Hochschule Esslingen
University of Applied Sciences

Forschungsarbeit

Anforderungen an eine robuste Car-2-Infrastructure Kommunikation für einen autonomen Fahrbetrieb zur erweiterten Umfelderfassung

im Studiengang Angewandte Informatik
der Fakultät Informationstechnik

angefertigt von
Patrizia Spinola

Stuttgart, 2. Januar 2022

Inhaltsverzeichnis

Abkürzungsverzeichnis

Symbole

3GPP 3rd Generation Partnership Project

A

AK Anwendungsklassen
AU Application Unit

C

C-ITS Cooperative Intelligent Transport Systems
C2C-CC Car to Car Communication Consortium
CAM Cooperative Awareness Messages
CAN Controller Area Network
Car2Car Fahrzeug-zu-Fahrzeug-Kommunikation
Car2I Fahrzeug-zu-Infrastruktur-Kommunikation
Car2X Fahrzeug-zu-X-Kommunikation

D

dBm Dezibel Milliwatt
DENM Decentraliced Enviornmental Notification Messages

E

ETSI European Telecommunications Standards Institute

G

GNSS Global Navigation Satellite System
GPRS General Packet Radio Service
GSM Global System for Mobile Communication

H

HSDPA High Speed Downlink Packet Access

I

IEEE Institute of Electrical and Electronics Engineers
IKT Informations- und Kommunikationstechnologie
IoT Internet of Things
ITS Intelligent Transport Systems
IVI Infrastructure to Vehicle Informationen

L

LTE Long Term Evolution

M

M2M Machine-to-Machine
MANET Mobile Ad Hoc Network
MAP MAP (topology) Extended Message

O

OBU On-Board Unit
OEPNV Öffentlicher Personennahverkehr
OEV Öffentlicher Verkehr
OFDM Orthogonal Frequency Division Multiplexing

P

PRE-DRIVE C2X Preparation for Driving implementation and Evaluation of C-2-X

R

RSU Road-Side Unit

S

SPAT Signal Phase and Timing

U

UC Use Case
UMTS Universal Mobile Telecommunications System

V

V2I Vehicle-to-Infrastructure
V2N Vehicle-to-Network
V2P Vehicle-to-Pedestrian
V2X Fahrzeug-zu-X-Kommunikation
VANET Vehicular Ad Hoc Network

W

WAVE Wireless Access in Vehicular Environments
WLAN Wireless Local Area Network

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

1 Megatrends „Vernetzte und automatisierte Mobilität“ und „intelligente Verkehrssysteme“

1.1 Aktuelle Herausforderungen und Chancen in der Mobilität

Die Vision von smarten Objekten, welche mit digitaler Logik, Sensorik und der Möglichkeit zur Vernetzung ausgestattet ein „Internet der Dinge“ (engl. Internet of Things, IoT), bilden, ist in der Praxis schon deutlich ausgeprägt sichtbar. Im IoT-Konzept, in dem der Computer als eigenständiges Gerät verschwindet, sollen die smarten Dinge (engl. Smart Products) die Anwender bei ihrer täglichen Routine unterstützen statt selbst Gegenstand der menschlichen Aufmerksamkeit zu sein. [vgl. Schneider 2007, S. 1-2]

In den letzten zwanzig Jahren hat das Internet die Welt erobert und ermöglicht heute die weltweite Vernetzung unter Nutzung der sich ebenfalls rasant entwickelnden Informations- und Kommunikationstechnologie (IKT). Hierdurch wachsen die physikalische und virtuelle Welt (engl. Cyber-Space) immer enger zusammen. Bei dieser Verschmelzung werden Objekte intelligent entwickelt, um direkt oder über das Internet mittels sogenannter Maschine-zu-Maschine-Kommunikation (engl. Machine-to-Machine, M2M) in Echtzeit zu kommunizieren. Dadurch können sie eigenständig Informationen austauschen, bestimmte Aktionen auslösen und einander steuern. Im Fachjargon wird mittlerweile vom „Internet der Dinge und Dienste“ gesprochen, da die schlauen Objekte ihre Fähigkeiten als intelligente Dienste anbieten [vgl. Kagermann 2011, S. 2]. Die Vorteile dieser technischen Möglichkeiten lassen sich vor allem aber im Verkehr und der Mobilität realisieren und bieten so neue Chancen. Auch hier ist die Digitalisierung, kurz „Mobilität 4.0“, längst angekommen. Die Entwicklung automatisierter und vernetzter Fahrzeuge, als wichtiger Bestandteil moderner Mobilität, hat begonnen. [vgl. Malleck 2015, S. 371]

Schon seit einigen Jahren zeichnet sich der Trend in Richtung Smart Mobility ab. Smart Mobility legt das Hauptaugenmerk auf die effiziente und sichere Gestaltung des Verkehrsflusses. Dabei wird die Fahrzeug-zu-Fahrzeug-Kommunikation (engl. Car2Car) und Fahrzeug-zu-Infrastruktur-Kommunikation (engl. Car2Infrastructure, Car2I) eingesetzt, bei der Fahrzeuge sowohl untereinander als auch mit ihrer Umgebung (z. B. Ampeln, Parkplätze, Verkehrssensoren) kommunizieren. Zusammengefasst werden die verschiedenen Kommunikationswege unter dem Oberbegriff Fahrzeug-zu-X-Kommunikation (engl.

Car2X oder V2X). [vgl. Wolter 2012, S. 528ff]

Die Mitteilung der Europäischen Kommission über eine EU-Strategie für die Einführung von kooperativen intelligenten Transportsystemen (engl. Cooperative Intelligent Transport Systems, C-ITS) kann einen enormen Beitrag leisten, die Mobilität der Zukunft sicherer und reibungsloser zu gestalten. Diese unterstützen unter Verwendung der Car2X-Kommunikation eine Reihe von Informations-, Warn- und Hilfsdiensten. Damit wird die Verkehrssicherheit erhöht, eine effektivere Nutzung der Infrastruktur ermöglicht und eine Reduzierung von Emissionen begünstigt. Dies stellt einen wichtigen Meilenstein auf dem Weg zu einer vernetzten und automatisierten Mobilität dar. [vgl. Kommission 2016, S. 2]

Ziel dieser Strategie ist es nicht nur Deutschlands Wettbewerbsfähigkeit zu stärken, sondern Städte mit ihrer gesamten Infrastruktur zu einer intelligenten Umgebung zu formen. (engl. Smart City) Im Mittelpunkt stehen hierbei die Kommunikation kooperativer, vernetzter und automatisierter Fahrzeuge sowohl direkt miteinander als auch mit der lokalen Straßeninfrastruktur. Dies dient dazu, um vor potenziell gefährlichen Situationen zu warnen (z. B. Stauenden) oder zur Optimierung der Geschwindigkeit durch Interaktion mit Verkehrsampeln. Vor allem aber geht es verstärkt darum, eine reibungslose Kommunikation zwischen Fahrzeugen, der Infrastruktur und den anderen Straßennutzern und eine verlässliche Übertragung der Informationen, deren Richtigkeit vorausgesetzt, zu gewährleisten. [vgl.Kommission 2016, S. 2ff]

Mit dem Einzug der C-ITS entsteht im urbanen Raum ein Paradigmenwechsel, bei dem öffentliche Verkehrsmittel autonom die Verkehrsroute bestimmen. Dieser Wandel bringt ganz neue Herausforderungen und eröffnet neue Möglichkeiten. Die bisher zentrale Steuerung der Fahrzeuge durch Menschen wird schrittweise durch die Einführung kooperativer intelligenter Verkehrssysteme ersetzt, bei denen Informations- und Kommunikationstechnologien im Straßenverkehr, einschließlich seiner Infrastrukturen, Fahrzeuge und Nutzer, eingesetzt werden. [vgl. BMVI 2013]

Aus diesem Grund verfolgt die Hochschule Esslingen mit verschiedenen Industriepartnern Ansätze, eine robuste und gleichzeitig sichere Verkehrs-Infrastruktur zu entwickeln, um Informationen von kooperativen, vernetzten und automatisierten Linienbussen empfangen und verarbeiten zu können. Dabei beschäftigt sich die Hochschule Esslingen gezielt mit der

- Veränderung der Infrastruktur zur optimierten maschinellen Wahrnehmung,
- Steigerung der autonomen Geschwindigkeit auf Basis unterstützender Ereignisstrategien (extern) und
- Untersuchung von Potential und Auswirkung eines autonomen Linienverkehrs im urbanen Mischverkehr,

um eine autonome Buslinie in der Stadt Waiblingen bereitstellen zu können. Konzentriert werden diese Ansätze im Rahmen des Forschungsprojekts namens „AMEISE“.

1.2 Vorstellung des Projekts „AMEISE“

Das Verbundforschungsprojekt „AMEISE“ thematisiert sechs große Arbeitspakete, die im Folgenden kurz beschrieben werden. Dabei wird die vorliegende Arbeit im Rahmen des ersten Arbeitspaketes „Infrastruktur Ausstattung“ verfasst.

Die ersten Entwicklungsschritte betreffen die infrastrukturelle Ausstattung. Hier wird eine Veränderung der konventionellen Infrastruktur durch den Einsatz externer Sensorik und unter Verwendung einer 5G-basierten Car2X-Kommunikation angestrebt. Dies betrifft die Straßenausstattung, Haltestellen sowie Markierungen und Beschilderungen.

In der zweiten Stufe wird eine Fahrzeugbeschaffungsstrategie entwickelt, um den autonomen Fahrbetrieb zu ermöglichen. Das umfasst sowohl die Linieneinrichtung als auch die Betriebsausstattung (z. B. Stellplatz, Ladesäulen etc.). Die Strategie zieht folgende Fahrzeuge in Betracht:

- e.Go, bietet futuristische Fahrzeuge für bis zu 15 Personen (People Mover)
- EFA-S Umrüstung, eine Umrüstung eines z. B. Sprinters mit offener Plattform zum Elektrobus inklusive sensorischer Ausstattung und
- Easy Mile, bietet eine Out of the Box Lösung für einen öffentlichen Personentransport.

Das nächste Arbeitspaket befasst sich mit der Erhebung, Verarbeitung und Ablage von Daten. Zunächst geht es um die Einrichtung eines 5G Campus-Netzes und die Anbindung an eine performante und sichere IT-Infrastruktur. Parallel dazu werden die Potenziale der existierenden Möglichkeiten zur Erfassung von Daten (Automative WLAN 802.11p, Cellular 5G, Wi-Fi 6. Gen. WLAN AX) untersucht. Auf Basis der erarbeiteten Potenziale wird anschließend eine Kategorisierung der Daten vorgenommen. Ziel ist es, durch die Beobachtung des Verkehrs und der kooperativen Zusammenführung von internen und externen Daten, eine Visualisierung dieser Daten zu ermöglichen. Damit soll den Fahrgästen über das Infotainmentsystem das autonome Fahren erlebbar gemacht werden.

Ein weiteres Arbeitspaket behandelt die verkehrsökonometrischen Auswirkungen eines autonomen Verkehrs. Hierbei wird sich mit der Verkehrsmodellierung (z. B. Modellierung neuer Angebotsformen, Netzplanung), Angebotsplanung sowie -optimierung (z. B. integrierte Planung OEV), Verkehrstechnik und Verkehrsflusssimulation beschäftigt. Die simulative Abbildung des Verkehrs spielt dabei eine wichtige Rolle. Anschließende Analysen können Aufschluss über die Gesamtbewertung und Nutzerakzeptanz für autonome Buslini-

en oder über Beschäftigungs- und Karriereperspektiven im automatisierten Fahren geben.

Im fünften Arbeitspaket wird sich damit befasst, was eine Automatisierung für den OEPNV bedeutet. Das umfasst beispielsweise Fragen wie: Welche (neuen) Job-Möglichkeiten wird es geben? Welcher Schulabschluss sowie welche Qualifikationen werden für den Beruf als Busfahrer benötigt? Inwieweit ist eine Schulung alter Mitarbeiter notwendig? Diese und viele weitere offene Fragen werden mit den Verbänden, Angestellten und Gewerkschaften geklärt werden.

Das sechste und somit letzte Arbeitspaket adressiert die Projektkoordination. Das betrifft Bereiche wie Kostenübernahme, Präsentation, Pressetermine und Außendarstellung. [vgl. AMEISE 2021]

Das folgende Bild zeigt das soeben beschriebene Konzept von „AMEISE“.

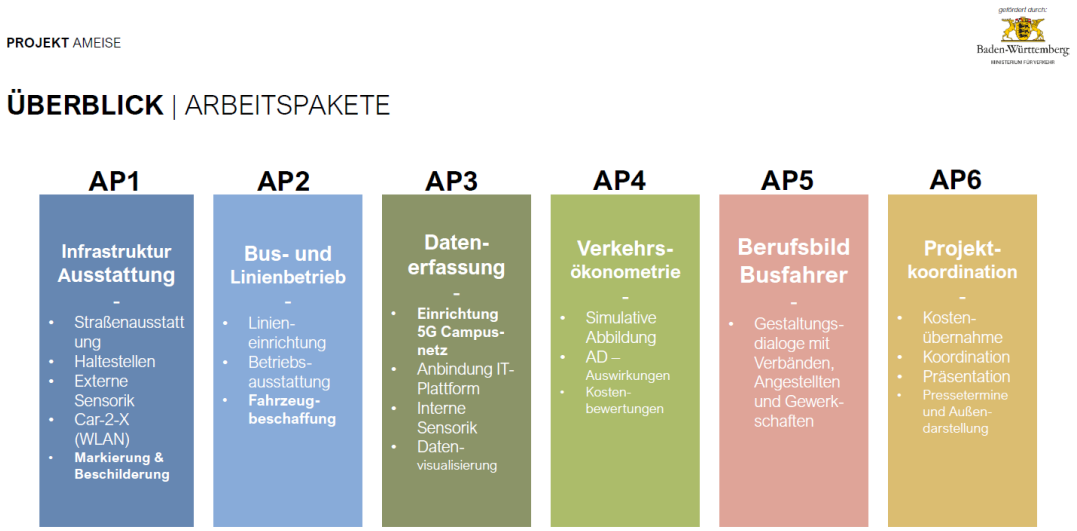


Abbildung 1.1: Konzept des Projekts AMEISE, [AMEISE 2021]

1.3 Transparente Car2I-Kommunikation als Lösung

Ziel dieser Forschungsarbeit ist die Ableitung von Anforderungen an eine robuste Car2I-Kommunikation für einen autonomen Fahrbetrieb im urbanen Raum. Im Fokus stehen dabei der Dateninhalt (aufbereitete-/ Rohdaten), Security Mechanismen sowie zeitliche Untersuchungen intrinsischer Use-Case Betrachtungen. Dabei wird der für eine WLAN-basierte Car2X-Kommunikation etablierte Standard namens ETSI ITS-G5 untersucht. In diesem Zuge wird auch kurz der WLAN-Standard IEEE 802.11p beleuchtet. Ein weiteres Ziel stellt die Analyse der Anforderungen an eine robuste und sichere Kommunikation dar. Für die Analyse sollen Anwendungsmöglichkeiten für Car2X betrachtet und daraus Anforderungen an die Latenz, Geo-Relevanz, die Bandbreite und die Datenart erhoben werden.

