Hamburg University of Technology (TUHH)
Institute of Communication Networks (ET6)

Master’s Thesis

Design and Failure Mode and Effects Analysis (FMEA) of a Vehicular Speed Advisory System

A Thesis by
Carsten Eric Nagel

05.06.2015

Examiners: Prof. Dr.-Ing. Andreas Timm-Giel
             Prof. Dr.-Ing. Carsten Gertz
Supervisors: Dipl.-Ing. Maciej Mühleisen
                Leo Krüger, M.Sc.
DECLARATION OF ORIGINALITY

I hereby declare that the work in this thesis was composed and originated by myself and has not been submitted for another degree or diploma at any university or other institute of tertiary education.

I certify that all information sources and literature used are indicated in the text and a list of references is given in the bibliography.

Hamburg, June 5, 2015

Carsten Nagel
ABSTRACT

“Design and Failure Mode and Effects Analysis (FMEA) of a Vehicular Speed Advisory System”
By Carsten Eric Nagel

The goal of this thesis is the development of a Vehicular Speed Advisory System, also called Green Light Optimized Speed Advisory (GLOSA) system. This GLOSA system displays a speed advice that a vehicle should drive to pass an upcoming traffic light at green. The purpose of this system is to optimize transportation efficiency and to save fuel and travel time through the use of Vehicle-2-X (V2X) communication technology.

The Systems Modeling Language (SysML) is used to analyze and design the GLOSA system. A System Analysis process sets up the stakeholders and requirements of the system, which are later used to design the system. The System Design process defines the blocks, interfaces and activities and visualizes them in SysML diagrams. The developed system uses V2X Communication technologies (e.g. IEEE 802.11p, ETSI ITS-G5) for the network communication and mobile communication for longer distances and emergency reaction.

Furthermore this thesis performs a Failure Analysis to evaluate how reliable the system would be. Therefor a Failure Tree Analysis (FTA) and a Failure Mode and Effects Analysis (FMEA) were done to analyze the reliability and possible failures.

A first demonstrator was implemented using Raspberry Pi’s. This demonstrator was designed to connect to a traffic light in Hamburg-Heimfeld later. Therefor the traffic light interface of Hamburg was examined, but since the GLOSA system was designed for a generic interface, there have been several changes necessary to connect the prototype to a real traffic light.

Furthermore the thesis gives a broad common overview about the topic, gives an outlook on potential future work and explains the next steps of the project.

Keywords
Green Light Optimized Speed Advisory (GLOSA), Speed Advisory System, Intelligent Transportation Systems (ITS), SysML, SYSMOD Approach, FTA, FMEA, Traffic Light Assist, System Engineering, V2X Communication, IEEE 802.11p, ETSI ITS, ITS-G5
TABLE OF CONTENTS

Abstract ........................................................................................................................................... III
Table of contents ............................................................................................................................ IV
Acronyms ........................................................................................................................................ VII
1. Introduction ................................................................................................................................. 8
   1.1. Motivation ............................................................................................................................ 8
       1.1.1. Advantages and Savings ............................................................................................... 9
       1.1.2. Hamburg Smart City ..................................................................................................... 9
   1.2. State of the Art ..................................................................................................................... 9
       1.2.1. Signal Guru .................................................................................................................. 10
       1.2.2. Audi Traffic Light Assist ............................................................................................ 10
       1.2.3. Green Light Optimized Speed Advisory (GLOSA) ....................................................... 10
       1.2.4. Cooperative ITS Corridor ........................................................................................... 11
   1.3. Thesis Structure ................................................................................................................... 11
2. Related Work .............................................................................................................................. 12
   2.1. V2X Communication ......................................................................................................... 12
       2.1.1. IEEE 802.11p Amendment .......................................................................................... 13
       2.1.2. ETSI ITS ..................................................................................................................... 16
   2.2. SAE J2735 Message Set ...................................................................................................... 20
       2.2.1. Message Types ........................................................................................................... 20
   2.3. Systems Modeling Language (SysML) .............................................................................. 22
       2.3.1. SysML Diagram Types ............................................................................................... 23
       2.3.2. SYSMOD Approach ................................................................................................. 25
   2.4. Failure Tree Analysis (FTA) .............................................................................................. 26
   2.5. Failure Mode and Effects Analysis (FMEA) ....................................................................... 27
   2.6. Existing Traffic Light Systems ......................................................................................... 28
       2.6.1. Traffic Light Interfaces ............................................................................................... 28
3. GLOSA System Analysis ............................................................................................................. 30
   3.1. Description of Project Context ......................................................................................... 30
   3.2. Possible System Architectures ........................................................................................... 31
       3.2.1. Central Service Center Approach .............................................................................. 31
       3.2.2. Local Infrastructure Approach .................................................................................. 32
   3.3. Requirement Analysis of the GLOSA system .................................................................... 33
       3.3.1. Stakeholders ............................................................................................................... 33
       3.3.2. Requirements ............................................................................................................. 34
   3.4. System Context Diagram ................................................................................................... 37
   3.5. Use Case Analysis .............................................................................................................. 38
4. GLOSA System Design ............................................................................................................. 40
   4.1. System Overview .............................................................................................................. 40
       4.1.1. Used Network Technologies ....................................................................................... 41
       4.1.2. Message Types ........................................................................................................... 42
   4.2. Central ITS Station ............................................................................................................. 42
# ACRONYMS

## Table 1: List of Acronyms

<table>
<thead>
<tr>
<th>Acronym Name</th>
<th>Acronym Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>AP</td>
<td>Access Point</td>
</tr>
<tr>
<td>ARINC</td>
<td>Aeronautical Radio Incorporated (Company)</td>
</tr>
<tr>
<td>ASN.1</td>
<td>Abstract Syntax Notation One</td>
</tr>
<tr>
<td>AU</td>
<td>Application Units</td>
</tr>
<tr>
<td>BER, DER</td>
<td>Basic Encoding Rules, Distinguished Encoding Rules</td>
</tr>
<tr>
<td>BSM</td>
<td>Basic Safety Message</td>
</tr>
<tr>
<td>BSS, BSSID, SSID</td>
<td>Basic Service Set, Basic Service Set Identification, Service Set Identification</td>
</tr>
<tr>
<td>BTP</td>
<td>Basic Transport Protocol</td>
</tr>
<tr>
<td>C2C-CC</td>
<td>Car-2-CAR Communication Consortium</td>
</tr>
<tr>
<td>CAM</td>
<td>Cooperative Awareness Message</td>
</tr>
<tr>
<td>COAP</td>
<td>Constrained Application Protocol</td>
</tr>
<tr>
<td>CSMA/CA</td>
<td>Carrier Sense Multiple Access/Collision Detection</td>
</tr>
<tr>
<td>DA</td>
<td>Destination Address</td>
</tr>
<tr>
<td>DCC</td>
<td>Decentralized Congestion Control</td>
</tr>
<tr>
<td>DENM</td>
<td>Decentralized Environmental Notification Message</td>
</tr>
<tr>
<td>DSRC</td>
<td>Dedicated Short Range Communication</td>
</tr>
<tr>
<td>ETSI</td>
<td>European Telecommunications Standards Institute</td>
</tr>
<tr>
<td>EVA</td>
<td>Emergency Vehicle Alert</td>
</tr>
<tr>
<td>FMEA</td>
<td>Failure Mode and Effects Analysis</td>
</tr>
<tr>
<td>FTA</td>
<td>Fault Tree Analysis</td>
</tr>
<tr>
<td>GLOSA</td>
<td>Green Light Optimized Speed Advisory</td>
</tr>
<tr>
<td>GUI, HMI</td>
<td>Graphical User Interface, Human Machine Interface</td>
</tr>
<tr>
<td>HS</td>
<td>Hot Spots</td>
</tr>
<tr>
<td>IBSS</td>
<td>IBSS (Independent BSS</td>
</tr>
<tr>
<td>IEEE</td>
<td>IEEE 802.11p Standard</td>
</tr>
<tr>
<td>ITS</td>
<td>Intelligent Transportation Systems</td>
</tr>
<tr>
<td>LDM</td>
<td>Local Dynamic Map</td>
</tr>
<tr>
<td>LLC</td>
<td>Logical Link Control</td>
</tr>
<tr>
<td>LSA</td>
<td>Lichtsignalanlage (Traffic Light Facility)</td>
</tr>
<tr>
<td>LSBG</td>
<td>Landesbetrieb Straßen, Brücken und Gewässer</td>
</tr>
<tr>
<td>MAC</td>
<td>Medium Access Control</td>
</tr>
<tr>
<td>MBSE</td>
<td>Model-Based Systems Engineering</td>
</tr>
<tr>
<td>OBU</td>
<td>On-Board Unit</td>
</tr>
<tr>
<td>PHY</td>
<td>Physical Layer</td>
</tr>
<tr>
<td>RA</td>
<td>Receiving STA Address</td>
</tr>
<tr>
<td>RPN</td>
<td>Risk Priority Number</td>
</tr>
<tr>
<td>RSU</td>
<td>Road Side Unit</td>
</tr>
<tr>
<td>SA</td>
<td>Source Address</td>
</tr>
<tr>
<td>SAE</td>
<td>Society of Automotive Engineers</td>
</tr>
<tr>
<td>SPAT</td>
<td>Signal Phase and Timing</td>
</tr>
<tr>
<td>STA</td>
<td>Station</td>
</tr>
<tr>
<td>SysML</td>
<td>Systems Modeling Language</td>
</tr>
<tr>
<td>SYSMOD</td>
<td>SYSMOD Approach</td>
</tr>
<tr>
<td>TA</td>
<td>Transmitting STA Address</td>
</tr>
<tr>
<td>TL, TLI</td>
<td>Traffic Light, Traffic Light Interface</td>
</tr>
<tr>
<td>VDA</td>
<td>Verband der Automobilindustrie</td>
</tr>
<tr>
<td>WAVE</td>
<td>Wireless Access in Vehicular Environments</td>
</tr>
</tbody>
</table>
1. **INTRODUCTION**

In our modern society traffic efficiency is getting more and more important. It will become even more essential in our future, since experts estimate that around 2050 75% of the world population will live in dense cities [1]. At the beginning of 2014 there have been roughly 62 million automobiles registered in the Federal Republic of Germany, with upward tendency [2].

This tendency led to the advent of cooperative Intelligent Transportation Systems (ITS), which could potentially improve road safety, traffic efficiency and also introduces new entertainment and business applications. The recent developments in wireless communication and particular in vehicular communication supports the advance of ITS systems. ITS include telematics and all kinds of communication in vehicles, between vehicles and between infrastructure and vehicles.

Over the past few years, the technology of connected and intelligent automobiles has been developed further very fast. This evolution of intelligent driving could for example also be observed at the Consumer Electronics Show (CES) 2015 in Las Vegas. This year, all the large and established car manufacturers announced their evolving plans of autonomous driving. **Audi** presented a self-driving model of the A7, which travelled 550 miles to Las Vegas almost autonomously [3]. **Mercedes** introduces their future of mobility, a concept car, which can drive completely independent and offers a lounge like interior with rotating chairs [4]. Earlier **Google** introduced a self-driving car completely without a steering wheel [5].

1.1. **MOTIVATION**

Not only autonomous driving is developing, but also vehicular communication is becoming more and more important. ITS could bring a huge amount of benefits for every vehicle driver and especially to all autonomous cars. This thesis will develop a Vehicular Speed Advisory System for traffic lights – also called Green Light Optimal Speed Advisory (GLOSA) – which is an important part of an efficient ITS.

The GLOSA system will calculate an optimal speed at which the vehicle could probably pass the next traffic light without stopping. When a vehicle approaches a traffic light, it receives information regarding the location of the intersection and the signal phase and timing. With this information and its own position the vehicle can calculate a speed advice. The vehicle could either notify the driver of the optimal speed or notify the autonomous car system directly. The main purpose of this system is to prevent a stop-and-go situation, reduce the fuel consumption and CO2 emission and thus lead to a more efficient infrastructure.
1.1.1. Advantages and Savings

The requirements for fuel consumption and CO2 Emission are becoming more and more strictly to fight the global climate change. The GLOSA system may help to accomplish those goals. Research shows that it can save up to 20% of fuel and 17% of stop time [6]. Vehicles could use the green wave to prevent stop and goes. The fuel and stop time savings depends on the density of vehicles that can receive the GLOSA information; a performance evaluation can be found in [7].

Minimizing the stop-and-go driving patterns can significantly improve the economic; about twenty percent more fuel will be consumed to accelerate from a full stop, than from eight kilometers per hour. A stop-and-go pattern is often caused by traffic lights; hence the improvement of traffic signalization is a key requirement for fuel-efficient driving.

Autonomous vehicles and navigation systems of the future could also use the signal timing information to derive faster routes. The start-stop-automatic of vehicles may become more efficiently and turn the motor on just before the traffic light switches green.

1.1.2. Hamburg Smart City

In 2014 a Memorandum of Understanding [8] of Hamburg (FHH), Hamburg Port Authority (HPA) and Cisco Systems was signed. The project Hamburg Smart City wants to make the city more innovative, intelligent and interconnected. The city needs to become more smart and citizen friendly.

Since Hamburg is already a very dense city it would not be possible to extend most of the available infrastructure. Hamburg Smart City should improve the present infrastructure to become more efficient and dynamic. Especially the harbor is a very important field, since it is a significant source for the economy of Hamburg. Hence the project smartPORT will develop the harbor further.

Example pilot projects of Hamburg Smart City are:

- An intelligent control of traffic lights and street lights
- An optimized acquisition and controlling of traffic
- An intelligent parking system for trucks in the harbor of Hamburg
- A more virtualized and interconnected communication with administrative bodies

1.2. State of the Art

There are already some ideas and prototype systems that offer some kind of intelligent traffic light assistance. Some interesting systems are described below. Further projects are for example: Project Travolution\(^1\), Project Kolibri\(^2\) and the Green Wave in Wolfsburg [9].

\(^1\) http://travolution-ingolstadt.de
\(^2\) http://www.kolibri-projekt.de
1.2.1. **Signal Guru**

One project that offers GLOSA capabilities is *Signal Guru* [10], a smartphone app of the *Massachusetts Institute of Technology* (MIT). The smartphone needs to be positioned directly behind the windshield and the app uses the smartphone camera to collect data of passed traffic lights. This data will be saved in a database and is used to determine the traffic light signal phases. When the system has collected sufficient data the app can display the remaining red time and an optimal speed to pass green lights.

1.2.2. **Audi Traffic Light Assist**

Audi’s *Traffic Light Assist* was introduced at the CES 2014, where journalists could participate in a live preview on the roads of Las Vegas. The *Traffic Light Assist* should help the driver to hit every green light and displays the optimal speed to pass a traffic light in the central car display. Furthermore, the system is also connected to the start-stop-automatic and will turn on the motor right before the traffic light switches green.

Audi informs that the system has been tested in Ingolstadt, Berlin, and Las Vegas, but right now they do not provide much information about the traffic light data collection. According to an IEEE paper [11] Audi designed a system that sends the signal timing from a central traffic light database via LTE. This concept is much faster to realize than a complete system that needs roadside infrastructure. Further disadvantages with devices at each traffic light are the necessity for long-lasting hardware and that the number of controlled intersections is very high. Since every city has its own communication protocols it will take some time to build a dense network of signaled traffic lights.

1.2.3. **Green Light Optimized Speed Advisory (GLOSA)**

Veselin Georgiev wrote his diploma thesis about the “Design and Implementation of a Green Light Optimized Speed Advisory Visualization for Android” [12]. The first part of the thesis is about the design and the evaluation of the graphical user interface on a smartphone (Section 4.4.5, Figure 32). Different GUIs have been evaluated by an opinion poll that regards readability, distraction, and understandability. Furthermore, the thesis explains the implementation and which message types are used.

The developed GLOSA app connects via WLAN to an onboard vehicle unit (a IEEE 802.11p Receiver) which forwards the necessary Vehicle-to-X (V2X) messages that it receives from the ITS. So the vehicle itself receives the data from the traffic light and the smartphone displays it to the driver. The underlying infrastructure of the system is explained in [13].

The thesis of Georgiev also establishes a first simple algorithm for the calculation of the optimal speed – which is further described in Section 4.4.6 – and a basic idea how to determine the lane on which the vehicle is approaching the intersection – described in Section 4.4.4.1.
1.2.4. COOPERATIVE ITS CORRIDOR

The Cooperative ITS Corridor [14] is a joint venture project of Germany, Austria and the Netherlands. It is a research and development test project that investigates cooperative ITS systems.

The system is based on the ETSI ITS infrastructure and the ETSI ITS-G5 communication standard, which are further explained in Section 2.1.2 and installs V2X communication units on a over 1300 km long street from Wien (Austria) over Frankfurt (Germany) to Rotterdam (Netherlands). These infrastructure units should supply vehicles with information about obstacles and hazards before they can see it. Furthermore it collects data about the traffic situation and provides a basis to optimize the traffic flow (includes traffic light optimization and GLOSA applications).

Many different companies (e.g. NXP Semiconductors, Siemens, Honda, Cohda Wireless, TÜV Süd) are working together to prepare the corridor for public use in 2015. For example the corridor include ITS test fields in Munich, Vienna and Helmond fitted with intelligent traffic infrastructure from Siemens.

1.3. THESIS STRUCTURE

The GLOSA system needs signalized intersections and traffic lights that transmit the MapData and the Signal Phase And Timing information (SPAT) for each direction of travel and each lane. The Vehicle-to-X Communications standards that are necessary for this information exchange will be described in Section 2, which summarizes the related work. This section also gives information about the modeling language SysML, the Failure Tree Analysis (FTA) and the Failure Mode and Effects Analysis (FMEA).

