles or events provided by the DEN basic service [48]. Thus at the receiver ITS-S the LDM is updated according to the received DENM and any authorized application can request the information for further processing.
The transmission of DENMs in the ITS network lasts as long as the event is present and is stopped either automatically after a predefined expiry time or manually by an ITS-Station. For this purpose the ITS-Station sends a special DENM that the event is over (Cancellation DENM by the originator ITS-S or Negation DENM by another ITS-S).

Figure 2.4 shows the general structure of a DENM. The ITS PDU Header is identical to the one in the CAM, containing the protocol version, the message type and the ITS-S ID of the originator ITS-S. The DENM payload consists of several containers, which are filled according to the event (the following list shows just a selection of data elements, for the complete list please refer to Annex B in [47]):

- **Management Container** is mandatory and consists of important DENM management data. It defines the type of the DENM (e.g. `isCancellation` or `isNegation`) and the ID of the event. The identifier of an detected event is the `actionID`. Every new DENM has to be assigned to an individual `actionID`, it consists of the originator ITS-Station ID and a sequence number which will be incremented each time the originator ITS-S detects an event. Furthermore, this container defines the duration of the DENM validity (`ValidityDuration`) as well as the event position and the relevant area.

- **Situation Container** defines the type of the detected event in more detail. A unique code (`eventType`) defines the reason of the hazard, to be more precise it is split in two data elements `causeCode` and `subCauseCode`. For example a vehicle breakdown because of lack of fuel has *causeCode=91* and *subCauseCode=1* (for the full cause code assignment table see p.29 of [47]).

- **Location Container** contains information about the event location. For example the data frame `traces` describes a set of consecutive positions leading to the event position. If multiple traces are leading to the event in the latest standard up to eight traces can be added to DENMs and each of them can consist of 40 positions (`PathPoint`).

- **Alacarte Container** contains additional information that is not included in the three previous containers. Depending on the use case application specific data can be transported here. An example is the `laneNumber` data element to inform about the lane position of the event.

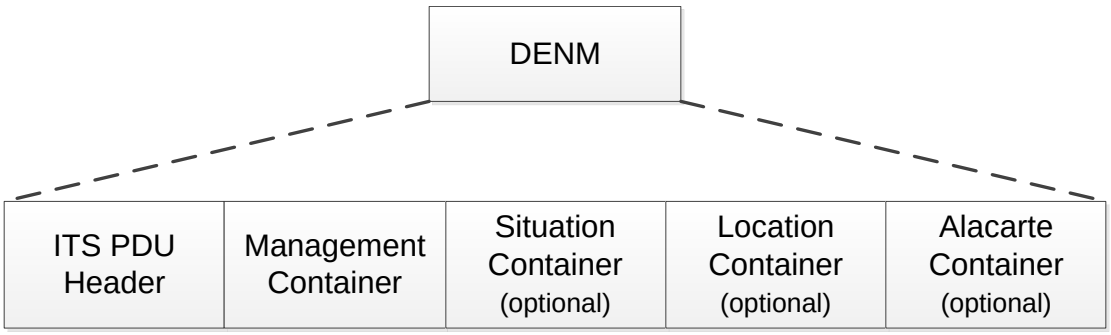


Figure 2.4: General structure of a DENM [47]

The ITS PDU Header and the Management Container have to be included in every DENM, the other ones are optional. But the Situation and Location Container are linked and should always appear together, except a cancellation or negation DENM should have none of them. The usage of the Alacarte Container depends on the application specification.

2.2.4.3 Infrastructural Message Sets

Besides CAM and DENM there are message sets which are closely linked to road infrastructure and is therefore also of big interest for road operators. The standardization work is performed by CEN and ISO in CEN/TC278/WG 16 and ISO/TC204/WG18, respectively. The technical specifications are still in progress, that is why the next lines just give an overview of the status quo. In general the infrastructural message sets are divided in three separate message sets:

SPaT, MAP, SRM and SSM

The Signal Phase and Timing (SPaT), Road Topology/Map Data (MAP), Signal Request Message (SRM) and Signal Status Message (SSM) sets inform road users about road topologies and traffic management information like the status and timing of traffic lights. In addition to a prioritization of special vehicles at specific road segments this should improve the traffic safety and efficiency [49]. A draft version of the specification CEN ISO TS 19091 is evaluated internally in the working group and should be reviewed in Q3/2014.

The SPaT message provides vehicles in an intersection with real-time information of the traffic lights signal phase state (red, yellow, green) and the remaining time before the traffic light changes to the next signal phase state (timing). Furthermore, it contains general operational states of the traffic controller and the right of way for each lane and direction. This just makes sense if the topology and the static physical geometry of an intersection and infrastructure area is known to the vehicles; this is provided with the MAP message. It contains all the lanes for vehicles, public transport system but also pedestrian crossings, approximately an area of 200m shall be covered.
The standardization CEN ISO TS 19091 is concluded with the definitions of Signal Request Message (SRM) and Signal Status Message (SSM), they are necessary to set individual priority and preemption at intersections. More details about the needs for SSM/SRM still have to be defined and investigated in the working groups.

The introduction of these message sets should give road users the opportunity to get through urban areas more efficiently and safely. For the road operators it provides more flexibility in managing the traffic flow and therefore decrease congestion and increase safety. Also public transport will benefit from this technology because it can request priority passages to become a faster and more convenient means of transportation than using the private car.

Probe Vehicle Data (PVD) and Probe Data Management (PDM)

The individual vehicle information which is sent via Car-2-X messages between cars and infrastructure is of crucial interest for road operators. It helps to investigate the current traffic situation, e.g. a traffic congestion can be identified and traffic reports

can be rapidly generated. In return for delivering the vehicle specific data to the road operator the road user gets detailed information of the situation on the road, for example that he is approaching a traffic congestion and gets a suggested alternative route. The standardization of PVD and PDM is still in a preliminary stage, but it is specified for day-one applications that vehicles provide the necessary information by means of CAM and DENM messages [49].
A critical point is when there is no Road Side Unit within the communication range, during this period the data has to be buffered and as soon as a Road Side Unit is within the communication range again the Probe Vehicle Data has to be delivered to the RSU. Furthermore, there is still the open question of privacy: Which information is really necessary for the road operator without invading someone's privacy? Who is responsible for the PVD and who is the owner? These questions have to be answered before the system can be established.

In-Vehicle Information (IVI)

A common problem for road users is the perception of traffic signs and other travel information. With In-Vehicle Information this information is also transmitted directly to the vehicles, where the driver can be informed visually or acoustically. This should increase the awareness of the road user significantly. In combination with a proper relevance check (see chapter 5.6.1) the driver just gets informed about road information which is necessary and applicable for him. In this way the risk of an information overload can be minimized.
In a further step also the road operator should benefit from this technology because the amount of signs along the roads can be reduced and temporary speed limits for example at road works can be applied faster and more easily.

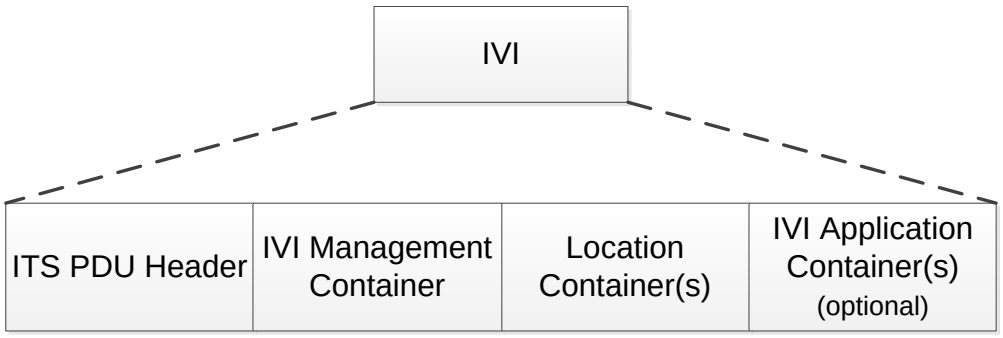


Figure 2.5: General structure of a IVI message [50]

The general structure of an IVI message is shown in Figure 2.5 and will be defined in the CEN/ISO TS 19321 specification. It consists of one mandatory IVI Management Container, one or more Location Container(s) including the essential information where

an IVI is relevant and one or more Application Container(s) [49].

The final draft of the specification is under internal investigation and is evaluated for first deployment support. Since the final version was not published before the end of this thesis the necessary data is transmitted within CAMs. For this purpose the `SpeedLimit` data element in the `SafetyCarContainer` is used, as described in more detail in 4.2.6 and 5.6.2.

CHAPTER 3

Hardware and Software

The main component of a Car-2-X system is the Car-2-X hardware platform. It handles the communication to the vehicle via the vehicle bus and the communication to other ITS-Stations via the ITS-G5 protocol as well as the reception of GPS data. This chapter introduces the platforms used within this thesis but also the software tools to develop embedded ITS applications for this Car-2-X module.

3.1 Hardware Platform

There are several Car-2-X platforms available on the market, but after a careful analysis of several commercially available Car-2-X systems in [14], MAGNA STEYR and Virtual Vehicle decided to use Cohda Wireless platforms. The tested Cohda MK2-platform fulfilled all criteria, like maturity phase, references and standards compliance.
Cohda Wireless [11] is a leading equipment vendor in the C-ITS market, based in Adelaide, Australia. It was founded in 2004 as a spin-off of the University of South Australia. Now even global players like Cisco and NXP Semiconductors are shareholders of Cohda and the objective is to advance Car-2-X communication.
A major milestone of Cohda Wireless and NXP Semiconductors was the development of the first market-ready chipset for Car-2-X communication, the **RoadLINK™ family**. The first two products of this chipset family are the Software-Defined Radio (SDR) baseband processor **SAF5100** and dual radio multi band RF transceiver **TEF5100** [51] [52]. The SDR approach of the SAF5100 gives customers the flexibility to support emerging standards across multiple regions and hence to deploy a global Car-2-X solution by simple firmware updates. It consists of two DSP cores, one ARM9 processor core, dedicated hardware for Viterbi decoding and interleaving as well as ADCs/DACs and filtering modules. The RF part is composed of necessary RF components (Low Noise Amplifier, Power Amplifier, antenna switch,...) and the RF transceiver chip. It fulfills Japanese (760MHz), US and European (5.9GHz) Car-2-X requirements as well as off-the-shelf Wi-

Fi (802.11abgn) and DSRC specifications.

Figure 3.1: Cohda Wireless MK4

The latest generation of Cohda's Car-2-X platforms, the **MK4** (see Figure 3.1), is already based on the RoadLINK^{TM} chipset (see hardware architecture in Figure 3.2). Because of using the integrated solution the size has been reduced by about two-thirds compared to the MK2 platform. Using the RoadLINK^{TM} chipset and Cohda's field proven network layer, facilities layer and applications layer software products, the MK4 should serve as a reference design for series-production of Car-2-X platforms [53].
The next generation (MK5) is currently under development and should be even smaller because of integrating transceiver and SDR baseband processor into one single chip.

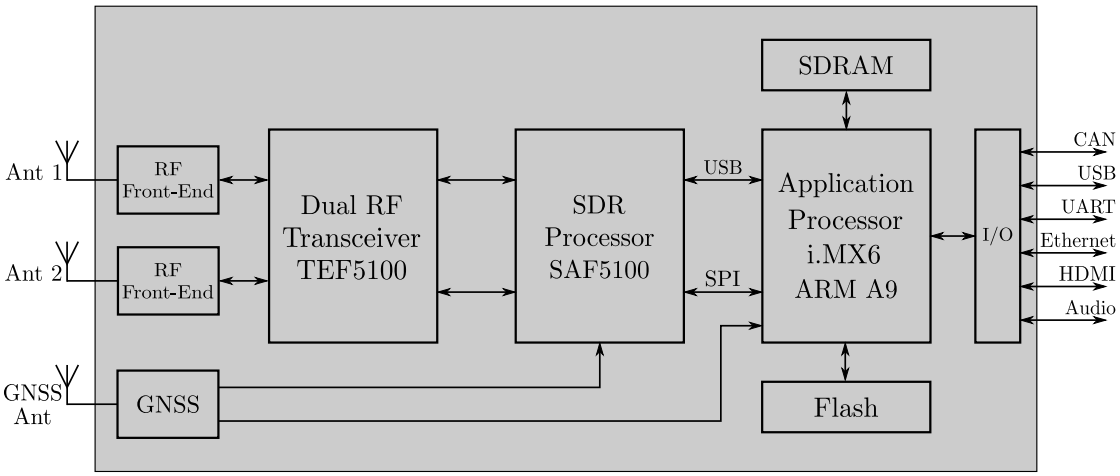


Figure 3.2: MK4 - Hardware architecture

Additionally to the RoadLINK^{TM} processors TEF5100 and SAF5100 an integral part of the 802.11p implementation in the MK4 platform is the 1GHz ARM Cortex-A9 based

application processor **i.MX6 solo** from Freescale Semiconductor [54]. A Linux operating system (3.0.68 Kernel, see Table 3.1) is installed and the ITS application software is running on this processor. It also includes 512MB onboard SDRAM and NAND flash memories, which stores

- Operating system files (including device drivers)
- ITS application software/data
- Calibration data
- Firmware images
- Secure data, such as MAC addresses and serial numbers. The secure data is placed in a read-only partition of the flash memory, which is not mounted by default.

The MK4 has a read/write Unsorted Block Image (UBI) file system mounted at */mnt/rw* to install user applications and provides a microSD card slot to extend the available memory for example to log real-time data. Any files edited and saved in a directory other than */mnt/rw* will just remain in the SDRAM, the changes will be lost after reboot or power off. To ensure that the files get written in the Flash memory it has to be synchronized with the *sync* command after saving the file.

3.1.1 I/O Interfaces

Managing the I/O interfaces of the MK4 platform is also a main task of the application processor (and its ITS application), the following interfaces are supported:

- **Two high-speed CAN interfaces**
  Dynamic vehicle data is received with these interfaces and the ITS application has to response accordingly, like updating the appropriate data elements in the CAM message (e.g. Speed or Status of Lights). The CAN data rate can be set arbitrarily up to 1Mbit/s in a configuration file (MK4: /etc/init/can0.conf; MK2: /opt/cohda/bin/rc.can) (see chapter 5.3). Both 'Base frame format' (CAN 2.0A) and 'Extended frame format' (CAN 2.0B) are supported. The Vehicle Interface Connector (VIC) on the MK4 digital connector side contains the ports for the two CAN interfaces but also three 12V-tolerant general purpose digital inputs to be used for other vehicle sensors.

- **USB interface**
  This high-speed USB 2.0 (480Mbit/s) interface is configured by default to emulate an Ethernet device (Ethernet over USB port - both interfaces are connected to the TCP/IP stack of the Linux kernel). So it is possible to set up a `ssh` connection also via the USB port. In the MK2 and MK4 modules the USB port is a USB On-The-Go port. It allows the module to also act as a host and to be able to plug-in other peripheral equipment, like a WLAN router, Bluetooth dongle, keyboard, etc. When operating as a host this port provides 5V supply voltage with a 500mA current limitation. In the USB Host mode it is not possible anymore to use the

USB port to connect to the PC and a dedicated USB On-The-Go cable/adapter is necessary.

- **UART/RS232 interface**
  The RS232 interface is, besides Ethernet and USB, a third opportunity to connect to the MK4 (115200bps, 8 data bits, no parity, no hardware flow-control) and it can be used to connect an external GPS receiver. But primarily the RS232 port is used for system development and debug operations to identify several problems at an early stage. For example during boot-up status messages are shown in the console and if modules could not be loaded successfully, this can be detected here. During developing the ITS applications it is possible to fill up the user memory `/mnt/rw` (MK4) and `/mnt/ubi` (MK2), respectively. If this happens it is not possible anymore to boot the MKx[1] device via Ethernet or USB. But via the serial port a factory image can be partly booted and to recover the MKx unit, the data which overfilled the memory, can be deleted[2].

- **Ethernet interface**
  Via the Gigabit Ethernet interface (10BASE-T/100BASE-TX/1000BASE-T) a `ssh` or `ftp` connection to the PC (similar to USB interface) or other IPv4 and IPv6 networks can be set up. Furthermore remote debugging of the ITS applications is possible with the SDK. The MKx module is configured to request the IP address from a DHCP server, but also statically setting the IP address is possible. During this work a local DHCP server (haneWIN DHCP server [55]) was established to maintain the IP addresses of the MKx modules.

- **HDMI v1.4 interface**
  The i.MX6 solo application processor has an integrated HDMI v1.4 interface for providing digital video output with a programmable resolution of up to 800x600. It can be used for example to connect a touch-screen as an Human Machine Interface (HMI). Since in this work the visualization is done on a tablet PC, connected via Bluetooth (see chapter 5.6), this interface was not used.

- **Audio line-out**
  The audio interface is used to warn the driver acoustically of an upcoming event. In this thesis the acoustical warning is performed by the vehicle itself (triggered by a CAN message from the MKx module). That is why this audio port was not used.

3.1.2 Antenna and GNSS ports

Two separate antenna ports for the 5.9GHz band are provided. So the MK4 module can operate in *single-channel* and *dual-channel mode* (two independent IEEE 802.11p radios). To improve radio performance, two-antenna diversity transmission and reception

[1] The term MKx is used for the Cohda Car-2-X platforms in general, that means the information is applicable to both, the MK2 and the MK4 platform.

[2] `http://mk2wiki.cohdawireless.com/pmwiki.php?n=Main.MKxRecovery`

is possible. In fact the MAC layer running on the ARM processor of the SAF5100 would allow even more operating modes like *single-radio with time-synchronized channel switching* for multi-channel operation. The MAC also includes radio channel measurements like per-channel statistics (number of successfully transmitted packets,..) or received signal levels. This is necessary for example for the Decentralized Congestion Control (see 2.2.3). The minimum transmit power is -10 dBm and the maximum transmit power is +22 dBm per antenna port. The transmit power can be controlled in 0.5 dB steps.

A precise position detection and timing is crucial for the success of a Car-2-X system. Hence, the MK4 is using an on-board GNSS receiver (support of GPS, Glonass and Galileo) and to assist during poor GNSS radio conditions (e.g. in urban areas with huge buildings) dead-reckoning is included. It makes use of a gyroscope or data provided by the CAN bus (e.g. steering angle sensor, odometer,...). The GNSS receiver of the MK4 provides position updates at rates up to **10 fixes/sec**, a horizontal accuracy of $<$**2m** and a timing accuracy of **25ns** is listed in the MK4 specifications [56]. As already mentioned in 3.1.1 the serial port can be used to connect an external GNSS receiver.
The MKx runs a `gpsd` server (GPS Daemon) in the background. It allows applications to access GPS data from the GPS receiver. It is also possible to access the GPS data from the command line to confirm that the GPS receiver is operating correctly.