Eine Implementierung einer robusten Car2I-Kommunikation über den WLAN-Standard ist ebenfalls Teil dieser Arbeit. Hierfür soll auch ein geeigneter Use-Case ausgearbeitet und umgesetzt werden. Vor allem aber soll eine geeignete Hardware für die Implementierung der Car2I-Kommunikation ausgewählt werden. Abschließend werden am konkreten Beispiel eines Verkehrssensors quantitative Messungen durchgeführt. Damit können auf Basis der Ergebnisse die abgeleiteten Anforderungen diskutiert und final definiert werden.

Im Allgemeinen soll eine robuste und sichere Fahrzeugkommunikation transparent gemacht werden, um die erhobenen Anforderungen evaluieren zu können. Dabei sollen die Daten, die zwischen dem Fahrzeug und der Infrastruktur ausgetauscht werden, auf einem mobilen Endgerät dargestellt werden.

Kapitel 2

2 Car2X-Kommunikation

In den Bereich der kooperativen intelligenten Transportsysteme (C-ITS) fällt die Vernetzung kooperativer, automatisierter Fahrzeuge sowohl direkt untereinander als auch mit der Straßeninfrastruktur. Diese Interaktion ermöglicht den Straßennutzern und Verkehrsleitstellen das Teilen und Verwenden bislang nicht verfügbarer Informationen miteinander sowie die Koordination ihrer Maßnahmen. Diese Vernetzung findet dabei mittels sogenannter Car2X-Kommunikation statt. [vgl. Kommission 2016, S. 3ff]

2.1 Architektur eines Car2X-Systems

Für die Realisierung einer robusten Car2X-Kommunikation ist es notwendig, sich mit dessen Funktionsweise auseinanderzusetzen. Hierbei werden die Kernkomponenten und ihre Interaktion erläutert. Für die Übertragung von Daten innerhalb C2X stehen prinzipiell drei verschiedene Technologieansätze zur Verfügung: WLAN-V2X, Cellular-V2X und Backend-V2X. Während erster genannte Varianten speziell für die Fahrzeugkommunikation entworfen und von den Funkstandards IEEE 802.11p und 3GPP LTE abgeleitet wurden, realisiert Letztere die Kommunikation zwischen Fahrzeugen und einem Backend über konventionelle öffentliche Mobilfunkstandards (4G LTE, 3G HSDPA, 2G GSM). Im Rahmen dieser Arbeit werden lediglich die WLAN und Cellular Technologie betrachtet.

2.1.1 Kommunikationstechnologien: WLAN-C2X vs. Cellular-C2X

Die WLAN-C2X Technologie nutzt für den Datenaustausch den durch IEEE (engl. Institute of Electrical and Electronics Engineers) spezifizierten WLAN-Standard 802.11p (kurz WLANp). Hierbei handelt es sich um eine Spezifikation, die speziell für den mobilen Einsatz optimiert ist und den Fokus auf die Fahrzeugkommunikation legt. Zudem ist 802.11p neben den garantierten niedrigen Latenzzeiten auch durch ihre Fähigkeit zur infrastrukturlosen Kommunikation charakterisiert. Damit eignet sie sich besonders für die direkte Kommunikation von Fahrzeugen untereinander über Entfernungen von wenigen hundert Metern (ca. 1000 m).

Im Gegensatz dazu basiert die zellulare Technologie auf den Mobilfunk-Standards der vierten und fünften Generation (4G bzw. LTE, 5G). Cellular-V2X, auch C-V2X genannt,

ist eine technische LTE-Spezifikation, die von 3GPP (engl. 3rd Generation Partnership Project) für die Fahrzeugvernetzung definiert wird. Sie ermöglicht sowohl die direkte Vernetzung von Fahrzeugen untereinander (infrastrukturlos) als auch über das Mobilfunknetz. Die Vorteile dieser Technologie beziehen sich vor allem auf hohe Zuverlässigkeit und kurze Latenzzeiten (bei 5G z. B. 1ms). Darüber hinaus erlaubt sie eine Kommunikation über große Distanzen.

Letztendlich werden beide Technologien, aufgrund ihrer Inkompatibilität, als Konkurrenten betrachtet. Möglich kann aus heutiger Sicht aber auch eine Koexistenz beider Lösungen der Fahrzeugkommunikation sein. [vgl. Protzmann u. a. 2018, S. 27ff]

2.1.2 Kernkomponenten: OBU vs. RSU

Kernkomponente: OBU

Damit ein Fahrzeug über Car2X kommunizieren kann, wird eine sogenannte On-Board Unit (OBU) benötigt. Diese ist für die Weiterleitung von Daten an OBU‘s anderer Fahrzeuge innerhalb einer Ad-hoc-Domäne (siehe 2.1.3) zuständig. Sie besitzt grundsätzlich die Fähigkeit zur drahtlosen Kommunikation über kurze Strecken auf Basis der IEEE 802.11p Funktechnologie. Dadurch kann das Senden, Empfangen und Weiterleiten von sicherheitsrelevanten Daten innerhalb einer Ad-hoc-Domäne ermöglicht werden. Zusätzlich kann eine On-Board-Unit zur Übertragung nicht-sicherheitsrelevanter Daten auch über andere Funktechnologien (z. B. IEEE 802.11a/b/g/n) verfügen und darüber auch eine Verbindung über öffentliche, kommerzielle oder private Hotspots mit Internetknoten oder Servern herstellen.

Kernkomponente: RSU

Road-Side Unit (RSU) sind stationäre Einheiten, die sowohl entlang von Straßen (z. B. an Lichtsignalanlagen), Autobahnen (z. B. an Autobahnbrücken) als auch an speziellen Orten wie Tankstellen oder Parkplätzen montiert werden können. So wie eine OBU, verfügt die RSU über die Fähigkeit zur drahtlosen Kommunikation im Nahbereich basierend auf der IEEE 802.11p Funktechnologie. Damit sind RSU‘s in der Lage, sowohl untereinander als auch direkt mit den OBU’s der Fahrzeuge zu kommunizieren. Ersteres ermöglicht die Zusammenarbeit mit anderen RSU’s bei der Weiterleitung oder Verteilung von Sicherheitsinformationen und Letzteres, um aktuelle Verkehrsinformationen zwischen OBU und RSU auszutauschen. Zudem kann die RSU den Kommunikationsbereich einer OBU, durch Weiterleitung seiner Daten, erweitern. Dieser Vorteil ergibt sich, wenn eine RSU die Daten direkt innerhalb einer Fahrzeugflotte von einer OBU zu anderen weiterleitet. Darüber hinaus kann eine RSU auch eine infrastrukturbasierte Kommunikation ermöglichen. Dabei lässt sie sich über ein Gateway mit dem Internet verbinden und kann darüber auch eine Internetverbindung für die OBU’s bereitstellen.

2.1.3 Domänen: In-Vehicle vs. Ad-Hoc vs. Infrastructure

Domäne: In-Vehicle

Im In-Vehicle-Domain-Netzwerk befinden sich eine On-Board Unit (OBU) und eine Anwendungseinheit (engl. Application Unit, AU). Bei einer AU handelt es sich um ein Gerät, das speziell für die Ausführung einer oder mehrerer Anwendungen entwickelt wurde. Sie wird mit der OBU verbunden und nutzt dessen Kommunikationsfähigkeit. Diese Verbindung kann sowohl kabelgebunden als auch drahtlos erfolgen (z. B. über Bluetooth). Die Anwendungseinheit kann fest im Fahrzeug integriert oder aber auch ein tragbares Gerät wie z. B. ein Laptop sein. Hierbei ist die Unterscheidung zwischen AU und OBU rein logisch. Beide Einheiten können sich auch auf einem einzigen physikalischen Gerät befinden.

Domäne: Ad-Hoc

Die Ad-Hoc-Domäne, auch als Vehicular Ad Hoc Network (VANET) bezeichnet, besteht aus Fahrzeugen, die mit einer OBU ausgestattet sind und aus den Road-Side Unit (RSU). Die Kommunikation läuft zwischen den Fahrzeugen ab. Dabei bilden die On-Board-Units ein mobiles Ad-Hoc Netzwerk(MANET). Das bedeutet, sie sind gleichberechtigt und können direkt eine Verbindung untereinander aufbauen, ohne dabei über eine zentralisierte Koordinationsinstanz kommunizieren zu müssen. Falls eine drahtlose Verbindung zwischen ihnen besteht, erfolgt die Kommunikation direkt. Zusätzlich ermöglichen spezielle Routing-Protokolle eine Multi-Hop-Kommunikation. Bei dieser werden die Daten solange zwischen den OBU's weitergeleitet, bis sie das Ziel erreichen. Diese Möglichkeit der Vernetzung kann nur innerhalb kurzer Reichweiten stattfinden und ist für die Verkehrssicherheit vorgesehen. Für die Vergrößerung der Kommunikationsreichweite werden die RSU's mit einbezogen.

Domäne: Infrastructure

In diese Domäne fallen die RSU's und Hotspots, die den OBU's Zugang zur Infrastruktur auf unterschiedliche Weise ermöglichen. Die OBU's können sich aber auch über zellulare Funknetze (4G, UMTS, GPRS) den Zugang zur Infrastruktur verschaffen, falls eine Verbindung über soeben genannte Zugangspunkte nicht möglich ist. [vgl. Baldessari u. a. 2007, S. 25ff] In *Abbildung 2.1* ist die Architektur von Car2X dargestellt.

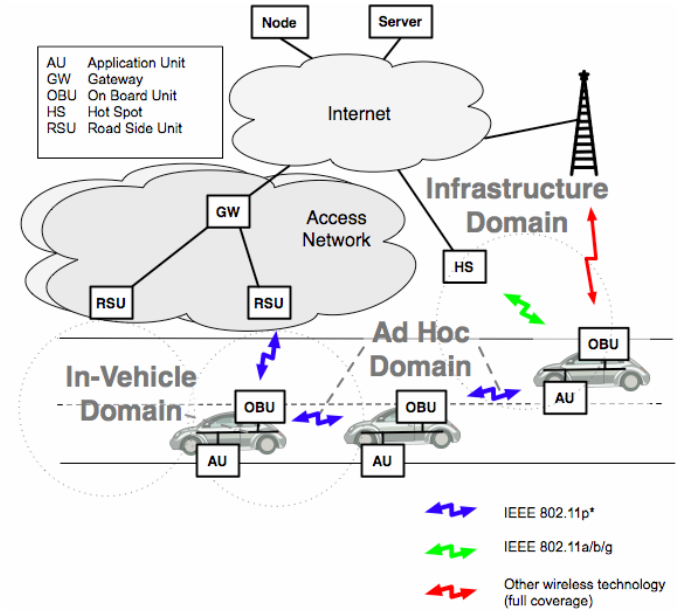


Abbildung 2.1: Architektur eines Car2X-Systems [Baldessari u. a. 2007, S. 27]

2.2 WLAN-basierte Car2X-Standardisierungen

Ein einheitlicher Kommunikationsstandard bildet die Voraussetzung für eine Car2X-Kommunikation, um eine Interoperabilität zwischen den kooperativen intelligenten Transportsystemen (C-ITS) zu erreichen. Da diese Forschungsarbeit eine Car2X-Implementierung über den WLAN-Standard vorsieht, wird dieser kurz beschrieben.

2.2.1 IEEE 802.11p

Das Institute of Electrical and Electronics Engineers ist ein weltweiter Berufsverband von Ingenieuren. Die IEEE bildet Gremien für die Standardisierung von Techniken, Hardware und Software. Eines dieser Standards ist der etablierte WLAN-Standard (802.11 Norm) für die Datenübertragung in Funknetzwerken. [vgl. Crow u. a. 1997, S. 3ff]

Für die Kommunikation zwischen Fahrzeugen untereinander und mit der Verkehrsinfrastruktur wurde ein besonderer Standard, 802.11p, entwickelt. Dieser wurde speziell für den Einsatz im mobilen Umfeld optimiert. Somit stellt er sich den Herausforderungen in der V2X-Kommunikation. Diese umfassen z. B. hohe Geschwindigkeiten, hohe Zuverlässigkeit bei niedrigen Latenzen oder die Möglichkeit zur Ad-Hoc Vernetzung. WLANp basiert auf dem 802.11a Standard und stellt eine Erweiterung dessen dar. Zum Beispiel wurde das Übertragungsverfahren OFDM (engl. Orthogonal Frequency Division Multiplexing) übernommen und entsprechend angepasst, sodass die Datenübertragung mehr Robustheit aufweist. Dafür wurde die Bandbreite des Kanals auf 10 MHz reduziert, wodurch sich auch die Datenrate auf 27 MBit/s minimiert. Der Standard operiert in dedizierten und speziell für C-ITS reservierten Frequenzbändern bei 5,9 GHz (5,855 - 5,925 GHz). Er ermöglicht

eine Sendeleistung von bis zu 33 dBm und erlaubt somit eine Reichweite von 500 bis ca. 1000 Meter. Eine weitere wichtige Eigenschaft ist die Fähigkeit der direkten Kommunikation (Peer-to-Peer), die keinen Access Point (AP) mehr voraussetzt. *Tabelle 2.1* fasst einige Eigenschaften des Standards zusammen.[vgl. KG 2021]

Tabelle 2.1: Ausgewählte Eigenschaften von IEEE 802.11p

Parameter	**IEEE 802.11p**
Frequenzbereich	5,855 - 5,925GHz
Kanalbandbreite	10 MHz
Datenrate	3 bis 27 Mbit/s (6 Mbit/s auf dem Steuerungskanal)
Modulationsverfahren	OFDM

IEEE 802.11p bildet dabei die Schlüsseltechnologie für den sogenannten amerikanischen „IEEE 1609 WAVE“-Standard (engl. Wireless Access in Vehicular Environments) und europäischen „ETSI ITS G5“-Standard, auf dem im Folgenden näher eingegangen wird.