Section 3 analyses the specification and requirements of the GLOSA system and Section 4 designs the actual system. The next step is the Failure Analysis including the FMEA and FTA in Section 5.

A part of this project was also to implement a first demonstrator of the GLOSA system that should later be connected to a traffic light in Hamburg-Heimfeld; it is described in Section 6. A first test run of the system was recorded and summarized in Section 7.

The last section outlines the results of the thesis and shows, which parts of the project should be part of future research.
2. RELATED WORK

This section presents related work and explains techniques that have been used by this thesis. It explains the basic principles of Vehicle-to-X (V2X) communication and gives details about the Institute of Electrical and Electronics Engineers (IEEE) 802.11p standard. Furthermore it describes an ITS proposed by the European Telecommunications Standards Institute (ETSI). Moreover it gives a short overview about the Systems Modeling Language (SysML), the Failure Tree Analysis (FTA) and the Failure Mode and Effects analysis (FMEA).

2.1. V2X COMMUNICATION

V2X communication is the interaction among cars, between infrastructures and cars, and vice versa. It can be split up in at least two variants: the first one is the Vehicle-to-Vehicle (V2V) communication, the second one the Vehicle-to-Infrastructure (V2I) communication. For V2X communication vehicles have to be equipped with special communications devices; but the communication can also be supported by mobile communication technologies like UMTS or LTE.

One of the first research institutions for V2X communication that was founded in Europe is the CAR 2 CAR Communication Consortium (C2C-CC). Their main mission is to develop an open European standard for ITS, develop a roadmap for the deployment of this ITS and associate a validation process that focuses on V2X systems. Furthermore the C2C-CC also supports several test projects in Europe.

The consortium published a manifesto in 2007 [15], which includes the prerequisites and constraints (e.g. anonymity and data security, an effective protected frequency band,
scalability and mandatory sensor data). It also describes the use cases of ITS and describes first requirements of the protocol stack (which is closely related to the ETSI protocol stack, Section 2.1.2). It also estimates that a V2X based ITS will be economic when 5% to 10% of cars are equipped with the necessary communication devices.

The main candidate for V2X road safety applications by the C2C-CC in Europe is the ETSI ITS standard, which is explained in Section 2.1.2. The C2C-CC is a main stakeholder of the ETSI ITS system. This standard relies on the IEEE 802.11p amendment (Section 2.1.1), but differs for example slightly in the frequency band.

Figure 1 describes the architecture of the proposed system by the C2C-CC, which is split up into three domains:

- The **In-Vehicle Domain** is represented by an *On-Board Unit* (OBU) and one or more *Application Units* (AU); the OBU is used for the incoming and outgoing communication of the vehicle, whereas the AU runs the application that uses the V2X data. Both units can be combined to one single unit.
- The **Ad-hoc Domain** consists of the OBUs and *Roadside Units* (RSU), which are infrastructure units that are located along the street and support/extend the V2X Communication. The *Ad-hoc Domain* describes the communication between these devices.
- The **Infrastructure Domain** describes the network structure behind the RSUs and Hot Spots (HS). RSUs are interconnected to each other and also allow a connection to the Internet.

### 2.1.1. IEEE 802.11p AMENDMENT

IEEE 802.11p is an amendment to the IEEE 802.11 standard, which adds wireless communication for vehicular system. It defines several enhancements that support ITS applications and is designed to exchange data between high-speed vehicles and roadside infrastructure. In the US it is also sometimes referenced to as *Dedicated Short-Range communication* (DSRC), but in Europe it falls under the ETSI ITS-G5 standard and DSRC is commonly known as a standard for toll collection.

The amendment defines several enhancements to IEEE 802.11 for vehicular communication. Vehicular environments set up new requirements to wireless communication systems. IEEE 802.11p is designed for 1000 m line of sight and a vehicle speed of up to 200 km/. Since the link between vehicles might only exist for a short amount of time it is necessary to keep latencies as short as possible. Further details can be found in the following two section that explain the Physical (PHY) and Medium Access Layer (MAC) layer.

IEEE 802.11p is the lower layer that is used by the ETSI ITS standard (Section 2.1.2) in Europe and the American *Wireless Access in Vehicular Environments* (WAVE, IEEE 1609) system. These two standards describe complete ITS systems and further specify their network architectures, routing algorithms and services.
Note: In March 2012 the IEEE has compiled a new version of the 802.11 standard, where all approved amendments that have been produced between 2007 and 2011 (including IEEE 802.11p) have been enrolled in the base standard. This new version is called IEEE 802.11-2012 and can be found in [16]. Due to this new version of 802.11, the 802.11p amendment is classified as superseded. In this thesis IEEE 802.11p means 802.11-2012, which is set up to communicate without associating to an Access Point, the details are explained in the following two sections.

2.1.1.1. MAC Layer of IEEE 802.11p

At first, this paragraph provides a short overview about setting up usual a IEEE 802.11 connection; afterwards the changes of the IEEE 802.11p amendment are described.

In a simplified description the IEEE 802.11 MAC layer arranges a set of radio stations to establish and maintain a group of communicating radios. These stations can freely communicate with each member of this group, but all transmissions from outside stations will be filtered out. Such a group is called a Basic Service Set (BSS) and there are many protocol mechanisms that provide secure and robust communication within a BSS.

IEEE 802.11 communication normally uses BSS to set up different wireless networks that are anchored by an Access Point (AP). The BSS allows MAC layer mechanisms to control the access to this network and also allows them to filter out transmissions from unrelated radios nearby. A radio first listens for beacons from an AP and then joins the BSS with a handshake mechanism, which includes the authentication and association.

The Ad-hoc mode of IEEE 802.11 follows a similar establishment process, which uses IBSS (Independent BSS). The Ad-hoc mode is however still too complex to be suitable for vehicular communication. These two infrastructure modes, the BSS and IBSS are shown in Figure 2.

The name of the wireless hotspot, which people can observe, is called the Service Set Identification (SSID). The SSID information is between 0 and 32 Bytes long.

The SSID should not be confused with the Basic Service Set Identification (BSSID), which is the identifier of a BSS at the MAC level and is a 48-bit long field (Length of MAC address).
Each BSS must have a unique BSSID (ensured by using the MAC address of the AP), which is shared by all members of the group. At the MAC level BSSID filtering is used to restrict all incoming frames to only those that have been transmitted from members of the same BSS.

Each IEEE 802.11 data frame (Figure 3) includes up to four address fields, which can contain the Source Address (SA), Destination Address (DA), Transmitting STA Address (TA), Receiving STA Address (RA) or the BSSID. The use of these four address fields differs according to the “To Distribution Service (DS)” and “From DS” bits in the Frame Control field and is further explained in Table 2.

<table>
<thead>
<tr>
<th>To DS</th>
<th>From DS</th>
<th>Address 1</th>
<th>Address 2</th>
<th>Address 3</th>
<th>Address 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>RA = DA</td>
<td>TA = SA</td>
<td>BSSID</td>
<td>N/A</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>RA = DA</td>
<td>TA = BSSID</td>
<td>SA</td>
<td>N/A</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>RA = BSSID</td>
<td>TA = SA</td>
<td>DA</td>
<td>N/A</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>RA</td>
<td>TA</td>
<td>DA</td>
<td>DA</td>
</tr>
</tbody>
</table>

The MAC layer of a station uses the content of the Address 1 field to perform an address matching for receiving decisions. If the Address 1 field contains a group address (e.g. broadcast), the BSSID is compared to ensure that the broadcast originated from a station in the same BSS.

For vehicular environments the standard IEEE 802.11 MAC operations are too time consuming. Vehicular use cases demand instantaneous data exchange capabilities. Because of this requirement authentication and association procedures are removed in IEEE 802.11p. Scanning for beacons of a BSS and executing handshakes to establish a connection, is too time consuming.

To allow a faster communication it is predefined that all IEEE 802.11p radios are configured to communicate on the same channel and with the same BSSID. A key amendment of IEEE 802.11p is that a station is allowed to transmit and receive outside the context of a BSS. For the BSSID the wildcard (all bits set to “1”) value is used. This means that two vehicles can immediately communicate with each other, without any additional overhead as long as they operate in the same channel and use the wildcard BSSID. Furthermore the “To DS” and “From DS” values of the Frame Control are both set to “0”.

By setting the MIB (Management Information Base) variable dot11OCBActivated to true, IEEE 802.11-2012 communication outside the context of a BSS is possible. When dot11OCBActivated is true, a data frame can be sent to either an individual or a group destination MAC address. Further explanations are in the IEEE 802.11-2012 standard [16] in Section 4.3.11. This type of communication allows an immediate exchange of data frames.
and avoids the management overhead that is used by the establishment of a network. Communication outside the context of a BSS enables exchange of data frames between stations that are not members of a BSS. This type of communication allows for immediate exchange of data frames, avoiding the latency associated with the establishment of a BSS.

The Logical Link Control (LLC) is not further explained in detail here, but it is defined by ANSI/IEEE Std. 802.2 and the mode of operation is set to Type 1: unacknowledged connectionless mode. The simplified CSMA/CA process is described [18]; more details about the channel access procedure are found in clause 9 of IEEE 802.11-2012 [16].

With the explained techniques a short latency can be achieved, to support vehicular safety applications. However since there is no network association, any required authentication and security services have to be provided by higher layers.

2.1.1.2. Physical Layer of IEEE 802.11p
The Physical Layer of IEEE 802.11p amendment was designed to make only the minimum necessary changes to the IEEE 802.11a Physical Layer. Large changes in the Physical Layer should be avoided because of necessary hardware changes; therefor more changes are made in the MAC layer, where they are practically software updates only and easier to deploy to the existing devices.

The Physical layer is similar to that of IEEE 802.11a and operates at 5.9 GHz, which is very close to the 5 GHz of IEEE 802.11a. More details about the frequency allocation in Europe can be found in Section 2.1.2.1.

The Physical layer of 802.11p also adopts the orthogonal frequency-division multiplexing (OFDM, clause 18 of IEEE 802.11-2012 [16]) transmission technique. However, the bandwidth of a single channel in IEEE 802.11p is scaled down to 10 MHz, which is half the bandwidth of 802.11a and leads to a maximum data rate of 27 Mbit/s (when 64-QAM modulation is used). The main reason for this change is to address problems with the increased delay spread in the vehicular environments. Studies [19] showed that the guard interval at the 20 MHz bandwidth is not long enough to offset the worst-case delay spread (e.g. the guard interval does not prevent inter-symbol interferences within a radios own transmissions).

2.1.2. ETSI ITS
The European Telecommunications Standards Institute (ETSI) establishes a European standard for vehicular communication. It published several papers that standardize a complete Intelligent Traffic System (ITS), which is basically based on the IEEE 802.11p amendment. Standardization is an essential part in vehicular communication systems that should prevent proprietary systems and ensure interoperability between different manufacturers. As always, there is also a system that tries to establish a different standard: the American competitor system WAVE (Wireless Access in Vehicular Environments, IEEE 1609).
The ETSI ITS describes a complete system for vehicular communication. It specifies the network architecture, the routing protocols and services that could be managed by the system and application. For this purpose it introduces a few protocols; a shortened protocol stack overview of the ETSI ITS is shown in Figure 4, further information can be found in [20].

### 2.1.2.1. **Access Technologies Layer**

The *Access Layer* of the ETSI ITS is heavily based on the IEEE 802.11 communication standard and particularly on the IEEE 802.11p amendment. In the ETSI ITS this communication functionality is called ITS-G5 and makes some minor changes to IEEE 802.11p (e.g. small changes of the frequency band). The *Access Layer* describes the Physical (PHY) and Medium Access (MAC) Layer. The specification can be found in [18], which already adopts the changes of the IEEE 802.11-2012 standard.

![Figure 4: Shortened ETSI ITS Protocol Stack](image)

It adapts the IEEE 802.11p standard to the European frequency spectrum: 5.855-5.925 GHz; for *ITS Road Traffic Safety* application there is one 10 MHz wide control channel (G5-CCH) and two 10MHz wide service channels (G5-SCH), for *ITS Traffic Efficiency* applications there are two 10 MHz wide service channels (Figure 5 and Table 3).
Table 3: ITS-G5 European channel allocation

<table>
<thead>
<tr>
<th>Channel Type</th>
<th>Frequency range (MHz)</th>
<th>IEEE channel number</th>
</tr>
</thead>
<tbody>
<tr>
<td>G5-CCCH</td>
<td>5896 to 5905</td>
<td>180</td>
</tr>
<tr>
<td>G5-SCH2</td>
<td>5885 to 5895</td>
<td>178</td>
</tr>
<tr>
<td>G5-SCH1</td>
<td>5875 to 5885</td>
<td>176</td>
</tr>
<tr>
<td>G5-SCH3</td>
<td>5865 to 5875</td>
<td>174</td>
</tr>
<tr>
<td>G5-SCH4</td>
<td>5855 to 5865</td>
<td>172</td>
</tr>
<tr>
<td>G5-SCH7</td>
<td>5470 to 5725</td>
<td>94 to 145</td>
</tr>
</tbody>
</table>

Mobile communication networks like UMTS or LTE are also considered as an access technology and may help to bridge long distances. This could be important in the early stage, when only a few cars are equipped with V2X communication units [7], but can also be used to interconnect the RSUs with a Central Control Center.

2.1.2.2. NETWORK AND TRANSPORT LAYER

The network and transport layer of the ETSI uses two main protocol stacks for communication. Which transport protocol stack is used, depends on the type of data and the route the data should take:

It uses either the usual protocols IPv4/IPv6 and TCP/UDP for all kind of data that is directly requested like Infotainment. Or the vehicular communication protocols Geonetworking and Basic Transport Protocol (BTP), which are used for safety and traffic efficiency applications, where data is broadcasted or directed to specific geographical areas. However traffic efficiency applications may also use both protocol stacks, depending on their specific needs.

Geonetworking [23] allows sending packets to geographical positions and areas. The protocol supports multi-addressing modes [22] (Point-to-Multipoint, GeoBroadcast,
GeoAnycast, see Figure 6) and specifies several routing protocols [24] (basic greedy forwarding and advanced forwarding scheme) to allow a multihop transmission. Geonetworking is used for all kinds of safety applications (CAM, DENM, see Section 2.1.2 and 2.2) and also partially for traffic efficiency applications.

The Basic Transport Protocol (BTP) [25] is a connection less protocol with low overhead. It is designed to work together closely with Geonetworking and supports applications by specifying ports and the length of data.

2.1.2.3. Facilities Layer

The Facilities Layer of the ETSI ITS supports the applications in several ways. This support is split into three parts: the Application Support, the Information Support and the Communication Support. The Application Support helps with services, messages, Local Dynamic Map (LDM) management and security accesses. The Information Support supplies access to the LDM, presents data, gives location referencing and the stations type and capabilities. The Communication Support helps with the addressing, Geonetworking and session support.

The Local Dynamic Map (LDM) stores and maintains all data that is received by the OBU of the vehicle. Applications can retrieve relevant data from the LDM instead of managing them by themselves. The LDM brings a standardized API for third party applications, which allows subscription/notification, but also direct queries. A filtering mechanism supports the applications by retrieving recent and relevant data. It can also store processed information from applications so that applications can use it to share results among themselves.

The Facilities Layer also defines the different message types of the ETSI ITS and how they will be sent (periodic, on-event, unicast, multicast, geocast). The ETSI ITS standard inherits messages from the SAE J2735 message set, but also defines two new messages: the Cooperative Awareness Message (CAM) [26] – a periodic heartbeat of every present car – and the Decentralized Environmental notification Message (DENM) [27] – an event-driven hazard warning (accidents, traffic, and weather). All other message types have been already defined by the SAE J2735 standard, which is explained in Section 2.2. This standard includes the Signal Phase and Timing (SPAT) and the MapData message, which are important for this project.

2.1.2.4. Applications Layer

The Applications Layer defines a basic set of applications and use-cases [28] for Road Safety, Traffic Efficiency and other applications. For example the Traffic Efficiency use case specifies a Traffic Light Optimal Speed Advisory System, which is comparable to the GLOSA system, this use case is further used during the requirement analysis (Section 3.3).

2.1.2.5. Quality of Service & Security

The ETSI ITS also specifies a Quality of Service (QoS) management, which is called Decentralized Congestion Control (DCC) [29]. The DCC is spread across all layers (the
Management layer) so that it can make adjustments to every communication layer. The DCC is a media dependent functionality and make optimizations based on the estimated data traffic volume. For example: Dynamic DCC profiles specify the message transmission, based on channel congestion level. It controls the transmission rate, transmission interval, transmission power, the channel and the traffic class. The security management of the ETSI ITS is also spread across all layers, however it is not important for this thesis and thus not further explained.

2.2. SAE J2735 Message Set

The ETSI ITS only specifies two message types, but uses a lot more. The Society of Automotive Engineers (SAE) defined the complete message set that is used by the system. It is a message set designed for Dedicated Short Range Communication (DSRC) and is also used by the American WAVE (IEEE 1609) standard for vehicular communication. It supports interoperability among DSRC applications and standardizes a message set, data frames and data elements by defining them with the Abstract Syntax Notation One (ASN.1) syntax.

There are several ways to encode ASN.1 messages. The messages are encoded with the Distinguished Encoding Rules (DER), which is a variant of the Basic Encoding Rules (BER). With the ASN.1 syntax and the DER it was possible to construct very compact messages, which meet the requirements of the ETSI ITS. A Basic Safety Message (BSM) for example is around 300 Byte long (comparable to the Cooperative Awareness Message, which is around 800 Byte). As much overhead as possible should be prevented because the data rate of IEEE 802.11p can be insufficient in highly populated areas, where a lot of cars send out a heartbeat message [30].