3.1.3 Power Supply

The MK4 unit is supplied with 6-36 VDC, so it can be directly connected to an automotive 12 V battery, e.g. via the cigarette lighter. If the power supply voltage drops below 6V the device will shut down.
Especially in combination with the electrically powered Quad from Virtual Vehicle the power consumption is important. The manufacturer states that in single channel operation the MK4 module (same values are in MK2 specification) will draw approximately 560 mA in receive mode and approximately 900 mA during transmission; in dual channel mode approximately 650 mA in receive mode and approximately 1100 mA during transmission (output power: 21 dBm).
During the field tests (3xMKx devices), see chapter 6, a current consumption of the MK2 module, which was also connected to the tablet via Bluetooth, of about **570 mA** was measured (single mode, just one antenna transmitting, maximum output power). Compared to the eQuad battery capacity of 16 Ah this is not negligible, although for example just the lighting system already draws 4 Ampere.

3.2 Software Tools

In this thesis basically two software packages have to be developed, one is the embedded ITS Application running on the Car-2-X hardware platform, the other one is the Android App for the tablet PC. Since the embedded application is implemented in **C** and the

App in **Java** it was barely possible to exploit synergies. Thus also two development environments are necessary. Both are described briefly in this section.

3.2.1 Cohda Wireless Tools

Cohda Wireless is providing a Software Development Kit (SDK) for all its customers to develop embedded software for the MKx modules in C. This SDK is a Virtual Machine (VM) running Ubuntu, with all necessary development tools for the MKx platform already installed and set up. **Eclipse**[3] is used as the Integrated Development Environment (IDE) which allows:

- Development of applications
- Cross-compilation of applications
- Downloading of applications to the MKx platform
- Remote debugging of applications on the MKx platform

The ITS applications are written as Linux application software in C and a framework from Cohda is used as a base for the ITS application covered in this thesis, see more about that in chapter 5.5.
On a regular basis of about one or two months, updates for the SDK and also the MKx firmware are available on the Cohda Customer FTP portal. In this work the SDK update of March 2014 is used, see in the next Table an overview of the version of each Cohda Wireless tool.

Software	Version
VM Operating System	Ubuntu 12.04.4 LTS
Eclipse IDE	3.7.2
MK4 Firmware	4.9.23174 (June 2014)
MK4 Operating System	Linux Kernel 3.0.68
MK2 Firmware	4.8.21611 (March 2014)
MK2 Operating System	Linux Kernel 2.6.28

Table 3.1: Overview Software version - Cohda Wireless tools

3.2.2 Android App Development Platform

The Android App is developed in Java with the development environment **MOTODEV Studio for Android 4.0.0**[4]. It is an free standalone IDE to develop Android Apps with Java. It is based on Eclipse. Also the Android Developer Tools plugin is installed, which is able to sign an Android application with a default debug key. This debug key is generated when running or debugging the application with the MOTODEV Studio (see

[3] `https://www.eclipse.org`
[4] `http://sourceforge.net/projects/motodev.motorola/`

chapter 5.4). With MOTODEV Studio it is easily possible to do remote debugging of the Android App on the tablet PC.
The operating system of the tablet is Android 4.1.2. and the API level of the Android App is 14 (see Table 3.2).

Software	**Version**
IDE	MOTODEV Studio 4.0.0
Android OS	4.1.2
API level	14

Table 3.2: Overview Software version - Android App

CHAPTER 4

Applications

4.1 Overview of Car-2-X Applications and Use Cases

The introduction of traffic lights or seat belts took decades [12] before they became an accepted and absolutely essential technology to increase the road safety. There will also be several challenges until vehicular communication could have a similar widespread acceptance in the population. But this is essential to take the full advantage of this promising technology and to further decrease the number of road accidents.
Complex cooperative safety applications, like cooperative driving [1] or intersection assistants, need a penetration rate [2] of up to 100%. At the market launch of vehicular communication this is an unrealistic scenario. Hence in the beginning the car manufacturers should focus on basic and simple applications which increase the drivers safety also with a low penetration rate of about 5-10% (Basic Warning Services). This also gives the customers enough time to build trust in the benefits of the new technology and will start a transition to more complex systems because of a higher market penetration (*Evolution instead of revolution*).

ETSI has published a Technical Report (TR 102 638 [57]) in 2009 containing applications and use cases for a deployment of Car-2-X which do not need a high penetration rate to provide a significant customer value (Basic Set of Applications). Furthermore an important achievement of the EU project PRE-DRIVE C2X was to set up a list of sixteen use cases which should be prioritized during European Field Operational Tests (FOTs). These results are used in the follow-up project DRIVE C2X [58]. In which seven field tests in Finland, France, Germany, Italy, Netherlands, Spain and Sweden have tested the functionality, suitability for daily use and efficiency of Car-2-X communication in real-life

[1] e.g. an automatic adaption of speed with the vehicle in front or automatic following of a leading human-driven vehicle (platooning).

[2] Percentage of vehicles on the road equipped with Car-2-X systems.

conditions. Probably the most famous one is sim^{TD} [3] in Hesse, Germany. In the project 17 partners from industry and research (e.g. AUDI AG, BMW AG, Daimler AG,...) worked together. The test fleet comprised 120 cars and three motorcycles; over 1,650,000 kilometers were clocked in the vehicles. The results are promising. From an economical point of view a full penetration with sim^{TD} functions in Germany could reduce the number of traffic accidents to save 6.5 billion euros of annual economic costs. Furthermore the increase in road efficiency and reduction of environmental pollution could bring a macroeconomic benefit of 4.9 billion euros [59].
As a result of sim^{TD} it is planned to start the transnational *Cooperative ITS Corridor Rotterdam - Frankfurt/Main - Vienna* project [60] in 2015. The responsible governmental departments of the EU Member States The Netherlands, Germany and Austria have signed on June 10th 2013 a Memorandum of Understanding. It says that a highway route between these three metropolises will be equipped with the necessary technology to start border-crossing deployment of Cooperative ITS. In Austria the Federal Ministry of Transport, Innovation and Technology (bmvit) and ASFINAG are involved [61].
Although the ITS Corridor should include only two cooperative ITS services at the beginning (Road works warning, Probe vehicle data), the already mentioned EU projects and field tests cover much more. Table 4.1 is a compressed overview of ITS applications and use cases covered in the ETSI report and the PRE-DRIVE project but also use cases added in the DRIVE C2X field tests. These use cases are intended to be deployed in the short- to medium-term. Please note that this list is the authors choice and makes no claims of being complete, in fact it is expected that further use cases will be added in the future. The ITS reference architecture suggests the use of three classes of applications: *Road Safety*, *Traffic Efficiency* and *Other Applications* [27]. This grouping will also be used throughout this chapter to assign each use case to one of these classes.

This thesis focuses mainly on *Road Safety*, for further breakdown this class is separated in two Applications:

- **Road Hazard Warning**
  The Road Hazard Warning application is improving the road safety by providing road users with information on road hazard events. The information is transferred with Decentralized Environmental Notification Messages (DENMs) throughout the vehicular network (see chapter 2.2.4.2). Each detected event is characterized by the event type, position, duration, severity (safety impact) and evolution (the event can vary both in time and in space).

- **Co-operative Awareness/Collision Avoidance**
  This application provides the driver with information about surrounding vehicles and situations in its vicinity such as an approaching emergency vehicle or to be aware of vulnerable road user like cyclists or pedestrians. Unlike the Road Hazard Warning it makes use of the Cooperative Awareness Message (CAM), which

[3] sim^{TD} stands for "Sichere Intelligente Mobilität - Testfeld Deutschland" (engl.: Save and Intelligent Mobility - Test Field Germany)

Class (Applications)	**Use case**
Road Safety (Road Hazard Warning, Co-operative Awareness / Collision Avoidance)	Emergency electronic brake lights
	Stationary vehicle warning (accident or car breakdown)
	Road works warning
	Traffic condition warning (e.g. traffic jam ahead)
	Obstacle warning (e.g. pothole or lost cargo)
	Wrong way driving warning
	Signal violation warning (e.g. traffic light or stop sign)
	Approaching emergency vehicle warning
	Slow vehicle warning (e.g. tractor)
	Approaching Motorcycle warning
	Vulnerable road user warning (e.g. pedestrian, cyclist,..)
	Weather warning
	Overtaking vehicle warning
	Lane change assistance
	Co-operative glare reduction
	Intersection collision warning (V2V or I2V)
	Co-operative forward collision warning
Traffic Efficiency (Co-operative Traffic Management)	Decentralized floating car data / Probe Vehicle Data
	In-vehicle signage (speed limits, stop sign,...)
	Green light optimal speed advisory
	Traffic information and recommended itinerary / detour notification
	Electronic toll collection
	Co-operative adaptive cruise control or platooning
Other Applications (Comfort and Entertainment, Vehicle/Service Life Cycle Management)	Point of Interest (POI) notification
	Automatic access control and parking management
	Transparent car leasing and rental
	SOS service
	Map download and update
	Vehicle software provisioning and update
	Stolen vehicle alert
	Remote diagnosis and just in time repair notification
	Insurance and financial services (e.g. after an accident)
	Optimization of fleet management across departments and companies

Table 4.1: Car-2-X use cases (Basic Set of Applications)

is broadcasted periodically by every ITS-Station. Additionally the ITS-Station could also send out complementary DENMs in order to provide the drivers with additional information or to inform vehicles over a longer distance, because CAMs are not forwarded throughout the network and hence have a limited range (see chapter 2.2.4.1).

Every use case has different requirements [62] for example in the use case *Green light optimal speed advisory* or *Traffic information and recommended itinerary* it is necessary that each Road Side Unit has a connection to a central ITS-Station (traffic control center). This is necessary to get information related to the traffic light phase, the local road traffic and recommended itinerary, respectively. Since there is no public access to these kind of data the number of realizable use cases within this thesis is limited. Nevertheless basically all of the 'day-one' use cases suggested by the Amsterdam Group [13], except the ones which do need access to a central ITS-Station, are covered. Table 4.2 gives a short overview of the implemented use cases and the following chapters introduce each of them in more detail regarding the implementation, requirements but also limitations.

Use Case	**Type**	**Message Type**	**Minimum Frequency**	**Maximum Latency**
Emergency electronic brake lights	C2C	CAM/DENM	10 Hz	100 ms
Stationary vehicle warning	C2C	CAM/DENM	10 Hz	100 ms
Road works warning	I2C	DENM	2 Hz	100 ms
Approaching emergency vehicle	C2C	CAM	10 Hz	100 ms
Motorcycle warning	C2C	CAM	2 Hz	100 ms
In-vehicle signage	I2C	CAM	1-10 Hz	100 ms

Table 4.2: Implemented Car-2-X use cases

4.2 Implemented Use Cases

4.2.1 Emergency electronic brake lights

Type: Car-2-Car Communication
Application: Road Hazard Warning

Description

In this use case a vehicle signals its hard braking to its local followers. This should reduce and attenuate rear-end collisions if a vehicle in front brakes suddenly. It is relevant especially in dense driving situations or if the driver does not see the vehicle directly (limited visibility).

Implementation

The vehicle ITS-Station needs access to the vehicle brake status. Two vehicles were available during the tests to read out the brake state from the CAN bus (see chapter 6). Actually one test car would also offer an emergency braking state or ABS state but for demonstration purposes it is easier to just use the brake state.
As soon as the vehicle is braking it sends periodic DENMs (additionally to the mandatory periodic CAMs) to vehicles within communication range. After the braking maneuver the originating ITS-Station terminates the *emergency electronic brake lights* event by sending out a cancellation DENM.
The receivers check if the originating vehicle is driving in front and in the same direction (see Relevance Check in chapter 5.6.1 and Figure 5.13) and a pop-up window appears for five seconds at the tablet PC (see Figure 4.1). Additionally in one test vehicle CAN messages are sent to the instrument cluster. The indicator lamp and the acoustical warning of the Forward Collision Assistant are used to signal the driver a braking vehicle ahead (see Figure 6.3 in chapter 6.2).

Cause code:	**99**	Description:	Dangerous situation
Sub cause code:	**1**	Description:	Emergency electronic brake lights

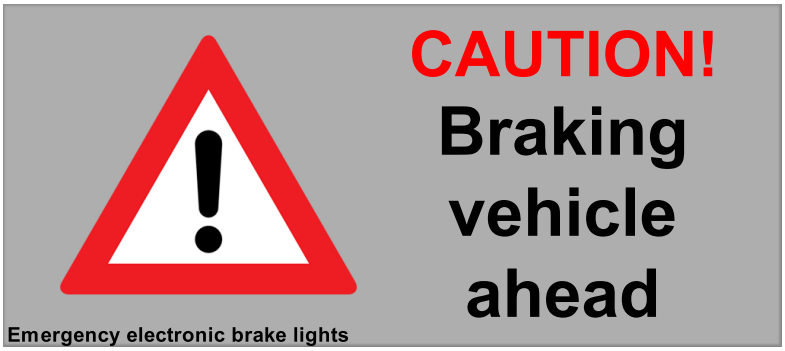


Figure 4.1: Pop-up window - Emergency electronic brake lights

4.2.2 Stationary vehicle warning (accident or car breakdown)

Type: Car-2-Car Communication
Application: Road Hazard Warning

Description

This function warns following road users about an upcoming disabled vehicle (consecutive to an accident, a breakdown or any other reason). The warning gives the driver the opportunity to slow down and avoid succession of collisions but also to protect the occupants of the broken vehicle.

Implementation

Since it is difficult to simulate a car breakdown, the status of the emergency flasher is used as an indicator. There is access to this status bit on the CAN bus of the *eQuad* and a second test vehicle.
The originator ITS-Station sends regular CAMs and as soon as the emergency flasher is activated and the speed of the vehicle is zero a Decentralized Environmental Notification Message is generated and sent to the surrounding vehicles. The generation of DENMs will be stopped when the driver switches off the emergency flashes or continues the journey. In this case the originating ITS-Station sends a cancellation DENM to stop the *stationary vehicle warning* event.
The receiving ITS-Stations check if the stationary vehicle is in front. If it is a pop-up window is activated for five seconds (see Figure 4.2) and also marker on the map indicating the stationary vehicle is presented.

Cause code:	**94**	Description:	Stationary vehicle
Sub cause code:	**2**	Description:	Vehicle breakdown

Figure 4.2: Pop-up window - Stationary vehicle warning

4.2.3 Road works warning

Type:	Infrastructure-2-Car Communication
Application:	Road Hazard Warning

Description

Road Side Units (RSUs) are mounted on road works and send periodic temporary messages (DENMs) to approaching vehicles. The driver gets aware of upcoming road works (e.g. road building, maintenance,..) early enough to reduce the speed and pass the road works safely. The aim is to reduce the likelihood of crashes during road works and to increase the safety of travelers and road workers.

Implementation

The ITS-Station is placed stationary at a road works and sends periodically DENMs to its vicinity. The DENM has to include event position and if necessary a certain lane which is blocked. Furthermore the duration of the road works should be defined and the originating ITS-Station stops sending messages after this amount of time. Receiving vehicles should detect and determine the relevance of the DENM (vehicle within relevance area and driving towards the road works) and if necessary a pop-up window appears for five seconds at the tablet PC (see Figure 4.3). During the validity period of the DENM a road works marker is visible on the Google map, afterwards it disappears automatically.

Cause code:	**3**	Description:	Roadworks
Sub cause code:	**1**	Description:	Major roadworks

Figure 4.3: Pop-up window - Road works warning

4.2.4 Approaching emergency vehicle warning

Type:	Car-2-Car Communication
Application:	Co-operative Awareness

Description

In this use case an emergency vehicle informs surrounding vehicles about its presence. This gives them the opportunity to clear the way at an early stage. The emergency vehicle can reach its target destination faster and also the risk of collisions between an emergency vehicle and other vehicles is reduced.

Implementation

The emergency vehicle sends periodic CAM messages. The receiving ITS-Stations have to process the CAMs and check the appropriate data elements (`StationType` (special vehicle=10), `VehicleRole` (emergeny=6) and `EmergencyContainer`) if it is an emergency vehicle [45]. In addition also a DENM could be used to warn other road users

that an emergency vehicle is approaching [62]. In this implementation no DENMs are sent, since the necessary information is already provided by the CAMs.
The receivers also check from which direction the emergency vehicle is approaching and show a pop-up window with an arrow indicating the direction of the emergency vehicle (see Figure 4.4). This window is visible for five seconds and a marker shows the current position of the emergency vehicle on the map.

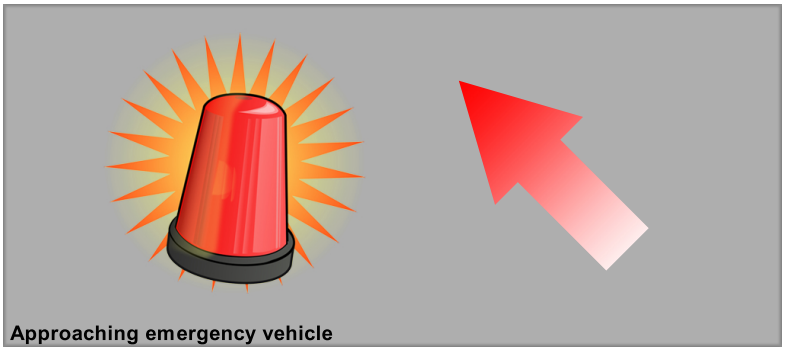


Figure 4.4: Pop-up window - Emergency vehicle approaching from front left

4.2.5 Motorcycle warning

Type:	Car-2-Car Communication
Application:	Co-operative Awareness

Description

Especially in case of reduced visibility a driver could overlook an approaching motorcycle. In many cases this results in a fatal accident or at least a severely injured motorcyclist. Hence in this use case surrounding vehicles should be informed of arriving motorcycles to avoid these kinds of accidents.

Implementation

In fact the procedure is very similar to the *Approaching emergency vehicle* use case. The periodic CAM messages from the motorcycle are processed by the receiving ITS-Stations and determined if the originating station is a motorcycle (`StationType`). Actually it should be used in addition to an accurate collision risk analysis, because otherwise the warning message would appear every time a motorcycle is in the vicinity. So in this implementation just a motorcycle symbol is shown on the map, but without a warning pop-up window. This use case can be extended to other vulnerable road users like pedestrians or cyclists (use case: *Vulnerable road user warning*).

4.2.6 In-vehicle signage (speed limits,...)

Type:	Infrastructure-2-Car Communication
Application:	Co-operative Traffic Management

Description

Road Side Units (RSUs) mounted on traffic signs along the roadway are broadcasting frequently for example a current local speed limits to approaching vehicles. This should increase the driver's safety in case a roadside traffic sign was overlooked. In case of variable sign information (e.g. lower speed limit if visibility is poor,...) the roadside ITS-Station should be connected to a central ITS-Station to get all the necessary information and updates.

Implementation

The roadside unit is mounted on a traffic sign and broadcasts the speed limits periodically to the surrounding vehicles via CAMs. The appropriate data elements in the CAM are `StationType` (RoadSideUnit=15) and `SpeedLimit` in the `SafetyCarContainer` of the `SpecialVehicleContainer`. The receiving ITS-Stations process the CAMs and show a popup window on the tablet for five seconds and warn the driver if he is going too fast (see Figure 4.5). Afterwards the sign will be shown in the Map as long as CAMs are received.
The specification of the In-vehicle signage messages (CEN ISO TS 19321) (see chapter 2.2.4.3) is still under development. That is why in a future step this use case will be extended to other traffic signs, like stop sign or yield sign as soon as the standard is available. In this implementation the workaround with the `SafetyCarContainer` is used because it includes a `SpeedLimit` data element.