2.2.2 ETSI ITS-G5

Das Europäisches Institut für Telekommunikationsnormen (ETSI) ist eine Organisation für die Erarbeitung von europaweiten Normen, Standards und Spezifikationen auf dem Gebiet der Telekommunikation. Das Institut arbeitet unter anderem an der Standardisierung der V2X-Kommunikation aufbauend auf dem 802.11p Standard im Bereich der kooperativen intelligenten Transportsystemen (C-ITS).

Dieser Abschnitt befasst sich mit der Beleuchtung des C-ITS Protokollstacks. Dieser besteht aus 6 verschiedene Schichten. In der untersten Ebene, der **Access** Schicht, findet die Bitübertragung statt. Darüber ist die **Networking & Transport** Schicht angesiedelt, in der die Daten in ein Protokoll gepackt und für das Versenden vorbereitet werden. Die darüber liegende Schicht **Facilities** stellt die Daten für die Protokolle (aus Networking & Transport) bereit. Dabei werden die soeben beschriebenen Schichten von der **Management** und **Security** Schicht umrahmt. In der obersten Schicht, namens **Applications**, befindet sich die eigentliche ITS-Anwendung (z. B. für die Verkehrssicherheit oder -effizienz). In *Abbildung 2.2* ist der Aufbau des C-ITS Protokollstacks dargestellt und anschließend kurz beschrieben.

Access: Diese Schicht unterstützt drahtlose Kommunikationstechnologien wie IEEE 802.11p bzw. ITS-G5, der auf 802.11p basiert und für den europäischen Raum angepasst wurde.

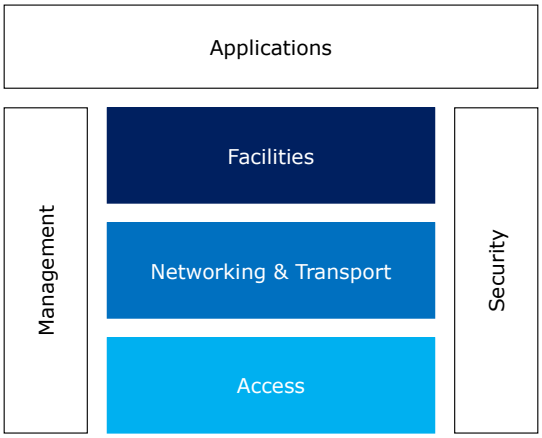


Abbildung 2.2: C-ITS Protokollstack, In Anlehnung an [ETSI 2010, S. 24]

Networking & Transport: Diese Schicht ist zuständig für den Transport der sogenannten CAM- & DENM-Nachrichten. Hierfür werden spezielle Netzwerkprotokolle verwendet, die für das Routing von Paketen in einem Ad-hoc-Netzwerk zuständig sind. Dabei werden die Nachrichten nicht direkt adressiert, sondern an eine Zielregion übermittelt.

Facilities: Sie definiert Nachrichten wie CAM und DENM, generiert Informationen für diese Nachrichten und gibt ihnen Zeitstempel sowie Positionsdaten. Zudem regelt diese Ebene auch, ob Nachrichten periodisch oder ereignisgesteuert generiert werden.

Applications: Diese Schicht spezifiziert die allgemeine Verwaltung von ITS-Anwendungen, die in Verkehrssicherheit, Verkehrseffizienz und Sonstige Anwendungen gruppiert sind.

Management: Diese Ebene kümmert sich um die Verwaltung der verschiedenen Anwendungen sowie um die Auslastung der Kanäle.

Security: Stellt sicherheitsrelevante Dienste für die Kommunikation bereit. Darunter fallen zum Beispiel die Verwaltung der Firewall oder die Erkennung von unerlaubten Zugriffen sowie deren Vermeidung. [vgl. ETSI 2010, S. 24ff]

An dieser Stelle folgt eine Übersicht der möglichen Nachrichtentypen.

CAM - Cooperative Awareness Messages: CAM-Nachrichten werden periodisch zwischen Fahrzeugen oder zwischen RSU's und der Verkehrsleitzentrale in unmittelbarer Nähe ausgetauscht. Sie geben Auskunft über den fließenden Verkehr und enthalten

dabei Statusinformationen wie Datum mit Uhrzeit, Position, Geschwindigkeit sowie zusätzlich weitere Informationen wie den Fahrzeugtyp oder die Rolle im Straßenverkehr (Krankenwagen, Polizeiwagen etc.).

DENM - Decentraliced Enviornmental Notification Messages: DENM-Nachrichten dienen zur Benachrichtigung der Verkehrsteilnehmer und werden ereignisgesteuert zwischen ITS-Stationen (RSU's) und Fahrzeugen versendet. Die Benachrichtigung erfolgt dabei nur zum betroffenen Gebiet, bei dem das Ereignis auftritt. Die Auslöser dafür können Unfälle, Falschfahrer, Warnungen (vor Baustellen, gefährlicher Straßenführung, Bremsvorgängen vorausfahrender Fahrzeuge etc.) sowie nasse und glatte Fahrbahnen sein.

MAP - MAP (topology) Extended Message: Sie enthalten Daten der Straßen-/Fahrspurtopologie (z. B. Parkplätze und Fußgängerüberwege) sowie Informationen zu zulässigen Manövern innerhalb eines Kreuzungsbereichs oder Straßensegments.

SPAT - Signal Phase and Timing: Sie dienen der Übermittlung von Echtzeitinformationen von Ampelsignalphasen und deren Timings bzw. Schaltzeiten. Zusätzlich verwendet sie Informationen aus MAP-Nachrichten, um eine Zuordnung der Lichtsignalanlagen zur Straßengeometrie zu ermöglichen. Aus diesem Grund werden MAP- und SPAT-Nachrichten zusammen übertragen.

IVI - Infrastructure to Vehicle Informationen: Dieser Nachrichtentyp liefert Informationen zu Verkehrszeichen oder Straßenarbeiten.

Kapitel 3

3 Analyse der Anforderungen an die C2X-Kommunikationstechnologien

Das folgende Kapitel beschäftigt sich mit der Analyse der grundsätzlichen Anforderungen an die zugrundeliegenden Car2X-Kommunikationstechniken (WLANp und Cellular). Zunächst soll ein grober Überblick über die aktuell verfügbaren Car2X-Anwendungen gegeben werden, um die Anforderungen entsprechend ableiten zu können. Anschließend wird ein Anwendungsfall beschrieben, der als Grundlage für die Umsetzung der robusten Car2I-Kommunikation dient. Im letzten Schritt der Anforderungsanalyse werden konkrete Anforderungen, ausgehend vom definierten Anwendungsfall, abgeleitet. Der Inhalt des folgenden Abschnitts beruht im Wesentlichen auf Protzmann u. a. 2018, (2018, S. 15-24), der kommunalen Aufgabenträgern und Entscheidern ein handhabbares Instrumentarium zur fundierten Entscheidungsfindung für die bestmögliche Ausschöpfung von V2X-Potenzialen an die Hand geben möchten.

3.1 Existierende C2X-Anwendungen

C-ITS verstehen sich als intelligente Car2X-Anwendungen, welche die Verkehrssicherheit erhöhen, die effektivere Nutzung der Infrastruktur ermöglichen und die Reduzierung von Emissionen begünstigen. Daraus ergeben sich eine Vielzahl von Anwendungen in den verschiedensten Anwendungsbereichen des Straßenverkehrs, die zusammen individuelle Anforderungen an die Kommunikationstechnologien stellen.

Solche Anwendungen sind heutzutage in modernen Fahrzeugen schon weit verbreitet, um unter Nutzung von Car2X-Kommunikation andere Verkehrsteilnehmer zu informieren und zu warnen. Gleichzeitig werden diese Car2X-Anwendungen von den Arbeiten der ETSI und des C2C-CC oder von Forschungsprojekten (z. B. PRE-DRIVE C2X) stark beeinflusst. Von dort entspringt auch die oftmals verwendete Aufteilung der Anwendungen in folgende Bereiche: **Sicherheit**, **Verkehrseffizienz** und **Komfort/Entertainment**. Zum Beispiel existieren heute bereits:

- Navigationslösungen mit Informationsaustausch zum Verkehrsaufkommen (z. B. von GoogleMaps),
- von Fahrzeughersteller eigens angebotene Services (z. B. Mercedes me) für Sicherheit, Navigation und Komfort,

- das automatische Notrufsystem eCall, das bei Verkehrsunfällen eine schnellere Meldung und Rettung ermöglicht und
- V2I-Service „Ampelinformation“ (Grüne Welle) von Audi für eine entspanntere und effizientere Autofahrt statt Stop-And-Go-Verkehr

sowie etliche weitere Anwendungslösungen. Daraus lassen sich eine Reihe von unterschiedlichen Anforderungen ableiten, von denen im Folgenden die bedeutendsten vorgestellt werden.

3.2 Ausgewählte Anforderungen der Anwendungen

Robuste Car2X-Kommunikationen sollten vor allem Sicherheits- und Effizienzanwendungen im Verkehrsumfeld ermöglichen. Vor diesem Hintergrund wird erwartet, dass die zugrundeliegenden Kommunikationstechnologien (WLANp und Cellular) mindestens die verschiedenen Anforderungen dieser Anwendungen erfüllen können. Nachfolgend werden die für diese Arbeit als wichtig erachteten Anforderungen beschrieben. Letztlich muss jedes Projekt individuell festlegen, welche Anforderungen bei der Untersuchung einer robusten V2X-Kommunikation erhoben werden.

Latenzanforderung: Unter Latenz wird die Zeit, welche für die Übermittlung eines Datenpakets von einem Gerät zu einem anderen benötigt wird, verstanden. Entsprechend bestimmt diese Anforderung die maximale Zeit der kommunizierten Daten.

Geographische Relevanz: Sie definiert die Reichweite in der die Kommunikation erfolgreich stattfinden muss.

Involvierte Endpunkte: Sie bezeichnen alle Entitäten mit dem sich ein Fahrzeug vernetzen kann. Darunter fallen, neben V2V, V2I (Vehicle-to-Infrastructure), V2N (Vehicle-to-Network) und V2P (Vehicle-to-Pedestrian). Entsprechend können die involvierten Endpunkte die Übertragungsart (z. B. Broadcast, Unicast) bestimmen.

Datenart: Die Datenart definiert die Form und Größe der zu übertragenden Daten. Sie kann von kleinen Paketen (für Kontrolldaten) bis zu großen Daten-Streams (für Video oder Sensordaten) reichen.

Datenrate: Die Datenrate gibt an (gemessen in Bit pro Sekunde), wie viel Daten in einer bestimmten Zeit übertragen werden können. [vgl. Protzmann u. a. 2018, S. 15-19]

3.3 Anwendungsklassen und deren Anforderungen

Wie dem (vgl. *Abschnitt 3.1*) zu entnehmen ist, existieren bereits eine Reihe von Car2X-Anwendungen in den Bereichen **Sicherheit**, **Verkehrseffizienz** und **Komfort/Entertainment**. Daraus lassen sich, nach den Überlegungen von Protzmann u. a. 2018, (2018,

S. 19-23), Anwendungsklassen (AK) je Bereich ableiten. Diese werden nun einschließlich ihrer Anforderungen vorgestellt.

3.3.1 Sicherheit

AK1 - Verkehrskritische Nahbereichswahrnehmung: Diese Applikationen tauschen regelmäßig Informationen von Statusdaten zwischen den Fahrzeugen aus, um die Sicherheit zu erhöhen und die Verkehrsteilnehmer zu schützen. Ziel ist es dabei Gefahrensituationen frühzeitig zu erkennen und bei relevanten Ereignissen zusätzlich zu kommunizieren (z. B. Kollisionswarnungen). Auch die Kommunikation mit Personen oder Radfahrern ist inbegriffen und schließt somit V2P mit ein. Demnach fordert der Informationsaustausch die geringsten Latenzen (Sehr hoch: bis 10 ms), bei naher geographischer Relevanz (Nah: <300 m). Bei den Daten handelt es sich hauptsächlich um Kontrolldaten, weshalb das Datenaufkommen entsprechend gering ausfällt.

AK2 - Sensordatenaustausch des lokalen Verkehrsgeschehens: Für den Austausch von Sensordaten durch Radar oder Lidar, wird ein viel größeres Datenaufkommen (Übertragung von Streams) produziert. Dabei stellt der Austausch hohe Anforderungen an die Latenz (Hoch: 10-100 ms).

Tabelle 3.1: Kommunikationsanforderungen Sicherheit

AK	Endpunkte	Latenz	Reichweite	Datenart	Datenrate
1	V2V, (V2P)	Sehr hoch (10 ms)	Nah (<300 m)	Datenpakete	-
2	V2I, V2N	Hoch (10-100 ms)	Nah (<300 m)	Breitbandige Streams	-

3.3.2 Verkehrseffizienz

AK3 - Verkehrsbegleitender Informationsaustausch im relevanten Umfeld: Sie können den Verkehrsfluss in Echtzeit schätzen und kurzzeitige Prognosen generieren mit dem Hintergrund Ressourcen- und Emissionsreduktion zu erzielen. Dabei werden hauptsächlich Kontrolldaten übertragen und zur Erhöhung der Informationsreichweite RSU's verwendet (V2I). Diese Art der Anwendung benötigt größere Reichweiten (Weit: > 1km) und toleriert gewisse Latenzen (Moderat: bis 1s).

AK4 - Teleoperiertes Fahren: Hier werden oftmals Status- und Sensordaten aber auch Kamera-Streams zwischen Fahrzeug und entfernten Datenzentren im Backend ausgetauscht. Diese Art von Anwendung hat die höchsten Kommunikationsanforderungen hinsichtlich der Latenz (Sehr hoch: bis 10 ms) und Datenart (Datenpakete und Streams). Aufgrund der Backend-Verbindung, ist die geographische Relevanz nicht definiert.

AK5 - Internetbasierte Streaming- und Cloud-Dienste: Grundsätzlich erfordern diese Anwendungen eine Internetverbindung und setzen moderate bis hohe Latenzanforderungen (Moderat 1s bis Hoch 50 ms), bei hohen Datenaufkommen (Streams) voraus. Zusätzlich können hier auch Anwendungen existieren, die eine Mischform zwischen Effizienz- und Komfort darstellen.