More important for this thesis are the two message types Signal Phase and Timing (SPAT) and MapData. The SPAT message supplies information about the actual traffic light state on each lane and after what time this state will change. The MapData delivers information about the topology and the physical structure of the intersection. SAE J2735 also proposes the transmission interval for these messages. However the ETSI ITS redefines these timings and may change them by estimating the actual data traffic (DCC).

2.2.1. Message Types

The SAE J2735 message set defines several messages that are listed in the Specification [31] and are further described in an Implementation Guide [32]. Along with the Implementation Guide SAE also provides the ASN.1 Specification and a Library for C. Of course it is also possible to use a suitable compiler to compile the ASN.1 specification to other languages like JAVA or C#.

One message is the Basic Safety Message (BSM), a periodic heartbeat that can prevent collision. However more important for this thesis are the SPAT message and the MapData message, which are explained in the following sections.
2.2.1.1. **Signal Phase and Timing Message (SPAT)**

The *Signal Phase and Timing* (SPAT) message contains the information about the signal light state. The structure of a SPAT message is drawn in Figure 7 and includes the message type and a list of *Intersection States*. Each *Intersection State* entry contains an *Intersection Id* the actual *status* and a sequence of *Movement States*. Optionally the *Intersection States* can contain a timestamp, the lane count, the priority and active preemption state data.

A *Movement State* is a possible move that a vehicle can make and contains a set of lanes and their light status (either for motorized lanes, pedestrian signal light states or special lane states). It also describes the remaining time of this light state and optionally information about the next state and a confidence.

### 2.2.1.2. MapData Message

The MapData message contains the geographical data of an intersection. It contains a *Reference Position* (longitude and latitude) and describes, which *Approaches* (and Egresses) the intersection has. An *Approach* describe how a vehicle can approach the intersection, an *Egress* how it can leave the intersection. It also contains the lane number and a node list that specifies the offset to the *Reference Position*. The structure of the message is shown in Figure 9 an example intersection is sketched in Figure 8.

The MapData message can also describe connections to other intersections, so that the vehicle can already find the next intersection and may request the data. Furthermore it can also describe road signs, roadside furniture and roadway geometry like curves.
2.3. **SYSTEMS MODELING LANGUAGE (SysML)**

The *System Analysis* and *System Design* of the GLOSA system is achieved by applying the *Systems Modeling Language* (SysML) [33]. This section will give a short explanation of SysML, which is used for the *System Analysis* and *System Design* process of the GLOSA system in Section 3 and 4.

SysML is a graphical modeling language for systems engineering applications, which is based on the *Unified Modeling Language* (UML) standard. It supports the visualization of the analysis of a system, and furthermore the specification, design, verification and validation. An overview of the SysML diagram types and the differences to UML 2 is shown in Figure 10.

The SysML dialect was drafted in 2003 as an open source project and has been adopted by the *Object Management Group* (OMG) as OMG SysML. Since then it has evolved into an
important standard for Model-Based Systems Engineering (MBSE) applications. SysML is for example supported and enhanced by Artisan Software Tools, IBM, Motorola and also Lockheed Martin or oose Innovative Informatik GmbH.

2.3.1. SysML Diagram Types

The following sections explain some of the SysML essentials, diagram types and relationships that have been used in this thesis. More information can be found in the SysML specification [33] or in the graphical notation overview of [34].

2.3.1.1. Stereotypes

A stereotype is an extensibility mechanism in SysML (also in UML), which allow extending the vocabulary of SysML in order to create new model elements, derived from existing ones. For example stereotypes allow to make new categories of blocks in the SysML Block Definition Diagram (Section 2.3.1.4), e.g. system, subsystem or routers. A stereotype is illustrated by guillemets (« » or « »). Actually block is already a stereotype that extends UML types.

2.3.1.2. Use Case Diagram

The Use Case Diagram is a graphical diagram that illustrates user interaction with a system. A Use Case Diagrams is a part of UML and not newly defined by SysML. It shows the user/actor and the different use cases in which he is involved. The diagram type is used by the GLOSA System Analysis in Section 3.5.

2.3.1.3. Requirements Diagram

The Requirement Diagram is a diagram type that is completely new defined by SysML. It describes the requirements of a system graphically and shows how they depend on each other. Several relations can be used to illustrate these relationships. A short example with descriptions is given in Figure 11.

![Figure 11: SysML Requirement Diagram [34]](image)

The Requirement Diagram was used in Section 3.3.2, to visualize and group the requirements of the GLOSA system. It is one of the main advantages to use SysML for systems engineering; UML does not provide a straightforward mechanism to capture system requirements.
2.3.1.4. **Block Definition Diagram**

A Block Definition Diagram visualizes the definition of system blocks and their relationships. It is an important diagram of SysML and is derived from the UML Class Diagram. A sample overview is drawn in Figure 12. The diagram type is used during the system design process in Section 4.1 to visualize the components of the GLOSA system.

*Figure 12: SysML Block Definition Diagram [34]*

Block Definition Diagrams are also used for the System Context Diagram (Section 3.4) and the System Overview (Section 4.1). They are proposed by the SYSMOD Approach – which is explained in the next section – and are expanded by stereotypes.

2.3.1.5. **Internal Block Diagram**

An Internal Block Diagram is used to visualize the internal structure of a block in terms of its parts, ports and connectors. The Blocks that have been defined by the Block Definition Diagram are here used as Parts, which are interconnected with each other. A sample diagram with several descriptions is illustrated in Figure 13.

*Figure 13: SysML Internal Block Diagram [34]*

The Internal Block Diagrams is used during the System Design of the GLOSA system in Section 4.2.1. It specifies how the internal parts of each subsystem are interconnected with each other.

2.3.1.6. **Activity Diagram**

An Activity Diagram is a graphical workflow representation of stepwise activities and actions. It supports choices, iterations and concurrents. Activity Diagrams are already defined in
UML, but further extended by SysML. An example overview is sketched in Figure 14. Some of the example actions are extended by stereotypes that have been added by the SYSMOD Approach (Section 2.3.2).

Activity Diagrams are used in Section 4.2.2 to visualize the execution process of the GLOSA applications/software. It was used because it simplifies the visualization of the parallel processes of the GLOSA system; the alternative choice was to use a Sequence Diagram.

![SysML Activity Diagram](image)

**Figure 14: SysML Activity Diagram [34]**

### 2.3.2. SYSMOD Approach

This thesis basically follows the proposed system development process of the SYSMOD Approach [35]. The SYSMOD Approach is a complete guideline how to develop and realize a system with SysML. The SYSMOD Approach proposes a System Analysis step and a System Design step, which are further explained in the following section; Section 3 and 4 use these proposed processes to realize the GLOSA system.

The SYSMOD Approach also extends the SysML dialect by defining several new stereotypes and derived diagrams. For example the System Context (Section 3.4) is a newly derived diagram, which uses a Block Definition Diagram to give a clearer overview of the system and its interaction with its environment. A graphical notation overview of the SysML and the SYSMOD Approach can be found in [34].

### 2.3.2.1. System Analysis Process

The first step of the System Analysis is to describe the Project Context or the scope of the project; afterwards it specifies the requirements of the system. For the requirements it is necessary to find the stakeholders – the persons who have interest in the system – and gather their needs and requirements to the system. After finding the requirements of the system the next step is to set up the System Context and start modeling the Use Cases. The
System Analysis process of the GLOSA system is performed in Section 3. The process is described graphically in Figure 15.

In parallel to the analysis and design process it would be obligatory to create a glossary, where all abbreviations and words that might need further description are listed.

![Figure 15: Simplified System Analysis (left) and System Design (right) [35]](image)

### 2.3.2.2. System Design Process

When the System Analysis step is completed the results can be used to actually realize the desired system. The Use Cases and the System Context are used to derive the system/actor interaction and the interfaces of the system. When this step is done the structures (Block Definition Diagram and Internal Block Diagram) of the system can be realized and further supported by a state or activity model. The System Design process of the GLOSA system is described graphically in Figure 15 and later performed in Section 4.

### 2.4. Failure Tree Analysis (FTA)

The Failure Tree Analysis (FTA) is a top down approach to analyze possible failures of a technical system. It is based on Boolean logic to calculate up the failure probability of a part or the complete system. A short example calculation is drawn in Figure 16. It is used to understand the logic (failure chain) that is leading to the top event and further to identify the best actions to reduce the failure probabilities.

The FTA is used by aerospace systems (SAE ARP 4761), nuclear power systems (NUREG–0492 [36]) and in car and other industries (e.g. IEC 61025 [37], DIN 25424 [38]). It is also used in software engineering for debugging purposes [39].
The procedure of the FTA is further explained in Section 5.6, which also performs an example FTA analysis of the GLOSA system and follows the process and methods that are described in [40].

2.5. FAILURE MODE AND EFFECTS ANALYSIS (FMEA)

The Failure Mode and Effects Analysis (FMEA) is an engineering method that helps to identify weak points during the concept and design phase of all kinds of products (hardware, software) and processes. It is mainly a qualitative analysis, which shows how reliable the designed system is.

The FMEA is widely used in development and manufacturing industries and there are plenty of different norms, e.g. Deutsche Gesellschaft für Qualität (DGQ) Band 13-11, SAE J1739 for the automobile industry or IEC 60812 for electronic devices. Many companies also develop their own FMEA sheets.

The FMEA can be split up into multiple types: e.g. Design-FMEA (DFMEA), System-FMEA (SFMEA), Hardware-FMEA, Software-FMEA and Processes-FMEA (PFMEA).

A successful FMEA helps to identify potential failures based on experience with similar products and processes – or based on common physics or logic. The goal of the FMEA is to prevent these failures as early as possible and before they happen. Potential failures could then be avoided by alternative designs or redundancies.

The FMEA should already be done in an early stage of a product development cycle, so already in following to the design stage. It is easier to solve problems as early as possible and the costs for problems will be much higher in later design stages. The process should involve as many different team members of different company departments as possible. Each department will have their own ideas and experiences.
The FMEA of this thesis is further described in Section 5. The proposed Failure Analysis starts with a Cause and Effect Diagram as a first method to find possible failure points and goes on with the 5 steps that the Verband der Automobilindustrie (VDA) proposes in their VDA Band 4 [41]. In this thesis an SAE J1739 FMEA sheet was used to analyze the GLOSA system.

2.6. EXISTING TRAFFIC LIGHT SYSTEMS

The existing operational infrastructure of traffic lights is not easy to summarize. There are many different road operators that manage the local infrastructure and even in Germany each city has its own implementation. Even in one specific city there may be several different types of traffic lights from different distributors, every city has its own local devices, proprietary interfaces and communication protocols.

Most of the supplied infrastructure in Germany is not designed for a GLOSA use case and does not have the possibility to get the necessary data from the intersection controller (e.g. map layout, forecast, remaining time). However, sometimes it is also possible to get full information (e.g. detector data and push button requests). A further problem is, that there is no nationwide traffic light central control center; there is sometimes even more than control center in one city.

This thesis was done in cooperation with the Landesbetrieb Straßen, Brücken und Gewässer (LSBG) and the traffic light operator Hamburg Verkehrsanlagen (HHVA). Therefore it used the traffic light system of Hamburg as an example, which is further explained during the description of the implemented demonstrator in Section 6.1.

2.6.1. TRAFFIC LIGHT INTERFACES

The simplest case for a traffic light is the fixed timing case where each cycle behaves the same. For this simple case it might be easily possible to create a forecast. However there are also more complex traffic lights that are equipped with pedestrian detectors, cameras or communicate with public transportation.

The GLOSA system requires a lot of data from a traffic light. The needed data can be derived from the SAE J2735 message set and consists for example of the following data:

**SPAT-Data:**
- Intersection ID
- Status and Timestamp
- Actual light state of all lanes
- Remaining time of the state of all lanes
- Red and green phase duration

**MapData:**
- MapData for the intersection
  - Position of Intersection
● Number and Position of Lanes
● Allowed speed on each lane
● Road link to next intersections/traffic lights

Since the *System Analysis* and the *System Design* should develop a general applicable system, this thesis assumes a “perfect” interface to the traffic lights. This interface is able to supply all necessary data to the GLOSA system.
3. **GLOSA System Analysis**

A GLOSA system is complex and highly interconnected system that includes several subsystems. For the realization of such a system it is essential to have a good development scheme and view to the system. This chapter will describe the design process of the GLOSA System. The *Systems Modeling Language* (SysML, see Section 2.3) is used to support and achieve the design process. It follows the already described *System Design* process of the *SYSMOD Approach* [35].

The first step of this project is the *System Analysis* process (explained in Section 2.3.2.1), which starts with setting up the description of the project context – a first rough explanation of the goals and scopes of the system – and continues with the definition of the stakeholders, their requirements and the main use cases. When this first step is done the *System Design* process (Section 4) can start, which realizes the use cases and describes the structures of the system. During this step the GLOSA system is split into three subsystems that are explained in separate sections: the *Central ITS Station*, the *Roadside Unit* (RSU) and the *Onboard Unit* (OBU).

3.1. **Description of Project Context**

The goal of this project is to develop a GLOSA system that calculates and displays a speed advice to the driver of a vehicle. With the help of this speed advice the driver can pass an upcoming traffic light at green without stopping the car. The system should also display the remaining time of a state. The purpose of the system is, to save fuel and stop time and lead to a more efficient traffic flow.

The interface of the traffic light will be used to connect the GLOSA system with the infrastructure. The system will send the green timing of the traffic light to the OBU in the vehicle, which will then calculate the speed advice and display the recommended speed to the driver.

The system has to be reliable and safe. Failures or delays should be detected and in case it is not reliable to display a speed advice or a remaining time, either an error message or no speed advice should be displayed. It should not display a speed advice above the speed limit and the algorithm should take the reaction time of the driver into account.

The system should also react to state changes of the traffic light. Traffic lights have either a fixed timing or they may react on changes, like pedestrians or preemption of public transportation. The GLOSA system should be updated, when in case of an emergency the timing of the traffic light is changed.

The customer prefers that the system will seamlessly adapt to the Internet of Things, which means that it provides a central service where data could be directly requested. For example
this allows navigation systems to request data about all traffic lights on a calculated route. The GLOSA system should use open communication standards, to enable interoperability between different manufacturers (e.g. the COAP protocol for IoT, ETSI ITS-G5 for V2X).

3.2. POSSIBLE SYSTEM ARCHITECTURES

Basically there are two fundamental system architectures for a GLOSA system that have to be compared. The first fundamental architecture is to link the vehicles via a central service center (mobile communication); the other one is to link the vehicles via local infrastructure (V2I, IEEE 802.11p, Section 2.1.1). Both kinds of systems have their own advantages and disadvantages.

3.2.1. CENTRAL SERVICE CENTER APPROACH

The service center based architecture has a centralized service station, which provides all cars with the necessary data. It produces the forecast for all traffic lights and adds required static data (e.g. intersection topology). A vehicle can request the data of the traffic light it is approaching via mobile communication.

Such a system would be faster and cheaper to realize than the local infrastructure approach. It would also be possible that the vehicle already requests data of the next upcoming traffic lights; a navigation system may use this data to calculate faster routes. In this case it is also very easy to integrate the system with Internet of Things idea; the central service center could map each traffic light to a separate resource URI.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Number of Vehicles (Inbound and outbound</td>
<td>6528 + 6045 = 12,573</td>
</tr>
<tr>
<td>vehicles at 60th street in New York [42])</td>
<td></td>
</tr>
<tr>
<td>One-way volume of vehicles per intersection</td>
<td>0.44 cars/s</td>
</tr>
<tr>
<td>Number of vehicles across all intersections</td>
<td>0.44 x 65 = 28.6 (29 cars/s)</td>
</tr>
<tr>
<td>Duration of an active SPAT session</td>
<td>10 seconds</td>
</tr>
<tr>
<td>Number of cars with active SPAT sessions over all</td>
<td>29*10 = 290 cars</td>
</tr>
<tr>
<td>intersections</td>
<td></td>
</tr>
<tr>
<td>SPAT rate per intersection</td>
<td>40 Kbit/s</td>
</tr>
<tr>
<td>Average bandwidth per vehicle (For 3 sector with</td>
<td>(3*150 Mbit/s) / 290 cars = 1.55 Mbit/s per car</td>
</tr>
<tr>
<td>each sector at 150 Mbps with 2x2 MIMO) (Based on</td>
<td></td>
</tr>
<tr>
<td>NEC LTE)</td>
<td></td>
</tr>
<tr>
<td>Maximum users per cell in 5 MHz band (Based on</td>
<td>200 x 3 = 600</td>
</tr>
<tr>
<td>NEC LTE)</td>
<td></td>
</tr>
<tr>
<td>Saturation ratio for SPAT traffic</td>
<td>290/600 = 48.3 %</td>
</tr>
<tr>
<td>Available bandwidth per vehicle with 2% loading by</td>
<td>0.98*450 Mbps / 290 cars = 1.52 Mbps per car</td>
</tr>
<tr>
<td>other traffic</td>
<td></td>
</tr>
<tr>
<td>Available bandwidth per vehicle with 50% loading by</td>
<td>0.50*450 Mbps / 290 cars = 776 kbps per car</td>
</tr>
<tr>
<td>other traffic</td>
<td></td>
</tr>
<tr>
<td>Available bandwidth per vehicle with 90% loading by</td>
<td>0.10*450 Mbps / 290 cars = 155 kbps per car</td>
</tr>
<tr>
<td>other traffic</td>
<td></td>
</tr>
</tbody>
</table>
A disadvantage is that the delay is higher than in systems with local infrastructure, because a request to the central service center is necessary and a longer transmission delay of mobile communication networks compared to IEEE 802.11p.