Figure 4.5: Pop-up window - In-vehicle signage (driver is going too fast in 30 km/h zone)

CHAPTER 5

Implementation

The aim of this thesis is to develop a full Car-2-X system demonstrator, including the ITS G5 transmission with the corresponding visualization of the Car-2-X data. The key features of the implementation as discussed in the introduction are:

- Integration of an embedded Car-2-X platform into a series vehicle via CAN bus.
- Graphical visualization of Car-2-X relevant information on an Android tablet PC.
- Detection and visualization of all vehicles within communication range equipped with Car-2-X technology.
- Interoperable and expandable implementation by using standard-compliant message types and interfaces.
- Implementation of six by the Amsterdam Group for day one deployment suggested Car-2-X use cases on the embedded platform.
- Compatibility with the electrically driven technology demonstrator *eQuad* of the industrial partner Virtual Vehicle.

Figure 5.1 presents the hardware architecture of the implementation. The Cohda Wireless Car-2-X hardware platform MKx is connected via a Controller Area Network (CAN) interface to the vehicle and reads or writes CAN messages from/to the vehicle CAN bus via the interface (see chapter 5.3). Manufacturer-specific CAN messages like speed, acceleration or yaw-rate are specified in a CAN matrix or a *.dbc-file. In the final tests (see chapter 6) CAN messages are sent to the instrument cluster to activate an indication lamp and an acoustical warning.

A tablet PC is connected via a Bluetooth interface to the MKx platform to visualize Car-2-X data. Since the MKx hardware has no Bluetooth an external Bluetooth dongle is used. The 802.11p transmission is controlled by the ITS application running on

the module. Furthermore a PC can be connected via Ethernet or RS232 (USB port is already occupied by the Bluetooth dongle) to capture debug messages and control the MKx module. This is helpful in the development process for debugging purposes.

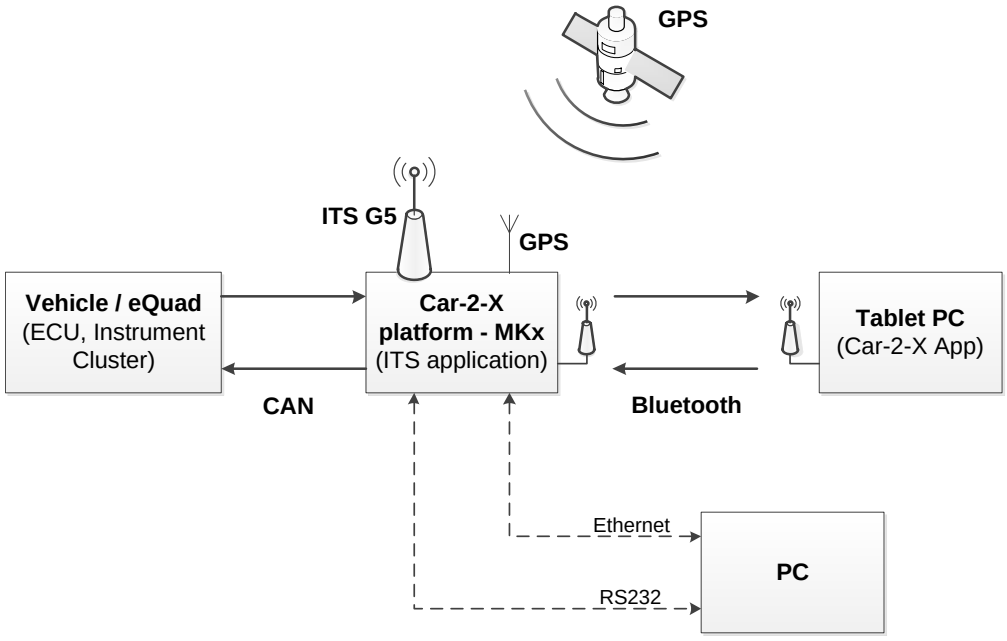


Figure 5.1: Car-2-X demonstrator - Hardware architecture

This chapter covers basically all the practical work done within this thesis. It starts with the configuration and initialization of Bluetooth for the MKx module. Figure 5.2 shows the concept of the Bluetooth connection. The ITS application (see chapter 5.5) forwards CAM and DENM packets to an ASN.1 encoder (see 5.2) to achieve an efficient data transfer between the module and the tablet computer.
The Serial Port Profile, described in 5.1.2, is used for the Bluetooth link. At the tablet computer the received ASN.1 bit stream is decoded and processed in the Android App (see 5.6). An integral part of the App is a Google map. The integration of it with the Google Maps API and its limitation is covered shortly in 5.4.

5.1 Bluetooth

The wireless technology standard *Bluetooth* is used to communicate between the Car-2-X hardware and the tablet. The reasons are that the tablet PC supports Bluetooth, libraries to implement Bluetooth are available for C (ITS Application) as well as Java (Android App) and Cohda Wireless ensured that the Linux operating system of the MKx module supports the Bluetooth stack.

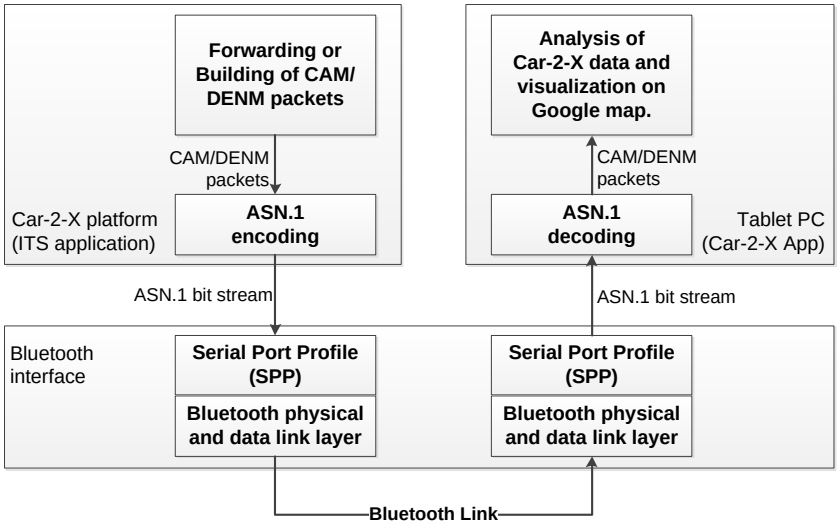


Figure 5.2: Bluetooth connection - Software architecture

As described in 3.1.1 the USB port is a USB On-The-Go port, so the Car-2-X module acts as a host. The tests have shown that it was not that straightforward to use Bluetooth in combination with the MKx modules. Later on also the support of Cohda Wireless confirmed that the USB Bluetooth functionality is not fully guaranteed and the customers have to update the Bluetooth driver and stack by themselves. Retrospectively the effort to get Bluetooth running was higher than expected and maybe it would have been an option to use an alternative communication protocol, like UDP in combination with a Wireless Router. Cohda Wireless has successfully tested the *TP-Link TL-WR702N 150Mbps-Wireless-N-Nano-Router*. This view was encouraged when the new MK4 module was delivered. Cohda Wireless announced that using Bluetooth with the MK4 will be easier compared to the MK2 because the operating system itself handles the USB port and hence also the Bluetooth connection. But in fact it was not possible to set up a Bluetooth connection with the MK4. The reason is that Cohda disabled the Bluetooth interface in the MK4 and some code and configuration files need to be changed, but also after multiple requests no firmware update was published which solves this issue. That is why for the rest of this thesis and also the final tests the MK2 module was used in combination with the tablet for the visualization. That is why the explanations of the Bluetooth stack and necessary adaptations covered in this chapter will mainly focus on the MK2 module.

5.1.1 BlueZ Bluetooth Stack

A very common implementation of the Bluetooth protocol suite in Linux is BlueZ[1], it supports all core Bluetooth layers and protocols [63]. BlueZ is an Open Source project

[1] `http://www.bluez.org`

written in C and licensed under the GNU General Public License (GPL). This Bluetooth stack is part of the official Linux kernel since version 2.4.6. Also the Linux kernel of the MK2 device includes all the necessary protocols to build up a Bluetooth connection between the Car-2-X module and the tablet computer. In case of missing files (the BlueZ tools are located in `/usr/bin`) or to update the BlueZ stack the command is

```
sudo apt-get install bluez python-gobject python-dbus
```

Besides the general BlueZ libraries also *bluez-utils* are useful tools to configure and test the Bluetooth connection (e.g. `hcitool` and `sdptool` which are explained in more detail later on) and because of its extensive APIs it is also easy to use Bluetooth within the ITS application in C (see chapter 5.5).

First of all the Bluetooth stick has to be detected by the MK2 module, for this purpose it is recommended to use a Bluetooth dongle which has indication lamps showing the status of the connection. Before connecting the Bluetooth stick with an USB On-The-Go cable the USB port of the MK2 module is configured in Host mode. An explanation on how to change the USB mode is available in the Cohda Wireless MKx Wiki [64].
The Bluetooth PIN is saved in `/usr/etc/bluetooth/pin` (read-only) and can be changed in `rc.bluetooth`[2] (*BT_LOCAL_PIN_DEFAULT*) or with `fw_setenv bt_local_pin "XXXX"`, per default it is 0000. Entering `rc.usb bluetooth` in the Terminal loads necessary Linux kernel modules like `bluetooth.ko`, `btusb.ko`, `l2cap.ko` and `rfcomm.ko`. Furthermore the Bluetooth Host Controller Interface Daemon (HCID) is started, basically it is responsible for managing the Bluetooth devices [65].

At this point the Bluetooth device should be already recognized, to check that the `hciconfig -a` command is used. The output looks similar to:

```
hci0:  Type:  USB
       BD Address:  00:00:00:00:00:00 ACL MTU: 0:0 SCO MTU: 0:0
       DOWN
       RX bytes:0 acl:0 sco:0 events:0 errors:0
       TX bytes:0 acl:0 sco:0 commands:0 errors:0
       ...
```

The Bluetooth address (BD Address)[3] is still not valid, for this purpose the command

```
hciconfig hci0 up
```

[2] Useful runlevel configuration files (rc), like `rc.cohda`, `rc.usb`, `rc.bluetooth`, `rc.can` or `rc.can` are saved under `/opt/cohda/bin`.

[3] The *Bluetooth address* or *device address* is a globally unique 48-bit address, comparable to the MAC address of Ethernet. It is assigned at manufacture time and is unique and remains static for the lifetime of the device. It is important to remember this address since it is necessary to tell the Android application with which of the available Bluetooth devices the tablet has to be connected.

opens and initializes the *hci0* HCI device, if it was successful `hciconfig -a` shows a valid BD Address, e.g.:

```
hci0:  Type:  USB
       BD Address:  00:09:DD:60:F1:91 ACL MTU: 310:10 SCO MTU: 64:8
       UP RUNNING PSCAN ISCAN
       RX bytes:718 acl:0 sco:0 events:26 errors:0
       TX bytes:348 acl:0 sco:0 commands:20 errors:0
       Features:  0xff 0xff 0x8f 0xfe 0x9b 0xf9 0x00 0x80
       Packet Type:  DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3
       Link policy:  RSWITCH HOLD SNIFF PARK
       Link mode:  SLAVE ACCEPT
       Name:  'Cohda_MK2'
       Class:  0x100100
       Service Classes:
       Device Class:  Computer, Uncategorized
       HCI Ver:  2.0 (0x3) HCI Rev:  0xc5c LMP Ver:  2.0 (0x3) LMP
       Subver:  0xc5c
       Manufacturer:  Cambridge Silicon Radio (10)
```

The configuration is set in `/etc/bluetooth/hcid.conf`, like the name of the device and if an authentication and encryption procedure is used. Now the Bluetooth initialization procedure is finished and with `hcitool scan` it is possible to search for Bluetooth devices within reach.

5.1.2 Serial Port Profile (SPP)

Before setting up a connection it must be determined which Bluetooth profile should be used and the MK2 module has to support this profile. A simple profile is the Serial Port Profile, it basically defines how to set up virtual serial ports and is the basis for several more complex Bluetooth profiles. SPP is introduced to replace previous wired RS-232 serial communications with Bluetooth technology. Base of this profile is the *RFCOMM* protocol, which is a set of transport protocols for serial port emulation. RFCOMM provides a simple reliable data stream to the other device, similar to TCP. If a packet cannot be delivered within a fixed time limit the connection is terminated and an error is returned. Although it was designed to emulate RS-232 it is now useable in many of the same scenarios as TCP. A typical application is a point-to-point connection over which the subscribers can reliably exchange streams of data [66], like the desired connection between the MK2 module and the tablet computer. Due to the analogy between TCP and RFCOMM, Bluetooth and Internet programming are close connected and basically the same socket programming techniques like in TCP/IP programming are used, see [67] [68].

If two Bluetooth devices intend to communicate with each other, first it has to be

determined which Bluetooth profiles are supported by both devices. Every Bluetooth device typically runs a Bluetooth Service Discovery Protocol (SDP) server that answers queries from other Bluetooth devices to discover what services each subscriber supports and what parameters have to be used to connect to them. Each service is identified by a 128-bit Universally Unique Identifier (UUID), if it is an official Bluetooth profile it is a 16-bit UUID (e.g. SPP: 0x1101). In BlueZ, the implementation of the SDP server/daemon is called *sdpd*, with the identically named command `sdpd`. It can be started and as a background process it handles all incoming SDP search requests. The Utility `sdptool` provides the interface for performing SDP queries on Bluetooth devices, so for example with the command `sdptool browse local` it is possible to discover all Bluetooth services the MK2 module currently supports, instead of `local` also the Bluetooth address of another device can be entered to get its available profiles. In the factory settings a message like

```
Failed to connect to SDP server on FF:FF:FF:00:00:00
```

will appear, this means that currently no Bluetooth profile is available on the MK2 device. For this purpose it is possible to add a Bluetooth profile manually, e.g. to add the Serial Port Profile enter

```
sdptool add -channel=1 SP
```

In this example the RFCOMM channel 1 is reserved for the SPP, in total RFCOMM allows up to 30 channels, which is a significant difference to TCP, where 65535 open ports on a single machine are supported. Checking again with the `sdptool browse local` command the SP profile is now listed, including the service name but also protocols involved in this Bluetooth profile (e.g. L2CAP and RFCOMM):

```
Service Name:  Serial Port
Service Description:  COM Port
Service RecHandle:  0x10000
Service Class ID List:
 "Serial Port" (0x1101)
Protocol Descriptor List:
 "L2CAP" (0x0100
 "RFCOMM" (0x0003)
 Channel:  1
...
```

The Bluetooth device is now initialized and configured to use for example `rfcomm listen` and `rfcomm connect` to set up a Bluetooth connection via the Linux terminal software *minicom* and the Windows software *HTerm*. After adding the necessary library `libbluetooth.a`, saved in `/mk2/tools/ltib/rootfs/usr/lib`, to the *Makefile* of the ETS-Shell project (see chapter 5.5) the sending of data between the MK2

module and the tablet computer within the C application, as described in [67] [68], is possible.

5.1.3 Initialize Bluetooth at boot-up

If field tests are performed, usually not every MKx module is connected to a PC to call all the commands mentioned in this chapter step-by-step. Hence, the Bluetooth port has to be initialized automatically at boot-up. In order to do this the system boot script `/opt/cohda/bin/rc.cohda` for MK2 and `/opt/cohda/bin/rc.mk4` for MK4 can be configured. To initialize Bluetooth for the MK2 at start up the following lines are added to `rc.cohda`:

```
rc.usb android
rc.usb bluetooth
hciconfig hci0 up
sdpd
sdptool add -channel=1 SP
```

After saving the file it is necessary to run the `sync` command otherwise changes are not written in the Flash memory and remain in the SDRAM, after reboot or power off the changes would be lost.

5.2 Abstract Syntax Notation One (ASN.1)

The purpose of the tablet is to visualize important Car-2-X data and warn the driver about dangerous upcoming situations. For example a vehicle in front is braking and sends an *Emergency electronic brake lights* DENM to its following vehicle, information like that have to be transferred immediately and with a low latency (*spontaneous data transfer*). On the other hand the tablet gets necessary information from the own vehicle (e.g. speed, position, heading,...) but also from surrounding vehicles. Because the demonstrator tracks the position of other vehicles to see which cars in the communication range do have a Car-2-X system. In general it is sufficient to send these kind of data in fixed intervals (*periodic data transfer*).

In the Android application which controls the eQuad of Virtual Vehicle, the tablet requests data from it every 100ms, this is fine for the vehicle-specific data but is of course not sufficient for the spontaneous data.
Furthermore the data was sent with a parser designed specifically for the eQuad, so depending on the used vehicle it has to be programmed a special parser, which is not useful for a re-usable Car-2-X system. A main requirement for the data format of the Bluetooth connection between the MKx module and the tablet PC is, that it is standardized and independent of the vehicle.
A format which has its advantages is XML, since it is both machine-readable and human-readable. Because of that it is easy to extend, but still every vehicle would have its own

XML-File, so it is still not vehicle-independent. But in this case the bigger issue is the enormous overhead of XML-Files. A XML-File, containing at least the most necessary data to establish a Car-2-X system, is bigger than 1kByte. The Bluetooth dongle in use supports Bluetooth 2.0 and thus a maximum data rate of 2.1Mbit/s, but in combination with the Serial Port Profile this will not be reached. Expecting a maximum data rate of about 1Mbit/s and vehicles sending CAM messages with 10Hz the channel will be overloaded already in a slightly dense traffic situation and an efficient data transfer between the Car-2-X hardware and the tablet is not possible. This was an exclusion criterion for XML.

As already discussed in 2.2.4.1 the ETSI ITS standard presents the message types in ASN.1, which is a formal notation used for describing complex data structures conveyed across a communication medium, regardless of language implementation and physical representation of these data. It is widely used in the specification of communication protocols. The data structures are defined in ASN.1 files which are compiled by a *ASN.1 Compiler* to produce a set of target language (e.g. C, C++, Java) files which contain the native type definitions for these abstractly specified structures. Also encoder/decoder functions to convert the parametrized data structures into/from a bit stream are generated. This compiling process has to be done in the development phase every time the ASN.1 file changes, on the other hand the encoder/decoder is working during run-time. Each time the ASN.1 defined data structures have to be transmitted over a network these encoding/decoding functions have to be called (see Figure 5.3). The ASN.1 standard supports several ways how to encode the ASN.1 data structures for the transmission (*Encoding Rules*), the most widely used ones are Basic Encoding Rules (BER), Canonical Encoding Rules (CER), Distinguished Encoding Rules (DER), Packed Encoding Rules (PER) and XML Encoding Rules (XER), where CER and DER are subsets of BER. Especially PER [46] is useful for applications with limited bandwidth, that is why it is used in the ETSI ITS standard.

The ASN.1 specification of a Cooperative Awareness Message (CAM) is defined in the ETSI standard EN 302 637-2 [45], the ASN.1 specification of a Decentralized Environmental Notification Message (DENM) in EN 302 637-3 [47] and the ASN.1 module ITS-Container[4] in TS 102 894-2 [70]. That is why in general every Car-2-X system has to understand and work with ASN.1. The Cohda framework uses the open source ASN.1 Compiler *asn1c*[5] to convert ASN.1 specifications into C source code. It supports all of the above mentioned Encoding Rules. The ASN.1 definitions of CAM, DENM and the ITS-Container are merged together in the file `ETS.asn1` stored under `~/mk2/stack/apps/ets-shell/src/asn/`.