AK6 - Schmalbandige Internetdienste: Das können Dienste zur Fernsteuerung des Fahrzeugs (via Backend) sein, welche sich durch die geringsten Kommunikationsanforderungen auszeichnen. Das betrifft sowohl Datenaufkommen (Kontrollpakete) als auch Latenzen (Gering: >1s). Je nach Realisierung können die schmalbandigen Internetdienste auch Effizienzanwendungen darstellen.

Tabelle 3.2: Kommunikationsanforderungen Verkehrseffizienz

AK	Endpunkte	Latenz	Reichweite	Datenart	Datenrate
3	V2V, V2I	Moderat (1 s)	Weit (>1 km)	Datenpakete	-
4	V2N	Sehr hoch (10 ms)	-	Breitbandige Streams, Datenpakete	-
5	V2N	Moderat (1 s) bis Hoch (50 ms)	-	Streams	-
6	V2N	Gering (>1 s)	-	Datenpakete	-

3.3.3 Komfort/Entertainment

AK5 - Internetbasierte Streaming- und Cloud-Dienste: Diese Dienste können breitbandige Multimedia-Dienste wie Web- oder Videokonferenzen oder Spiele repräsentieren. Die Anforderungen wurden bereits dargestellt.

AK6 - Schmalbandige Internetdienste: Auch hier können schmalbandige Internetdienste vertreten sein, dessen Anforderungen bereits besprochen wurden.

AK7 - Umfangreiche Downloads, Updates und Upgrades: Hier werden größere Datenmengen wie Fahrzeugsoftware- und Sicherheitsupdates oder Kartenaktualisierungen übertragen. Der Datenaustausch erfolgt hier auch über eine Backend-Infrastruktur. Dieser verursacht ein hohes Datenaufkommen (Streams), um große Software-Pakete zu übertragen (Downlink). Hingegen werden im Uplink kleinere Kontrolldaten übermittelt (Datenpakete). Diese Anwendungen besitzen geringe bis moderate Latenzanforderungen (Gering > 1s bis Moderat 1s).

3.4 Gewählter Use Case und dessen Anforderungen

Eine Car2X-Kommunikation halt vielfältige Anwendungsmöglichkeiten. Zusammen mit den Industriepartnern wurden eine Reihe von Anwendungsfällen betrachtet, wobei alle Beteiligten sich letztlich auf ein Anwendungsfall im Car2I-Umfeld einigen konnten.

Tabelle 3.3: Kommunikationsanforderungen Komfort/Entertainment

AK	Endpunkte	Latenz	Reichweite	Datenart	Datenrate
5	V2N	Moderat (1 s) bis Hoch (50 ms)	-	Streams	-
6	V2N	Gering (>1 s)	-	Datenpakete	-
7	V2N	Gering (> 1 s) bis Moderat (1 s)	-	Datenpakete, Breitbandige Bulk Daten,	-

3.4.1 Dokumentation des gewählten Use Cases

Ein Anwendungsfall, auch als Use Case (UC) bekannt, dient der Dokumentation der wichtigsten Systemfunktionalitäten aus Sicht der Anwender und kann somit einen Überblick über das Gesamtsystem verschaffen. Zusammenfassend beschreibt ein Use Case das sichtbare Verhalten eines Systems und repräsentiert dabei die Ziele der Nutzer. [vgl. Kleuker 2013, S. 63-64] Die visuelle Darstellung des definierten Use Cases namens **„Gefahrenwarnung (Vorwarnanhänger/Anzeigetafel)“** ist in *Abbildung 3.1* dargestellt.

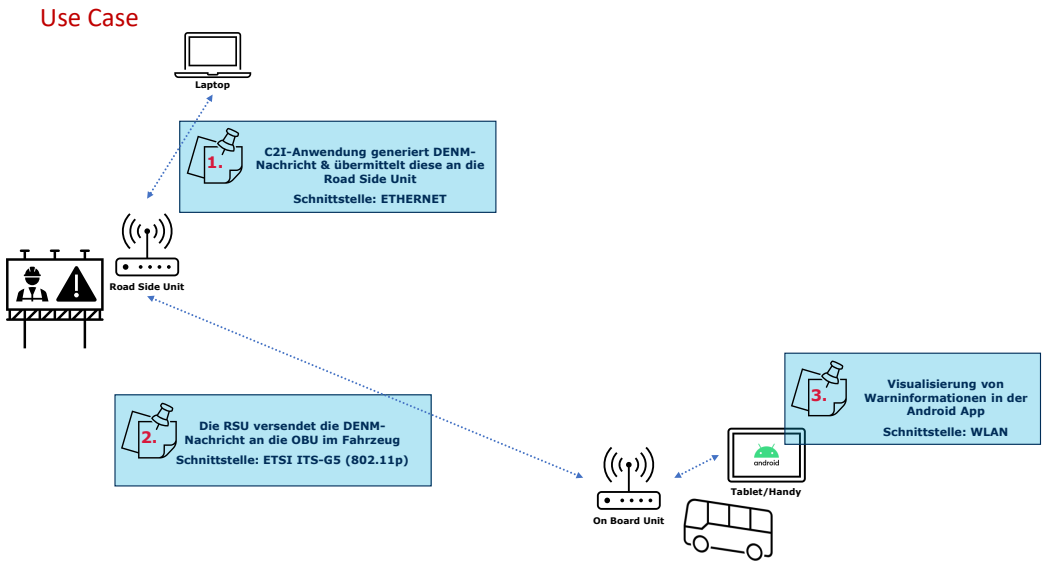


Abbildung 3.1: Visualisierung des UC's „Gefahrenwarnung (Vorwarnanhänger/Anzeigetafel)“

Nach der Definition des Anwendungsfalls **„Gefahrenwarnung (Vorwarnanhänger/-Anzeigetafel)“**, gilt es diesen zu dokumentieren. An dieser Stelle folgt eine detaillierte Beschreibung des UC's in einzelnen Schritte.

Tabelle 3.4: Dokumentation des Use Cases „Gefahrenwarnung (Vorwarnanhänger/Anzeigetafel)“

Name des Use Cases	**Gefahrenwarnung (Vorwarnanhänger/Anzeigetafel)**
Kurzbeschreibung	Informationen über Gefahrenstelle werden via Ethernet an die RSU übermittelt und über das ETSI ITS-G5 Protokoll an die OBU ins Fahrzeug versendet.
Akteure	Laptop, Anzeigetafel, RSU, OBU, Fahrzeug, HMI-Lösung
Normaler Ablauf	1. Es liegen Infos (z. B. Ort) der Gefahrenstelle vor, die Fahrzeuge über ihre Umgebung erhalten sollen. 2. Ein Laptop übermittelt diese Informationen via Ethernet an die Anzeigetafel. 3. Eine C2I-Anwendung (auf dem Laptop) generiert die entsprechende DENM-Nachricht und übermittelt diese via Ethernet an die RSU. 4. Die RSU versendet die DENM-Nachricht über das ETSI ITS-G5 Protokoll an die OBU im Fahrzeug. 5. Das Fahrzeug erhält die Information. 6. Diese Info kann auf dem HMI betrachtet werden.
Alternativer Ablauf	
Annahme	
Auslöser	Eine Gefahrenstelle, welche die Verkehrssicherheit/-effizienz, betrifft, liegt vor.
Vorbedingungen	Daten müssen vom Laptop an die Anzeigetafel übermittelt werden.
Nachbedingungen	

3.4.2 Ableitung der Anforderungen aus dem definierten Use Case

Im letzten Schritt der Anforderungsanalyse sollen konkrete Anforderungen, ausgehend vom soeben definierten Anwendungsfall, abgeleitet werden. Diese beziehen sich auf das eigentliche Systemverhalten. Dabei erfolgt die Definition der Anforderungen technologieneutral auf Basis der Erkenntnisse aus *Abschnitt 3.3*.

Prinzipiell dienen Baustellenwarnungen der Erhöhung der Verkehrssicherheit. Folglich lässt sich der definierte Use Case in die Anwendungsklasse **„AK1 - Verkehrskritische Nahbereichswahrnehmung“** einordnen und stellt somit sehr hohe Latenzanforderungen bei naher geographischer Relevanz. Andererseits bleibt das Datenaufkommen nahezu gering, da hauptsächlich Kontrolldaten übertragen werden.

Die nachstehende Auflistung umfasst die für diese Arbeit als wichtig erkannten Anforderungen des Use Cases „Gefahrenwarnung (Vorwarnanhänger/Anzeigetafel)“ an die Kommunikationstechnologien, wobei durchaus weitere wichtige Anforderungsmetriken berücksichtigt werden können. Letztendlich muss für jeden Anwendungsfall individuell festgelegt werden, welche Metriken Priorität genießen.

Tabelle 3.5: Kommunikationsanforderungen des definierten Use Cases

Endpunkte	Latenz	Reichweite	Datenart	Datenrate
V2I	Sehr hoch (10 ms)	Nah (<300 m)	Datenpakete	-

Kapitel 4

4 Implementierung der Car2I-Kommunikation

Dieses Kapitel beschreibt den Aufbau der Car2I-Kommunikation mit den ausgewählten Hard- und Softwarekomponenten. Ein Ziel war zu validieren, dass die Car2I-Kommunikation wie vorgesehen umgesetzt und damit der ausgewählte Use Case „Gefahrenwarnung (Vorwarnanhänger/Anzeigetafel)“ abgebildet werden kann. Insbesondere sollte geprüft werden, ob die für den ausgewählten Use Case wichtigen DENM-Nachrichten wie erwartet generiert werden und ob die RSU, das ETSI ITS-G5 Protokoll und die OBU problemlos zusammenspielen. Des Weiteren sollte die Implementierung helfen, praktische Probleme zu erkennen und es ermöglichen, ausgewählte Konfigurationen auszuprobieren sowie die übermittelten Nachrichtentypen und deren Inhalt zu analysieren.

Die Car2I-Implementierung wurde mit zwei On-Board Units der australischen Firma Cohda Wireless, die mit ihrer OBU Komplett-Lösung für das Fahrzeug einen V2X-Stack zur Verfügung stellt, den IEEE 802.11p Standard (IEEE 1609 WAVE und ETSI ITS G5) unterstützt und in der Industrie weit verbreitet ist, umgesetzt. Eine vollständige Neuimplementierung eines V2X-Stacks war im Rahmen dieser Arbeit weder möglich noch sinnvoll. Zum Testen wurde neben den beiden Cohda OBUs ein Samsung Galaxy S7 als HMI-Gerät verwendet, um audio-visuelle Warnungen und Alarme ausgeben zu können. Auf diesem ist Android als Betriebssystem und die von Cohda bereitgestellte App „CohdaASDDVI-6.2“ installiert.

4.1 Cohda Wireless MK5 OBU

Cohda Wireless

Cohda Wireless ist ein Gerätehersteller im Bereich der kooperativen intelligenten Verkehrssysteme. Das Unternehmen stellt Hardware-Produkte für die C2X-Kommunikation her. Eins dieser Produkte ist die MK5 OBU, auf der eine Beispiel-Anwendung namens „exampleETSI“ implementiert ist, die als Grundlage zum Senden und Empfangen von ETSI-Nachrichten wie CAM, DENM, MAP/SPAT und IVI verwendet werden kann. Sie kann sowohl als OBU als auch als RSU konfiguriert werden, weshalb die Verwendung zweier OBU MK5s zum einen als OBU und zum anderen als RSU möglich ist und im Rahmen dieser Arbeit sinnvoll war. Dies liegt zum einen an den zum Zeitpunkt der Erstellung dieser Arbeit vorhandenen Lieferengpässen und zum anderen daran, dass OBUs dieses Typs der

Fakultät Informationstechnik bereits zur Verfügung standen. Alternativ ließe sich die C2I-Kommunikation auf Basis einer MK5 OBU und MK5 RSU als komplette Outdoor-Lösung realisieren [vgl. Wireless 2021]. Aus Hardware-Sicht spielt es aber keine Rolle, ob es sich bei der RSU-Hardware um eine MK5 RSU oder MK5 OBU handelt. Das Verhalten der exampleETSI-Anwendung wird durch auf der Hardware hinterlegte Konfigurationsdateien definiert. Die exampleETSI-Anwendung wird als Binärdatei bereitgestellt. Um eigene Anwendungen schreiben oder die exampleETSI-Anwendung ändern zu können, muss das Cohda SDK gekauft werden. Dieses enthält den Quellcode der Beispiel-Anwendung.

MK5 OBU

Die MK5 OBU besitzt zwei 802.11p Antennenanschlüsse und einen Anschluss für eine GNSS-Antenne für die Positionsbestimmung. *Abbildung 4.1* zeigt eine MK5 OBU mit den drei Antennenanschlüssen.

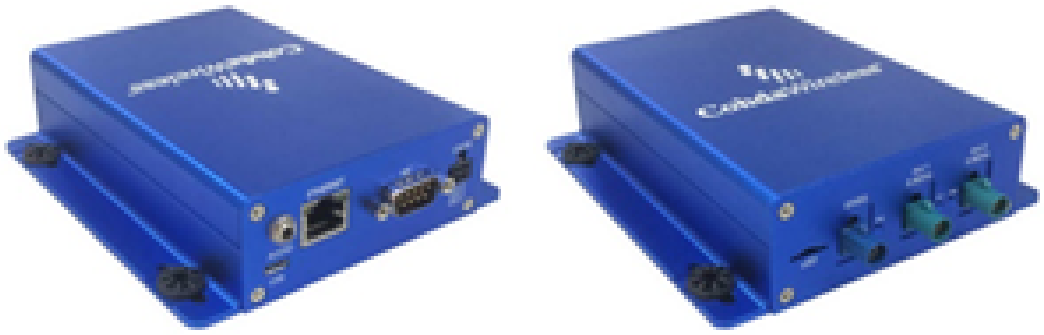


Abbildung 4.1: Cohda MK5 OBU

Zudem besitzt die MK5 OBU einen USB 2.0 Port, einen Ethernet-Anschluss, CAN-Anschlüsse und einen microSD-Karteneinschub. Dort lässt sich eine microSD-Karte einlegen, auf welcher die von der exampleETSI-Anwendung generierten Log-Dateien gespeichert werden können. Diese können z. B. nach einer Testfahrt mit dem von Cohda bereitgestellten Tool Wireshark (mit Cohda Plugin) betrachtet und ausgewertet werden. Für die Spannungsversorgung ist ein 12V Gleichstromanschluss im Gehäuse integriert, wodurch die MK5 sowohl mit +12V als auch mit +24V betrieben werden kann. Der Ethernet- sowie der USB-Anschluss eignen sich für die Kommunikation mit einem Laptop. Darüber lassen sich eine SSH-Verbindung zur MK5 OBU aufbauen und unter anderem Konfigurationen für die exampleETSI-Anwendung vornehmen sowie die exampleETSI-Anwendung starten. Wird die exampleETSI-Anwendung als OBU konfiguriert, so übertragt die Beispiel-Anwendung CAMs. CAMs enthalten Position, Geschwindigkeit, Richtung usw. des Fahrzeugs. Die Daten stammen von GNSS. Als RSU konfiguriert, überträgt die Beispiel-Anwendung DENMs, z. B. Road Works oder Unwetterwarnungen, MAP/SPAT, die von Ampeln gesendet werden, oder IVI, z. B. Geschwindigkeitsbegrenzungen, Richtungen usw. Diese Daten stammen aus den Konfigurationsdateien.