Further the data rate of LTE may not be high enough in really high-populated areas [30]. Every car has its own connection to the service center, whereas in the local infrastructure approach shares bandwidth, because the information is broadcasted to multiple vehicles at once. Since mobile communication normally does not allow the broadcasting of data. Existing communication networks (LTE) do not have enough bandwidth to supply vehicles with SPAT (and MapData) information in highly populated cities. A calculation about the bandwidth worst-case is found in Table 4. For every car 1.55 Mbps is necessary, this could only be achieved when 100% of the network bandwidth could be used; a stand-alone LTE would be necessary. When LTE could be use broadcasting to transmit the SPAT data the bandwidth would be sufficient [30].

3.2.2. LOCAL INFRASTRUCTURE APPROACH

The local infrastructure approach requires equipping the local infrastructure (traffic lights) with IEEE 802.11p communication units. Each intersection will generate the necessary information by itself and communicate it to approaching vehicles.

For the network communication a dedicated IEEE 802.11p (or ITS-G5) infrastructure is used, which will provide several advantages. IEEE 802.11p has less delay than mobile communication and allows broadcasting of information. Further a dedicated network does not need to share bandwidth with other units, e.g. smartphones and infotainment, and will therefore be more consistent.

In the future an IEEE 802.11p infrastructure will likely be realized, since it creates plenty of advantages to vehicles. Vehicles can communicate with other vehicles and share important information to prevent collision. They can directly share information with each other to optimize the traffic flow. Since IEEE 802.11p is an important technology for next generation vehicles, the GLOSA system has to support it to be ready for the future.

However there are also some disadvantages. One point is that the number of intersections that have to be equipped is very high; the manufacturing and installation of such a system will cost plenty of money. The next point is that the hardware cycles of traffic infrastructure are very long (decades) and that sometimes not everything could be changed by software updates; hardware updates can become very expensive in this case. Another problem with this solution is the question: when will IEEE 802.11p communication be integrated in cars? A solution that uses UMTS or WLAN would be much earlier to realize and easier to integrate in old vehicles by using smartphones.

A further disadvantage is that the intersection controllers are often not designed for this use case. They may not have probable interface that allow extraction of the needed data.
Furthermore the controllers may also miss important data, e.g. the physical map layout of the intersection.

A reasonable solution is to combine both types of systems in such a way, that the advantages are shared and the disadvantages are minimized. The local infrastructure approach for fast and future safe communication and the mobile communication approach to support navigation systems and traffic flow optimization applications. This thesis will design the GLOSA system and recognizes this approach; the detailed design will be explained in the following.

3.3. **Requirement Analysis of the GLOSA System**

An important step in a system development process is the specification of the requirements. The *SYSMOD Approach* proposes to identify the stakeholders first and afterwards determine which requirements they have to the GLOSA system.

3.3.1. **Stakeholders**

*Stakeholders* include all persons that might have interest in the GLOSA system. That might be users or actors who are actually using the system, but also persons that might not be directly in touch with the system, e.g. insurance companies, laws or suppliers of the system.

The *Stakeholders* found for the GLOSA system are listed in Table 5. The table also defines the priority of the *Stakeholder* and shortly describes their interests (1 = high priority, 3 = low priority). The priority depends on how much the *Stakeholder* interacts with the system and if he might be hurt.

Sometimes the priority is not straight forward, e.g. the priority of pedestrians. Pedestrians are not directly in touch with the GLOSA system - of course they can require preemption by pressing the button of the traffic light, but it is the traffic light they are using and not the GLOSA system. Furthermore the driver is liable for his vehicle and he has to comply with the road traffic act (StVO), like it is the case when using a navigation system. However a high priority was chosen, because pedestrians might get hurt if the GLOSA system shows wrong data.

<table>
<thead>
<tr>
<th>Stakeholder</th>
<th>Priority</th>
<th>Comment / Interests</th>
</tr>
</thead>
<tbody>
<tr>
<td>Driver</td>
<td>1</td>
<td>Wants to save fuel and travel/stop time.</td>
</tr>
<tr>
<td>Traffic Light Control</td>
<td>1</td>
<td>Sends data to the GLOSA system, must provide an interface. Have its own requirements to the system. GLOSA System must not break traffic lights.</td>
</tr>
<tr>
<td>Positioning system</td>
<td>1</td>
<td>Might be liable of costs (e.g. GPS should be free of charge) Operator might change accuracy (e.g. US army for GPS)</td>
</tr>
<tr>
<td>Emergency vehicles</td>
<td>1</td>
<td>Traffic lights may be changed in case of an emergency or when a bus wants preemption.</td>
</tr>
<tr>
<td>Public Transportation</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
### Stakeholders

<table>
<thead>
<tr>
<th>Stakeholder</th>
<th>Priority</th>
<th>Comment / Interests</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pedestrian</td>
<td>1</td>
<td>Gets hurt if GLOSA system is wrong. Presses the button and can therefore change the remaining time of a traffic light</td>
</tr>
<tr>
<td>Communication channel vendor</td>
<td>2</td>
<td>(In case of mobile radio operator) Wants money for his communication service.</td>
</tr>
<tr>
<td>Lawmaker</td>
<td>2</td>
<td>What laws are affecting system, e.g. it is not allowed to touch a smartphone while driving</td>
</tr>
<tr>
<td>Client</td>
<td>2</td>
<td>Wants integration to a navigation system and Internet of Things (COAP) support</td>
</tr>
<tr>
<td>Other Drivers</td>
<td>2</td>
<td>The system may cause traffic because of slow driving.</td>
</tr>
<tr>
<td>Insurance Companies</td>
<td>3</td>
<td>The driver might be distracted by the system. Will they pay?</td>
</tr>
<tr>
<td>Support/Marketing</td>
<td>3</td>
<td>Might need knowledge or special access to the system.</td>
</tr>
<tr>
<td>Hardware Manufacturer</td>
<td>3</td>
<td>Wants money for delivered devices (RSU, OBU)</td>
</tr>
<tr>
<td>System Attacker</td>
<td>3</td>
<td>Wants to attack the system or use it for his advantage.</td>
</tr>
</tbody>
</table>

3.3.2. Requirements

The list of Stakeholders from Table 5 was used to determine which requirements the Stakeholders have to the GLOSA system. Requirements can be grouped into various categories, which help to clearly arrange the diagram, but also help to discover as many as possible. One example for categorizing requirements is the FURPS model [19], it is recommended by the SYSMOD approach; the categories of this model will also help to visualize the requirements in the requirement diagram:

- **Functionality** — functional requirements.
- **Usability** — usability requirements, e.g., usage concepts, corporate identity.
- **Reliability** — reliability requirements, e.g., failure frequency.
- **Performance** — quantifiable requirements, e.g., response times, speed, bandwidth.
- **Supportability** — testability, configurability, installation, service.

There are other categorization catalogs that could be used to group requirements. SysML does not specify requirement categories, but they can be derived from the standard requirements by using stereotypes. Requirements also have a hierarchy, which you can show by drawing the SysML Requirement diagram and group them with the containment relationship. Later during the process it is also possible to specify which requirement is satisfied by which component of the designed system and they can be verified by test cases.

As already written the ETSI ITS specifies a basic set of applications [28], which also describes a similar system. The given main requirements for such a system are:

- Capability of a RSU (traffic light) to broadcast the SPAT data periodically
- Broadcast detailed topological and geometrical information about the intersection
- Capability of vehicles to receive broadcasted SPAT messages and processing them.
- Minimum frequency of the periodic message: 2 Hz
- Maximum latency time: 100 ms
- Minimum positioning accuracy: better than 5 m.

These requirements have been extended for the GLOSA system, are graphically presented in Figure 17 and further explained in the following paragraphs:

The main goal of the GLOSA system is to save fuel, save time and optimize the traffic flow. This request is however too abstract for a requirement, so it was split up to the functional requirement that the system displays a speed advice and the remaining time and that it needs a suitable speed algorithm, which saves fuel and stop time.
The system has to ensure, that the driver does not drive over a red light. It has to consider the reaction time of the driver for the speed advice. Furthermore the system has to comply with speed limits and needs a suitable refresh time for the speed advice and the remaining time. This implies that all data from the traffic light or the actual position of the vehicle must be up to date.

A very important requirement is that the system needs some kind of safety/reliability detection. It should be detected if the calculation of the speed advice is not safe to display. When the calculation is unreliable the system should either display an error message or display no information (blank screen) to the driver. It should for example detect that the vehicle position is not accurate or that the SPAT data is too old (also MapData, but longer expiration time).

The GLOSA system must be usable while driving the vehicle. The system needs an intuitive Graphical User Interface (GUI) and no inputs should be made while driving, in particular if a Smartphone is used as the display for the speed advice. Without these requirements it would also never satisfy the requirement that it would be accepted by law. The application should automatically connect to the corresponding intersection and receive its data.

Furthermore it has to be possible to remote maintain the system and observe possible errors. The system needs to be sustainable, robust and reliable even under harsh weather conditions. Requirements for range and refresh times are already set up by the ETSI ITS. The range should be at least 300 m otherwise the system will not be economic [7]. The SPAT Interval should be 1 Hz-5Hz according to the DCC.

The requirements for the traffic light interface have already been explained in Section 2.5. In the diagram this requirement is called traffic light synchronization, but also means that the system receives the data over the traffic light interface.

There are further requirements that are self-evident and not further listed, e.g. that the system should fit into a car. However this should be listed and more specified, when there appear actual requirements during a broader design process (e.g. that the on-board computer should fit into a specific DIN compartment). Another requirement that was desired by the client was that the system uses standard/open-source protocols to allow interoperability with other systems.

3.3.2.1. Essential Requirements

The indicated requirements can be split into essential and technical requirements. It is a different view to the found requirements, which gives another overview and helps to review, if all requirements have been found. The essential requirements are, what the client or project manager really wants. All technical requirements are derived by them and are necessary to realize the system.
The essential requirements of the GLOSA system are shown in Figure 18, the technical/functional requirements are not listed again, as they are already listed in Figure 17. In a complete diagram they would be connected to the essential requirements via a derive relationship.

3.4. SYSTEM CONTEXT DIAGRAM

The System Context Diagram in Figure 19 shows how the GLOSA system is connected to its environment. The Diagram includes the main actors who use the system, systems that are used by the user and other external systems that deliver data or might influence it. For the System Context Diagram the main actors of the GLOSA system have been identified as:

- The Driver, who wants to see the speed advice and the remaining time
- A Positioning System that delivers the position of the vehicle (and therefore also the speed of the vehicle)
- The Traffic Light Control who can change the program of the traffic light (and logs the states of the traffic lights)
- Public transportation, emergency vehicles and pedestrians can influence the traffic light states and demand for preemption.
- Other external entities (power supply, interferences, system attacker, temperature)

In this early project phase it is already known which type of information is exchanged and who participates in the exchange. However the exact protocol to the external systems is not yet decided. The here specified information in the System Context Diagram is initially very abstract, but helps to get an overview of the system. The System Context diagram is not a predefined SysML or UML diagram, but part of the SYSMOD Approach. It is formally and correctly composed of standard SysML/UML elements. The System Context is drawn as a Block Definition diagram or an Internal Block diagram, an Internal Block diagram would specify much more details like interfaces and data flows, but may be confusing for a first system overview.

There are a few possibilities how emergencies may be propagated in the system: by a Central Station (LTE), by a traffic light (RSU) or the vehicles can directly receive emergencies (via SAE J2735 Emergency Vehicle Alert messages). This thesis assumes that an RSU is in knowledge of emergencies and changes the light states; and therefor also the SPAT data, which is used to calculate the speed advice.
3.5. **Use Case Analysis**

After finishing the *System Context Diagram* the use cases of the GLOSA system can be identified. The use cases represent the services the GLOSA system provides, which means they are central elements of the requirements analysis. All functional requirements have to be directly satisfied by a use case and have a high priority. All other requirements, like delays, range or usability are of qualitative or supportive nature.

From a high level point of view the main use case could be described as “Driver wants to save fuel and time”, but this is too abstract and does not help to design the system. Therefor the core use case of the GLOSA system is to display the information of the traffic light to the driver. This use case is a generalization of the two use cases of displaying the speed advice and the remaining time. It furthermore includes the use cases of starting and stopping the usage of the system.

This core use case will receive the data from the Traffic Light Interface (TLI) and the *Positioning System*. The *Positioning System* will be used to determine the actual speed of the vehicle and the distance to the traffic light. The TLI data will be used to synchronize the GLOSA system with the traffic lights state. This use case also discovers changes in the traffic light timings that are triggered by pedestrians or public transportation.
The core use case may already include the emergency reaction since the traffic lights will react to emergencies (as already described in Section 3.4) and change their light timing and therefor the SPAT data. When the SPAT data is changed, also the vehicle will recognize the change. There is basically no existing difference between emergencies and preemption for public transportation and pedestrians, which will also be recognized. Nevertheless this use case is included in the diagram, because it is an important requirement and has to be checked again if a local traffic light system treats emergencies differently.

Further use cases of the system are, that the operator at the traffic light central can change the program and map data of a traffic light. Traffic light program changes can either be the creation of new programs or the updating or deletion of existing ones.

In a future design more use cases can be created, e.g. for the Internet of Things like getting the speed advice, remaining time or MapData directly (for navigation systems). It is also preferable to log what is going on inside the system. Later the GLOSA system can also be combined with Infotainment and more important with road safety or crash warnings.
4. GLOSA SYSTEM DESIGN

The next step of this project is to design the actual GLOSA system from the results of the System Analysis in the last chapter. Now the use cases have to be realized and the system structures and blocks have to be set up. For this step the GLOSA system is split into three subsystems: the Central ITS Station, the Roadside Unit (RSU) and the Onboard Unit (OBU). The System Overview describes, how the three subsystems are interconnected and interact with their environment.

4.1. SYSTEM OVERVIEW

The System Overview gives an impression of the GLOSA system and its three subsystems. It is not a standardized diagram type of SysML, but it is proposed by the SYSMOD Approach and could be drawn like a Block Definition Diagram. The overview is shown in Figure 21.

In the real system there are of course more than one traffic light, more than one vehicle and maybe also more than one control center. Therefor the multiplicities are printed near the relations of the blocks.
The Central ITS Station is the operation and observation point of the GLOSA system. The operator can program the RSUs with new traffic light programs and create or update the map of the Intersection. It is also responsible for logging and detection of failures that have to be repaired by the operator. The RSU is connected to the traffic light via the TLI and thus synchronizes the SPAT and Map data. This data is then send to the OBU, which can calculate the speed advice via its actual position and speed. The RSU might also use a Positioning System to deliver a correction signal to support the OBU with a DGPS correction signal (see Section 4.4.3).

![GLOSA System Block Diagram](image)

Each subsystem that was sketched in the System Overview consists of further blocks. These Blocks are shown in the Block Definition Diagram in Figure 22. The diagram as shown here is a reduced version; a more complete version with a function description for each block is shown during the FMEA in Figure 39.

4.1.1. Used Network Technologies

As analyzed in Section 3.2 the best way is to combine V2X communication and mobile communication.

Under normal circumstances the IEEE 802.11p devices will transmit SPAT and MapData, with them it is possible to broadcast the data, what saves bandwidth (e.g. LTE alone can not deliver SPAT to many cars). It can be used for a range of up to 1000 m (300 m was required) and Geonetworking may further extend this range by multi-hop transmission.

Mobile communication (e.g. LTE) can support the GLOSA system in special cases. For example when the complete bandwidth of IEEE 802.11p is used (in highly populated cities the bandwidth may be used completely by CAM messages, which have priority to prevent crashes), or for longer distances when multihop is not possible (in early stages when there are not enough vehicles/stations to allow multihop). Both examples are especially important during emergencies.

Mobile communication is also a solution that makes IoT applications much easier to realize; e.g. it should be possible to request SPAT and Map data from the Central ITS Station via direct request.
4.1.2. **MESSAGE TYPES**

The GLOSA system uses the SAE J2735 message set that have been explained in Section 2.2. The SPAT messages are used to communicate the state and timing of the traffic light. The MapData message is used to communicate the geographical information of the intersection.

The RSU generates the SPAT and MapData for all existing lanes and broadcast it to; the OBU receives these broadcasts and decides, which data (of which intersection and lane number) it really needs for the calculation of the speed advice.

4.2. **CENTRAL ITS STATION**

The *Central ITS Station* is the general control center of the GLOSA system. It is used for remote maintenance, logging and controlling. The operator can enter and update traffic light programs and administer the MapData for every Intersection. It is also used to display system errors to the operator. System errors can be for example a broken RSU or a lost connection to the traffic light, which may be broken.

![Figure 23: Central ITS Station with Interfaces](image)

Figure 23 shows the *Central ITS Station* block and its interfaces. The underlying blocks and subsystems have already been displayed in the *Block Definition Diagram* of the GLOSA system in Figure 22. The IHMI interface is the human machine interface that allows the operator to change traffic light programs and map data. The Ethernet interface is connected to the RSU and the UMTS interface allows communication with vehicles.

Another function of the central is that it provides services via the mobile network. One service is that a vehicle can request SPAT and MapData information for an intersection. With this service it is possible to plan a route and already gain data for future upcoming traffic lights. It could also be used to determine the most probable intersection by a mobile request. This is useful if the next traffic light is out of range of the IEEE 802.11p connection or in case of a network failure as a redundancy.