[4]The **ITS-Container** module contains the ITS data dictionary with a list of ASN.1 data type specifications of data elements (DE) and data frames (DF) which are used within the CAM and DENM construction (e.g. StationID, Longitude, Latitude, CauseCode,...).

[5]http://lionet.info/asn1c/compiler.html

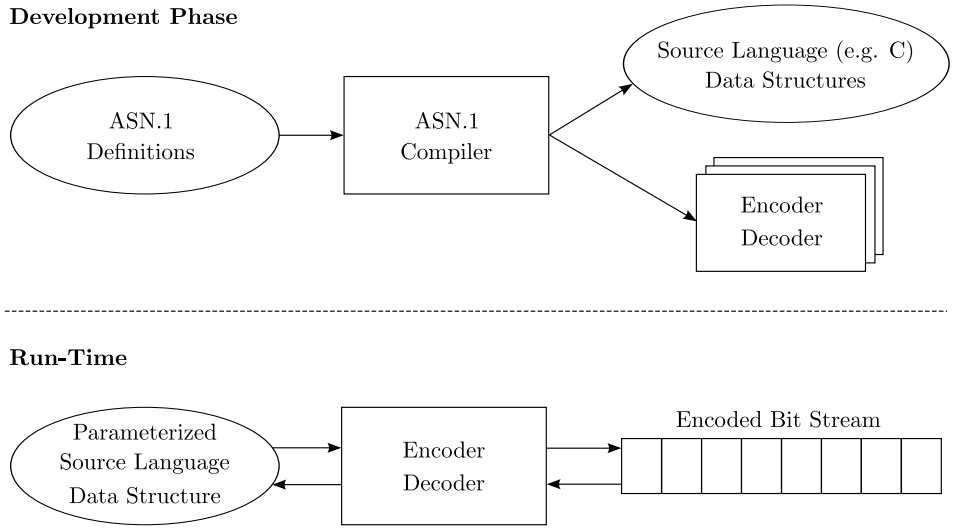


Figure 5.3: Usage of ASN.1 Compiler and Encoder/Decoder [69]

Since all the necessary tools to use ASN.1 are already implemented in the MKx modules it is also interesting to use it for the Bluetooth communication with the tablet PC. The first idea is to generate an asn.1 file which contains all the essential data for the Android application, but also in this case the module needs to have this vehicle-dependent file. Optimally no additional files and parsers should be necessary on the Car-2-X system and the already installed tools should be sufficient. A decision guidance was that Cohda did not implement a Local Dynamic Map (LDM) [48] in the Car-2-X framework and does not plan to do that in the near future. So the option was to simply forward all Car-2-X messages not just over the 802.11p channel but also over Bluetooth and build up a LDM-like database directly in the Android App (see chapter 5.6). At this point it is worth to mention that this Java database is not standard-compliant with ETSI EN 302 895, but basically fulfills the same requirements, like maintaining useful information for active ITS applications, e.g. attributes of surrounding vehicles and DENM events.

The usage of ASN.1 for the Bluetooth communication was therefore already implemented but the free asn1c ASN.1 compiler does not support Java as target language, but this is necessary to use it within the Android application. During investigation of available Java ASN.1 Compiler it turned out that it is not that common to use Java in combination with ASN.1 and because of that the number of providers is limited:

- **unigone** `http://www.unigone.com/`
  A month-long trial version was investigated and all the ASN.1 features are supported. The ASN.1 files provided by the ETSI ITS standard could be directly converted to the Java data structures. But a full license costs 7500 € and because of that this Java ASN.1 Compiler is no option.

- **BinaryNotes** `http://www.latestbit.com/binarynotes`
  BinaryNotes is open source but does not support AUTOMATIC tagging and the project has been suspended.

- **PowerASN** `http://powerasn.ncottin.net/javaPowerASN.php`
  According to the webpage the ASN.1 Compiler is not ready so far and because there was no update since 2007 the project is probably stopped.

- **ASNlab** `http://www.asnlab.com/asnjc/overview.html`
  The amount of limitations of this ASN.1 Java Compiler excluded this tool.

- **ASN.1 Compiler** `http://www.asn1c.com/asn1c/products.html`
  According to the website all the necessary ASN.1 tools are supported and a University license was requested several times, but with no response. Anyway, the University license would just allow internal use of the Compiler and runtime libraries within the university network.

- **openMUC jASN1** `http://www.openmuc.org/index.php?id=60`
  jASN1 is a free Java ASN.1 tool developed at the Fraunhofer Institute for Solar Energy Systems in Freiburg, Germany. The asn.1 file could not be directly converted to the Java data structures and also just Basic Encoding Rules (BER) are supported.

Every free Java ASN.1 Compiler has its limitations, but all together the Fraunhofer jASN1 tools are the best compromise. Since it was not possible to directly convert the ETSI asn.1 files to the associated Java classes the asn.1 files have to be adapted to overcome each of the limitations. The following list shows all the not supported ASN.1 commands and a workaround to make it compatible with the Fraunhofer ASN.1 tools.

- IMPORTS
  IMPORTS are used to import ASN.1 elements from other ASN.1 modules. This is not supported, so each module has to be saved in a separate asn.1 file and all the elements which are imported from other modules have to be copied in each file. In the case of CAM and DENM both ASN.1 definitions have to be saved in a separate asn.1 file (`CAM.asn1` and `DENM.asn1` in Figure 5.4) and the necessary elements of the ITS-Container have to copied in both files.

- Type constraints
  With constraints it is possible to set a lower and upper bound of allowed range of values. For example the following ASN.1 specification defines a Temperature element which is limited to -60 to 67°C.

  ```
  Temperature ::= INTEGER {oneDegreeCelsius(1)} (-60..67)
  ```

  These constraints are not supported in jASN1 and have to be deleted in the asn.1 file.

- AUTOMATIC Tagging
  The structured types SEQUENCE and CHOICE consist of a list of simple types (e.g. a list of INTEGER elements) and to avoid ambiguous descriptions each of them has to be tagged. With AUTOMATIC Tagging this is performed automatically by the ASN.1 Compiler, but this is not supported by the Fraunhofer Compiler. That is why this has to be done manually with the IMPLICIT keyword. The tagging begins with 0 from top to bottom. The next lines show the ASN.1 element `CoopAwareness` first in the AUTOMATIC Tagging version afterwards in the IMPLICIT Tagging version which is understood by the Fraunhofer jASN1 Compiler.

```
CoopAwareness ::= SEQUENCE {
 generationDeltaTime GenerationDeltaTime,
 camParameters CamParameters
}
```

```
CoopAwareness ::= SEQUENCE {
 generationDeltaTime [0] IMPLICIT GenerationDeltaTime,
 camParameters [1] IMPLICIT CamParameters
}
```

- CHOICE
  In general the structured ASN.1 type CHOICE is supported, but there is an undefined problem with the Identifier of CHOICE elements. A simple workaround is to change the CHOICE elements to SEQUENCE and set its entries as OPTIONAL. So that SEQUENCE and CHOICE are identical, the programmer has to make sure that just one entry is used during runtime. Please see a snippet of the `SpecialVehicleContainer` element definition and how it has to be changed to be compatible with jASN1.

```
SpecialVehicleContainer ::= CHOICE {
 publicTransportContainer PublicTransportContainer,
 specialTransportContainer SpecialTransportContainer,
...
```

```
SpecialVehicleContainer ::= SEQUENCE {
 publicTransportContainer PublicTransportContainer OPTIONAL,
 specialTransportContainer SpecialTransportContainer OPTIONAL,
...
```

As already mentioned a disadvantage of the Fraunhofer jASN1 Compiler is that it includes only Basic Encoding Rules (BER), but the ETSI ITS standard is using Unaligned Packed Encoding Rules (PER) which is more efficient and achieves a more compact bit stream than BER. One reason is that it uses the lower and upper limits of the Constraints parameter to represent the data using a minimum number of bits, on the other hand the

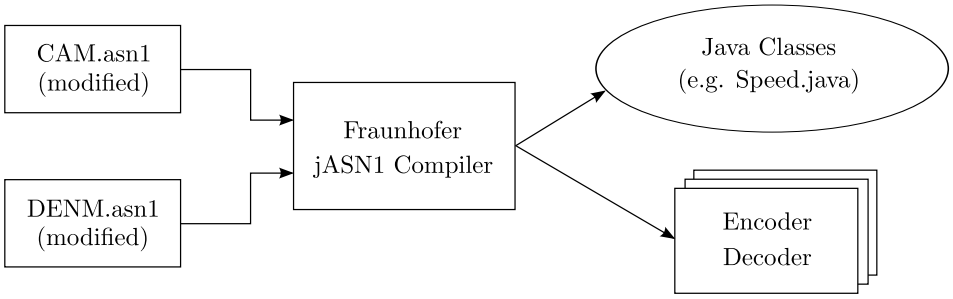


Figure 5.4: Compile process of (modified) CAM/DENM asn.1 files with jASN1 Compiler

BER ignore the constraints and always use the maximum amount of bits, e.g. three bytes are necessary to encode a simple Boolean data element. A comparison of BER and PER has shown that the payload of a CAM message is **52 Bytes** with PER and **175 Byte** with BER, respectively. But in comparison to XML (∼**4.5kB**) BER is still much more efficient.

5.3 Controller Area Network (CAN)

In 3.1.1 it was discussed that the MKx modules have two integrated high-speed CAN interfaces (up to 1Mbit/s) to connect directly to a vehicle's CAN bus and receive dynamic vehicle data. The Cohda framework uses SocketCAN [71], an open source implementation of CAN protocols for Linux and designed by Volkswagen Group Electronic Research. Similar to the Bluez Bluetooth stack (see 5.1.1) SocketCAN uses the Berkeley socket API (or BSD sockets)[6], the standard Linux network stack. The communication with CAN is therefore very similar to the TCP/IP implementation and this allow programmers, familiar with network programming, to easily use CAN within a C application. The big advantage of SocketCAN compared to other CAN implementations for Linux is that it allows multiple applications to access one CAN device simultaneously and a single application is able to access multiple CAN networks in parallel. Very helpful command line tools from the SocketCAN *can-utils*[7] to test a CAN interface are `candump` which shows all messages being exchanged on the CAN bus and `cansend` to send CAN frames.

5.3.1 Change CAN bit rate

Depending on the vehicle CAN bus different data rates are in use, in the automotive industry 500kbit/s and 125kbit/s are very common, but the eQuad uses the maximum data rate of 1Mbit/s. The CAN interfaces of the MKx modules can be set arbitrarily up to 1Mbit/s in a configuration file. On the MK4 the appropriate files for the two CAN interfaces are `can0.conf` and `can1.conf` in `/etc/init/`, a CAN rate of 500kbit/s can be set with

[6] http://en.wikipedia.org/wiki/Berkeley_sockets
[7] https://gitorious.org/linux-can/can-utils

```
env BITRATE="500000"
```

On the MK2 module it is more complicated to set the CAN bit rate. According to the CAN specification the bit time is divided into four segments: the synchronization segment (fixed to one time quantum), the propagation segment, the phase buffer segment 1 and the phase buffer segment 2. The length of each of these segments[8] influences the resulting CAN bit rate. The appropriate parameters can be set in `/sys/devices /platform/FlexCAN.x/` or via the configuration file `/opt/cohda/bin/rc.can`, whereby the latter is recommended. The necessary parameters and also the method to calculate the resulting CAN rate are shortly described here, for further details please refer to Chapter 28 of [72] and Chapter 24 of [73]:

- br_clksrc
  This parameter selects the clock source to the CAN protocol interface (CPI). It is set to `bus` which means that the CAN engine clock source is the bus clock ($f_{CANCLK} = 66.5\ MHz$).
- br_presdiv (Prescaler Value)
  Defines the ratio between the CPI clock frequency and the Serial Clock (SCLK) frequency, which defines the time quantum of the CAN protocol: 1 Time Quantum = 1 SCLK period.
  $$f_{SCLK} = f_{Tq} = \frac{f_{CANCLK}}{Prescaler\ Value} \tag{5.1}$$
- br_propseg
  Configures the length of the propagation segment. Valid values are 1-8.
- br_pseg1
  Configures the length of phase buffer segment 1. Valid values are 1-8.
- br_pseg2
  Configures the length of phase buffer segment 2. Valid values are 2-8.

The CAN bit rate is defined as

$$Bit\ Rate = \frac{f_{Tq}}{Number\ of\ Time\ Quanta\ (\#Tq)} \tag{5.2}$$

The Number of Time Quanta is defined by the length of the four segments:

$$\#Tq = 1 + br\_propseg + br\_pseg1 + br\_pseg2 \tag{5.3}$$

The parameters now have to be chosen to reach the desired CAN bit rate. But not every combination is valid, for the CAN standard compatible time segment settings see Figure

[8] CAN uses the term *time quantum Tq* as the basic time unit. Each segment consists of a specific, programmable number of time quanta. The sum of all time quanta results in the bit time.

24-19 and Table 24-20 in [73]. Since a CAN bit rate of 500kbit/s and 1Mbit/s are the most important ones in this thesis the used combinations are listed here:

500 kbit/s:

$br\_presdiv = 7$
$br\_propseg = 5$
$br\_pseg1 = 5$
$br\_pseg2 = 8$

$$\#Tq = 1 + br\_propseg + br\_pseg1 + br\_pseg2 = 19 \tag{5.4}$$

$$f_{Tq} = \frac{f_{CANCLK}}{Prescaler\ Value} = \frac{66.5\ MHz}{7} = 9.5\ MHz \tag{5.5}$$

$$Bit\ Rate = \frac{f_{Tq}}{Number\ of\ Time\ Quanta\ (\#Tq)} = \frac{9.5\ MHz}{19} = 500\ kbit/s \tag{5.6}$$

1 Mbit/s:

$br\_presdiv = 6$
$br\_propseg = 3$
$br\_pseg1 = 4$
$br\_pseg2 = 3$

$$\#Tq = 1 + br\_propseg + br\_pseg1 + br\_pseg2 = 11 \tag{5.7}$$

$$f_{Tq} = \frac{f_{CANCLK}}{Prescaler\ Value} = \frac{66.5\ MHz}{6} = 11.0833\ MHz \tag{5.8}$$

$$Bit\ Rate = \frac{11.0833\ MHz}{11} = 1.007576\ Mbit/s \tag{5.9}$$

A configuration to get a CAN rate of exactly 1 Mbit/s was not possible, but 1.007576 Mbit/s was within the tolerance to receive all CAN packets on the CAN bus of the eQuad.

5.4 Google Maps

In the future probably the already existing HMIs and navigation systems of a car will be extended with a functionality to visualize Car-2-X-relevant data and warnings. The tablet PC, which acts as HMI in this Car-2-X demonstrator system, has to include the basic capability of such a system and an integral part is a map to mark the spots where a specific Car-2-X event occurred. For this purpose Google provides a *Google Maps Android API v2* [74] to embed Google maps into an Android App for free. Just a Google account, a Google API-Key and the Google Play Services SDK[9] are necessary. Google

[9] `https://developer.android.com/google/play-services`

Maps API requests are sent directly to Google from the Android device and for the verification a SHA-1 certificate fingerprint is used. In fact this fingerprint is required to receive the Google API key from the Google APIs Console[10] and this key has to be added to the `AndroidManifest.xml` of the Android project. From there, the Google Maps API reads the key value and passes it to the Google Maps server, which confirms that the App has access to Google Maps data.
Furthermore the SHA-1 fingerprint is needed to sign an Android application to be able to publish it on an App marketplace such as Google Play. Because so far it is not intended to open the Car-2-X App to the public, the default debug key was used to automatically sign the Android Application Package (APK). The debug key is generated when running or debugging the application by using Eclipse with the Android Developer Tools (ADT) plugin installed (e.g. the used IDE MOTODEV Studio for Android). Self-signed certificates have an expiration date of 365 days from its creation date.

The used tablet PC has no contract for data services, so no internet connection is possible. That is why a smartphone is used as a mobile hotspot to receive Google Maps data. But the tablet also buffers the Google Maps data, so in case of no hotspot at least the last presented section of the map can be shown.

5.4.1 Limitations of Google Maps

A big disadvantage of the Google Maps API is that it is not possible to get street names and information about the course of the road out of the map. Although Google provides the *Google Geocoding API*[11] to convert geographic coordinates into a human-readable address it has the limitation of 2,500 hits per day, which is exceeded quite fast by the App if it continuously requests street information from the Google Maps server moreover it is highly resource demanding.
The lack of free geographical data limits the performance of a *Relevance Check* of the Car-2-X messages significantly. As an example an *Emergency electronic brake lights* DENM should just result in a warning to the driver if a vehicle in front is braking, in general it does not affect the driver if a vehicle on the other lane is braking. But in the case that the DENM-sending vehicle in front hast just passed a sharp curve, a comparison of the heading parameter and ignoring the course of the road would predicate that the vehicle is driving in an opposite direction and is therefore not relevant, which is definitely a misinterpretation (see chapter 5.6.1). A remedy could provide Open Data map providers, like OpenStreetMap[12] or basemap.at which offer free usable geographical data. For lack of time they could not be investigated during this thesis and just Google Maps was used (see outlook in chapter 7.1).

[10] `https://code.google.com/apis/console`
[11] `https://developers.google.com/maps/documentation/geocoding`
[12] `http://www.openstreetmap.org`

5.5 ITS-Application on MKx Module

The C-ITS application running on the MKx module has the following main tasks

- Managing CAN, GPS and Bluetooth interface.
- Managing the CAM and DENM transmission and reception via ITS-G5.
- Forwarding received DENMs and CAMs to the tablet PC via Bluetooth.
- Logging of transmitted and received Car-2-X data.

It is programmed in *C* with the IDE Eclipse 3.7.2 (Indigo). Cohda Wireless provides a Framework for ITS applications using the European ETSI ITS standard [75], which was also used as a base for this thesis. Basically it consists of two packages:

- **ETS App (ETSA)**
  The `etsa` application implements the ETSI ITS protocol stack and is provided as an executable file, which is already installed on the MKx devices, from Cohda Wireless. It is the interface between the ITS application and the ITS-G5 Transceiver. Since the source code of the ETS App is not open the only way to configure it is via a configuration file[13]. As an example the radio configuration, like the used ITS-G5 channel number (`ItsG5CchChanNum`), can be set.

- **ETS Shell**
  The `ets-shell` application is the actual ITS application. The source code of this software package is available and has to be adapted to achieve the desired Car-2-X functionality. The path of the Eclipse project is `home/duser/mkx/stack/apps/ets-shell`. The `ets-shell` application has to be built (`Make`) for the target architecture on which it is intended to run (MK2: ARM11, MK4: ARM Cortex-A9, Virtual Machine: x86).