4.2 Versuchsaufbau

Der Versuchsaufbau besteht aus zwei Teilen. Die eine Seite bildet das Fahrzeug, die andere Seite stellt die Infrastrukturkomponente bzw. den/die Vorwarnanhänger/Anzeigetafel dar. Im Rahmen dieser Arbeit steht zwar ein echtes Fahrzeug zur Verfügung, aber kein/keine Vorwarnanhänger/Anzeigetafel. Aus diesem Grund wird eine Cohda MK5 OBU, konfiguriert als Road-Side Unit, allein als „Vorwarnanhänger/Anzeigetafel“-Seite betrachtet. Die andere Cohda MK5 OBU (als On-Board Unit konfiguriert) hingegen wird zusammen mit dem Samsung Galaxy S7 als HMI-Gerät als „Fahrzeug“-Seite betrachtet.

Nach Anschluss aller notwendigen Komponenten wird auf beiden Seiten ein Laptop über den vorhandenen Ethernet-Anschluss angeschlossen. Über diesen wird jeweils eine SSH-Verbindung zur MK5 OBU aufgebaut. Darüber muss auf beiden OBUs die exampleETSI-Anwendung installiert werden, damit ETSI-Nachrichten bei erfolgreicher Installation zwischen RSU und OBU ausgetauscht werden können. Wenn die exampleETSI-App ausgeführt wird, verwendet sie bestimmte Konfigurationsdateien. Diese befinden sich im Ordner exampleETSI. Je nachdem, ob die Anwendung als OBU oder RSU ausgeführt wird, werden folgende Konfigurationsdateien verwendet (* steht für RSU oder OBU):

- *.cfg: enthält die Konfigurationsparameter für die Anwendung. Dort können verschiedene Arten von zu sendenden/empfangenden Nachrichten eingestellt werden
- *.conf: wird verwendet, um die Standard-Stack-Parameter zu überschreiben

Zum Starten der exampleETSI-Anwendung muss in das Verzeichnis gewechselt werden, indem die exampleETSI-Anwendung gespeichert ist. Über den Befehl *„cd /mnt/rw/exampleETSI“* wird in das *exampleETSI*-Verzeichnis gewechselt. Der Befehl *„./rc.exampleETSI start obu“* startet die Beispiel-Anwendung. Damit verhält sich die MK5 OBU als On-Board Unit, überträgt CAM- und empfängt auch CAM-, DENM-, MAP/SPAT- und IVI-Nachrichten. Mit dem Befehl *„./rc.exampleETSI start rsu“* simuliert die MK5 OBU eine Road-Side Unit und überträgt und empfängt DENM-, MAP/SPAT- und IVI-Nachrichten. Diese Nachrichten werden in der „rsu.cfg“-Datei konfiguriert. Diese Textdatei kann mit jedem Editor z. B. vi, der auf der MK5 vorinstalliert ist, bearbeitet werden. Über den Befehl *„./rc.exampleETSI stop“* wird die Beispiel-Anwendung beendet. Auf beiden Seiten werden bei Start der exampleETSI-Anwendung in *„/mnt/rw/log/current“* ein Log-Verzeichnis erstellt. Dort befinden pcap-Dateien, die die Sende- und Empfangspakete sowie die GPS-Informationspakete enthalten. Weiterhin befindet sich dort eine Datei Namens „*.conf“, die die verwendete Konfiguration enthält sowie eine Log-Datei namens ‚stderr‘. Um die pcap-Dateien (in */mnt/src/log/*) ansehen zu können, müssen sie zunächst auf den Laptop hochgeladen und die Anwendung Wireshark gestartet werden. Dabei muss es sich um Wireshark mit Cohda Plugin handeln. Das „normale“ Wireshark enthält die Dissektoren für das ETSI-Protokoll nicht und kann diese demnach nicht dekodieren.

Zu diesem Zeitpunkt war es nötig im Rahmen jeder Testfahrt einen Laptop über Ethernet-Anschluss an die „Vorwarnanhänger/Anzeigetafel“- und „Fahrzeug“-Seite anzuschließen und über SSH manuell jeweils die exampleETSI-Anwendung zu starten. Um dem zu entgehen, wurden beide Cohda MK5s so konfiguriert, dass diese die exampleETSI-Anwendung nach dem Hochfahren automatisch starten. Damit ist es möglich, die Testfahrten ohne Laptop durchzuführen, sobald die MK5 OBUs mit Strom versorgt werden. Zudem wurden auf „Fahrzeug“-Seite (auf MK5 OBU und dem Samsung Galaxy S7) entsprechende Einstellungen vorgenommen, um die OBU mit dem Samsung Galaxy S7 zu verbinden und die Cohda-App für audio-visuelle Warnungen und Alarme während der Testfahrt nutzen zu können. Alle Konfigurationen sind im Anhang zu finden.

Fahrzeug

Zur „Fahrzeug“-Seite gehört eine MK5 OBU, 802.11p Antennen, eine GNSS-Antenne und eine KFZ-Spannungsversorgung. Zu Debug-Zwecken wird ein Laptop über Ethernet verwendet. Um die empfangenen DENM-Nachrichten zu veranschaulichen wird ein Samsung Galaxy S7 mit Android Betriebssystem und Cohda-App verwendet. In *Abbildung 4.2* sind die Komponenten für den Aufbau des Fahrzeugs schematisch und in *Tabelle 4.1* tabellarisch dargestellt.

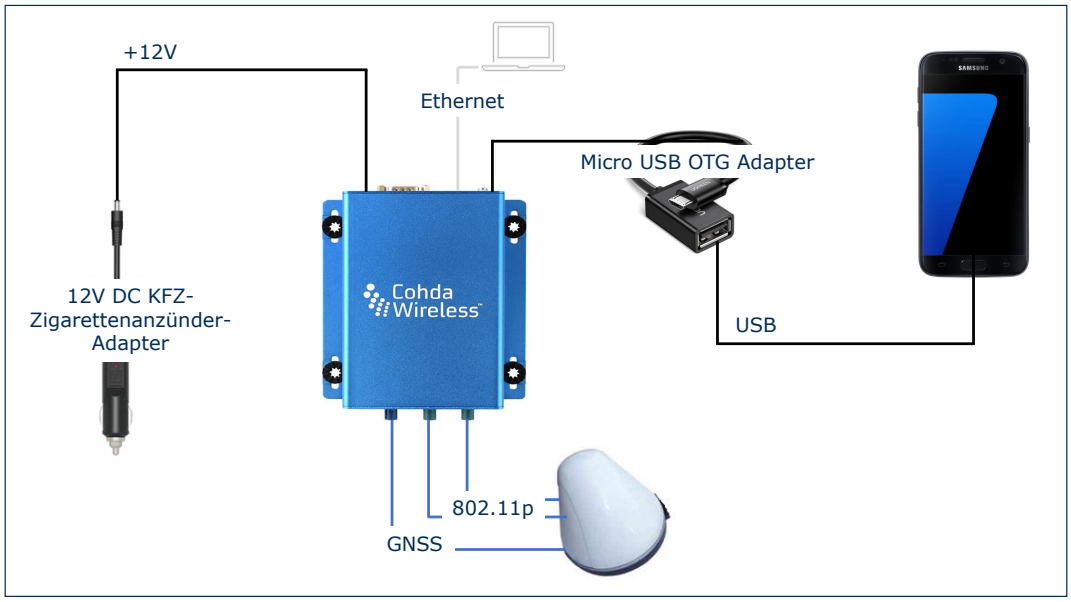


Abbildung 4.2: Aufbau Komponenten Fahrzeug

Tabelle 4.1: Komponentenliste Fahrzeug

Komponente
Cohda Wireless MK5 OBU (als OBU konfiguriert)
GNSS/802.11p Antennen
KFZ-Netzteil 12V DC für die Testfahrten
Tischnetzteil 12V DC zum Inbetriebnehmen am Arbeitsplatz
HMI-Gerät zur Visualisierung
Laptop zur Inbetriebnahme/Debug-Zwecken
Ethernet-Kabel
Testfahrzeug
Micro-USB OTG Adapter
Micro-USB Kabel

Vorwarnanhänger/Anzeigetafel

Die „Vorwarnanhänger/Anzeigetafel“ besteht wie das Fahrzeug aus einer MK5 OBU, einer GNSS Antenne, 802.11p Antennen und einer Spannungsversorgung. Auch hier wird zu Debug-Zwecken ein Laptop über Ethernet verwendet. Additiv wird nach ausgewählten Use Case (vgl. 3.4) künftig auch ein/eine Vorwarnanhänger/Anzeigetafel verwendet, die an die MK5 OBU angeschlossen ist. In *Abbildung 4.3* sind die Komponenten für den Aufbau der Vorwarnanhänger/Anzeigetafel-Seite schematisch und in *Tabelle 4.2* tabellarisch dargestellt.

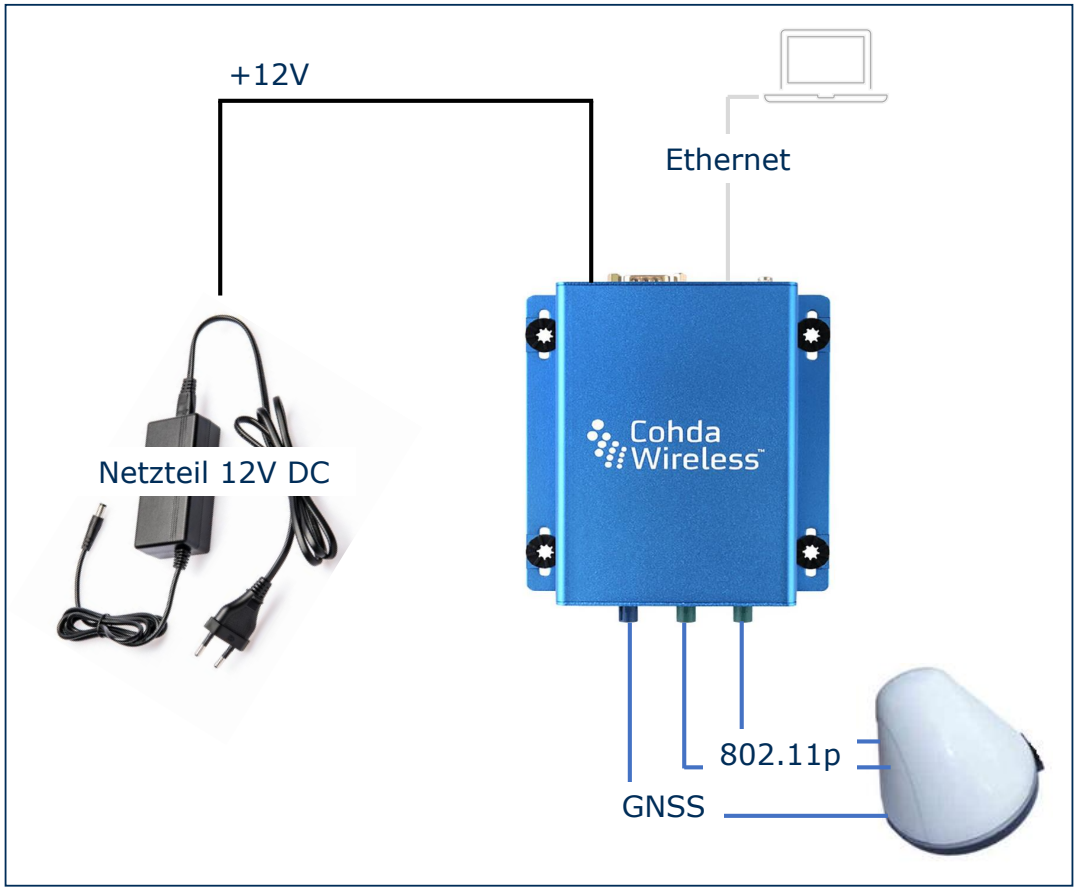


Abbildung 4.3: Aufbau Komponenten Vorwarnanhänger/Anzeigetafel

Tabelle 4.2: Komponentenliste Vorwarnanhänger/Anzeigetafel

Komponente
Cohda Wireless MK5 OBU (als RSU konfiguriert)
GNSS/802.11p Antennen
Tischnetzteil 12V DC zum Inbetriebnehmen am Arbeitsplatz
Laptop zur Inbetriebnahme/Debug-Zwecken
Ethernet-Kabel
Vorwarnanhänger/Anzeigetafel

Gesamtsystem

In *Abbildung 4.4* ist der Aufbau des Gesamtsystems aufgezeigt. Dieses besteht aus dem Fahrzeug und dem/der Vorwarnanhänger/Anzeigetafel. Beide kommunizieren über den ETSI ITS-G5 Standard, der auf IEEE 802.11p basiert, miteinander.

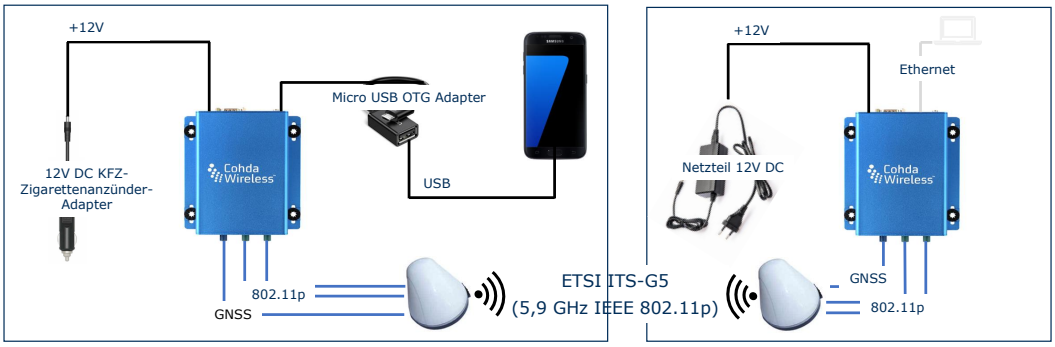


Abbildung 4.4: Aufbau Gesamtsystem

Im Rahmen der Testfahrten konnte in keinem Fall nachgewiesen werden, dass die beiden MK5 OBUs Nachrichten miteinander austauschen. Zwar ließen sie sich schnell und einfach in Betrieb nehmen, aber sie erwiesen sich als Hardware für eine Car2I-Kommunikation über ETSI ITS-G5 als ungeeignet. Das Hauptproblem bestand darin, dass die mit dem Fahrzeug und der MK5 OBU (als OBU konfiguriert) gefahrene Route sich nicht auf dem Weg der durch die MK5 OBU (als RSU konfiguriert) versendeten DENM-Nachrichten befindet. *Abbildung 4.5* verdeutlicht dieses Problem. Weitere Probleme und Lösungsansätze, die leider zu keinem positiven Ergebnis führten, sind im Anhang nachzulesen.