In the future the *Central ITS Station* should be connected to the control centers of the traffic light operators. It would bring plenty of improvements when both data sources are connected, even more in the future when both systems gets combined and define a completely new solution for ITS.
4.2.1. INTERNAL BLOCK DIAGRAM

The Central ITS Station is composed of four internal parts, which are shown in the Internal Block Diagram in Figure 24. The main component is the ITS computer or processor that runs the stations program. It is connected to all other components like the Ethernet block, UMTS block and the HMI.

The Ethernet block is used to communicate with the RSUs and send them the updated programs or MapData. The HMI is an interface that waits for input of the operator or display him information about the system. The UMTS block is used to provide service via mobile communication.

![Central ITS Station Internal Block Diagram](image)

Figure 24: Central ITS Station Internal Block Diagram

4.2.2. ACTIVITY DIAGRAM

The scope of this thesis does not include all desired functions of the Central ITS Station. Only the basic activities are further described: updating of traffic light programs and MapData.

![Central ITS Station activity diagram](image)

Figure 25: Central ITS Station activity diagram

Figure 25 describes the processes that run inside the Central ITS Station. The program starts with the initializing action Start Usage and then launches three asynchronous processes that
wait for actions. The first process waits for modifications of the traffic light programs, the
second process waits for MapData changes and the third one waits for the end of usage.

The application that is running at the Central ITS Stations is event driven. It reacts to the
operators input for creating and updating of traffic light programs and MapData, which will
then be sent to the corresponding RSU.

The Central ITS Station in this design does not specify the following things: A central log
database is required for insurance reasons. SPAT and MapData may be saved in a central
service, where direct data requests could be made (or use data to optimize traffic light
programs).

Start and End Activities are not further explained here, for example they include
initializations and deallocation of memory.

4.3. ROAD SIDE UNIT (RSU)

The device that will be located at each traffic light is basically a usual ITS RSU, which is
connected over a traffic light interface. An ITS RSU is an infrastructure device near the road,
which extends the IEEE 802.11p communication range and could be used for all kinds of ITS
communication. A RSU could also be independent from a traffic light, e.g. it could be a road
sign. However in case of the GLOSA system the RSU is connected with the traffic light via a
traffic light interface to collect SPAT data, which could be send to approaching vehicles.

![Figure 26: GLOSA RSU with interfaces](image)

Figure 26 shows the GLOSA RSU and its interfaces. The underlying blocks and subsystems
have already been displayed in the Block Definition Diagram of the GLOSA system in Figure
22. The Ethernet interface is the connection to the Central ITS Station, the TLInterface
connects to the traffic light and the IEEE 802.11p interface sends SPAT and Map data to the
vehicles.

4.3.1. INTERNAL BLOCK DIAGRAM

The RSU is composed of four internal blocks that are shown in the Internal Block Diagram in
Figure 27. The main component is the RSU computer or processor, which runs the
components program. It is connected to all other components like the traffic light interface,
the IEEE 802.11p device and the Ethernet block.
The Ethernet block is used to communicate with the Central ITS station and receive the updated programs or MapData. The traffic light interface synchronizes the RSU with the connected traffic light. The IEEE 802.11p device communicates with nearing vehicles.

4.3.2. Activity Diagram

The application that is running on the RSU includes several asynchronous processes. The RSU has to receive data from the Central ITS station, synchronize its data with the connected traffic light and broadcast it to approaching vehicles.

Figure 28 describes the processes that run inside the central. The program starts with the initializing action **Start Usage** and then launches five asynchronous processes that wait for actions; one of them waits for the end of usage.
The first process combines the reception of traffic light timing and map data over the *Traffic Light Interface*. To make the *System Design* as general as possible a “perfect” interface is assumed as described in Section 2.6.1.

Two more processes broadcast the SPAT and MapData via the IEEE 802.11p device. The default broadcast interval is to send every 20 ms, this interval may change according to the actual network usage; this may be controlled in future implementations by the DCC, which was explained in Section 2.1.2.5.

The last process synchronizes the RSU with the state of the traffic light. This could either mean that the system gets all necessary data from the traffic light interface or that is has to synchronize and update an internal state machine, which reimplements the traffic light (e.g. as describe in Section 6.1.5).

### 4.4. VEHICLE ONBOARD UNIT (OBU)

Each vehicle will be equipped with an *Onboard Unit (OBU)* that allows the vehicle to communicate with its environment. The OBU collects and send V2X-Messages and also collects the position for further processing.

![Figure 29: OBU with interfaces](image)

The interfaces of the OBU are presented in Figure 29. The underlying blocks and subsystems have already been displayed in the *Block Definition Diagram* of the GLOSA system in Figure 22. The OBU has an interface to the driver, an interface to the positioning system (GPS) and two communication interfaces (IEEE 802.11p and UMTS).

The OBU receives the SPAT and MapData messages from the traffic light traffic light. It collects the position and speed of the vehicle and calculates a speed advice to pass the traffic light when it is green.

The OBU described in this design, combines the OBU and AU in one single unit. Another idea is that the OBU is just a relay that forwards IEEE 802.11p messages via standard WLAN to a smartphone, which will then display the information to the driver. Thus the display block could also be seen as an interface to a smartphone. The smartphone could either use its own GPS or be supported by the GPS of the OBU.
4.4.1. **Internal Block Diagram**

The OBU is composed of five blocks, which are shown in the *Internal Block Diagram* in Figure 30. The main component is the OBU computer or processor that combines all data and runs the necessary calculations.

![Figure 30: OBU Internal Block Diagram](image)

The IEEE 802.11p device is used to communicate with other vehicles or infrastructure – in the GLOSA system case mainly with the traffic lights. The *GPS Receiver* is used to collect the vehicles position and track the actual speed. The *UMTS module* could be used to send requests to the *Central ITS Station*. The *Display* shows the calculated data (of the OBU computer) to the driver of the vehicle.

4.4.2. **Activity Diagram**

The application that is running at the RSU includes several asynchronous processes that are described graphically in Figure 31. The program starts with the initializing action *Start Usage* and then launches four independent processes. The *End Usage* process was already explained in earlier sections.

The continuous *V2X-Receiving* process is an event driven process. Every time a V2X-Message is received the described sequence will be executed. The received messages will be identified, if it is a SPAT, a MapData or an unknown message. Unknown message will be discarded. SPAT and MapData information will be saved to the corresponding database (with a time stamp). For the moment the application has to set up its own database (which ensures interoperability), but in the future it should use the already provided LDM of the ETSI ITS (Section 2.1.2.3). However there are currently no devices on the market that have already implemented the LDM service. Furthermore many systems are designed for the American WAVE system and only adapted to support the European frequencies, but do not provide the LDM service.

The continuous *Position Update* process uses the *GPS Receiver* to determine the position of the vehicle. The positioning system is also used to track the vehicle speed. Before the position is saved the process has to check if the position is plausible and accurate enough.
Many GPS- Receivers provide an accuracy that useful for the GLOSA system (5 m accuracy was required), but it should also be checked if the position is jumping around or lead to impossible speeds. It also saves the time stamp of the last successful received position, so that the application can later decide if the position is usable. The Position update process could either be an interval-triggered process, which polls the positioning system (GPSD) or event triggered where the GPS Position notifies the thread.

![OBU Activity Diagram](image)

The last continuous process uses the saved inputs from the other two processes to execute the actual speed advice calculation. This calculation is done twice a second. The first step of this sequence is to identify the intersection the vehicle is approaching. The identification is only possible if the vehicle position is clear and when SPAT and MapData is ready. When an intersection is identified, the next step is to identify the correct lane. This could be more than one lane because the OBU does not have information if the car will turn or not. The information for each possible lane has to be displayed to the driver of the vehicle.

When at least one lane is identified, the actual calculation can start. The algorithm can calculate the speed advice that depends on the distance to the traffic light, the vehicles
position, the remaining green/red time and the actual speed of the vehicle. When the calculation is complete, the data will be displayed to the driver.

4.4.3. POSITIONING SYSTEM (GPS)

The Positioning System is used to determine the position and speed of the vehicle, which is used to calculate the distance to the traffic light [12]. For this first design the distance is just calculated as the straight direct line. A future solution should also consider a map of the street.

For a more accurate position the GPS Receiver could use a Differential GPS (DGPS) correction signal that is created by the RSU; this idea is further explained in Section 5.9.2.1.

4.4.4. IDENTIFY THE APPROPRIATE INTERSECTION

It is not trivial to identify the corresponding Intersection from the saved MapData in the Map database. The easiest solution right now is to choose the closest intersection that is in a ±45° angle in front of the car. For this approach only the location of the vehicle and the moving direction is necessary.

However a future solution should also consider an actual map, because the first proposed algorithm has problems with curves and street that actually turns to a different intersection. Furthermore the algorithm could analyze the road links that are provided in the MapData message.

4.4.4.1. CHOOSING THE APPROPRIATE LANE

When choosing the appropriate lane, the same problem occurs. The easiest solution is to calculate the angle between the lanes offset vector and the vector between the traffic light and the car. If the angle is inside the range of ±45° the appropriate lane was found.

This algorithm was already used in [12] and evaluated to have a success rate of 80%. However there are still some problems with difficult intersections and curved streets. This problem can also only be solved when the algorithm uses a map.

Another problem is that it is not possible to know if the driver wants to turn left or right, so that the OBU has to show all lanes that may be of interest for the driver. This could be solved if the OBU has a navigation route set up or a connection to the indicator of the vehicle.

4.4.5. GRAPHICAL USER INTERFACE

The development of a user interface for the OBU is not part of this thesis. However the already proposed of [12] is a great starting point; it is shown in Figure 32. The interface is very clear and easy to understand. At the top it displays a traffic light with the actual state and the remaining time. At the bottom it also displays the speed limit, actual speed and the distance to the traffic light. In the lower half of the screen it shows the actual speed advice for each lane. It is possible to display information of all possible lanes.
The speed advice can be calculated as a range; when the driver drives too fast or too slow he cannot pass the traffic light. However in [12] it was analyzed that showing a range would be too complex and distracting for drivers. It was proposed to only show a red and a green area. When the speed of the vehicle is in the green area, it will pass the traffic light; if the speed is in the red area it will not pass the traffic light. As long as the driver drives in the green area and close to the proposed speed (the line where the red and the green area meet) he will pass the traffic light. The speed advice is limited by the speed limit and also a lower bound is proposed (20 km/h when 50 km/h is allowed). The GUI should always show the highest possible bound to allow a higher speed and therefore a more efficient traffic flow.

This user interface could be either displayed on an onboard display or on a smartphone, which is connected to the OBU via WLAN. More requirements for a user interface are explained in [43].

4.4.6. SPEED ADVICE ALGORITHM

A first simple speed advice algorithm was also explained in [12]. In a simplified manner it uses the actual traffic light state and divides the distance by the remaining time of the state \( v = \frac{s}{t} \). The Failure Analysis shows that this algorithm is not suitable, since it does not consider the reaction time of the driver or the actual speed of the vehicle. An extended algorithm loop is explained in Section 5.9.3.

4.5. CONCLUSION

SysML and the SYSMOD Approach give many advantages and ensure a more structured realization. With the aid of the developed design the first prototype could be realized much faster. However the first demonstrator that was realized, needed a few different solutions to the developed design; the differences and simplifications are explained later in Section 6.
Furthermore it helped to perform the Failure Analysis, which is done in the next section. The design already includes some necessary steps of the FTA and FMEA.

The Central ITS Station, the RSU and the OBU may need further functions and requirements, especially in future applications and when the system is not used for GLOSA only. Further functions for the GLOSA system have to be elaborated together with the stakeholders LSBG and HHVA.

SysML could go even more into technical details. It is possible to describe all interfaces and message types, ports and the physical structure of all devices in a much higher level of detail. However it is not possible to show this complexity in a written thesis and it is not necessary for understanding the design. For a more complex design also a more advanced SysML tool should be used (Microsoft Visio was used in this thesis) to specify, visualize and check the SysML model. In the ongoing project SysML tools can also support testing and verification.
5. Failure Analysis

This section describes the Failure Analysis of the GLOSA system. It first starts with a general Cause and Effect Diagram and continues with a Failure Tree Analysis (FTA) and a Failure Mode and Effects Analysis (FMEA) according to the proposed process by the VDA.

Research shows that SysML can close the gap between a product development and the Failure Analysis. It can prevent redundant work tasks and saves investigation time, e.g. it saves the breakdown to system levels step of the FMEA (see Section 5.7.1) and helps to do the FTA.

During a real project there are however plenty of challenges:

- High quality SysML (or UML) models are not always available.
- In agile development these models, the requirements and the design, evolve during the development cycle.
- SysML is often a simplified view of the component behavior. Design-vs.-Code alignment studies of flight software show that faithful UML representations may only capture 40% to 60% of the implementation [44].
- Automated processes necessary in large projects because of efficiency reasons. However automated FMEA synthesis on incomplete information could be misleading it can generate large amounts of data requiring manual intervention.

5.1.1.1. Definitions

Fault: An incorrect step, process, or data definition, which causes the system to perform in an unintended or unanticipated manner. It is an inherent weakness of the design or implementation, which might result in a failure. A fault might be present and latent in the systems and lead to a failure when the exact scenario is met.

Failure: The inability of a system or component to perform its required function within the specified requirements

Error: A discrepancy between a computed, observed, or measured value or condition and the true, specified, or theoretically correct value or condition.

Failure Cause: Could be a defect or broken part, which is the underlying cause of a failure

Failure Mode: “A single event, which causes a functional failure.” (SAE JA 1011/SAE JA 1012)

Failure Effect: Effect of a Failure Mode

Failure Cause -> Failure Mode -> Failure Effect

These definitions differ according to where the line is drawn between effect and cause. When describing Failure Modes you are in fact always describing Failure Effects and not “the cause” (you can always ask an extra "why?"). As a result there is not one right definition for
Failure Mode. The definition depends on the need of the project and what goals it wants to achieve with defining Failure Modes.

5.2. FAILURE ANALYSIS PROCEDURE

Directly after the completion of the GLOSA System Analysis (Section 3) it was possible to start with the first steps of the Failure Analysis. The procedure starts with the groundwork that should have been done already before the actual System Design process starts. The groundwork shows the first common failures of the analyzed system and may save one – time-consuming – review cycle.

After the System Design the FTA and FMEA shall be accomplished. The three steps of the system development (System Analysis, System Design and Failure Analysis) should be done in cycles (or quasi in parallel) as shown in Figure 33. When the Failure Analysis is completed, the System Analysis and the design have to be updated to transfer necessary changes. During the process new requirements may be derived or it may be necessary to add actors (e.g. sources of interference or system attackers) or to modify use cases. During the whole process, possible failures that get discovered should be added to the list of failures.

5.3. GROUNDWORK

The groundwork of the Failure Analysis step is a starting point that should already be done at the beginning of the System Design process or at least during the requirement analysis. The groundwork can consist of a Cause and Effect diagram and a list of possible failures. It is a first brainstorm and helps to find general ideas of failures that might happen and helps to find requirements that have to be considered during the first design cycle.

This analysis also prevents general failures that a FMEA may not find or find later, because it is closely bound to the system specification and parts.
5.3.1. **Cause and Effect Diagram**

The Cause and Effect Diagram is a graphical diagram that could be used as a first brainstorm for the Failure Analysis. It may be used in team meetings as a first guideline. Probably the most common one is the *Ishikawa diagram*, also called *fishbone diagram*, which was used for the GLOSA system in Figure 34. The categories help to find the corresponding failures. Categories can be changed to match the requirements of the project.

![Figure 34: Cause and Effect Diagram](image)

5.3.2. **Table of Possible Failures**

This work can also be summarized in a table of possible failures, which is shown in Table 6. In a table it is easier to add comments or first possible solutions. It may be used to group the findings of the *Cause and Effect diagram* that was done in a team meeting.

<table>
<thead>
<tr>
<th>Table 6: Table of Failures</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Failure</strong></td>
</tr>
</tbody>
</table>
| Driver drives through a red light | • Before it turns green (driver trusts GLOSA, can not break)  
• Driver starts car too early (TimeToChange not correct)  
• GLOSA system shows wrong state or speed  
• Reaction time of the driver was not considered |
| Wrong speed advice | • Speed advice unsteady  
• No speed advice |
| Data from traffic light | • Traffic Light defective  
• Connection lost  
• Data is outdated or wrong  
• Not synchronized with traffic light |
| Driver gets distracted | • GUI too complex, No input allowed while driving  
• GUI shows error message (input of driver necessary)  
• GUI frozen |
| Encourages speeding | • E.g. Audi does not show the real remaining time (because the temperament of Italian people will lead to speeding)  
• No speed advices above speed limit |
| GPS | • No connection  
• Wrong position, positions unsteady |
| Network connection | • Connection lost |
5.4. Hazard Declaration

Before the FTA and FEMA are performed the main hazards and catastrophic failures of the system need to be further declared. For the GLOSA system there are mainly two possible failures that have to be analyzed in more detail: “Wrong speed advice displayed” and “No speed advice displayed”.

A wrong speed advice is very risky for the user of the system, because it can lead to crashes and fatal injuries of people, e.g. if the driver trusts the wrong information of the GLOSA system and drives through a red light.

If in doubt, it is safer to display no speed advice. Therefore a wrong speed advice should not be displayed to the user and should be detected as best as possible. During the FMEA analysis a higher Severity was assigned to failure modes that lead to a wrong speed advice. A higher Severity will lead to a higher RPN.

5.5. Relationship between FMEA and FTA

The FMEA and the FTA are different procedures, however they are closely related as illustrated in Figure 35.