The `etsa` and `ets-shell` application use the Basic Transport Protocol (BTP) to communicate with each other (see Figure 5.5). This interface is part of the Networking & Transport layer (see chapter 2.2) and provides an end-to-end, connection-less transport service specifically for VANETs [14] [76]. The BTP destination port that the ETS App opens for the ETS Shell (`ItsBtpShellDestPort`) has to be identical for both applications (see [77]).
The GPS information is managed by the GPS Daemon (GPSD), which is a background process that handles data from a GPS receiver and provides the data for other applications. During the development process and if no GPS signal is available the `gpsfake` function can be used to stream GPS data from a before recorded log-file, an example is saved in `~/x86/NMEA.txt`. The CAN and Bluetooth interface can be used from the `ets-shell` application via the sockets as described in 5.1 and 5.3, respectively.

[13] Default etsa configuration files are saved under `~/mk2/stack/tools/configs/etsa` in the VM or under `/opt/cohda/bin/test` on the MKx module. Cohda Wireless has to be contacted for a full list of all the available configuration parameters.

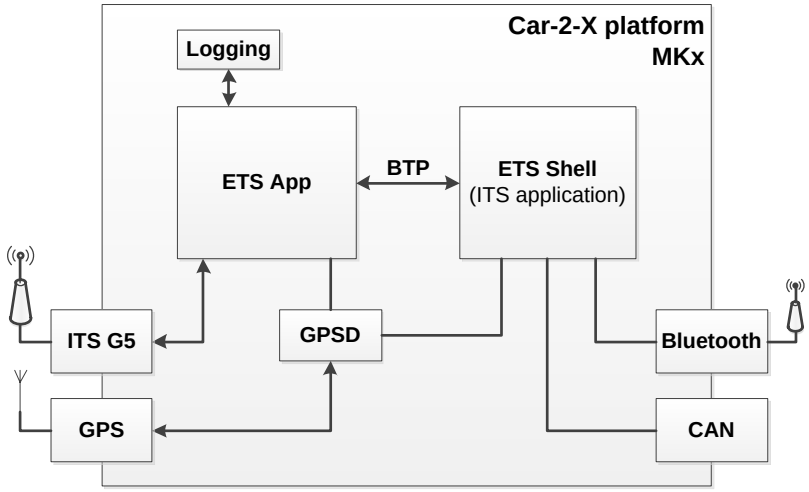


Figure 5.5: ETSA and ETS-Shell System architecture

An important fact is, that both, the ETS App and the ETS Shell, implement the Facilities layer for example to build CAM and DENM messages, so in fact also the ETS App could be used as a standalone ITS application, but because of the limitation with the configuration file this is not flexible enough to implement the use cases of this thesis, that is why the ETS App Facilities layer is disabled (`ItsFacilitiesCAMDENMEnabled`) and the ETS Shell Facilities layer is enabled (`ItsFacilitiesShellEnabled`) in the `etsa` configuration file so that only the `ets-shell` Facilities layer messages are sent.

5.5.1 Structure and Flow Diagram

Since the Cohda Wireless Framework *ETS Shell* is used as a base, also the naming of the files and functions were retained unchanged. But of course files (e.g. `if-bluetooth.c`) and functionality was added. The main program is in `ets-shell.c` and the corresponding header file `ets-shell.h` contains the `TargetStation` enumeration, which defines the type of the ITS-Station the application has to be compiled for, because the ITS application is used in several configurations, as a Road Side Unit, On Board Unit and also in combination with different vehicles (eQuad, test vehicle). Furthermore the `BT_AVAILABLE` flag marks if the Car-2-X hardware is used with a Bluetooth dongle and if it is connected to the tablet PC. If the flag is true the Bluetooth connection is initialized after the definition of necessary variables and structures. Also `StationID` is set to zero, because the Car-2-X Android application has to know if the incoming packets are from the own vehicle, see chapter 5.6. If the station does not need a Bluetooth connection (e.g. road works, speed limit,...) the StationID can be set arbitrarily.
Afterwards interfaces, which are independent of the type of ITS-Station, are initialized, like the GPS, CAN, UDP and IPv6 interfaces. Each initialization function returns a file descriptor (integer variable) to access the interfaces. In the case no real information is available on the CAN interface default values are loaded to avoid crashing of the soft-

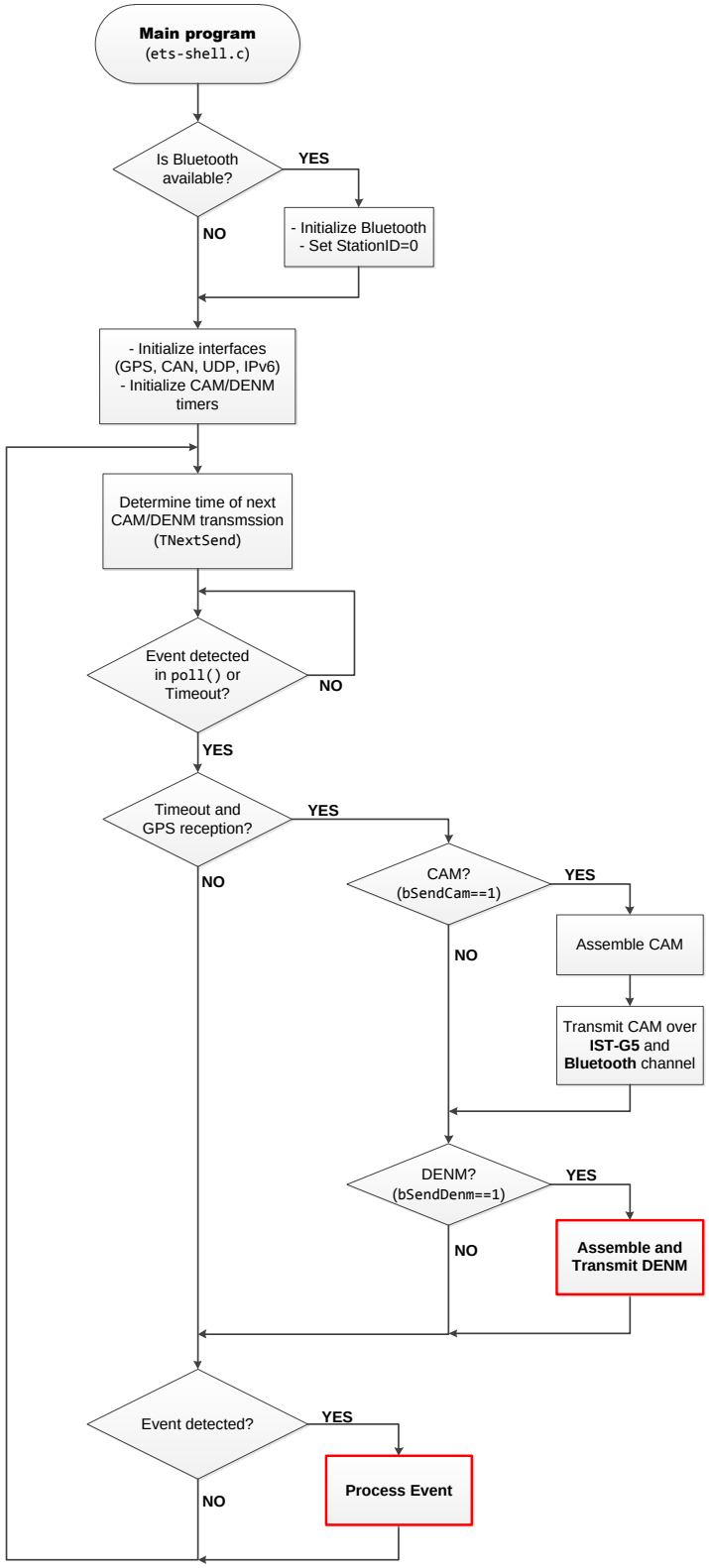


Figure 5.6: Implemented ITS Application - Flow diagram (Overview)

ware. The full flow diagram of the ITS application running on the MKx module is shown in Figure 5.6. Certain parts are grouped in one sub-block to keep it clear and the flow diagrams and description of them are covered apart and in a separate paragraph (marked with red frame in Figure 5.6).

The original ETS Shell application is just able to send CAMs and DENMs with the same fixed rate and it was not possible to stop and restart the transmission. But this is a prerequisite to implement the use cases, that is why these limitations were eliminated in a first step, which is explained in the following lines.
The `ets-shell` application uses the `poll` function to wait for some input from the interfaces (file descriptors). The third parameter of this function is `timeout`, if no event has occurred on any of the defined file descriptors, `poll()` waits at least `timeout` milliseconds for an event to occur. After the timeout it stops polling and returns a zero value. This mechanism is used for the transmission of CAM and DENM messages. Two timers control the CAM (`TLastSend_CAM`, `TNextSend_CAM`) and another two the DENM transmission (`TLastSend_DENM`, `TNextSend_DENM`). The difference between them is determined by the transmission rate (`ETS_OPTS_CAM_PERIOD_DEFAULT` and `ETS_OPTS_DENM_PERIOD_DEFAULT` in `ets-opts.h`). If no CAM or DENM has to be sent the respective value is set to zero otherwise to a non-zero period in milliseconds. Depending on the type of the next Car-2-X message (CAM or DENM) to be sent, the `TNextSend` timer is either set to `TNextSend_CAM` or `TNextSend_DENM`. Moreover the appropriate flag `bSendCam` or `bSendDenm` is set or both if at the same time a CAM and a DENM should be transmitted. The `timeout` parameter of the poll function contains the difference of the `TNextSend` timer value and the current time (`TNow`) and if this time expires the poll function returns and the CAM or DENM transmission is initiated. Since the main routine is in an endless loop the value of the timers and the `timeout` parameter are recalculated after every event or timeout.

But the other option that the `poll` function returns from its wait-state is that an event occurred on any of the file descriptors. That is why it first has to be determined if it was because of a timeout (CAM/DENM transmission) or because an interface has received data. In the case of a timeout the transmission of the Car-2-X message will be continued if the GPS reception is sufficient otherwise it will be skipped. The flags `bSendCam` and `bSendDenm` mark which type of Car-2-X message has to be sent over the ITS G5 (802.11p) channel. Is it a CAM the transmission will be handled by the `FL_PeriodicAction` function in `facility.cc`. In this function the Cooperative Awareness Message is assembled and sent over the 802.11p channel as well as the Bluetooth channel. This function was just changed slightly to automatize the CAM assembly depending on the use case and add the Bluetooth functionality.

Sub-block: Assemble and Transmit DENM

If the message to be transmitted is a DENM it has to be distinguished which kind of use case has to be activated, because depending on that the description of the DENM

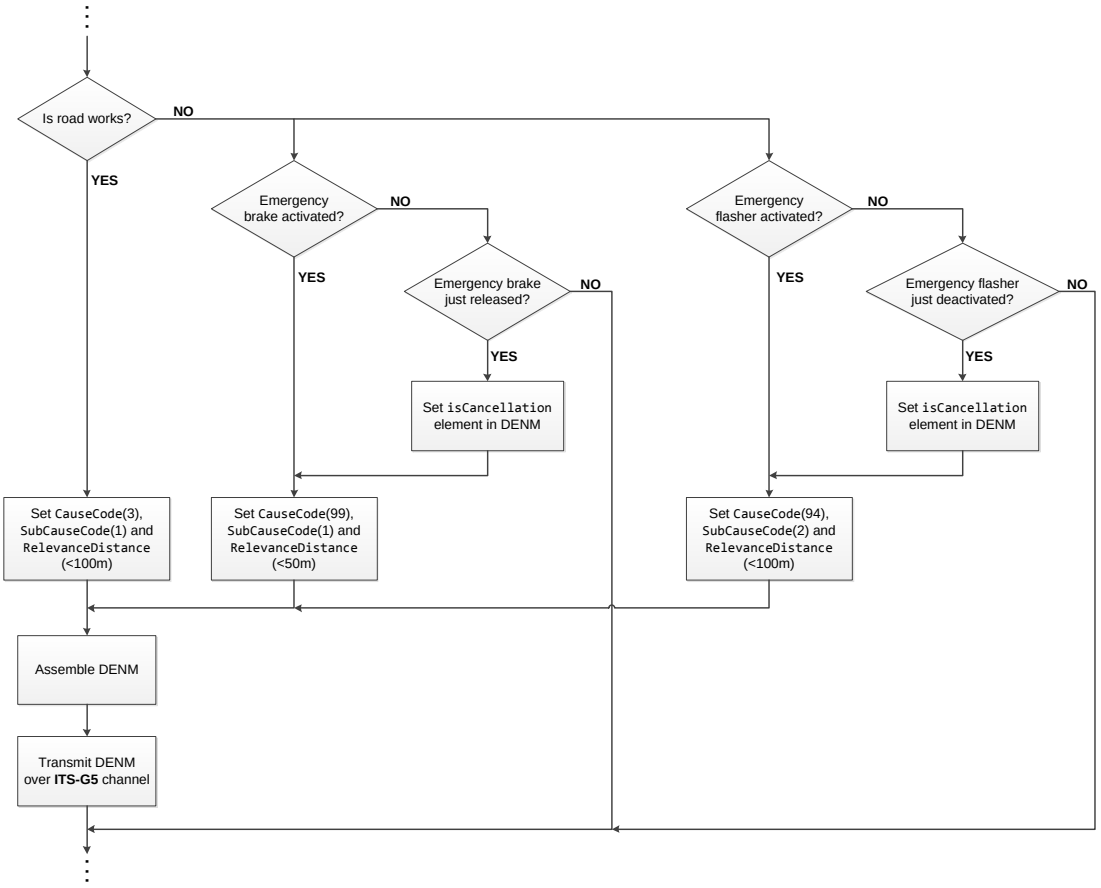


Figure 5.7: Implemented ITS Application - Flow diagram (Assemble and Transmit DENM)

event with the data frame `eventType` of type `CauseCode`, consisting of the data elements `causeCode` and `subCauseCode`, has to be set accordingly. Furthermore the `RelevanceDistance` data element determines the distance in which the event information is relevant for the receiver ITS-S [47]. Three of the six covered use cases make use of DENMs (see chapter 4.2.1- 4.2.3). See the following list for the appropriate codes and also the relevance distance. Please note that the value for the distance is just a recommendation as it is used within this thesis, but can be set as required.

- Emergency electronic brake lights
  Cause code: 99 (Dangerous situation)
  Sub cause code: 1 (Emergency electronic brake lights)
  Relevance distance: $<$50m

- Stationary vehicle warning
  Cause code: 94 (Stationary vehicle)

Sub cause code: 2 (Vehicle breakdown)
Relevance distance: <100m

- Road works warning
  Cause code: 3 (Roadworks)
  Sub cause code: 1 (Major roadworks)
  Relevance distance: <100m

The ITS-Station acts as a road works if `TARGET_STATION` is set to *roadworks* in `ets-shell.h`. After setting the data elements listed above, the `FL_EventAction` function is called to assemble the DENM and transmit it via ITS-G5 (802.11p) to other road users (see Figure 5.7). A transmission via Bluetooth to the tablet PC is not necessary because the DENM was produced by the own ITS-Station. Furthermore the CAN bus is checked if the emergency brake is activated, this would correspond to the *Emergency electronic brake lights* use case. The releasing of the brake (transition from high to low in the CAN message) also has to be detected to send a *cancellation DENM*, for this purpose the boolean variable `isCancellation` in the `ManagementContainer` of the DENM is set to high (see 2.2.4.2). If the ITS-Station stops ($Speed = 0$) and the button to activate the emergency flasher is pressed a *Stationary vehicle warning* DENM is sent to vehicles in the vicinity. And as soon as the emergency flasher is deactivated a cancellation DENM is transmitted once.

Sub-block: Process Event

Another reason that the application returns from polling is that events on the file descriptors were detected. The first possibility is that data was received at the CAN interface (see Figure 5.8). If this is the case the `CANUpdate` function in `if-can.c` is called to process the received CAN messages. This function and also the header file `if-can.h` has to be updated with the CAN information of the vehicle in use.

If a CAM or DENM message of another ITS-Station was received the UDP file descriptor detects this as an event and calls the `FL_BTPDataInd` to handle the received data. In this function also the forwarding of the received messages to the tablet PC is performed. Since the Car-2-X messages are PER encoded (see ASN.1 chapter 5.2) they first have to be decoded and encoded again with Distinguished Encoding Rules (DER). This is because the ASN.1 Compiler of the Java Android Application just supports DER (actually BER, but DER is a subset of BER, see 5.6). This DER encoded packet is sent over the Bluetooth channel.
The other two interfaces are GPS, which processes received GPS data and updates the corresponding variables, and the IPv6 interface. Both of them were inherited from the Cohda Wireless Framework and retained unchanged, that is why they are not described in more detail here.

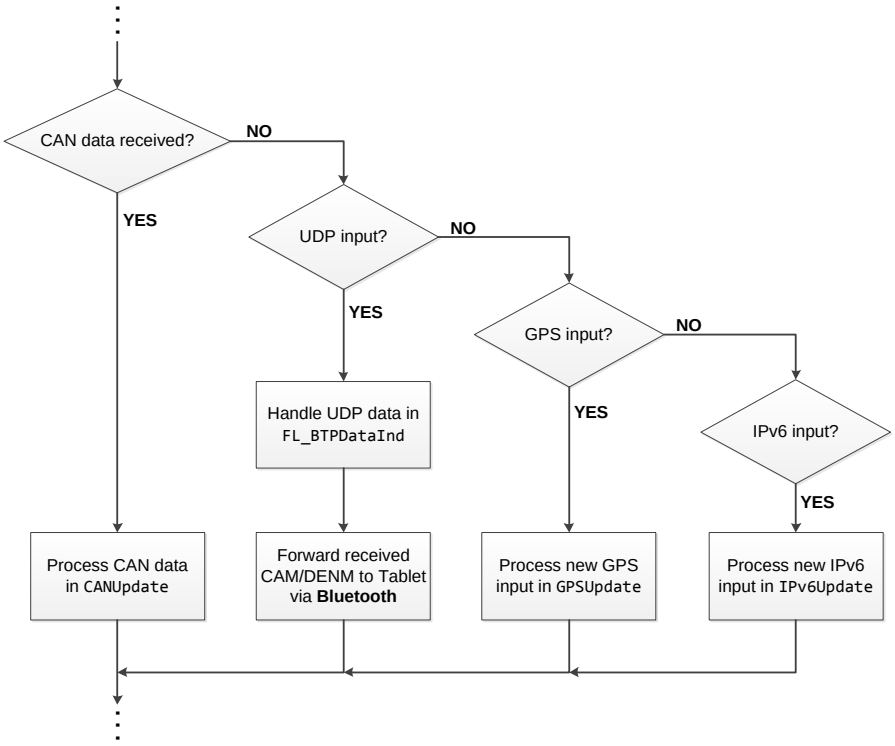


Figure 5.8: Implemented ITS Application - Flow diagram (Process Event)

5.6 Car-2-X Android Application

An integral part of a Car-2-X system is the visualization of Car-2-X-relevant data on a Human Machine Interface (HMI). A requirement for this Car-2-X demonstration system is that it has to be transferable from one vehicle to another one easily and should work with the eQuad. That is why the same tablet PC[14] which already controls the eQuad via Bluetooth is used as a HMI. The operating system of the tablet is *Android 4.1.2*. An Android tablet PC has the advantage that it is portable and several software packages are available to program Android applications in Java.
The already developed Android application to control the eQuad is an in-house development of Virtual Vehicle, but this thesis is also in cooperation with MAGNA STEYR Engineering AG & Co KG. Hence a completely new Android application was developed, which both companies could use for demonstration purposes of the Car-2-X technology.