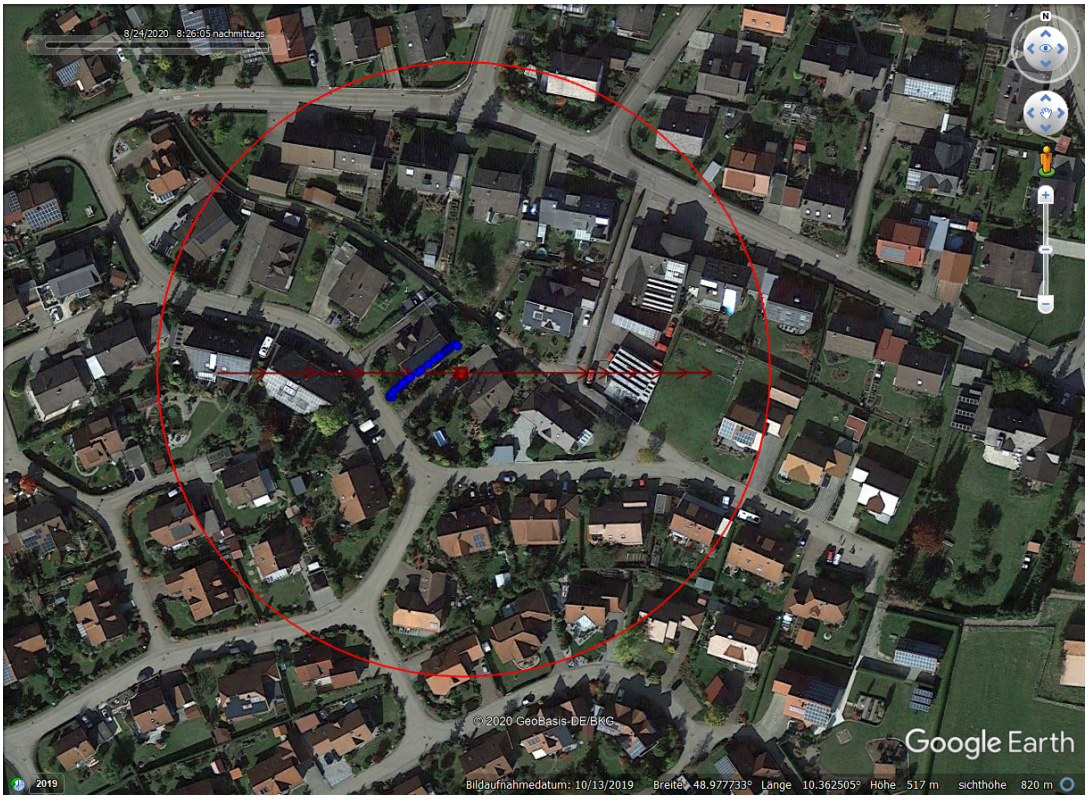


Abbildung 4.5: Gefahrene Route (OBU) und Ausbreitung DENM-Nachrichten (RSU)

Die blauen Punkte: Es wird für jede gesendete CAM-Nachricht (kommt von der OBU)

ein blauer Punkt erzeugt. Damit zeigen die blauen Punkte quasi den Fahrtweg des Fahrzeugs mit angeschlossener OBU an. Dass sich die blauen Punkte auf dem Hausdach befinden, zeigt, dass das GNSS nicht sehr genau ist. Der eigentliche Fahrtweg war selbstverständlich rechts davon auf dem Weg. Das Problem scheint die Position zwischen den beiden Häusern zu sein. Eine andere Route löste das Problem aber auch nicht, da sich auf jeder Route die blauen Punkte von der roten Linie unterschieden.

Rote Kasten in der Mitte bildet die Position der RSU, also die Stelle an der die MK5 OBU (als RSU konfiguriert) physisch aufgebaut wurde. Diese Position wurde auch als Event-Position in den Konfigurationsdateien auf der MK5 OBU manuell erfasst.

Die rote Linie: Die RSU versendet periodisch eine DENM-Nachricht, die eigentlich von der OBU empfangen wird. Der horizontale Weg links von der RSU/Event-Position stellt den Annäherungsweg an das Event dar. Der horizontale Weg rechts von der RSU hingegen stellt die Ausdehnung des Events dar. Auf der gesamten Linie werden DENM-Nachrichten versendet. Die beiden beschriebenen Wege lassen sich in den Konfigurationsdateien manuell mit festen Koordinaten erfassen, um der RSU mitzuteilen, auf welcher Route sie ihre Nachrichten versenden soll.

Wie sich zeigt, befindet sich der Fahrtweg (die blauen Punkte) nicht auf dem Weg der DENM-Nachrichten (die rote Linie), die von der RSU versendet werden. Selbst das manuelle Erfassen von Koordinaten in den Konfigurationsdateien der RSU, um den Sendeweg der DENM-Nachrichten (die rote Linie) zu bestimmen, konnte dieses Problem nicht lösen. Letztlich konnten somit die OBU und RSU keine ETSI-Nachrichten miteinander austauschen, sondern nur für sich ihre Nachrichtentypen versenden. Zumindest konnte verifiziert werden, dass die exampleETSI-Anwendungen je nach Konfiguration die entsprechenden Nachrichtentypen erzeugen und laut pcap-Dateien auch über die Funkgeräte senden. Das heißt, als OBU konfiguriert überträgt die MK5 OBU CAM- und empfängt auch CAM-, DENM-, MAP/SPAT- und IVI-Nachrichten. Als RSU konfiguriert überträgt und empfängt die MK5 OBU DENM-, MAP/SPAT- und IVI-Nachrichten.

Kapitel 5

5 Abschließende Betrachtung aller Ergebnisse und Ausblick

In diesem Kapitel werden die grundlegenden Erkenntnisse aufgegriffen und zusammengefasst. Ferner sollen Themengebiete für weitere Untersuchungen aufgezeigt werden.

Abschließende Betrachtung aller Ergebnisse
Das Ziel dieser Forschungsarbeit war die Analyse der Anforderungen an eine robuste und gleichzeitig sichere Car2I-Kommunikation, um die zugrundeliegende Kommunikationstechnologie IEEE 802.11p auf die Erfüllung der Anforderungen hin untersuchen zu können. Dazu erfolgte eine Analyse des Car2X-Kommunikationssystems und bereits existierender Car2X-Lösungen, um darauf aufbauend Anforderungen an die zugrundeliegende Kommunikation stellen zu können.

Zuerst wurde die Architektur eines Car2X-Kommunikationssystem, einschließlich aller Kommunikationstechnologien, Kernkomponenten und Domänen, untersucht. Diese Untersuchung hat ergeben, dass grundsätzlich drei Technologien für die Umsetzung einer Car2X-Kommuniaktion existieren: WLAN-V2x, Cellular-V2X und Backend-V2X. Für die Implementierung kamen nur WLAN-V2X und Cellular-V2X in Frage. Im Rahmen dieser Arbeit hat die Realisierung der Car2X-Kommunikation auf Basis des WLAN-Standards stattgefunden. Daraus ergab sich die Betrachtung des europäischen ITS-G5 Standards für den funkbasierten Datenaustausch zwischen Fahrzeugen und anderen Verkehrsteilnehmer. Das ergab, dass der Austausch von Informationen im Verkehrsbereich auf Basis standardisierter Nachrichten (CAM, DENM, IVI, MAP/SPATEM) erfolgt, um eine europaweite Interoperabilität zu gewährleisten.

Vor der eigentlichen Car2I-Implementierung wurde eine Analyse der Anforderungen bereits existierender Car2X-Anwendungen an die Kommunikation durchgeführt. Hierfür wurden vorhandene Anwendungen für Car2X betrachtet und festgestellt, dass sie sich in drei häufig verwendete Bereiche einteilen lassen: Verkehrssicherheit, Verkehrseffizienz und Komfort/Entertainment. Aus der Vielzahl der heute verfügbaren Car2X-Anwendungslösungen lassen sich eine Reihe von unterschiedlichen Anforderungen ableiten. Daraus wurden, auf Basis der Überlegungen von Protzmann u. a. 2018, (2018, S. 19-23), Anforderungsmetriken definiert, die für diese Arbeit als wichtig erachtet wurden. Diese umfassen dabei die involvierten Endpunkte (V2V, V2I, V2N, V2P), die Kommunikationslatenz, die Reichweite der kommunizierten Daten, die Datenart (von Datenpakete bis Streams) und die

Datenrate, welche die Menge der übertragenen Daten in einer bestimmten Zeit (Bit pro Sekunde) angibt. Anschließend wurden die Anforderungen je Bereich (Sicherheit, Effizienz und Komfort) mit Hilfe der durch Protzmann u. a. 2018 definierten Anwendungsklassen festgehalten. Abschließend wurde der ausgewählte Anwendungsfall beschreiben. Damit war es möglich, grundsätzliche Anforderungen (vgl. *Tabelle 5.1*) an die Kommunikation zu stellen.

Tabelle 5.1: Kommunikationsanforderungen des gewählten Use Cases

Endpunkte	**Latenz**	**Reichweite**	**Datenart**	**Datenrate**
V2I	Sehr hoch (10 ms)	Nah (<300 m)	Datenpakete	-

Die Realisierung des Anwendungsfalls erfolgte mit einer Cohda Wireless MK5 als OBU und einer weiteren Cohda Wireless MK5 als RSU. Diese bilden keine gute technische Basis für die Umsetzung des Use Cases. Zwar ließen sie sich schnell und einfach in Betrieb nehmen, aber es konnte in keinem Fall nachgewiesen werden, dass RSU und OBU Nachrichten miteinander austauschen. Aus diesem Grund erwies sich diese Hardware als ungeeignet für eine Car2X-Kommunikation über ETSI ITS-G5.

Insgesamt konnte nicht verifiziert werden, ob die angestrebte Car2I-Kommunikation alle Anforderungen, die spezifiziert wurden, erfüllt.

Ausblick

Das Ziel, eine robuste und sichere Car2I-Kommunikation auf die Erfüllung definierter Anforderungen hin zu bewerten, wurde im Rahmen dieser Forschungsarbeit nicht erreicht. Deshalb bietet diese Arbeit Möglichkeiten für weitere Forschungen. Das betrifft in erster Linie die erfolgreiche Implementierung einer Car2I-Kommunikation mit einer geeigneten Hardware.

Für die Umsetzung des Anwendungsfalls „Gefahrenwarnung (Vorwarnanhänger/Anzeigetafel)“ wurde die Hardware von Cohda Wireless verwendet. Diese weist Defizite in Hinblick auf den Nachrichtenaustausch, die vorhandene Dokumentation und den Support auf. Daher gilt es sich zu überlegen, andere Hersteller in Betracht zu ziehen, um einen effizienten Car2I-Kommunikationsaufbau zu ermöglichen.

Ferner sollte das System um eine Vielzahl anderer Use Cases erweitert werden, da die Vernetzung des Linienbusses in Zukunft auch mit anderen Verkehrsteilnehmern erfolgen muss. Dies beinhaltet, neben der Infrastruktur, andere Linienbusse oder Fahrzeuge (V2V), Radfahrer oder Fußgänger (V2P) und möglicherweise entfernte Server (V2N).

Zuletzt besteht die Notwendigkeit das System auf Sicherheitsmechanismen und -aspekte zu untersuchen, da die Vernetzung Angriffsflächen für Manipulationen und Hacker bieten.

Vor allem ist hierbei die Fahrsicherheit bedroht. Deshalb ist es ohne Zweifel sinnvoll, für weitere Forschungen den Fokus auf die Sicherheit zu legen, indem beispielsweise die Kommunikation verschlüsselt wird. Alternativ wäre es auch denkbar, Möglichkeiten zur bidirektionalen Authentifizierung über einen gemeinsamen Zugangspunkt in Betracht zu ziehen.

Anhang

Inbetriebnahme

Inhalt

Überblick - Cohda Wireless MK5‘s

- Die MK5 OBU ist eine Komplett-Lösung für das Fahrzeug

- Die MK5 RSU ist eine komplette Outdoor-Lösung

  Sie besitzt dieselbe Platine wie die OBU, nur das Gehäuse und die Schnittstellen unterscheiden sich zur OBU.
- Beide sind folgendermaßen ausgestattet:
  - 802.11p radio
  - GNSS receiver
  - Dual-core ARM9, 1GB RAM, 4GB flash
  - Linux for stack & application
  - Ethernet
  - Antennen
- Unterschied der RSU zur OBU:
  - NEMA 4 Housing
  - Power over Ethernet, 802.3af oder 802.at Type 1 oder 2 Compliant
  - Optional RS232, CAN, USB (kann verfügbar gemacht werden)
- Die OBU besitzt zusätzlich:
  - USB, GPI, CAN
  - 7V- 36V Stromversorgung
  - Antenne & Stromkabel

Erste Schritte

Die MK5 unterstützt standardmäßig keine Maus und kein Display. Um sich mit der MK5 verbinden zu können, muss eine SSH-Anwendung wie Putty oder TeraTerm verwendet werden.