The FTA is a top down approach that analyzes the details of the event at the top. In this thesis it was done for two things: “No speed advice displayed” and “Wrong speed advice displayed”.

The FMEA goes bottom up and analyzes the effect chain of a failing part.
5.6. Fault Tree Analysis (FTA)

The FTA is a systematic method for System Analysis. It examines a system from the top to the bottom (top down) and draws it with graphical symbols for ease of understanding. It is used to investigate potential faults, its modes and causes and to quantify their influence (calculated with Boolean logic) to the system unreliability. The process of the FTA is described below [40]:

1. Define the undesired event to be analyzed (the focus of the FTA):
   - “No speed advice displayed”
   - “Wrong speed advice displayed”

2. Define the boundary of the system (the scope of the FTA):
   - The GLOSA system and its components as shown in Figure 22.
   - However it does not cover the Central ITS Station (it is only used for controlling the system)

3. Define the basic causal events to be considered (the resolution of the FTA):
   - Marked as circles in the FTA. Part failures are not analyzed in detail (e.g. power loss, because of a broken resistor)

4. Define the initial state of the system:
   - “No speed advice displayed”: Could be the initial start of the system
   - “Wrong speed advice displayed”: System already in use for some time, but error not yet detect (e.g. old SPAT/MapData/Position)

5.6.1. No Speed Advice Displayed

The first presented FTA investigates the case that “No speed advice is displayed” by the GLOSA system. This might happen when a complete system block is broken or the communication channel is constantly blocked. The resulting fault tree is shown in Figure 36.

![Failure Tree Analysis: “No speed advice displayed”](image-url)
5.6.2. **Wrong Speed Advice Displayed**

The second FTA investigates the case that a “Wrong speed advice is displayed” to the driver. It shows, what might cause incorrect data to be displayed. The resulting fault tree is shown in Figure 37.

The speed advice can also be wrong if a system block fails, however the first failure tree has already covered these failures (broken causes not included again); the system should detect missing data when the security interval expires.

![Diagram of fault tree analysis for wrong speed advice displayed](image)

**Figure 37: Failure Tree Analysis: "Wrong speed advice displayed"**

5.7. **Failure Mode and Effects Analysis (FMEA)**

The basic idea of the FMEA process was already explained in Section 2.5. For the German automobile industry the process of the *Verband der Automobilindustrie* (VDA) applies. The VDA approach is further explained in [41]. It proposes five steps that are explained and realized in the following sections.

5.7.1. **Product Breakdown to System Levels**

The first step is about the definition of the system, to break down the system into several levels. It analyses the structure of the system and shows all subsystems and components in a tree view like diagram. The output of this step is a system structure net. This step also specifies the scope of the analysis; it shows which parts of the system have to be analyzed.
This step is similar to the already derived Block Definition Diagram of the GLOSA system in Figure 22. The diagram already includes all necessary subsystems that have to be analyzed.

5.7.2. Functional Description of the System

The second step is the function analysis. Each system element from the first step will be analyses for its function and characteristics. Every component gets at least one function, more are possible. All functions combined, describe how the system work. This step extends the diagram of steps 1 with function fields for all components.

![Figure 39: FMEA step 2, Complete GLOSA Block Definition Diagram](image)

The function description was also already done during the System Design, however the Block Definition diagram did not show it for readability reasons. The diagram in Figure 39 now shows the functions/operations of the blocks (furthermore also values/variables can be included in the block definition).

5.7.3. Failure Analysis

The Failure Analysis is the third step in the VDA process. Every function that was found in step 2 will be associated with possible failures. In many cases the failure could be described as a not functioning component or a not working function of the component. The diagram from step 2 will be extended again and is shown in Figure 40.
The diagram also shows an alternative depiction for the Failure Modes, which is to show them as additional blocks. These blocks are extended by the stereotypes/categories for: “FailureMode”, “FailureEffect” and “FailureCause”, and can give much more information about the failure mode.

Furthermore this step associates the failures to other failures. Every failure may trigger another failure or is triggered by itself. Failures may propagate multiple times and reach the top of the system. Therefore the failures should be arranged to a failure chain. At the top of the chain stands at least one failure and at the bottom at least one fault.

The result of this step is similar to the FTA that was already done. Therefore only a short example is shown in Figure 41.
5.7.4. **Risks Evaluation**

Step 4 analyses the risks of the system. As a basis for this step a corrective or discovering action for each fault should be found. Some faults could be completely prevented or removed; others should be at least discoverable to define how the system should react in such a case.

With these actions it is possible to analyze the risks for the system by calculating the *Risk Priority Number* (RPN). The goal of the RPN is to prioritize the failures and the importance of the corrective actions. Potential failures have to be analyzed and classified by three different numbers: The *Severity* $S$, the *Occurrence* $O$ and the *Detection* $D$. The *Severity* $S$ describes the importance of the failure or the consequences of a failure mode. It considers the worst potential consequence of a failure, determined by the degree of injury, property damage, system damage and/or time lost to repair the failure. The *Occurrence* $O$ gives a description about the probability of the event occurring. The *Detection* $D$ qualifies the probability that the event would not be detected before the user was aware of it. The three Numbers are in whole numbers from 1 to 10. The RPN is calculated by $RPN = S \times O \times D$ and therefore ranges from 1 to 1000.

5.7.5. **Risks Optimization**

The last step is to define the corrective actions in more details. According to the RPN the most important failure has to be solved first. Corrective actions could prevent, remove, forecast or give tolerances for the faults. The goal of this step is to decrease the RPN and recalculate it in a next evaluation round.

When corrective actions change the *System Design (and Analysis)* all five steps of the FMEA have to be repeated. This should prove that the corrective actions take effect and may also detect new failures. If the corrective actions try to discover and avoid failures, only step 4 and 5 have to be repeated.

**Optimization Methods:**

**Fault Prevention** should prevent faults by using development methods and implementation techniques. They could either remove development faults or remove faults during use.

**Removal** should remove development faults by verification of the design, before a system is put into production. A system that is already in use should record failures so that they could be removed in maintenance cycles.

**Fault Forecasting** should predict faults in a running system, so that they could be removed or their effects can be weakened.

**Fault Tolerance** mechanisms should deal with faults in such a way, that the system could still deliver the required service (maybe degraded) even in presence of faults.
5.8. FMEA Results

The FMEA results can be listed in a table. There are many different formats for an FMEA table; it is possible to develop a more suitable variant for different cases. In this thesis a format close to the SAE J1739 was chosen. The results are shown in Table 7. Normally this table also has a further column, which reviews the Severity, Occurrence, Detection and RPN after the recommended actions have been implemented; in this thesis they are explained in the Section 5.9.

<table>
<thead>
<tr>
<th>Ref</th>
<th>Failure Mode</th>
<th>Failure effect</th>
<th>Sever.</th>
<th>Cause(s)</th>
<th>Occur</th>
<th>Current Design Controls</th>
<th>Detect</th>
<th>RPN</th>
<th>Recommended Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Human error</td>
<td>Drives Wrong speed</td>
<td>10</td>
<td>Reaction Time Driver Distracted Does not watch</td>
<td>8</td>
<td>10 800</td>
<td>Check GPS, if user reacts to speed advice Check if driver will cause a crash (drives over red)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Wrong Position by OBU GPS</td>
<td>Wrong speed advice Speed advice unsteady</td>
<td>10</td>
<td>No reception/view</td>
<td>7</td>
<td>10 700</td>
<td>Position plausibility checks needed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>RSU 802.11p Packet error/loss/delay</td>
<td>Wrong speed advice</td>
<td>10</td>
<td>Interferences, Bit error not detected</td>
<td>5</td>
<td>10 500</td>
<td>Replace bit error detection</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>OBU 802.11p receives defective data</td>
<td>Wrong speed advice</td>
<td>10</td>
<td>Interferences, Bit error not detected</td>
<td>5</td>
<td>10 500</td>
<td>Replace bit error detection</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Emergency not detected</td>
<td>Wrong speed advice</td>
<td>10</td>
<td>TL does not change Timings TL has no emergency information</td>
<td>4</td>
<td>10 400</td>
<td>Central emergency service (redundancy) OBU should receive and use EVA messages</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Time/Clock failure</td>
<td>Wrong speed advice (Delay subtracted from remaining time)</td>
<td>10</td>
<td>Clocks of OBU and RSU not synchronized</td>
<td>5</td>
<td>8 400</td>
<td>Use more reliable clock source (Internet, GPS)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>TLI returns wrong data</td>
<td>Wrong speed advice</td>
<td>10</td>
<td>TL internal failure Wrong data input</td>
<td>2</td>
<td>10 200</td>
<td>Check a central database average (redundancy)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>RSU 802.11p fails to send data</td>
<td>No speed advice</td>
<td>7</td>
<td>Channel blocked (too many cars transmitting)</td>
<td>5</td>
<td>No data received Partly received</td>
<td>3 105</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>OBU 802.11p fails to receive data</td>
<td>No speed advice Does not display actual information</td>
<td>7</td>
<td>Out of Range or blocked Hidden Station</td>
<td>5</td>
<td>No data received Partly received</td>
<td>3 105</td>
<td>Larger range (More efficient Antennas)</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Algorithm error/failure</td>
<td>Wrong speed advice (Above Speed limit Not doable)</td>
<td>10</td>
<td>Acceleration not used in calculation</td>
<td>2</td>
<td>5 100</td>
<td>Check/research algorithm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>System attacked</td>
<td>Wrong speed advice No speed advice</td>
<td>10</td>
<td>Interferes the system Modifies data</td>
<td>1</td>
<td>10 100</td>
<td>Security has to be added</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Central ITS Station failure</td>
<td>Modification Error Wrong speed advice</td>
<td>10</td>
<td>Operator Input HMI error Ethernet error</td>
<td>3</td>
<td>3 90</td>
<td>Check user input</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>RSU computer frozen</td>
<td>No speed advice (or first wrong)</td>
<td>7</td>
<td>Software failure</td>
<td>2</td>
<td>No data received Partly received</td>
<td>6 84</td>
<td>Watchdog Check CPU load</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>OBU computer frozen</td>
<td>No speed advice, Slow update rate</td>
<td>7</td>
<td>Software failure</td>
<td>2</td>
<td>6 84</td>
<td>Watchdog Check CPU load</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>No Position by OBU GPS</td>
<td>No speed advice</td>
<td>7</td>
<td>Unit broken Internal Part failure</td>
<td>5</td>
<td>Old Positions detected</td>
<td>2 70</td>
<td>Better GPS antenna</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>TLI fails to connect</td>
<td>No speed advice</td>
<td>7</td>
<td>No physical connection</td>
<td>3</td>
<td>Detected by RSU</td>
<td>1 800</td>
<td>Do not sent out old SPAT data</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Issue Description</td>
<td>Error Code</td>
<td>Error Description</td>
<td>Error Code</td>
<td>Error Code</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>----------------------------------------------------------------------------------</td>
<td>------------</td>
<td>-----------------------------------------------</td>
<td>------------</td>
<td>------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>RSU computer broken No speed advice (or first wrong)</td>
<td>7</td>
<td>Broken Internal part No power</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>OBU computer Unit broken</td>
<td>7</td>
<td>Broken Internal part No Power</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>OBU Display defective No speed advice</td>
<td>7</td>
<td>No Power/Connect. Broken Internal Part</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>RSU Ethernet fails to connect</td>
<td>2</td>
<td>No physical connection</td>
<td>3</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>RSU Ethernet Packet error/loss/delay</td>
<td>2</td>
<td>Defective TL Programs Wrong speed advice</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>SPAT/Map data wrong Wrong speed advice</td>
<td>10</td>
<td>User Input</td>
<td>2</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

On important point of the *Failure Analysis* is, that it also has to think about the user of the system. In the case of the GLOSA system the analysis must not stop at the display, it also has to consider the reaction time of the user or that he does not even watch the display.

### 5.9. Main Issues of the GLOSA System

The Failure Analysis discovered several unresolved issues, which are summarized and further highlighted in the following sections.

#### 5.9.1. Network Technologies

The range performance of IEEE 802.11p varies widely and depends on the test situation and arrangement. The results of a PER vs. communication range test (line of sight) in an urban environment was measured in [30]. The results are shown in Figure 42; it analyzed the two protocols UDP and WSM (Wave short message protocol, which is comparable to the Geonetworking protocol of the ETSI ITS) and shows that the effective communication range in an urban environment is around 420 meters (when the PER reaches 12%). For a 1500 byte WAVE Short Message this corresponds to a BER of $1 \times 10^{-5}$.

![Figure 42: Tall Urban Canyon vs. Distance for Urban Environment [30]](image)

These measured results are in the requirements (Table 8) that ARINC has specified for SPAT applications. However there are further studies that do not satisfy the requirements, e.g. the
results of [46]. The requirements could only be met with the transmission of further message retries.

Table 8: Failure Rate Allocation, Source: ARINC April 2012 [30]

<table>
<thead>
<tr>
<th>Element</th>
<th>Failure Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traffic signal controller provision of SPAT information</td>
<td>$1 \times 10^{-5}$</td>
</tr>
<tr>
<td>Generation of a SPAT message</td>
<td>$1 \times 10^{-5}$</td>
</tr>
<tr>
<td>Communication of the SPAT message</td>
<td>Based on minimum established, 0.7 probability of packet delivery, BER = $2.5 \times 10^{-4}$</td>
</tr>
<tr>
<td>Decoding of the SPAT message</td>
<td>$1 \times 10^{-5}$</td>
</tr>
<tr>
<td>Assessment of vehicle state (speed and position) relative to the application decision point</td>
<td>$1 \times 10^{-5}$</td>
</tr>
<tr>
<td>Human execution of the application action. (Note: varies with age, complexity of Task and stress level; FR provided is typical)</td>
<td>$0.75 \times 10^{-3}$</td>
</tr>
</tbody>
</table>

5.9.1.1. Bandwidth Considerations

In a V2X scenario it could be possible that the complete bandwidth is used by the vehicle heartbeat messages (CAM/BSM). The document however specified a typical bandwidth of 6 Mbit/s for IEEE 802.11p, since prior tests showed this bandwidth as a result at a usable BER and application range. 6 Mbit/s may not be sufficient in highly populated areas. A calculation example can be found in Table 8.

Table 9: Capacity Analysis for DSRC [30]

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of vehicles crossing per second per direction at intersection:</td>
<td>0.5 cars/s</td>
</tr>
<tr>
<td>Carrier sensing range (2* TX Range)</td>
<td>2*250=500m</td>
</tr>
<tr>
<td>Average speed of vehicles</td>
<td>30mph</td>
</tr>
<tr>
<td>Average duration of a vehicle in sensing vicinity</td>
<td>1000m/30mph=75 seconds</td>
</tr>
<tr>
<td>Number of vehicles in the intersection area (assuming 2 streets i.e., 4 directions-2 way)</td>
<td>2<em>2</em>75*0.5=150 vehicles</td>
</tr>
<tr>
<td>Packet interval for Safety Message</td>
<td>100 milliseconds</td>
</tr>
<tr>
<td>Data rate per intersection (Safety related)</td>
<td>0.4 Kbit/s</td>
</tr>
<tr>
<td>Aggregate offered load (BSM Part 1, 339 Byte @ 10Hz)</td>
<td>4.09 Mbit/s with 150 vehicles</td>
</tr>
<tr>
<td>Aggregate offered load (BSM Part 1+ Part 2, 2091 byte @ 10Hz)</td>
<td>25.5 Mbit/s with 150 vehicles</td>
</tr>
<tr>
<td>Aggregate offered load (ETSI CAM message, approx. 800 Byte @ 10Hz)</td>
<td>9.6 Mbit/s with 150 vehicles</td>
</tr>
<tr>
<td>DSRC Max throughput</td>
<td>6 Mbit/s</td>
</tr>
</tbody>
</table>

5.9.2. GPS Accuracy

The GLOSA system uses GPS as the positioning system. It is possible that the resulting position (heading, speed) may be inaccurate. The position can also be old or unsteady. Therefore the GLOSA system should provide checks that verify the position, e.g. it should check for impossible jumps or vehicle speeds.

The percentile impact of the position to the speed advice is calculated by $impact = \frac{accuracy}{distance} \times 100$. For example when a distance of 100m to the traffic light is assumed, the
remaining time is 8s (result in a speed advice of 45 km/h), the impact to the speed advice will be ±2.25 km/h (±5%). Especially at short distances the position has a high impact and may lead to an unsteady speed advice.

5.9.2.1. **DIFFERENTIAL GPS (DGPS)**

Differential GPS (DGPS) uses correcting data to enhance the accuracy of GPS. It uses a fixed station to broadcast the difference between the position that is indicated by the GPS and the known fixed position. Receivers will correct their positions by subtracting this difference.

The DGPS technique can eliminate many GPS errors, but the corrections lose validity with time and with the displacement to the reference station. A more complete study about the accuracy of DGPS can be found in [47].

In the GLOSA system each RSU can for example generate such a DGPS signal and send it to approaching vehicles. Every RSU has to know its exact position and can calculate the difference to the GPS position.

5.9.3. **ALGORITHM IMPROVEMENTS**

The algorithm that the design proposed in Section 4.4.6 is not suitable. The proposed algorithm misses several problems:

1) Algorithm too simple (v=s/t would display a higher/lower speed than needed). The actual vehicle speed is not considered. Acceleration/Deceleration need some time.
2) Reaction time of driver missing, Safety zone for driver reaction. Probably the human delay is the one with the biggest influence
3) Only considers one signal cycle. If speed advice is above the speed limit, it may be possible to pass the light in the next cycle.