After starting the Android application package *Car2X* on the tablet the *MainActivity*[15] creates a tab for the *VEHICLE STATUS* and one for the *MAP*. The components of the Vehicle Status tab are controlled by the *VehicleStatusActivity* and the Map tab by the *MapActivity*. Figure 5.9 shows the flow diagram of the Android application. Similar to the ITS Application running on the MKx module certain parts are grouped in a sub-block (marked with red frame) and explained separately.

5.6.1 Vehicle Status Tab

The Vehicles Status tab presents User Interface objects to control the Bluetooth connection to the MKx device (connect/disconnect, quit and send text to MKx), start/stop logging and shows vehicle specific data received by the CAM messages (see Figure 5.10). Actually this tab is not part of a final Car-2-X system because it basically just shows data which should not concern the end user, but during development this is very helpful. In the `onCreate()` callback function of VehicleStatusActivity this tab is initialized. It creates handles to the GUI elements and the vehicle parameters are set to default values in the `initVehicleStatus()` class method. The `VehicleStatus` class variables hold all these parameters concerning the own vehicle (e.g. 4D position, Speed, Heading,...), basically each of them is visualized on a screen element in the Vehicle Status tab. Also the LDM-like database, shortly mentioned in 5.2, is initialized here. It consists of two `Vector` class objects saving properties of surrounding vehicles and DENM events. Furthermore the Bluetooth device is prepared and checked if Blueooth is available and switched on at the tablet PC.

If the *Connect* button is pressed the application tries to establish a connection to the MKx module using a RFCOMM socket. The Android application works as a client, this

[14] Acer Iconia B1-A71

[15] An *Activity* is a component of an Android App that provides a window in which to draw its user interface. The Activity also supports life-cycle callback methods, like `onCreate()` which is called when the activity is first created or `onDestroy()` which is called before the activity is destroyed.

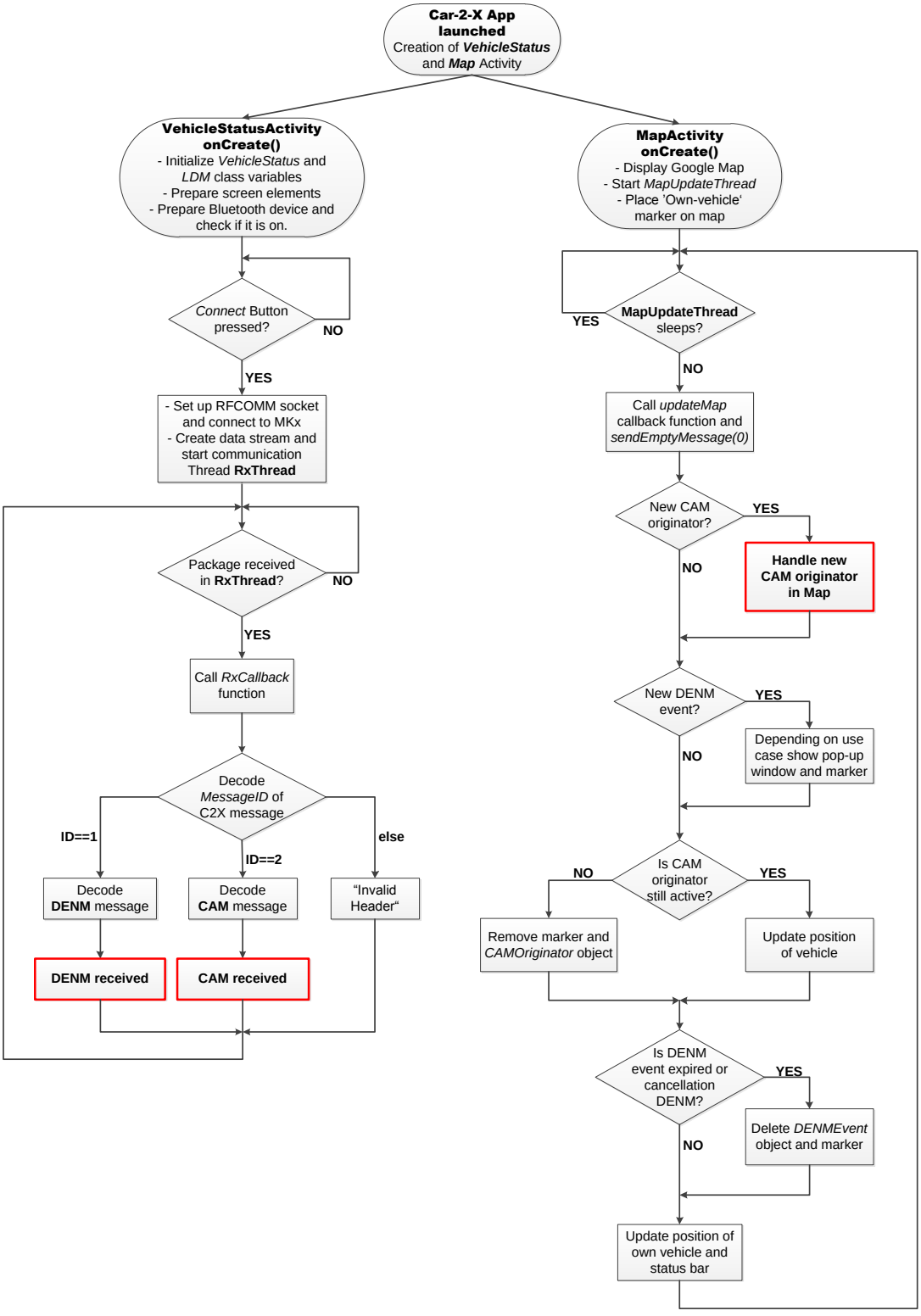


Figure 5.9: Car-2-X Android Application - Flow diagram (Overview)

means that the MKx module already has to be in *Wait-for-connection* mode, so that the tablet PC can connect to it. Since the ITS application on the MKx module is started automatically the module is in listen mode after boot-up and waits for the tablet to connect. If the Bluetooth dongle of the MKx is not ready or the device is switched off a warning box will appear.

After the successful connection buildup a data stream to/from the MKx module is created and a `start` command is sent to test the connection. Furthermore the communication thread `RxThread` is started which continuously listens to the Bluetooth channel if a packet is available. If there is one available the callback function `RxCallback` of the VehicleStatusActivity is called where the packet will be processed further. First the header of the ITS packet, to be precise the `messageID` element, is checked if it is a CAM ($messageID = 2$, see paragraph CAM received) or DENM ($messageID = 1$, see paragraph DENM received) message or maybe it even consists of an invalid header. Depending on the type the processing differs significantly.

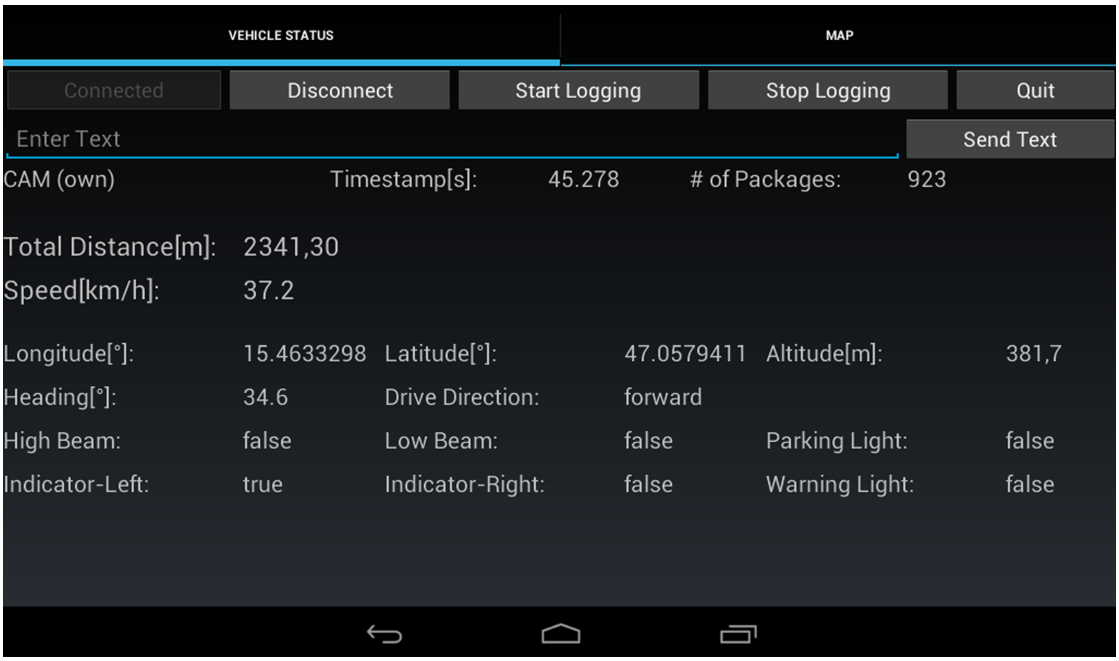


Figure 5.10: Vehicle Status Tab

Sub-block: CAM received

The decoding of the received packet is performed by the `decode()` function which is automatically generated by the ASN.1 Compiler from the asn.1 file. In the case of a CAM message the function is called `CamPdu.decode()`. If the packet is error-free and the decoding process was successful the appropriate Java structures are filled with the

decoded data.
Since every CAM or DENM sent or received is directly forwarded to the tablet PC it has to be defined which data belongs to a certain ITS-station. For that purpose the `StationID`[16] data element is used. A $stationID = 0$ is used and reserved for the vehicle containing the tablet PC. So if a CAM with a $stationID = 0$ is received, the corresponding `VehicleStatus` class variables have to be updated to have the latest vehicle and position data in the Car-2-X Android application (see Figure 5.11).

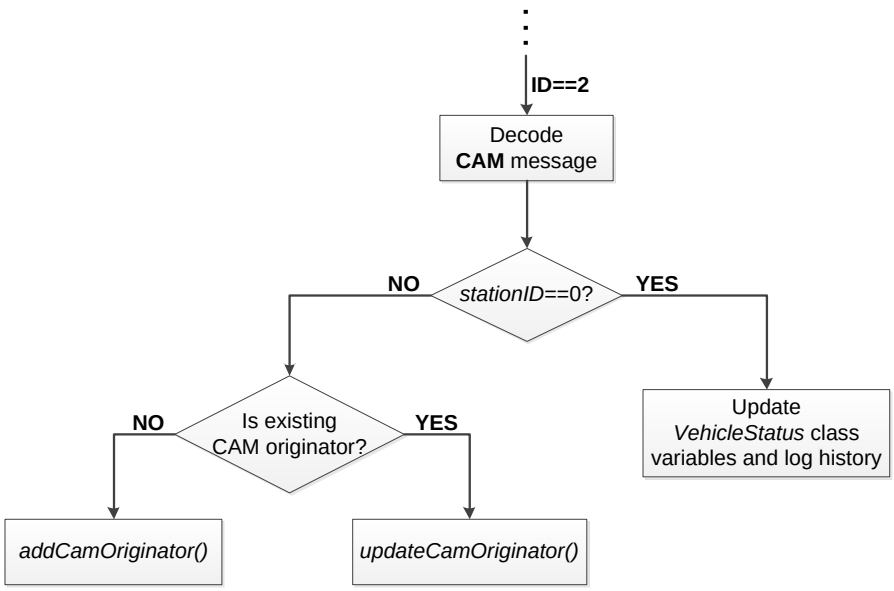


Figure 5.11: Car-2-X Android Application - Flow diagram (CAM received)

Additionally if the *Start Logging* button was pressed a `LogItem` object containing the most important data elements (timestamp, position, speed, heading and total distance) is saved and after pressing the *Stop Logging* button all the `LogItem` objects are written to a log-file called `fahrt#.log` in `/storage/sdcard0/MKx`.
If $stationID \neq 0$ a CAM from another ITS-Station was received. It first has to be determined if the ITS-Station is already existing in the LDM. If it is, the corresponding `CamOriginator` object has to be updated with the new parameters and if not a new object has to be created and loaded with the just received data.

Sub-block: DENM received

Is the received message a DENM ($messageID = 2$), it has to be decoded with the generated `DenmPdu.decode()` function. Afterwards the `RelevanceDistance` data element is investigated. Is the distance to the DENM originator below this relevance distance, the DENM will be processed otherwise it will be discarded, see the flow diagram of the DENM processing in Figure 5.12.

[16] The identifier of the ITS-Station that generates the ITS message [70].

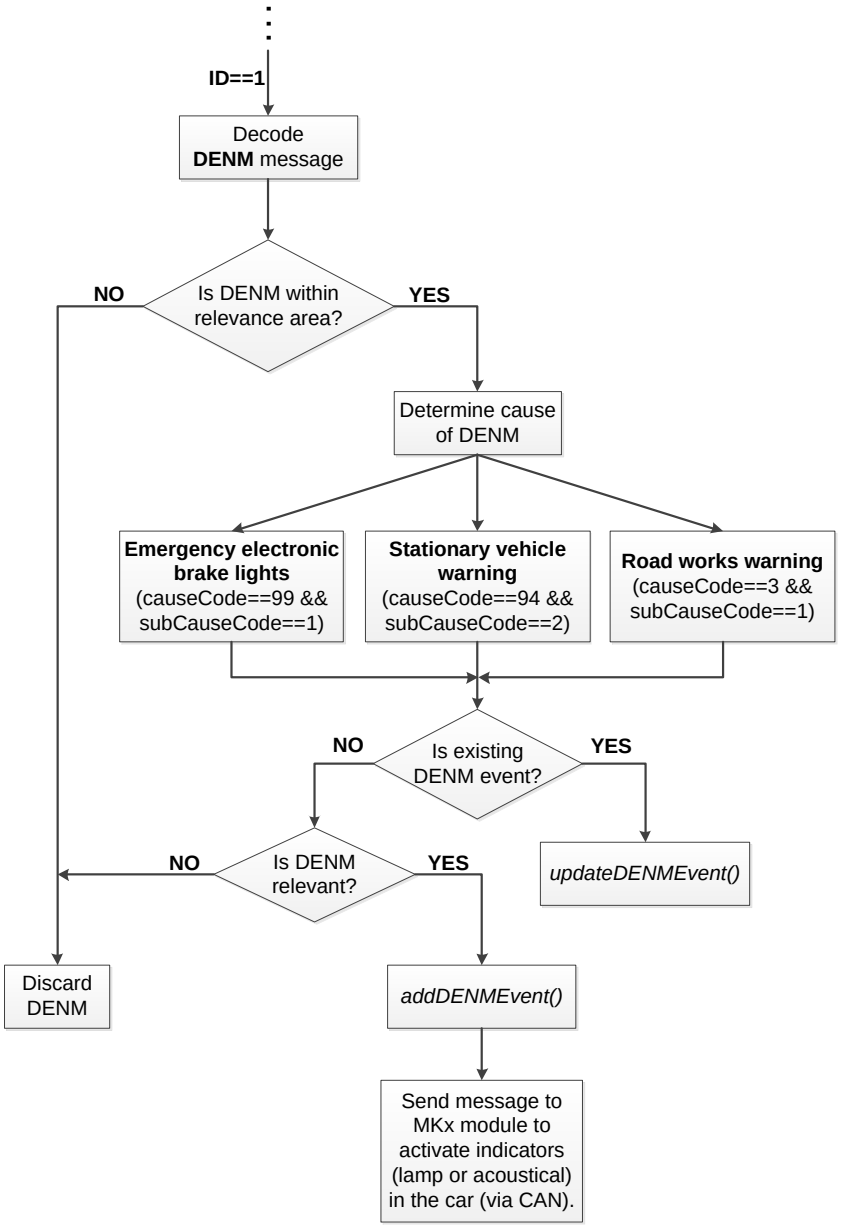


Figure 5.12: Car-2-X Android Application - Flow diagram (DENM received)

Similar to the CAM originator the LDM saves the status of valid DENM events. After determining the `CauseCode` date frame it is checked if the DENM event from this ITS-Station (`StationID`) is already known to the LDM, if yes the `DENMEvent` object will be updated (`LDM.updateDENMEvent()`) otherwise created (`LDM.addDENMEvent()`). But before creating a new `DENMEvent` object the relevance of the DENM has to be investigated. The next paragraph describes the implemented **Relevance Check**.

A commercial and ready for sale Car-2-X system needs a proper Relevance Check of Car-2-X messages, which means that warnings should be shown to the driver just when they are relevant and necessary. For example a warning that a vehicle is braking on a different lane or even a different road does not increase the road safety. Quite the opposite could happen, that the driver is confused and irritated by the huge amount of Car-2-X warnings and at worst switches off the system. As shortly described in 5.4.1 the use of Google Maps limits the complexity of a Relevance Check algorithm already significantly, this is why in this thesis a simple method was used. Basically it compares the *Heading* and *Bearing* parameters. The difference between these two angles (in degrees East of true North) is that Heading is the direction the vehicle is currently pointing to whereas Bearing is pointing directly to the DENM event (see figure 5.13). If the following condition is fulfilled

$$Heading - 90° < Bearing < Heading + 90° \tag{5.10}$$

the DENM event is in front of the user and the implemented use cases are relevant. Additionally in the use case *Emergency electronic brake lights* it has to be checked if the vehicles are driving in the same direction, for this purpose the Heading parameter of the own vehicle and the DENM originator have to be compared:

$$Heading(own) - 90° < Heading(originator) < Heading(own) + 90° \tag{5.11}$$

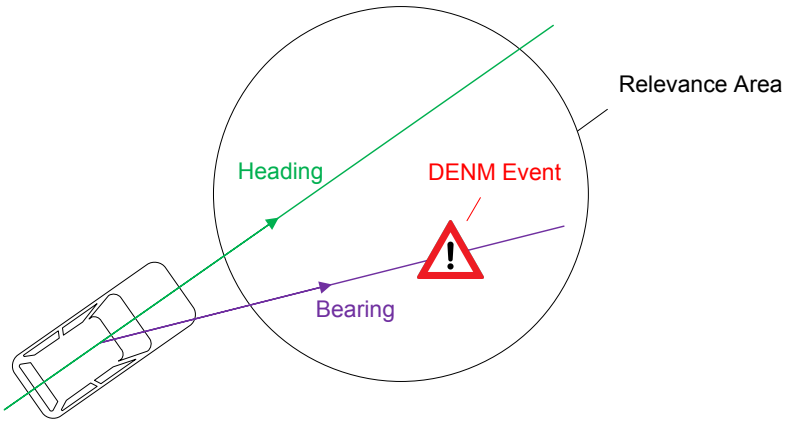


Figure 5.13: Comparison of Heading and Bearing

If the Relevance Check reveals that the conditions are not fulfilled the DENM is discarded otherwise a new `DENMEvent` object is generated and if necessary a Bluetooth message is sent to the MKx module to activate indicators in the vehicle (e.g. indication lamps in the instrument cluster or an acoustical warning). The ITS application on the MKx module runs a thread which is continuously listening to the Bluetooth channel for incoming packets. If the corresponding command to activate the indicators in the vehicle was received the `CANWrite` function in `if-can.c` is called to write the message

on the CAN bus. The requirement is that a vehicle with access to the CAN bus and the possibility to write on it is available. In one of the vehicles for the final tests this was possible (see chapter 6) .
Furthermore if the *Disconnect* or *Quit* button is pressed in the Android App a command is sent to the MKx module that it has to stop sending packets and it goes back to the Bluetooth *Wait-for-connection* mode, so that it can be re-connected with the tablet PC without switching the Car-2-X platform off and on.