Folgende **Schritte** sind hierzu notwendig:

- Installiere einen bevorzugten SSH-Client:
  - Für Windows z. B. (Putty oder TeraTerm)
  - Unter Linux ist eine Command Line mit SSH-Unterstützung vorinstalliert

- Wähle eine physikalische Verbindungsmethode, **RJ45 (LAN)** oder **USB**. Der Hersteller bevorzugt eine **LAN-Verbindung (Ethernet)**.
- Wähle ein bevorzugtes Schema für die Netzwerkadressierung:
  - ***LAN***
    - ***Link-local IPv6 Adressierung*** (vom Hersteller bevorzugt). Kann in lokalen Netzwerken ohne Konfigurationsaufwand verwendet werden ((erfahrungsgemäß gab es unter Windows ständig Probleme mit der Verbindung. Über mein MacBook war die Verbindung problemloser)
    - ***DHCP Adressierung***. Am einfachsten, wenn man Zugriff auf den DHCP-Server hat & ihn nach zugewiesenen Adressen abfragen kann
    - ***Static IPv4 Adressierung***. Erfordert die erstmalige Anmeldung mit einer anderen Methode & die Zuweisung und Speicherung der Adresse im persistenten Speicher mit dem Befehl „fw_setenv“ (erfahrungsgemäß die beste Methode!)
  - ***USB***
    - *Ethernet over USB*. Benötigt den Windows „RNDIS“ Treiber. Der Netzwerkadapter muss auf die Netzwerkadresse 10.1.1.2 gesetzt werden

Für die Inbetriebnahme wurde ausschließlich eine Verbindung via LAN (Ethernet) genutzt. Für Informationen zur Verbindung via USB siehe Support-Seite.

Verbindung via LAN (Ethernet)

Link-local IPv6 Adressierung (Vom Hersteller bevorzugt)

Suche den Seriennummernaufkleber auf der Gehäuse-Rückseite der MK5 OBU/RSU. Die Seriennummer der MK5 ist auch seine Ethernet-MAC-Adresse & kann in eine Link-Local IPv6-Adresse umgewandelt werden. Seriennummer Beispiel auf Gehäuse-Rückseite der MK5: ***04:E5:48:20:1d:48***.

- Verwende für die Konvertierung von der Ethernet-Mac-Adresse *04:E5:48:20:1d:48* zu Link-local IPv6 Adresse folgenden Konverter
  ODER
- Verwende folgendes Präfix: „*fe80::6e5:48ff:fe*“ & füge die letzten 6 Ziffern/Zahlen (20:1d48) der Seriennummer/Mac-Adresse hinzu → *fe80::6e5:48ff:fe**20:1d48***
- „fe80::6e5:48ff:fe20:1d48“ ist nun die IPv6-Adresse der MK5. Diese wird als Host-Adresse bei der Verbindung über SSH eingetragen

Nun muss der Ethernet-Port ermittelt und an die verbindungslokale IPv6-Adresse angehängt werden. Dazu sind folgende Schritte notwendig:

- Öffne ein Terminal-Fenster
  - drücke Windows-Taste,
  - tippe „*cmd*“ &
  - *drücke ENTER*
- Im Terminal-Fenster tippe folgendes:
  - *ipconfig*
- Es sollte nun unter „*Ethernet-Adapter Ethernet*“ in der „*Verbindungslokalen IPv6-Adresse*“ die Ethernet-Port-Nummer zu sehen sein. Diese folgt nach dem „%“-Zeichen. z. B.:
  - `Verbindungslokale IPv6-Adresse fe80::6e5:48ff:fe20:1d48%11`
  - **Die Nummer 11 stellt die Ethernet-Port-Nummer dar**
- Teste die Verbindung zur MK5 durch einen Ping im Terminal-Fenster:
  - *ping -6 fe80::6e5:48ff:fe201d48%11*

DHCP Adressierung

Gebe in einem Browserfenster die Standard Router-Adresse ein (z. B. *fritz.box*, wenn der WLAN-Router eine Fritzbox ist). Dort anmelden und den Reiter LAN-Adressen aufrufen. DHCP-Adressen können mit der Option „Reserve-Adressen“ im Router persistent gemacht werden. Teste die Verbindung mit der MK5 über den ‚`ping`‘-Befehl.

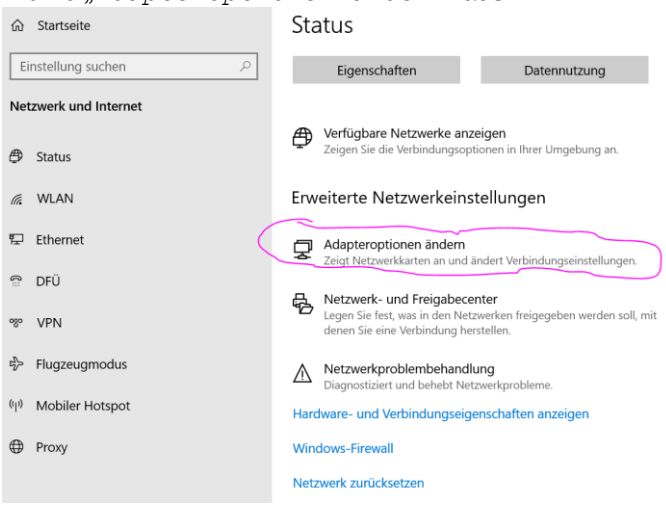
Static IPv4 Adressierung (erfahrungsgemäß die beste Option unter Windows)

Nach erstmaliger Anmeldung durch eine der anderen Methoden, kann eine feste IPv4-Adresse im persistenten Speicher der MK5 gespeichert werden. Dies erfolgt mit den folgenden Befehlen ‚*`fw_setenv`*‘ & einem anschließendem ‚*`reboot`*‘.

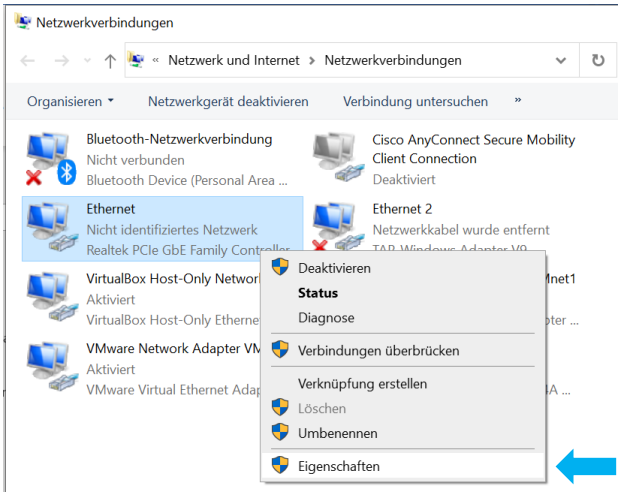
```
sudo fw_setenv static_ip_addr "192.168.52.79"
sudo fw_setenv static_ip_mask "255.255.255.0"
sudo fw_setenv static_ip_bcast "192.168.52.255"
sudo fw_setenv static_ip_gw "192.168.52.1"
sudo fw_setenv static_ip_ns "192.168.52.2"
reboot
```

Dann sollte in Windows unter Netzwerkverbindungen dem Ethernet-Adapter eine feste IPv4-Adresse zugewiesen werden, die zur gesetzten IPv4-Adresse der MK5 (192.168.52.79) passt.

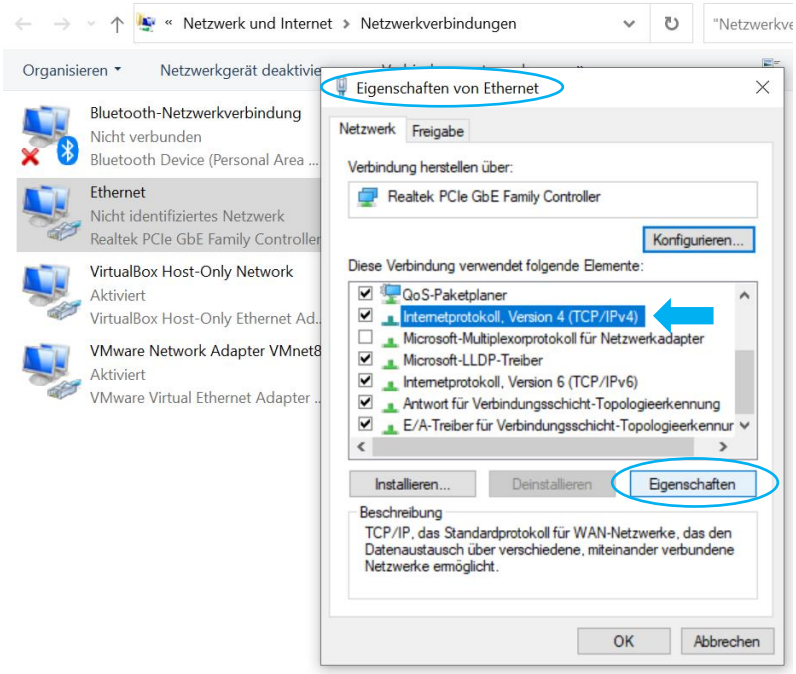
- drücke die Windows-Taste,
- tippe „`Netzwerkstatus`“ *&*
- *drücke ENTER*
- *Wähle „`Adapteroptionen ändern`“ aus*



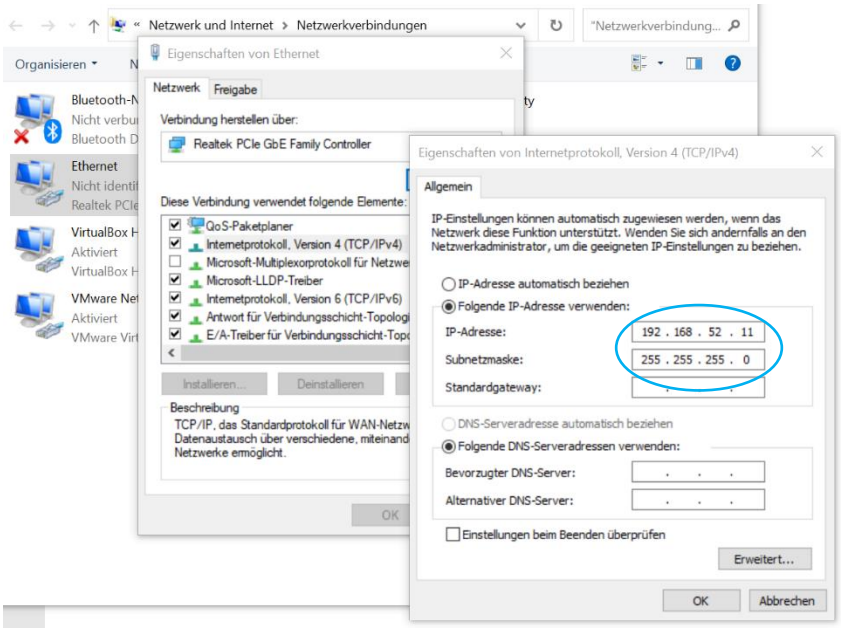
- *Dann rechts-Klick auf den aktiven Ethernet-Adapter & „`Eigenschaften`“ auswählen*



- Nun in den „Eigenschaften von Ethernet“: Scrolle zu „`Internetprotokoll, Version 4 (TCP/IPv4)`“ & klicke auf „`Eigenschaften`“ (unten rechts)



- Nun muss eine feste IP-Adresse vergeben werden, z. B. `192.168.52.11` & die Subnetzmaske `255.255.255.0`



Link-local IPv4 Adressierung

Die link-lokale IPv4-Adressierung wird nicht empfohlen. Dennoch werden im Folgenden einige Informationen dazu gegeben.

Eine virtuelle Schnittstelle / Alias eth0:1 wird verwendet, um Netzwerkunterstützung innerhalb des IPv4 Link-Local-Adressraums 169.254.0.0/16 zu bieten - die zugewiesene Adresse hat die Form 169.254.ABC.DEF, wobei ABC.DEF den letzten beiden Oktetten der Seriennummer des Geräts im Dezimalformat entspricht.

Zum Beispiel wie unten gezeigt: 01-04E54801325C -> 169.254.0x32.0x5c = 169.254.50.92

```
root@MK5:~# ifconfig
.....
.....
eth0:1 Link encap:Ethernet HWaddr 04:e5:48:01:32:5c
 inet addr:169.254.50.92 Bcast:169.254.255.255 Mask:255.255.0.0
 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
```

Unter Windows:

Die IPv4-Adresse auf dem Windows-PC wird wie folgt angezeigt:

```
C:\Users\xyz>ipconfig
Windows IP Configuration
Ethernet adapter Local Area Connection:
Connection-specific DNS Suffix . :
 Link-local IPv6 Address . . . . . : fe80::e0e8:8bdf:4f1e:5ede%11
 Autoconfiguration IPv4 Address. . : 169.254.94.222
 Subnet Mask . . . . . : 255.255.0.0
 Default Gateway . . . . . :
```

Ping die MK5 von Windows aus:

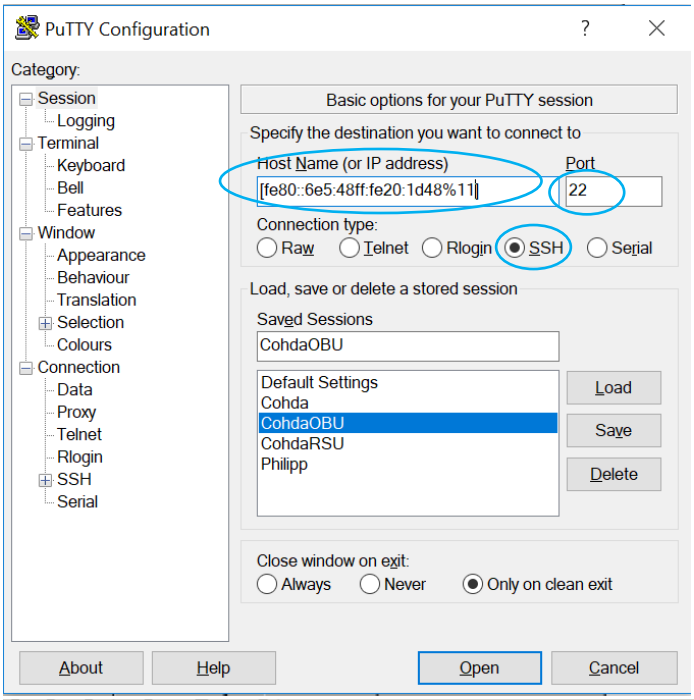
```
C:\Users\xyz>ping 169.254.50.92
Pinging 169.254.50.92 with 32 bytes of data:
Reply from 169.254.50.92: bytes=32 time=1ms TTL=64
Reply from 169.254.50.92: bytes=32 time<1ms TTL=64
...
```

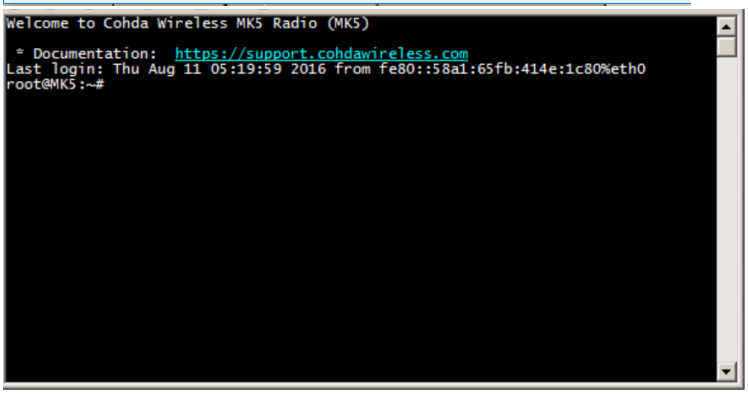
Um SSH von Windows aus durchzuführen, kann Tera-Term oder Putty verwendet. Im Abschnitt "Windows SSH-Login (mit Putty)" ist die Verwendung mit Putty beschrieben.