5.9.3.1. **SPEED ALGORITHM LOOP**

The speed algorithm has to consider that the traffic light has 4 different signal cases, which can be split into more specific cases:

**Traffic Light is green (Calculate Speed with the time when traffic light switches to red)**

- Speed advice is higher than the actual speed of the car → the vehicle has to accelerate and can pass the light while it is green.
- Speed advice is lower than car speed → vehicle has to decelerate
- Speed advice is above the speed limit → the traffic light could not be passed while it is green. Maybe it could be passed after the next red phase.

**Traffic Light is red (Calculate speed with time when traffic light will switch to green)**

- Remaining red time is short → Car can decelerate to pass intersection
- At longer distance, driver may accelerate to pass during next green phase (again with time when switches to red)
- It is not possible to pass traffic light (speed advice higher as the speed limit, or lower than the minimum speed advice)
Traffic Light is yellow or red-yellow

- The yellow state should be seen as a red light, so that the driver has a safety distance before the light switches to red
- The red-yellow state can be seen as green. Unfortunately the driver does not have a safety distance here.

The newly designed speed advice algorithm is a loop that runs until the speed advice is smaller than the allowed speed. It returns the speed advice as a string that shows if the driver has to drive smaller or faster than the advised speed. It sums up the state times to calculate the time to the next green phase.

The code for the speed algorithm loop is shown in Figure 43. Normally the speed advice should be a range. However this algorithm finds the upper limit of the speed advice; the one that is closer to maximal allowed speed. It than appends “<” or “>” to specify if you have to go slow or faster than this speed. This was designed in association to the proposed GUI (Section 4.4.5), which shows a red and a green area for a lane.

```java
// Speed Algorithm

time = RemainingTime;
do {
    if (state == RED || state == YELLOW) {
        if (state == YELLOW) {
            time += getRedTime(laneNumber);
        }
        speedAdv = distance / time * 3.6;
        retString = "<";
        time += REDYELLOWTIME;
        state = REDYELLOW;
    } //if green or red yellow
    else {
        if (state == REDYELLOW) {
            time += getGreenTime(laneNumber);
        }
        speedAdv = distance / time * 3.6;
        retString = ">";
        time += YELLOWTIME;
        state = YELLOW;
    }
} while (speedAdv > maxSpeed);
```

For this kind of algorithm the red and green timings are necessary. However the SPAT Message of the SAE Message set does not have these fields (see Section 2.2.1.1).

The proposed algorithm considers the yellow state as a state, where it is not allowed to drive. Therefore the driver has a security distance in which he can decide if he trusts the system or not. For the red state there is however no security distance/time, this distance/time has to be considered by the algorithm, so that the driver can observe that the state already switched to green, before he passes the intersection.
5.9.3.2. **REACTION TIME AND SECURITY DISTANCE**

The developed algorithm has to be further extended. It has to consider the reaction time of the driver, the time for acceleration/deceleration and a safety margin. The safety margin is a distance that is needed to stop the car if the traffic light state of the GLOSA system is not the same as the real traffic light state. It consists of reaction time and the way, which is needed for an emergency braking.

\[
v_{adv} = \frac{s - s_{dec} - 2 \times s_{react} - s_{sec}}{t - t_{dec} - 2 \times t_{react} - t_{sec}}
\]

Equation 1: Advanced Speed Algorithm

The expanded speed advice equation is described in Equation 1 and an example approach to a traffic light in illustrated in Figure 44. Since some of the values depend on the actual speed of the vehicle, it has to be transformed in a more usable format.

Furthermore the algorithm has to consider a lower speed limit. A speed advice that is under this limit (20 km/h was proposed at a speed limit of 50 km/h) should not be shown to the driver, because it could produce traffic or provokes other drivers to risky overtaking. When the system does not display a recommendation, the driver will just go with the normal traffic flow.
6. **DEMONSTRATOR IMPLEMENTATION**

After the design of the GLOSA system a first prototype was implemented. The prototype was developed to connect to a traffic light in Hamburg-Harburg. However it was necessary to make several changes from the original design. One reason was the traffic light system of Hamburg (Section 6.1) and the other one, that it was not possible to use real IEEE 802.11p hardware. The realized prototype is described in Section 6.2.

### 6.1. Traffic Light System of Hamburg

The Traffic Light System in Hamburg is managed by the *Landesbetrieb Straßen, Brücken und Gewässer* (LSBG) and operated by Hamburg Verkehrsanlagen. 1750 traffic lights (traffic light) are scattered over Hamburg, whereof 500 are pedestrian traffic lights only. The traffic light system is controlled by 7 central traffic computers.

Each traffic light is connected by a unique 2-wire cable and a modem to the central traffic center; so the system builds a star topology. Hamburg uses a proprietary protocol for the communication between the traffic lights and the central traffic computer, which is further explained in 6.1.2.

![Hamburg Traffic Light assembly](image)

Each traffic light is quasi independent and does not need a connection to the central to continue operation. A traffic light has various timing programs, which describes how the traffic light behaves. The central traffic computer cannot edit these programs, but it can choose which hard-coded program the traffic light should use\(^3\). The traffic light sends back a status message to the central traffic computer every second, which is mainly used for logging purposes and failure detection. A Central Database logs all information, but is delayed by roughly 3min.

The assembly of a traffic light and its connection to the central traffic computer is shown in Figure 45. It also outlines how the system detects connection losses and wire breaks. The

---

\(^3\) Mainly based on the actual time of the day; so there are week and day plans.
central traffic computer also sends a periodic sync impulse that synchronizes its time with the traffic light. Newer traffic lights however have their own GPS clocks that give a more accurate time.

Even though it would have lots of benefits for Hamburg the system will probably not be modernized during the next 10 years. Maybe this project can give a first starting point to modernize the system and make it more flexible for future use cases.

### 6.1.1. Traffic Light in Hamburg-Heimfeld

For the demonstrator a simple 4-way traffic light close to the University was chosen. The traffic light is located at the intersection of *Heimfelder Straße* and *Lohmanssweg*; the map is sketched in Figure 46. It uses a fixed timing; the traffic light timing plan depends on the weekday and the hour of the day, a program plan from the LSBG could be used to determine the currently running plan. Furthermore the traffic light does not have pedestrian detectors or bus acceleration.

![Figure 46: Map of the traffic light in Hamburg-Heimfeld](image)

An example program plan of the traffic light is shown in Figure 47. When the line is dark, the traffic light is red. At the white box-like line it is green. The crosses out boxes are the yellow and the red-yellow states. K1-K4 are the traffic light groups, which direct the vehicle traffic; F5-F8 are pedestrian lights. The column $An$ specifies the time when the traffic light on, so it is the time when it switches from red to red yellow; $Ab$ is the time when the traffic light
turns off (switches from green to yellow). The Time $TF$ specifies the length of green phase; $TU$ is the length of a cycle period.

Unfortunately there is currently no special program plan for emergencies in Hamburg. Furthermore the traffic light protocol does not give information about emergencies. Therefore the GLOSA system cannot react to emergencies.

**6.1.2. TRAFFIC LIGHT PROTOCOL**

Between the central traffic server and the traffic lights a special protocol is used, but because of privacy reasons the exact protocol [50] will not be specified here; however a short overview is given.

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Start sequence</td>
</tr>
<tr>
<td>2</td>
<td>Length</td>
</tr>
<tr>
<td>3</td>
<td>Control byte</td>
</tr>
<tr>
<td>4</td>
<td>Address byte</td>
</tr>
<tr>
<td>5</td>
<td>Actual Time in seconds</td>
</tr>
<tr>
<td>6</td>
<td>Message ID</td>
</tr>
<tr>
<td>7</td>
<td>Program</td>
</tr>
<tr>
<td>8</td>
<td>Status</td>
</tr>
<tr>
<td>9</td>
<td>Actual time in program cycle</td>
</tr>
<tr>
<td>10</td>
<td>Defective Lights</td>
</tr>
<tr>
<td>11</td>
<td>States of detectors</td>
</tr>
<tr>
<td>12</td>
<td>State of Traffic Lights</td>
</tr>
<tr>
<td>13</td>
<td>Special status</td>
</tr>
<tr>
<td>14</td>
<td>Type of traffic light</td>
</tr>
<tr>
<td>15</td>
<td>Checksum</td>
</tr>
<tr>
<td>16</td>
<td>End sequence</td>
</tr>
</tbody>
</table>

The protocol outlines several message types. For the GLOSA system the most useful one is a log message, which is displayed in Table 10. This message is sent from a traffic light to the central traffic server once a second and is used to log the traffic lights state to a central
database (this database is unfortunately delayed by a few minutes, so it could not be used directly for SPAT information).

The message contains for example the state of the traffic light and the actual time in the current program cycle. It also contains data of the detectors, which detect pedestrian button presses or busses that want preemption for the intersection.

Unfortunately the message does not contain the remaining time of a state. This time could either be predicted by a learning database or the RSU must know the specific program plan and calculate the remaining time itself (by using the actual time in the current program cycle). Furthermore it is also not possible to gather MapData via the protocol.

**6.1.3. TRAFFIC LIGHT INTERFACE OF HAMBURG (TLI)**

One big problem was, how the RSU would synchronize its data with a traffic light in Hamburg. The first idea was to gain the data from the data line between the traffic light and the central traffic server and decode the protocol. A thinkable solution would be to connect between the modem and the internal controller of the traffic light, which is basically an unmodulated V.24 or RS232 connection. A RS232 spy circuit (Figure 48) could be used to gather the data. With this type of interface it would also be possible to send data from the GLOSA system back to the traffic light.

With this solution there is still one problem: there is no direct information about remaining time of the light state. However the actual time (in seconds) in the ongoing program cycle (inapplicable when there is an error with a red light) could be used to calculate the remaining time.

This first idea was however not realized, because of responsibility and insurance problems. It is of course not allowed to place or connect something inside a traffic light. The allowed solution was to place a light sensor inside the red light of a traffic light. With this information it was possible to synchronize a state machine that runs in the RSU. It was however not possible to realize a test on a real traffic light during the thesis anymore.
6.1.4. MapData

Even with a real traffic light interface it would probably not be possible to collect the map of the traffic light from its controller. The data has to be copied manually from the construction plans, which can maybe done automatically later. Maybe it could also be done with crowdsourcing, similar to the idea that Signal Guru introduced in Section 1.2.1. These cars could drive around and scan each intersection.

Since some problems existed with the compiling of the SAE J2735 messages it was only possible to use SPAT or MapData messages alone. Therefor only the SPAT data was used and the MapData was hardcoded into the OBU.

6.1.5. State Machine of Traffic Light

To realize the solution with a light sensor, it was necessary that the RSU knows the traffic light program. The idea for the demonstrator was that the RSU runs a state machine, which simulates the complete traffic light.

For the demonstrator only one single program of the traffic light in Hamburg-Heimfeld was implemented. To realize the state machine, each traffic light group was implemented as an own traffic light as shown in Figure 49. A superior state machine controls a list of light state machines and iterates over it every second to switch to the next state. This state machine is synchronized by the red light sensor and resets all corresponding traffic lights when necessary, which is further describe in Section 6.2.1. The State Machine is updated every second, since a real traffic light only changes once per second (and also sends the log message once a second).

![Traffic Light State Machine](image)

Figure 49: Traffic Light State Machine

6.1.5.1. Complex State Machine

There are also traffic lights that are much more complex. Theses traffic lights could have detectors for pedestrians, vehicles or communicate with public transportation to speed up busses.

The specification for these traffic lights was different to the specification for the simple traffic light in Heimfeld. In addition to the cycle diagram the specification for these traffic lights also includes something that is similar to a state machine, an example is shown in
Figure 50. The Cycle diagram in Figure 51 looks similar to the simple one, but phase sequence and length can vary.

![Complex traffic light state machine](image)

For this kind of traffic lights another kind of state machine has to be developed. An idea that follows the specification is that each state describes, which traffic light groups are turned on and which are off; so a state (of the state machine) describes the state of all traffic light groups (which groups are turned on) and is followed by transition to next state.

![Complex traffic light cycle diagram](image)
A problem is that the triggers for the transitions between the states are not really specified in the specification document by the LSBG. The development of a general state machine or traffic light interface has to be part of a future thesis.

6.2. **Demonstrator**

The demonstrator for the GLOSA system was implemented on two *Raspberry Pi’s*; a low cost, credit card sized computer running the standard *Raspbian* Linux (Debian). Both applications were implemented in C++ that was supported by further libraries like the *BOOST* library and the SAE J2735 message library.

![Figure 52: OBU in test vehicle](image)

Unfortunately it was not possible to use real IEEE 802.11p network communication. There are some modules on the market\(^4\), but they are expensive and availability is not clear. To simulate the behavior of IEEE 802.11p the monitor mode for WLAN was used. With the monitor mode enabled it is possible to receive all packets that the antenna can receive from the air, without filtering out messages of unrelated radios (it does not filter BSS, see Section 2.1.1.1). With the monitor mode it was possible to receive data from every wireless networks, but when the network is secured the message will be encrypted. The usage of the monitor mode should imitate an IEEE 802.11p network device, where it is not necessary to authenticate at an access point. With this solution the RSU can set up a wireless ad-hoc

---

\(^4\) List of suppliers: Arada Systems LocoMate OBU, RENESAS ELECTRONICS, NEC, NXP, Cohda Wireless, Cisco, Denso, Delphi, Savari, Kapsch (Raspberry Pi and Evaluation Kit EVK-3300), ITRI
network and broadcasts SPAT messages into the air. The OBU can use the monitor mode to receive these packets without the authentication to the network.

There are further attempts to simulate IEEE 802.11p under Linux; e.g. [52] describes in detail which parameters have to be changed. These attempts try to simulate more details and use for example 10 MHz channels. Special hardware (e.g. Unex DCMA-86P2 Atheros-based, GNU Radio [53]) and modified drivers have to be used. The approximated IEEE 802.11p is however limited in its transmit power, which results in a less robust signal and a lower communication range.

Furthermore the SAE J2735 message library provided by SAE made some problems; it was not possible to compile when using the SPAT and MapData message in one project (Linker Error: Redefinition of symbols). A future solution may be to compile the ASN.1 files that describe the message structure to another language like C++ or Java. Unfortunately most of these ASN.1 compilers are not open source.

![Figure 53: RSU Photos](image)

Therefor it was only possible to use the SPAT messages (MapData was hardcoded into the OBU) as designed. They have been encoded with the *Distinguished Encoding Rule* (DER), a standard ASN.1 encoding format; these encoding and decoding functions were provided by the SAE library. Also red/green times are not include in the SPAT message, they are important for the speed advice to consider the next state cycles. For the red and green timings other fields have been used.

The prototype also implemented a first mobile communication interface that provides the wanted data for the Internet of Things support. The realized implementation is further explained in Section 6.2.3. and could be used by the IPhone App explained in Section 6.2.4.
6.2.1. RSU

The RSU application prototype was implemented on the first Raspberry Pi that was equipped with a WLAN Stick and a light sensor. The RSU runs the state machine that was already explained in Section 6.1.5. An UML Class Diagram is sketched in Figure 54.

The demonstrator RSU was using a power bank, so that it could be attached to the traffic light in Heimfeld (Section 6.1.1) later. With this battery the Raspberry Pi could survive for roughly one day.

![RSU Class Diagram](image)

The RSU sets up a WLAN Ad-hoc network, which is used to broadcast the SPAT messages. It also runs a DHCP (Dynamic Host Configuration Protocol) Server, which allows it to connect a Laptop to the network, get an automatic IP address and enables a SSH connection to the Raspberry.

The **SPAT Message** is a shared object between the **COAP** and the **SPATSender**. The **IntersectionSM** is the Simple State Machine that was already describe in Section 6.1.5. It manages a list of **TrafficLightSM** objects, which represents a traffic light group state machine. It basically reimplements the traffic light and simulates its light states.

The **SPATSender** is a boost **TimerTask**, which is executed every 500ms. It uses the **IntersectionSM** object to update the **SPATMessage** with the actual traffic light state. This task also broadcasts the **SPAT Messages** over the WLAN interface. It uses a basic **UDPSender** that sends out the messages in the ad-hoc network.

The **TLI** class is the actual interface to the traffic light. It reads out the light sensor and synchronizes the **IntersectionSM**. The light sensor is connected to the Serial Peripheral Interface (SPI) of the Raspberry Pi. It is composed of a phototransistor that is connected to an Analog-to-Digital-Converter. In the software a simple threshold was be used to determine
if the red light is on. The C library for Broadcom BCM 2835 for the Raspberry Pi was used to communicate with the sensor via SPI.

```c
if(trafficLightIsRed() != (StateMachineTrafficLight == RED)) {
    // wait tolerance time and check again
    wait(100ms);
    if(trafficLightIsRed() != (StateMachineTrafficLight == RED)) {
        printf("Out of sync\n");
        // wait until traffic light turns red again, then reset state machine
        // traffic light already red? Wait for not red
        while(trafficLightIsRed()) {
            printf("Wait for LSA to change from Red to sth else\n");
            wait(100ms);
        }
        while(!isRed()) {
            printf("Wait for LSA to turn red again\n");
            wait(100ms);
        }
        // reset State Machine
        printf("reset traffic lights");
        intersectionSM.doSync(0);
    }
}
```

Figure 55: Sync Algorithm

A simple synchronization algorithm is shown in Figure 55. The TLI class is a separate thread that executes the algorithm every 100 ms. It checks if the traffic light and the state machine are red on the same time. If not it is checked again after a security/tolerance interval. When it is still not synchronized a synchronization process will be started. The algorithm will wait to the point when the red period starts and resets the IntersectionSM.