5.6.2 Map Tab

The Map tab contains the Google Map and all the Car-2-X-relevant information are visualized here. After starting the App in the `onCreate()` callback function the Google Map is displayed, a `MapUpdateThread` is started and a marker, representing the own vehicle, is placed on the map. After a position change this marker of the own vehicle is always placed in the center of the map (see flow diagram in Figure 5.9).
The `MapUpdateThread` controls the update of the Map with a fixed rate, basically it just sleeps but after a certain amount of time (200ms is a good tradeoff between resource efficiency and a smooth and judder-free map update) the `updateMap()` callback function of the *MapActivity* is called. This function just sends a message with `Handler.sendEmptyMessage(0)` to a Handler of the UI thread (main thread), which runs UI objects such as TextView and Button objects. It is not possible to have access to UI objects from a different thread in Android otherwise the *Not on the main Thread* error is thrown. Thus the dummy message has to be sent between both threads so that the UI thread handles the update of the map.

Sub-block: Handle new CAM originator in Map

If the *VehicleStatusActivity* recognizes a new surrounding vehicle (CAM originator) and adds a new `CamOriginator` object to the LDM the *MapActivity* detects that it has to add an additional marker to the map, which represents the new vehicle. Figure 5.14 shows a detailed flow diagram of this sub-block.
Depending on the `StationType` and `VehiceRole` data elements the according icon has to be used as a marker. If the surrounding vehicle is a passenger car ($stationType = 5$ & $vehicleRole = 0$) or a motorcycle ($stationType = 4$ & $vehicleRole = 0$) just the marker is created but no warning appears. Actually detecting that there is a motorcycle is part of the *Motorcycle warning* use case, but as described in 4.2.5 because of the missing collision risk analysis no pop-up window appears.
But if the CAM originator is an emergency vehicle ($stationType = 10$ & $vehicleRole = 6$) it has to be determined from which direction the vehicle is approaching by calculating the bearing to the emergency vehicle. Depending on the heading of the own vehicle an arrow in the pop-up window shows the driver where the emergency vehicle is coming from, so that he can pay his attention to this side (see *Approaching emergency vehicle warning* use case in chapter 4.2.4).

As described in 4.2.6 the ITS standard so far does not support the In-Vehicle Information (IVI) message set (see 2.2.4.3) that is why the workaround with the CAM message and the `SafetyCarContainer` is used. This container includes a `SpeedLimit` data element. If the current speed is higher than the speed limit a *Slow Down* pop-up window (see Figure 4.5) appears otherwise just the current speed limit is presented. At the position of the CAM originator a marker showing the speed limit is placed.

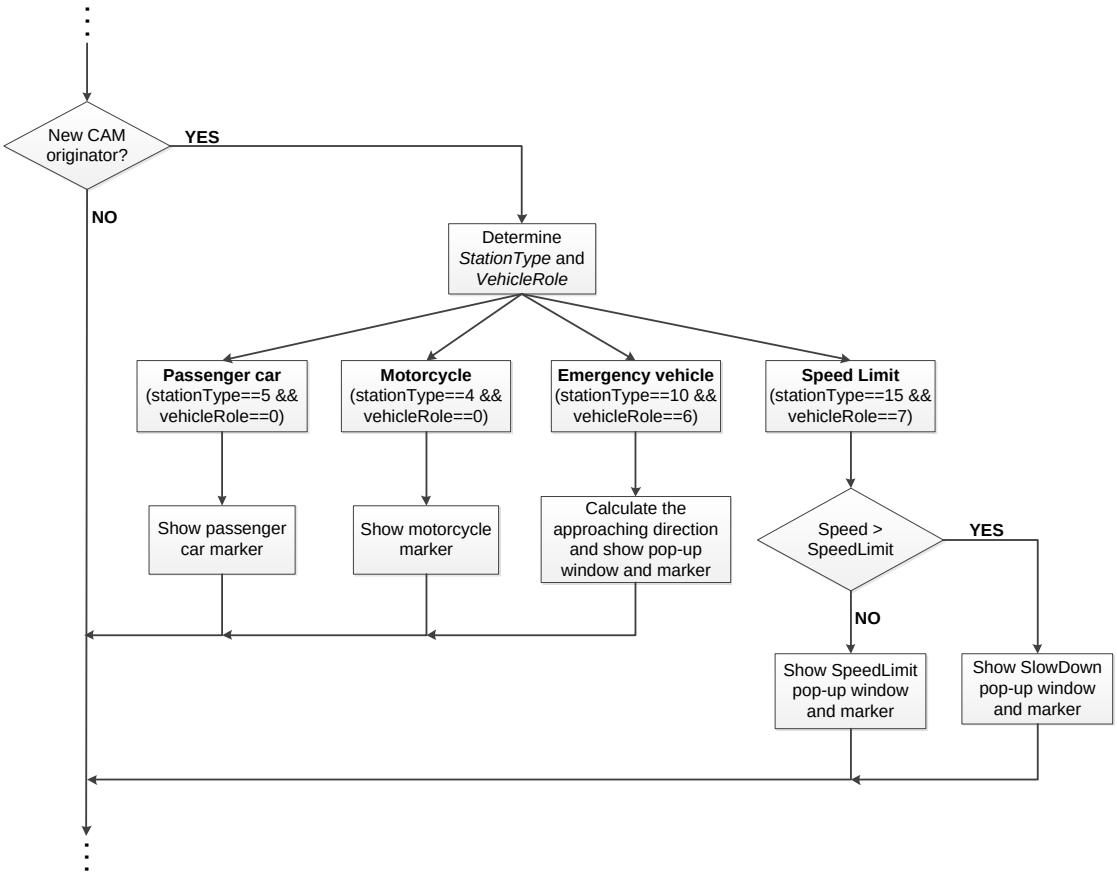


Figure 5.14: Car-2-X Android Application - Flow diagram (Handle new CAM originator)

Similar to a new surrounding vehicle the *MapActivity* recognizes that a new DENM event was detected by the *VehicleStatusActivity* and has to react accordingly. Depending on the `CauseCode` the appropriate pop-up window appears and for the *Road works warning* and *Stationary vehicle warning* use case also a marker is displayed on the map (see chapter 4.2.1- 4.2.3).

If a CAM originator is not in communication range anymore the marker but also the `CamOriginator` object in the LDM has to be deleted. For this purpose all the surrounding vehicles (CAM originators) are checked every time the map is updated if

packets are still received from them. After a timeout of three seconds without a new CAM the marker and the `CamOriginator` object are permanently deleted. Is the vehicle again within communication range a new object and a marker are initialized.
For the DENM events the validity check is more complicated. Every DENM contains a `ValidityDuration` data element, to set how long the DENM event persists, after this time the event is expired and will be deleted. But a DENM event could also be stopped if the originating ITS-Station sends a *cancellation DENM*. If $isCancellation = true$ the `DENMEvent` object and the marker are immediately deleted.

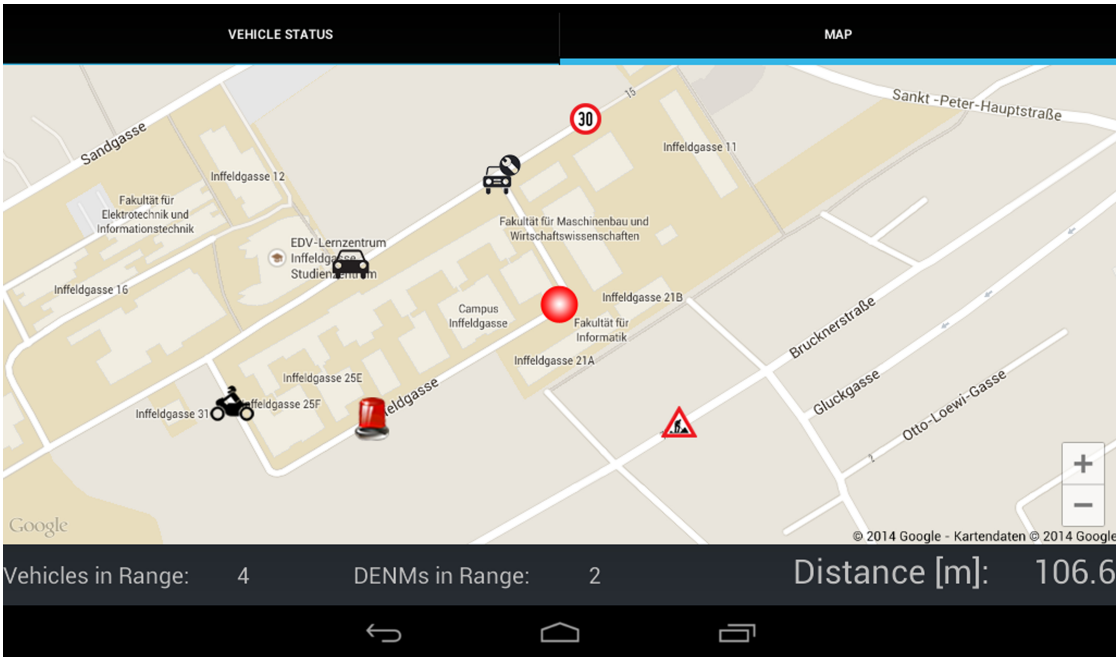


Figure 5.15: Map Tab

The last step is to update the position of the own vehicle and set it to the center of the map. Also update the status bar, which contains the number of vehicles and active DENM events within communication range and also the distance to the latest DENM event in meters, see Figure 5.15. In the example shown in the figure, six ITS-Stations are within communication range, four of them are vehicles (passenger car, motorcycle, stationary vehicle and emergency vehicle) and two of them are sending DENMs (stationary vehicle and road works). Since the speed limit is using the `SafetyCarContainer` per definition it would be a vehicle, but because this is just a workaround as described earlier it is defined in this application as a Road Side Unit, which is sending CAMs.

CHAPTER 6

Final Tests

The research done on the current state of the ETSI ITS standard and the practical work was described in the previous chapters. Finally, the thesis is completed with a field test on the TU Graz campus.
The focus of the tests was in the installation and integration of the Car-2-X platform in the car and the cooperation between the Car-2-X-equipped vehicles. In total five MKx platforms (2xMK2, 3xMK4) and three test vehicles (eQuad and two passenger cars) were available. Access to the CAN bus was, with limitations, possible in all of them. Since the eQuad was developed by Virtual Vehicle all the CAN data is accessible. The *.dbc file and the vehicle network topology was available for one car (*test-car 1*), so basically all the CAN data was accessible, even access to the instrument cluster was possible. On the other vehicle (*test-car 2*) no *.dbc files were available, but at least we got information about the most important CAN messages, like the current speed and the state of braking. For reasons of confidentiality the manufacturer and the version of the vehicle will be not revealed and thus just called *test-car 1* and *test-car 2* throughout this chapter.

6.1 Test Condition

Channel Configuration

The channel allocation is strictly defined by the ETSI ITS standard and covered in chapter 2.2.2.2. All of the tested use cases are of the *Road Safety* class (see Table 4.1) that implies that the devices have to communicate in the **ITS-G5A band**. Because of the usage order defined in [43] the **G5-CCH** channel (IEEE 802.11 Ch.No.: 180) is the default channel. In fact *In-vehicle signage* is a Traffic Efficiency use case, but as already discussed several times CAMs are used as a workaround to implement it. Hence, the same frequency band was used.
The channel number is not the only important setting in the `etsa` configuration file.

The following table shows an overview of the channel configuration used in these tests. The first column is a description of the setting, the second is the name of the parameter in the configuration file and the last one the assigned value.

Description	Parameter	Value
IEEE 802.11 Channel number	ItsG5CchChanNum	180 (≙G5-CCH)
Transmit power level	ItsG5CchTxPwrLevel	33 (≙33 dBm)
Data rate	ItsG5CchTxDataRate	12 (≙6 Mbit/s)

Table 6.1: Final Test - Channel configuration

Please note that the transmit power level of 33dBm is just a theoretical value defined by the standard but according to the MKx specification the maximum transmit power is +22dBm per antenna port 3.1.2.

Use cases under test

All the six use cases described in the chapter 4 were tested, although the *Motorcycle warning* was not tested as a separate use case. Instead the equipment was mounted on a motorcycle and on the map the presence of the motorcycle was shown as soon as it was within communication range.
Depending on the use case the ITS application running on the MKx modules were recompiled with the corresponding settings (e.g. `TARGET_STATION` in `ets-shell.h`) and uploaded on the module.

Test site

The tests were performed on the campus of the Technical University Graz and surrounding streets. Road works in the 'Händelstraße' were used to test the *Road works warning* use case. Figure 6.1 shows the test scenario. In the front of the picture is the road works traffic sign with the Car-2-X equipment and the antenna. The MKx hardware is supplied with a usual car battery.
The 30 kph zone in the 'Brucknerstraße' was used to test the *In-vehicle signage* use case, for this purpose the Car-2-X platform was mounted on a 30 kph traffic sign. Since the other use cases are Car-2-Car communications they were directly performed at the TU Graz Campus Inffeldgasse with the two test vehicles.

6.2 Test Procedure

The *test-car 1* where all the necessary CAN data was accessible was used as the main test object. In this vehicle the MK2 module was connected with the CAN bus (see Figure 6.2a) and the tablet PC as well as a GoPro camera to capture the tests was mounted in the vehicle.

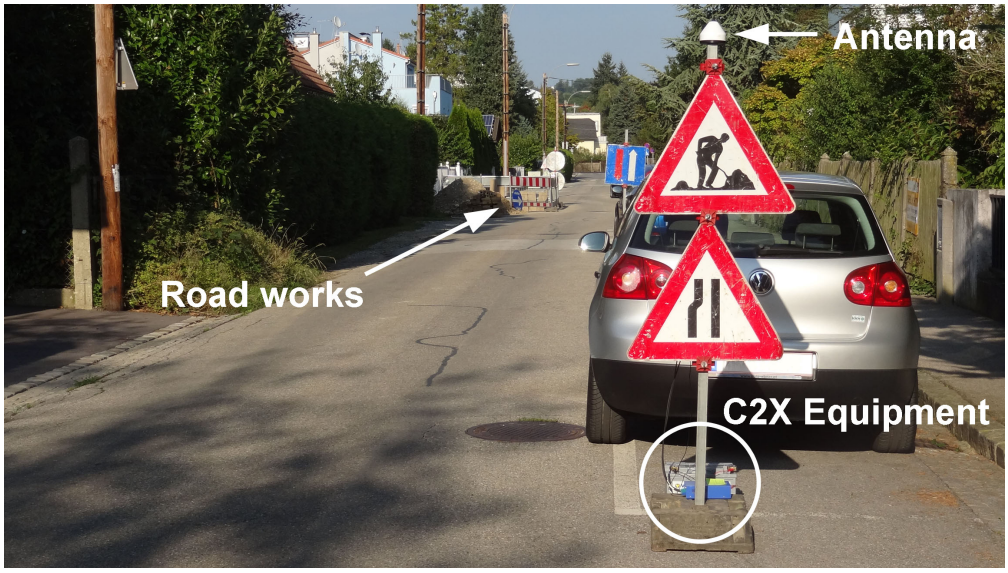


Figure 6.1: Road works equipped with Car-2-X hardware

The MK2 module of *test-car 2* was also connected to the vehicle CAN bus (see Figure 6.2b). Because of the limited CAN access this vehicle was just used to simulate a braking vehicle in the *Emergency electronic brake lights* use case, a stationary vehicle and an emergency vehicle.

(a) (b)

Figure 6.2: Cohda Wireless MK2 box in a) *test-car 1*, b) *test-car 2*

In Figure 6.3 the dashboard of the *test-car 1* during the *Emergency electronic brake lights* use case is shown. The tablet PC is mounted in a clearly visible position. As soon as the vehicle in front is braking the warning pop-up window (see Figure 4.1 appears on the screen. In addition an indication lamp on the instrument cluster (visible in Figure 6.3) is activated as well as an acoustical warning in the vehicle.

Figure 6.3: Field Test - Dashboard during *Emergency electronic brake lights* use case

6.3 Test Results

Use Case	Status
Emergency electronic brake lights	√
Stationary vehicle warning	√
Road works warning	√
Approaching emergency vehicle warning	√
Motorcycle warning	√
In-vehicle signage	√

Table 6.2: Tested use cases

All the six use cases were successfully tested. Since all use cases (except the ones which do need access to a central ITS-Station, see chapter 4.1) suggested by the Amsterdam Group are implemented, this Car-2-X demonstration system is in principle ready for deployment. This was the objective and main goal of this thesis.
Because of the limited availability of the test cars no tests of the communication range have been performed, please refer to [14].

6.4 Conclusion

The functionality and usability of the Car-2-X demonstration system based on the Cohda Wireless MKx platforms as well as the integration in the vehicle were successfully tested. Furthermore the potential of vehicular communications was demonstrated and also a video to promote the system was recorded.

CHAPTER 7

Conclusion and Outlook

This chapter summarizes each previous chapter shortly and a outlook / next steps for further works is proposed.

The first tasks of this thesis was to get an overview of the current status of the standardization and to analyze how the standards have to be applied to implement a full standards-compliant Car-2-X demonstration system.
In February 2014 at the 6th ETSI workshop on ITS in Berlin a first release of standards for Cooperative Intelligent Transport Systems (C-ITS) was published. The results are discussed, applied and examined within this thesis. Moreover the capability of LTE for vehicular communications is analyzed.

The available Car-2-X platforms on the market are development platforms and not suitable for a serial production, mainly because of size and costs. The cooperation between Cohda Wireless and NXP Semiconductors made a major step forward by introducing the first market-ready chipset for Car-2-X communication, the RoadLINK^{TM} family. The first two products of this chipset family are the Software-Defined Radio (SDR) baseband processor **SAF5100** and the dual radio multi band RF transceiver **TEF5100** [51] [52]. The Cohda Wireless platform MK4 already uses these processors and significant improvements in size and performance could have been achieved. The next generation (MK5) will just need a single chip and with this platform from a hardware perspective the necessary conditions for the deployment of C-ITS are created.

On the MKx modules (MK2 and MK4 are used) runs a Linux operating system which allows to implement ITS applications for the platform as Linux application software in C. For this purpose Cohda Wireless provides a Virtual Machine running Ubuntu and all the necessary development tools as well as a framework for the implementation of ITS applications.

In 2009 ETSI has published a list of possible applications and use cases for a 'day-one' deployment of Car-2-X. This list was reviewed and edited in the EU projects PRE-DRIVE C2X and DRIVE C2X. Finally the Amsterdam Group suggested a few 'day-one' use cases in a Roadmap document [13]. Basically all of them were implemented and demonstrated within these thesis, except the ones which do need access to a central ITS-Station.

The main component of a Car-2-X system is the Car-2-X hardware platform. It handles the communication to the vehicle via the vehicle bus and the communication to other ITS-Stations via the ITS-G5 protocol as well as the reception of GPS data. The usage of the hard- and software interfaces is controlled by the ITS application running on the platform. A framework which can be used as a base is provided by Cohda. This application is just capable of sending CAMs and DENMs with a fixed rate continuously, but no scheduling is implemented. For example it is not possible to initiate the spontaneous sending of DENMs after an event occurred (e.g. brake was activated in the vehicle). Also stopping the event automatically is not provided. So this framework was extended and parts of it are completely renewed to be capable of performing the desired use cases.

Also the interface to the Human Machine Interface (HMI), which is the second integral part of the Car-2-X system, was implemented on the MKx module. The HMI visualizes Car-2-X-relevant information and informs the driver about upcoming hazards. In this thesis an Android tablet PC was used as a HMI. The reasons are that it is portable and several software suites are available to program Android applications in Java.
The author has selected Bluetooth as the interface between the Car-2-X hardware and the tablet PC. Because of the limited data rate of Bluetooth it is important to find an efficient way to transfer the Car-2-X messages (CAMs and DENMs) received by the MKx platform to the tablet PC. It was decided to use ASN.1 because of its efficiency, platform independence and because the ETSI ITS standard already makes use of it.

A full Car-2-X Android App was developed during this thesis to visualize the Car-2-X relevant data. The App has to be programmed in Java and not like the MKx ITS application in C, which resulted in a significant extra effort. For example the Cohda framework for the ITS application already supported ASN.1 but for the Java application it was hard to find an appropriate ASN.1 Compiler. At the end an open-source software of Fraunhofer Institute for Solar Energy Systems in Freiburg, Germany was adapted to support the ASN.1 files of the ETSI ITS standard.
The Android App consists of an Tab to visualize the status (4D position, status of lights, status of brake,...) of the own vehicle, which contains the tablet PC. And of a second Tab which includes a Google map to visualize the current position of the own vehicle but also the position of DENM events as well as all vehicles in the communication range which are equipped with Car-2-X hardware.
Basically all the Car-2-X messages received by the MKx module are just forwarded to the tablet PC and all the analysis is performed in the Java App.

Finally the developed Car-2-X demonstrator was investigated in field tests with two passenger cars. All the six implemented use cases have been successfully tested. The potential of vehicular communications have been demonstrated also a video to promote the system was recorded.

7.1 Outlook and Next Steps

The integration of the Car-2-X platform in the vehicle was demonstrated successfully. In case that car manufacturers will be legally obliged to integrate Car-2-X functionality in their vehicles probably Bluetooth will not be used as the interface between the Car-2-X platform and the HMI. A potential interface is Ethernet. The usage of Ethernet within this Car-2-X demonstration system is a possible extension and improvement.

Furthermore in chapter 5.4.1 the limitations of Google Maps are already discussed. It is impossible to get street names and information about the course of the road out of the map. This limits the performance of a Relevance Check significantly. An option is to use open map providers like OpenStreetMap or basemap.at. In combination with a suitable Map Matching algorithm to overcome the GPS inaccuracy a sophisticated collision risk analysis can be implemented.

This thesis implemented a fully compliant Car-2-X system according to current state of standardization. Potential standard changes and updates have to be incorporated in the Car-2-X demonstration system. As an example GeoNetworking to disseminate information over a defined geographical area [78] was not specifically addressed throughout this document. Also security & legal issues are still not fully solved and the validation and certification of Car-2-X hardware and software is under discussion [79].

APPENDIX A

List of Abbreviations

ABS	Anti-lock Braking System
ADC	Analog-to-Digital Converter
ADT	Android Developer Tools
API	Application Programming Interface
APK	Android Application Package
ASECAP	European Association of Tolled Road Infrastructures Concessionaires
ASN.1	Abstract Syntax Notation One
BER	Basic Encoding Rules
BSD	Berkeley Software Distribution
BSS	Basic Service Set
BTP	Basic Transport Protocol
C2C	Car-2-Car
C2C-CC	CAR 2 CAR Communication Consortium
C2I	Car-2-Infrastructure
C2X	Car-2-X
CA	Cooperative Awareness
CAM	Cooperative Awareness Message
CAN	Controller Area Network
CEDR	Conference of European Directors of Roads
CEN	European Committee for Standardization
CENELEC	European Committee for Electrotechnical Standardization
CER	Canonical Encoding Rules
CPI	CAN protocol interface

C-ITS	Cooperative Intelligent Transport Systems
DAC	Digital-to-Analog Converter
DCC	Decentralized Congestion Control
DENM	Decentralized Environmental Notification Message
DER	Distinguished Encoding Rules
DFS	Dynamic Frequency Selection
DSC	DCC Sensitivity Control
DSP	Digital Signal Processor
EDGE	Enhanced Data rates for GSM Evolution
ESO	European Standardization Organization
ETSI	European Telecommunications Standards Institute
FOT	Field Operational Test
G5-CCH	ITS-G5 Control Channel
G5-SCH	ITS-G5 Service Channel
GPS	Global Positioning System
GPSD	GPS Daemon
GSM	Global System for Mobile Communications
GUI	Graphical User Interface
HCI	Host Controller Interface
HCID	Host Controller Interface Daemon
HMI	Human Machine Interface
HSPA	High Speed Packet Access
I2V	Infrastructure-to-Vehicle
ICS	ITS Central Station
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
ISO	International Organization for Standardization
ITS	Intelligent Transport Systems
ITS-S	ITS-Station
IVI	In-Vehicle Information
LDM	Local Dynamic Map
LTE	Long-Term Evolution
MAC	Media Access Control
OBU	On Board Unit
OEM	Original Equipment Manufacturer

OSI	Open Systems Interconnection
PDM	Probe Data Management
PDU	Packet Data Unit
PER	Packed Encoding Rules
PVD	Probe Vehicle Data
RFCOMM	Radio Frequency Communication
RSU	Road Side Unit
SAE	Society of Automotive Engineers
SCLK	Serial Clock
SDK	Software Development Kit
SDP	Service Discovery Protocol
SDR	Software-Defined Radio
SHA	Secure Hash Algorithm
SPaT	Signal Phase and Timing
SPP	Serial Port Profile
SRM	Signal Request Message
SSM	Signal Status Message
TAC	Transmit Access Control
TCP	Transmission Control Protocol
TDC	Transmit Datarate Control
TPC	Transmit Power Control
TRC	Transmit Rate Control
UBI	Unsorted Block Image
UDP	User Datagram Protocol
UMTS	Universal Telecommunication System
UUID	Universally Unique Identifier
V2I	Vehicle-to-Infrastructure
V2V	Vehicle-to-Vehicle
VANET	Vehicular Ad-Hoc Network
VIC	Vehicle Interface Connector
VM	Virtual Machine
WAVE	Wireless Access in Vehicular Environment
WHO	World Health Organization
WLAN	Wireless Local Area Network
XER	XML Encoding Rules
XML	Extensible Markup Language

Bibliography

[1] Euro NCAP. 2020 Roadmap - European new car assessment programme. 2014.

[2] World Health Organization WHO. The 10 leading causes of death in the world, 2000 and 2012. Fact sheet. `http://www.who.int/mediacentre/factsheets/fs310/en/`. [Online; accessed 09-Oct-2014].

[3] World Health Organization WHO. Global status report on road safety. 2013.

[4] World Health Organization WHO. Road traffic injuries. Fact sheet. `http://www.who.int/mediacentre/factsheets/fs358/en/`. [Online; accessed 09-Oct-2014].

[5] European Union. EU transport in figures - Statistical Pocketbook 2013. 2013.

[6] European Union. Roadmap to a Single European Transport Area – Towards a competitive and resource efficient transport system. Whitepaper. 2011.

[7] World Health Organization WHO. World report on road traffic injury prevention. 2004.

[8] Centre for Economics and Business Research (Cebr). Economic and environmental costs of gridlock. 2013.

[9] The Center for Internet and Society at Stanford Law School. Human error as a cause of vehicle crashes. 2013.

[10] G. Karagiannis, O. Altintas, E. Ekici, G. Heijenk, B. Jarupan, K. Lin, and T. Weil. Vehicular Networking: A Survey and Tutorial on Requirements, Architectures, Challenges, Standards and Solutions. *IEEE Communications Surveys & Tutorials*, 13(4):584–616, 4th Quarter 2011.

[11] Cohda Wireless Pty Ltd. Website. `http://www.cohdawireless.com`. [Online; accessed 07-Aug-2014].

[12] R. Popescu-Zeletin, I. Radusch, and M.A. Rigani. *Vehicular-2-X Communication, State-of-the-Art and Research in Mobile Vehicular Ad hoc Networks*. Springer-Verlag Berlin Heidelberg, online edition, 2010.

[13] Amsterdam Group. Roadmap between automotive industry and infrastructure organisations on initial deployment of Cooperative ITS in Europe. Version 1.0, 2013-06.

[14] C. Payerl. Integration von Car-to-X Kommunikation in die E/E-Architektur von Fahrzeugen. Master's thesis, Graz University of Technology, 2013.

[15] Amsterdam Group. Website. https://amsterdamgroup.mett.nl. [Online; accessed 11-Feb-2014].

[16] Amsterdam Group. Flyer. 2012.

[17] CAR 2 CAR Communication Consortium. Website. Mission & Objectives. http://car-2-car.org. [Online; accessed 11-Feb-2014].

[18] ASECAP. Website. Mission. http://www.asecap.com. [Online; accessed 17-Feb-2014].

[19] Conference of European Directors of Roads. Website. http://www.cedr.fr. [Online; accessed 11-Feb-2014].

[20] POLIS. Website. About Us. http://www.polisnetwork.eu/about/about-polis. [Online; accessed 20-Feb-2014].

[21] ERTICO - ITS Europe. Website. http://www.ertico.com. [Online; accessed 17-Feb-2014].

[22] Tona P. Compass4D - Working towards deployment of C-ITS. In *Proceedings of the Amsterdam Group Workshop, September 25, Brussels, Belgium*, 2013.

[23] European Commission - Directorate General Communications Networks, Content and Technology. Website. http://ec.europa.eu/dgs/connect. [Online; accessed 20-Feb-2014].

[24] European Commission - Directorate General Enterprise and Industry. Standardisation mandate M/453, 2009.

[25] European Commission. Press Release. New connected car standards put Europe into top gear, February 2014.

[26] ETSI. Content Release 1 - Rev 1. http://docbox.etsi.org/Workshop/2013/201302_ITSWORKSHOP/Release1_rev1.pdf, 2013. [Online; accessed 26-Feb-2014].

[27] ETSI EN 302 665 V1.1.1. Intelligent Transport Systems (ITS); Communications Architecture. 2010-09.

[28] ISO/IEC 7498-1. Information technology - Open Systems Interconnection - Basic Reference Model: The Basic Model. 1994.

[29] ETSI TS 102 723-Part1-11. Intelligent Transport Systems (ITS); OSI cross-layer topics; Part 1-11.

[30] ETSI EN 302 663 V1.2.0 Draft. Intelligent Transport Systems (ITS); Access layer specification for Intelligent Transport Systems operating in the 5 GHz frequency band. 2012-11.

[31] IEEE 802.11. IEEE Standard for Information technology - Telecommunications and information exchange between systems. Local and metropolitan area networks - Specific requirements. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Amendment 6: Wireless Access in Vehicular Environments. 2010-07.

[32] W.-Y. Lin. A comparison of 802.11a and 802.11p for V-to-I communication: a measurement study.

[33] Y. Li. An Overview of the DSRC/WAVE Technology. In *7th International ICST Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness*, 2010.

[34] K. Sjöberg. Standardization of Wireless Vehicular Communications within IEEE and ETSI. In *IEEE VTS Workshop on Wireless Vehicular Communications, Halmstad University, Sweden*, 2011.

[35] ETSI TR 102 962 V1.1.1. Intelligent Transport Systems (ITS); Framework for Public Mobile Networks in Cooperative ITS (C-ITS). 2012-02.

[36] K. Trichias. Modeling and Evaluation of LTE in Intelligent Transportation Systems. Master's thesis, University of Twente, 2011.

[37] A. Vinel. 3GPP LTE Versus IEEE 802.11p/WAVE: Which Technology is Able to Support Cooperative Vehicular Safety Applications? *IEEE Wireless Communications Letters*, 1(2):125–128, April 2012.

[38] T. Mangel, T. Kosch, and H. Hartenstein. A Comparison of UMTS and LTE for Vehicular Safety Communications at Intersections. In *IEEE Vehicular Networking Conference*, 2010.

[39] G. Araniti, C. Campolo, M. Condoluci, A. Iera, and Molinaro A. LTE for Vehicular Networking: A Survey. *IEEE Communications Magazine. Topics in Automotive Networking and Applications*, pages 148–157, May 2013.

[40] European Commission. Digital Agenda Scoreboard. Trends in European broadband markets 2014, 2014.

[41] Qualcomm Technologies, Inc. LTE Direct Always-on Device-to-Device Proximal Discovery. 2014.

[42] ETSI EN 302 571 V1.2.0 Draft. Intelligent Transport Systems (ITS); Radiocommunications equipment operating in the 5 855 MHz to 5 925 MHz frequency band; Harmonized EN covering the essential requirements of article 3.2 of the R&TTE Directive. 2013-05.

[43] ETSI TS 102 724 V1.1.1. Intelligent Transport Systems (ITS); Harmonized Channel Specifications for Intelligent Transport Systems operating in the 5 GHz frequency band. 2012-10.

[44] ETSI TS 102 687 V1.1.1. Intelligent Transport Systems (ITS); Decentralized Congestion Control Mechanism for Intelligent Transport Systems operating in the 5GHz range; Access layer part. 2011-07.

[45] ETSI EN 302 637-2 V1.3.0 Draft. Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part2: Specifications of Cooperative Awareness Basic Service. 2013-08.

[46] ITU-T X.691 Recommendation. Information technology - ASN.1 encoding rules: Specification of Packed Encoding Rules (PER). 2008.

[47] ETSI EN 302 637-3 V1.2.0 Draft. Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part3: Specifications of Decentralized Environmental Notification Basic Service. 2013-08.

[48] ETSI EN 302 895 V1.0.0 Draft. Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Local Dynamic Map (LDM). 2014-01.

[49] P. Spaanderman. Infrastructural Message Sets Standardization at CEN and ISO. In *Proceedings of the 6th ETSI ITS Workshop, February 12-13, Berlin, Germany*, 2014.

[50] H. Berninger. In-vehicle Information (IVI). In *Proceedings of the 7th Car 2 Car Forum, November 19-20, Munich, Germany*, 2013.

[51] NXP Semiconductors. NXP Delivers First RoadLINK Product. Press release. 2013.

[52] NXP Semiconductors. NXP Pushes Car-to-X market with Launch of New RoadLINK Solution. Press release. 2014.

[53] Cohda Wireless Pty Ltd. NXP and Cohda unveil new V2X device for connected vehicles. Press release. 2013.

[54] Freescale Semiconductor. Website. `http://www.freescale.com/webapp/sps/site/taxonomy.jsp?code=IMX6X_SERIES&cof=0&am=0`. [Online; accessed 08-Aug-2014].

[55] haneWIN DHCP Server für Windows. Website. `http://www.hanewin.de/dhcp-d.htm`. [Online; accessed 08-Aug-2014].

[56] Cohda Wireless Pty Ltd. CohdaMobility MK4A Radio Specification. 2013-11.

[57] ETSI TR 102 638 V1.1.1. Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Definitions. 2009-06.

[58] DRIVE C2X. Website. Use Cases. http://www.drive-c2x.eu/use-cases. [Online; accessed 09-May-2014].

[59] simTD. Field operational test carried out within research project simTD proves: Car-to-x communication is ready for everyday use. Press release. 2011.

[60] Federal Ministry of Transport and Digital Infrastructure BMVI. Cooperative ITS Corridor Joint deployment. 2013.

[61] AustriaTech - Federal Agency for Technological Measures Ltd. ITS corridor from Vienna to Rotterdam. www.smart-mobility.at, 2013. [Online; accessed 10-Feb-2014].

[62] ETSI TS 102 637-1 V1.1.1. Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part1: Functional Requirements. 2010-09.

[63] BlueZ - Official Linux Bluetooth protocol stack. Website. Supported Profiles. http://www.bluez.org/profiles. [Online; accessed 02-Sept-2014].

[64] Cohda Wireless Pty Ltd. MKx Wiki. Switching USB 2.0 OTG interface between Ethernet-Gadget Mode and Host Mode. http://mk2wiki.cohdawireless.com/pmwiki.php?n=Main.SwitchingUSBInterface. [Online; accessed 07-Aug-2014].

[65] Ubuntu. Manuals. hcid - Bluetooth Host Controller Interface Daemon. http://manpages.ubuntu.com/manpages/hardy/man8/hcid.8.html. [Online; accessed 07-Aug-2014].

[66] A. Huang. 2.3. Choosing a transport protocol. http://people.csail.mit.edu/albert/bluez-intro/x95.html. [Online; accessed 03-Sept-2014].

[67] A. Huang. Chapter 4. Bluetooth programming in C with BlueZ. http://people.csail.mit.edu/albert/bluez-intro/c404.html. [Online; accessed 02-Sept-2014].

[68] A. Huang. 4.2. RFCOMM sockets. http://people.csail.mit.edu/albert/bluez-intro/x502.html. [Online; accessed 02-Sept-2014].

[69] D. Steedman. E.1 What is ASN.1? http://www.bgbm.org/tdwg/acc/Documents/asn1gloss.htm. [Online; accessed 05-Sept-2014].

[70] ETSI TS 102 894-2 V1.1.1. Intelligent Transport Systems (ITS); Users and applications requirements; Part2: Applications and facilities layer common data dictionary. 2013-08.

[71] Volkswagen Group Electronic Research. Readme file for the Controller Area Network Protocol family (aka SocketCAN). https://www.kernel.org/doc/Documentation/networking/can.txt. [Online; accessed 11-Sept-2014].

[72] Freescale semiconductor. i.MX35 PDK 1.6 Linux. Reference Manual. 2009-11.

[73] Freescale semiconductor. i.MX35 (MCIMX35) Multimedia Applications Processor. Reference Manual. 2010-11.

[74] Google Inc. Google Maps Android API v2. https://developers.google.com/maps/documentation/android/start. [Online; accessed 25-Sep-2014].

[75] Cohda Wireless Pty Ltd. MKx Wiki. ETS-Shell: An MK2 example application. http://mk2wiki.cohdawireless.com/ets-shell/html/index.html. [Online; accessed 22-Sep-2014].

[76] ETSI EN 302 636-5-1 V1.2.0 Draft. Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 5: Transport Protocols; Sub-part 1: Basic Transport Protocol. 2013-10.

[77] Cohda Wireless Pty Ltd. MKx Wiki. ETS-Shell Usage. http://mk2wiki.cohdawireless.com/ets-shell/html/a00063.html. [Online; accessed 22-Sep-2014].

[78] ETSI EN 302 636. Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 1 - Part 6. 2013.

[79] Buburuzan T. Roadmap for joint deployment of cooperative ITS. In *Proceedings of the Amsterdam Group Workshop, September 25, Brussels, Belgium*, 2013.