The light sensor for the synchronization has to be placed carefully inside the red light. The lights are waterproof and the cable may destroy the sealing. A threshold was used to determine if the light is red, to prevent sun reflections that might disturb the light sensor, the traffic light group in the north was chosen.

The traffic light program cycle had to be shifted to start with a point where the traffic light turns from yellow to red. The shifted cycle diagram for the traffic light is shown in Figure 56.

Figure 56: Sync Algorithm

---

5 http://www.airspayce.com/mikem/bcm2835/
6.2.2. OBU

The OBU of the demonstrator was implemented on the second Raspberry Pi, which was furthermore equipped with a 2.5” touch screen and an USB GPS-Receiver. In the car the Raspberry Pi was powered by the 12V lighter output or another power bank. The UML Class Diagram of the OBU is sketched in Figure 57.

![OBU Class Diagram](image)

Figure 57: OBU Class Diagram

The Position, MapData and the SPAT Message are shared objects that are used by the three other tasks.

The PositionUpdateTask reads out the GPS Receiver and updates the position object. It is derived from the TimerTask of the boost library, so basically a task that is started every 200 ms. An USB GPS Receiver and the Linux GPSD service was used to provide the data to the application. Therefore a C++ API library was used. The Navilock NL-602U GPS-Receiver (u-Blox 6) supports a baud rate of 115200, which leads to an update rate of 200 ms.

The V2XReceiving class is a completely separate thread, which uses a TP-Link TL-WN722N WLAN Stick to receive the SPAT messages in monitor mode. The PCAP library was used to set the up monitor mode and filter the received messages. At the moment it filters port and IP only (later it should also check the DSRCmsgID). It also removes the overhead and converts the message into a SPAT message that could be used by the application.

The ShowSpeedAndTTCTask is also a TimerTask that is executed every second. It calculates the speed advice and displays it to the console. A small GUI was written that uses the curses library to display important data like speed advice and the remaining time. The graphical user interface that was explained in Section 4.4.5 was not implemented yet and only the information of one single lane is displayed. The current GUI is shown in Figure 58; for debug purposes it shows data all the time, even in case of a detected error (e.g. the SPAT data was old during the screen shot, the real GLOSA system should not display a speed advice in such a case).
This task uses a simplified algorithm for the speed advice; currently it only divides the distance by the remaining time. No algorithm for the intersection finding was implemented. The demonstrator only supports one intersection (as a result it also displays a speed advice when moving away from the intersection). A simplified algorithm for lane finding was used. It chooses the lane by calculating if the car is in a $\pm45^\circ$ angle to the offset vector of the lane.

All data that the OBU calculates and shows to the driver was logged to a file, to visualize it graphically later.

### 6.2.3. CENTRAL ITS STATION

For the first prototype it was not necessary to implement a CentralITSStation. However a first basic application was implemented to access the SPAT data over mobile communication. In mobile communication it is not possible to broadcast information, therefor the OBU application has to request the SPAT Data at the CentralITSStation. To realize these needs only a small part of the functionality was implemented. Basically the implemented station is a COAP Server with a SPAT Message resource that runs on a computer at home, it was possible to access this resource with a DynDNS URL.

The RSU updates the SPAT resource in a specific time interval and the OBU can request the updated data from the CentralITSStation. With the implemented CentralITSStation an IPhone app (Section 6.2.2) could access the SPAT data over UMTS and display a speed advice to the user. The COAP class in Figure 59 sets up a COAP Server and provide a SPAT message resource.
6.2.4. IPHONE APP (OBU)

During the thesis also a first IPhone app was developed. The idea of this was that a smartphone is already equipped with all necessary devices for the GLOSA system. It has a built-in GPS-Receiver, a display, WLAN, UMTS and a power source. However it does not allow using the monitor mode to simulate a IEEE 802.11p device.

The app uses the COAP protocol to request the necessary SPAT messages from the Central ITS Station. With the built in GPS-Receiver it could calculate the speed advice similar to the OBU from the last section. As Figure 60 shows, it also displays GPS information, the remaining time, the distance and the actual state.

![Figure 60: IPhone App Screenshot](image)

The app was written in Swift, a language for developing apps for iOS. However the decoding (BER decoding) of the SAE messages and the calculation of the speed advice is done in plain C. The Swift language is able to use C code and build a bridge function that could be used in the Swift code. This C code is basically the same as it was already used in the OBU demonstrator application.
7. **DEMONSTRATOR TESTING**

To find new failure modes, algorithm errors and software bugs the demonstrator should be tested as early as possible. The test drive was done to find further new *Failure Modes*, which have not been considered during the *System Design* or overlooked during the *Failure Analysis*.

7.1. **BASIC OPERATION TEST**

A basic operation test was made on a straight street in an industrial area without traffic. It was made on the Street “Neuer Höltigbaum”, a map of the area is shown in Figure 61. The red dot marks the position of the RSU (traffic light); the blue line shows the driven route (max. distance 390 m).

For this test no physical traffic light was used, to eliminate the effects of the traffic light synchronization. The test showed that the application runs, displays a useful speed advice and the remaining state time, which allow passing the traffic light at green.

An example Graph is shown in Figure 62, more data is displayed in the appendix. In the diagram the speed advice has gray upper bars, which means that the driver had to drive faster than the displayed speed (e.g. >33 km/h). In other cases in the appendix it has lower bars to show that the driver had to drive slower (e.g. <21 km/h).

7.1.1. **GPS**

The test also showed that the speed advice is very unsteady at short distances. Sometimes the lane switched shortly in front of an intersection and showed a wrong state and speed advice.

I also observed that the GPS position seems to be delayed. Sometimes I passed the traffic light, but the GPS said that it is still 30 m away. This is unfortunately not visible in the logged data and has to be further investigated. Furthermore the update rate of the GPS was not 200 ms as programmed; the
data showed that the position only switched once a second.

7.1.2. Algorithm

Even though the reaction and acceleration time was not considered in this first algorithm, I observed that it was very usable. Most of the time at 380 m a SPAT message was received and a first speed advice was displayed. When I took this speed I passed the traffic light at green.

However it may be necessary to calculate a speed advice range. Sometimes the Speed advice switched from “>49 km/h” to “<14 km/h”. The lower bound should be displayed earlier. However the proposed GUI would make the use and driving much more intuitive.

Another problem was that there was no security distance/time in when the state switched from red to green. It was not possible for the driver to decide if he can trust the system.
7.1.3. **Range**

At a 380 m line of sight it was often possible to receive the first SPAT messages and display a first speed advice. However the range decreased dramatically, when there were vehicles that blocked the line of sight. The connection was not really reliable before 200 m.

7.1.4. **Error Handling**

Errors have been detected and displayed when the system was out of range and no actual SPAT data was available (SPAT 3 s old or remaining time negative). Errors are shown as yellow diamonds in the diagram.

Unfortunately the *Raspberry Pi’s* do not have a real-time clock. Therefor it was not possible to detect delayed messages or subtract the transmission delay from the SPAT data (remaining time). The OBU could have used the GPS receiver as a time source, however the RSU did not have one. A better suitable time source should be used in future prototypes.

7.2. **Demonstrator Summary**

The first operation tests of the demonstrator showed that the prototype does not have enough range to meet the requirements. According to tests the maximum reliable range was around 200m; at least 300m (1000 m line of sight) was required by the specification, otherwise the system would not be economic [7]. This should later be solved by using IEEE 802.11p technology.

Even though the explained problems the GLOSA demonstrator was quite usable. Most of the time it was possible to trust the first displayed speed advice and just take this speed roughly for the complete distance. It would be even more intuitive with the proposed GUI. However it has to be observed how much the GPS delay will affect the usability on a real traffic light.
8. Thesis Conclusion

The development and realization of a reliable GLOSA system is a relevant topic for future traffic systems. It is a promising technique to save up to 20% of fuel, to save stop time and optimize the traffic efficiency.

The first step of this thesis explained the underlying techniques that have been used. It showed the basics of V2X Communication, system development techniques and failure analysis steps.

During the main part of the thesis the specifications for the GLOSA system have been investigated. SysML was used to analyze stakeholders, requirements and use cases and furthermore to design the systems hardware blocks, interfaces and activities.

FTA, FMEA was used to analyze possible Failure Modes and a first prototype was realized on two Raspberry Pi'. It was observed that the basic concept and the used techniques can work, but that there are still many points that need further research.

8.1. Recommendations for Future Research

The Failure Analysis and the developed demonstrator show that there is still plenty of research, that has to be done, to realize a reliable GLOSA system. This section lists some of the ideas and problems that need further research and gives a short outlook for future steps.

8.1.1. SysML

The SysML model saved a few steps of the Failure Analysis and helped to realize the prototype much faster. However more advanced SysML development tool have to be used, which could check the design rules and generate requirement tables automatically. For example there are tools that can verify if requirements are satisfied. An open source tool that provides support for SysML is for example the Eclipse Papyrus Plugin.

8.1.2. Algorithm

Altough the simple algorithm was working quite well for the demonstrator, there is further research necessary for the speed advice algorithm. As already mentioned it has to be investigated if the proposed enhancements (e.g. consider reaction time, acceleration time, safety distance) solve the discovered problems or if there is further research necessary. For example [54] gives a good overview about different speed advice algorithms and introduces a new genetic algorithm, which takes all following traffic lights on a route of a vehicle into account. It calculates a speed advice for every road segment and allows to select preferences, like minimization of traveling time or fuel consumption.
Furthermore the algorithms for finding intersection and driving lanes have to be improved. The algorithm should be supported by a map, to consider curves and other obstacles (this also important for the speed advice calculation). Also some further data may be required by these algorithms, e.g. the correct lane can be chosen, if the system has a sensor that is connected to the vehicles turn light. It could not be proven that the GLOSA system saves fuel and travel/stop time. This requirement needs further extensive testing, when a more reliable prototype is developed.

8.1.3. Scalability

The scalability of the system has to be investigated. It has to be possible to use the system in different cities and countries; it has to be interoperable with the American WAVE (IEEE 1609) system and its protocols. Therefor the different traffic light interfaces have to be analyzed. Sometimes it will be possible to gather the necessary data, other times there have to be found different solutions. The State Machine solution that was used in the demonstrator is not realizable for a large and complex traffic light system.

One of the biggest problems of the GLOSA system is the gathering of the traffic light data (SPAT and Map data). Either new traffic light controllers are necessary that can deliver this data, or solutions have to be found, which creates these data automatically. At the moment the process to gather the traffic light data is very complex. A State Machine has to be programmed manually for each traffic light and is not straightforward (state diagrams have to be shifted, MapData has to be translated and so on).

8.1.4. Security Concerns

In IEEE 802.11p the Mac layer authentication is removed, so it has to be investigated how security mechanisms have to be implemented in the system. It is not risky that an Attacker reads out the SPAT and MapData, but the manipulation is. An Attacker could manipulate SPAT or MapData data to provoke crashes and injure people.

Furthermore there is research about privacy in ITS necessary. It may be possible to track vehicles and create movement profiles.

8.1.5. Demonstrator Recommendations

In the next phase of the project the demonstrator should be connected to the traffic light in Hamburg-Heimfeld and extensively tested; therefor the implemented synchronization process of the RSU has to be verified at the real traffic light.

Later the demonstrator has to be realized and investigated using real IEEE 802.11p hardware that allow more range and a more reliable connection. Furthermore the smartphone app and the mobile communication solution have to be further developed and tested.
The graphical user interface, which was proposed in Section 4.4.5, has to be implemented and tested during a real drive. Moreover the demonstrator should also implement more intersections and lanes.

### 8.1.6. General Environment

Mobile devices allow plenty of advantages. They already have all necessary interfaces (WLAN, GPS), are cheaper to use and easier to update than expensive integrated units. However there is still an OBU necessary that forwards the IEEE 802.11p messages to the Smartphone. It has to be investigated if the already installed LTE networks can sustain the requirements by V2X systems or if there has to be set up an additional/jurisdictional network for vehicle communication.

At last the GLOSAs has to be extended to receive and analyze Emergency Vehicle Alert messages, which realize a faster reaction to emergencies. It could also analyze CAM messages to prevent crashes. All these V2X communication applications have to be combined into one single, intuitive application. This application should allow: navigation, GLOSAs, crash warning and emergency warning.

### 8.2. Outlook

In the future, the GLOSAs system may give feedback to the traffic lights, which can then optimize their state cycle. Therefor the RSUs may analyze the CAM messages of nearing vehicles and use the traffic light interface to change the timing.

Beyond this, a new generation of traffic light systems has to be developed, which combines the RSU and the traffic light and allow an easier use of the data for GLOSAs systems or other applications. This newly designed ITS brings plenty of advantages and allow an efficient traffic flow for future generations.
9. Bibliography


[41] VDA, Band 4 Ringbuch, Sicherung der Qualität in der Prozesslandschaft. 20011.


## 10. List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>IEEE 802.11 infrastructure modes [17]</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>IEEE 802.11 MAC frame format [16]</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>Shortened ETSI ITS Protocol Stack</td>
<td>17</td>
</tr>
<tr>
<td>5</td>
<td>ITS-G5 frequencies [21]</td>
<td>17</td>
</tr>
<tr>
<td>6</td>
<td>Geonetworking examples [22]</td>
<td>18</td>
</tr>
<tr>
<td>7</td>
<td>SAE J2735 Signal Phase and Timing Message</td>
<td>21</td>
</tr>
<tr>
<td>8</td>
<td>Example Map of an Intersection</td>
<td>21</td>
</tr>
<tr>
<td>9</td>
<td>SAE J2735 MapData Message</td>
<td>22</td>
</tr>
<tr>
<td>10</td>
<td>SysML diagram types [33]</td>
<td>22</td>
</tr>
<tr>
<td>11</td>
<td>SysML Requirement Diagram [34]</td>
<td>23</td>
</tr>
<tr>
<td>12</td>
<td>SysML Block Definition Diagram [34]</td>
<td>24</td>
</tr>
<tr>
<td>13</td>
<td>SysML Internal Block Diagram [34]</td>
<td>24</td>
</tr>
<tr>
<td>14</td>
<td>SysML Activity Diagram [34]</td>
<td>25</td>
</tr>
<tr>
<td>15</td>
<td>Simplified System Analysis (left) and System Design (right) [35]</td>
<td>26</td>
</tr>
<tr>
<td>16</td>
<td>Failure Tree Analysis examples</td>
<td>27</td>
</tr>
<tr>
<td>17</td>
<td>Requirements Diagram</td>
<td>35</td>
</tr>
<tr>
<td>18</td>
<td>Essential Requirements</td>
<td>37</td>
</tr>
<tr>
<td>19</td>
<td>System Context Diagram</td>
<td>38</td>
</tr>
<tr>
<td>20</td>
<td>Core Use Cases</td>
<td>39</td>
</tr>
<tr>
<td>21</td>
<td>GLOSA System Overview</td>
<td>40</td>
</tr>
<tr>
<td>22</td>
<td>GLOSA System Block Diagram</td>
<td>41</td>
</tr>
<tr>
<td>23</td>
<td>Central ITS Station with Interfaces</td>
<td>42</td>
</tr>
<tr>
<td>24</td>
<td>Central ITS Station Internal Block Diagram</td>
<td>43</td>
</tr>
<tr>
<td>25</td>
<td>Central ITS Station activity diagram</td>
<td>43</td>
</tr>
<tr>
<td>26</td>
<td>GLOSA RSU with interfaces</td>
<td>44</td>
</tr>
<tr>
<td>27</td>
<td>RSU Internal Block Diagram</td>
<td>45</td>
</tr>
<tr>
<td>28</td>
<td>RSU activity diagram</td>
<td>45</td>
</tr>
<tr>
<td>29</td>
<td>OBU with interfaces</td>
<td>46</td>
</tr>
<tr>
<td>30</td>
<td>OBU Internal Block Diagram</td>
<td>47</td>
</tr>
<tr>
<td>31</td>
<td>OBU Activity Diagram</td>
<td>48</td>
</tr>
<tr>
<td>32</td>
<td>Graphical User Interface example [12]</td>
<td>50</td>
</tr>
<tr>
<td>33</td>
<td>Failure Analysis Process based on [45]</td>
<td>53</td>
</tr>
<tr>
<td>34</td>
<td>Cause and Effect Diagram</td>
<td>54</td>
</tr>
</tbody>
</table>
11. LIST OF TABLES

Table 1: List of Acronyms ................................................................................................................... VII
Table 2: IEEE 802.11 data frame address field content ................................................................. 15
Table 3: ITS-G5 European channel allocation .................................................................................. 18
Table 4: LTE Unicast Analysis [30] .................................................................................................. 31
Table 5: Stakeholders ....................................................................................................................... 33
Table 6: Table of Failures ................................................................................................................. 54
Table 7: FMEA Table according to SAE J1739 ............................................................................. 61
Table 8: Failure Rate Allocation, Source: ARINC April 2012 [30] .............................................. 63
Table 9: Capacity Analysis for DSRC [30] ....................................................................................... 63
Table 10: Traffic Light Protocol log message .................................................................................. 69
12. APPENDICES

12.1. FURTHER TEST DRIVE DATA
12.2. Content of the Attached CD

- PDF version of the Thesis
- Traffic Light Documents
- Code of the Prototype
  - Central ITS Station (C++)
  - RSU (C++)
  - OBU (C++)
  - IPhone App (Swift)
  - PCAP test application (C++)
  - SAE Message Set (C)
- Visio Graphics (SysML stencil link)
  - SysML Diagrams and other Figures
- Test drive data, plots and photos