Windows SSH-Login (mit Putty)

Das folgende Beispiel illustriert die Nutzung mit der unter *Verbindung via Link-local IPv6 Adressierung ermittelten I*Pv6-Adresse*. Wenn eine Verbindung via Static IPv4 Adressierung genutzt werden möchte, dann einfach im Feld „Host Name (or IP address)“ die auf der MK5 festgelegte IPv4-Adresse verwenden (192.168.52.79, siehe* Static IPv4 Adressierung (erfahrungsgemäß die beste Option unter Windows)

- Öffne Putty und stelle sicher, dass der „Connection type“ auf SSH eingestellt ist.
- Unter „Host Name (or IP adress)“ muss die IPv6-Adresse **innerhalb eckiger Klammern** eingetragen werden → [*fe80::6e5:48ff:fe**20:1d48%11***]
- Port-Nummer bleibt die 22 (Standard-Port für SSH-Verbindungen).
- Nun auf „Open“ klicken. Anmeldename: ‚`user`‘, Passwort: ‚`user`‘.





ExampleETSI - Installing & Running

Die Beispiel-Anwendung „ExampleETSI“ kann zum Übertragen von ETSI-Nachrichten wie CAM (Cooperative Awareness Messages), DENM (Decentralized Notification Message), MAP/SPAT (MAP/Signal Phase And Timing) & IVI (In Vehicle Information) verwendet werden. Sie kann sowohl als OBU als auch als RSU konfiguriert werden.

- Als OBU konfiguriert, überträgt die Beispiel-Anwendung CAMs. CAMs enthalten Position, Geschwindigkeit, Richtung usw. des Fahrzeugs. Die Daten stammen von GNSS.
- Als RSU konfiguriert, überträgt die Beispiel-Anwendung DENMs, z. B. Road Works oder Unwetterwarnungen, MAP/SPAT, die von Ampeln gesendet werden, oder IVI, z. B. Geschwindigkeitsbegrenzungen, Richtungen usw. Diese Daten stammen aus einer Konfigurationsdatei.

Aus Hardware-Sicht spielt es keine Rolle, ob es sich bei der Hardware um eine RSU oder OBU handelt. Das Verhalten der Beispiel-Anwendung wird durch die App-Konfiguration definiert.

Die exampleETSI-App wird als Binärdatei bereitgestellt. Um eigene Anwendungen schreiben oder die exampleETSI-App ändern zu können, muss das SDK gekauft werden. Dieses enthält den Quellcode der Beispiel-Anwendung.

Running ExampleETSI – Voraussetzungen

- Sicherstellen, dass die GNSS-Antenne angeschlossen ist
- Sicherstellen, dass die GNSS-Antenne zum Himmel zeigt
- Sicherstellen, dass GNSS-Fix vorhanden ist (Neben der GNSS-Antenne blinkt die 1PPS-LED, sobald das GNSS-Fix vorhanden ist. Mit ‚`gpspipe-r`' kann überprüft werden, ob gültige NMEA-Datensätze empfangen werden)

Der ETSI-Stack erfordert einen gültigen 3D-Fix & wird nicht gestartet, wenn der GNSS-Fix fehlt

- Nutze ‚`chronyc tracking`', um time lock für PPS zu bestätigen
- Jedes Mal, wenn die ExampleETSI-Anwendung ausgeführt wird, wird ein Log-Verzeichnis erstellt. Da dieses sehr groß werden kann, wird eine SD-Karte zum Speichern empfohlen:
  - Lege die SD-Karte in den SD-Karten-Slot der MK5 ein
  - Checke, ob das Betriebssystem die SD-Karte erkennt: `cat /proc/partitions` `(Dies sollte mmcblk2p1 anzeigen, das ist die microSD-Karte)`
  - Die SD-Karte ist in `/mnt/src` „gemountet"
  - Erstelle nun ein Log-Verzeichnis auf der SD-Karte: `mkdir /mnt/src/log`
  - Erstelle eine Verknüpfung des Log-Verzeichnisses der SD-Karte in `/mnt/rw`
    - `ln -s /mnt/src/log /mnt/ubi/log`

Die pcap-Datien im Log-Verzeichnis können mit Wireshark angesehen werden. Wireshark ist in der Cohda SDK installiert. Um die Dateien unter Windows ansehen zu können, muss Wireshark mit Cohda Plugins installiert werden.

Installing exampleETSI

Das ExampleETSI-Archiv ist auf dem Image im Verzeichnis `/opt/cohda/application/` zu finden. Zunächst logge dich via Putty auf der MK5 ein (Benutzername: `user`, Passwort: `user`)

- Navigiere zum Verzeichnis, in dem das exampleETSI-Archiv liegt
  - `cd /opt/cohda/application/`
- Dort entpacke das exampleETSI-Archiv:
  - `tar -xzf exampleETSI-mk5-*.tgz`
- Nun navigiere zum Verzeichnis /mnt/rw/ & entpacke dort das example-ETSI-Archiv
  - `cd /mnt/rw/`
  - `tar -xzf /opt/cohda/application/exampleETSI-mk5-*.tgz`

Running exampleETSI

Starte die exampleETSI-Anwendung stets innerhalb des `/mnt/rw/exampleETSI` Ordners. Wenn die exampleETSI-App ausgeführt wird, verwendet sie bestimmte Konfigurationsdateien. Diese befinden sich im Ordner `exampleETSI`. Je nachdem, ob die Anwendung als OBU oder RSU ausgeführt wird, werden folgende Konfigurationsdateien verwendet (* steht für RSU oder OBU):

- `*.cfg`: enthält die Konfigurationsparameter für die Anwendung. Dort können verschiedene Arten von zu sendenden/empfangenden Nachrichten eingestellt werden
- `*.conf`: wird verwendet, um die Standard-Stack-Parameter zu überschreiben

Es müssen Positions- & Zeitangaben eingeholt werden, um die exampleETSI-Anwendung ausführen zu können! Falls es nicht möglich ist, die GPS-Antenne unter freien Himmel zu montieren, müssen die

GPS-Daten von einer externen gpsd-Quelle bereitgestellt werden. Dazu müssen folgende Parameter in den Konfigurationsdateien `obu.conf` oder `rsu.conf` im Verzeichnis `/mnt/rw/exampleETSI` verändert werden:

- `Cohda_GPSD_HostName = <IP address of external source>`
- `Cohda_GPSD_Port = <port number>;1,65535`

Das Datum kann manuell mit folgendem Befehl eingestellt werden:

- `sudo date -s "Thurs February 20 05:53:59 UTC 2020"`

Running exampleETSI als OBU

- Navigiere zum Verzeichnis, in dem die exampleETSI-Anwendung liegt
  - `cd /mnt/rw/exampleETSI`
- Starte die Beispielanwendung mit folgendem Befehl:
  - `./rc.exampleETSI start obu`
    Die MK5 überträgt CAMs und empfängt auch CAM, DENM, MAP/SPAT & IVI Nachrichten

In ‚`/mnt/rw/log/current`' wird ein Log-Verzeichnis erstellt. Dort befinden pcap-Dateien, die die Sende- & Empfangspakete sowie die GPS-Informationspakete enthalten. Weiterhin befindet sich dort eine Datei Namens ‚`*.conf`', die die verwendete Konfiguration enthält sowie eine Log-Datei namens ‚`stderr`'.

Die Meldung „`LOG_RedirectUDP FNEND: (2,<NULL>,0) SEHEN: UDP_OpenPort FAIL`", kann ignoriert werden.

- Stoppe die Beispielanwendung mit folgendem Befehl:
  - `./rc.exampleETSI stop`

Running exampleETSI als RSU

- Navigiere zum Verzeichnis, in dem die exampleETSI-Anwendung liegt
  - `cd /mnt/rw/exampleETSI`
- Starte die Beispielanwendung mit folgendem Befehl:
  - `./rc.exampleETSI start rsu`
    Die MK5 übertragt und empfängt DENM, MAP/SPAT und IVI Nachrichten. Diese Nachrichten werden in der ‚`rsu.cfg`'-Datei konfiguriert. Diese Textdatei kann mit jedem Editor z. B. vi, der auf der MK5 vorinstalliert ist, bearbeitet werden.

In ‚`/mnt/rw/log/current`' wird ebenfalls ein Log-Verzeichnis erstellt.

- Stoppe die Beispielanwendung mit folgendem Befehl:
  - `./rc.exampleETSI stop`

Log-Dateien (pcap-Files)

Um die pcap-Dateien (in `/mnt/src/log/`) ansehen zu können, müssen sie zunächst auf den PC hochgeladen und die Anwendung Wireshark gestartet werden. Dabei muss es sich um Wireshark mit Cohda Plugin handeln. Das „normale" Wireshark enthält die Dissektoren fürs ETSI-Protokoll nicht und kann diese demnach nicht dekodieren.

Das Log-Verzeichnis enthält folgende Dateien:

- `rx.pcap`
  Vom Radio empfangene Nachrichten
- `tx.pcap`
  Vom Radio versendete Nachrichten

- `gps.pcap`
  Von gpsd-Programm empfangene GPS-Informationen. Das gpsd-Programm erhält die NMEA-Datensätze und extrahiert Position, Kurs, Geschwindigkeit usw.
- `stderr`
  Log-Datei mit Errors, Warnings & Debug Nachrichten
- `conf`
  Stack-Konfiguration
- `cfg`
  Konfiguration der Anwendung

Die Länge der ‚`rx.pcap`' beträgt 14590 & die der ‚`tx.pcap`' 138130 Bytes. Dies zeigt an, dass die MK5 Pakete empfängt & versendet. Mit dem Befehl ‚`watch -d ll /mnt/rw/log/current/`' kann überprüft werden, ob die rx.pcap & tx.pcap-Dateien wachsen. Dies ist ein Hinweis dafür, dass die MK5 sendet & empfängt.

App: Verbindung OBU mit einem Android Gerät

Das Android-Gerät (Smartphone oder Tablet) wird als HMI-Gerät für die OBU eingerichtet. Somit können audio-visuelle Warnungen & Alarme betrachtet werden. Dazu muss die Cohda HMI App ‚`CohdaASDDVI-6.2.zip`' unter Box.com heruntergeladen und als System App auf dem Android-Gerät installiert werden.

- Installiere einen Datei-Explorer (wie z. B. ES FileExplorer) auf dem Android-Gerät. Samsung-Geräte haben bereits einen Datei-Explorer installiert (im Samsung Ordner namens ‚`My Files`').
- Verbinde das Android-Gerät mit Hilfe eines USB-Kabels mit dem PC. Öffne den Windows Datei-Explorer und navigiere zum Download-Ordner des Android-Geräts.
- Kopiere die ‚`CohdaASDDVI-debug.apk`'-Datei in den Download-Ordner des Android-Geräts. Die apk-Datei befindet sich im heruntergeladenen `CohdaASDDVI-6.2.zip`-Ordner.
- Auf dem Android-Gerät: navigiere zu „Einstellungen→Sicherheit“ & aktiviere „Unbekannte Quellen“ für „Installation von Programmen aus unbekannten Quellen“
- Öffne den Datei-Explorer auf dem Android-Gerät und navigiere zum Download-Ordner. Dort installiere die ‚`CohdaASDDVI-debug.apk`'-Datei.
- Starte die Anwendung, nach erfolgreichem Installieren & Öffnen der DVI-App, sollte folgendes zu sehen sein:

- Tippe auf den Bildschirm und dann auf die drei Punkte in der oberen Leiste. Dort die Einstellungen antippen. In Einstellungen:
  - „Automatische Abschaltung“ deaktivieren (nicht aktivieren)
  - Aktiviere „Statusindikatoren“ (standardmäßig aktiviert)

Einrichten der OBU & des Android-Geräts

Um ein Android-Gerät mit der OBU verbinden zu können, wird ein USB OTG Adapter – USB-A to micro-USB – benötigt.

OBU

Auf der OBU muss die Firmware Umgebungsvariable ‚usb_mode' auf ‚android' gesetzt werden.

- Um die Umgebungsvariablen aufzulisten, muss folgendes eingegeben werden
  - sudo fw_printenv
- Um die Umgebungsvariable ‚usb_mode' zu setzen, muss folgendes eingegeben werden
  - Sudo fw_setenv usb_mode android
- Anschließend muss die OBU neu gestartet werden. Dazu tippe
  - reboot
- Nach dem Reboot, überprüfe, ob die Umgebungsvariable erfolgreich gesetzt wurde
  - sudo fw_printenv (schaue nach ‚usb_mode')
- Um die Umgebungsvariable ‚usb_mode' zu deaktivieren muss folgender Befehl verwendet werden
  - sudo fw_setenv usb_mode"
    Anschließend muss neu gestartet werden: reboot

Android-Gerät

Auf dem Android-Gerät muss in „Einstellungen" unter „Tethering and Mobile Hotspot" die Option „USB tethering" aktiviert werden.

Wenn das USB-Kabel an der OBU angesteckt ist & die Umgebungsvariable ‚usb_mode' auf der MK5 auf ‚android' gesetzt ist, tippe folgendes in der OBU: ifconfig usb0. Folgende IP-Adresse wird angezeigt (192.168.42.101).

```
root@MK5:~# ifconfig usb0
usb0      Link encap:Ethernet  HWaddr 02:00:5a:0a:32:38
          inet addr:192.168.42.101  Bcast:192.168.42.255  Mask:255.255.255.0
          inet6 addr: fe80::5aff:fe0a:3238/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:6 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:544 (544.0 B)  TX bytes:732 (732.0 B)
```

Um die USB IP-Adresse auf dem Android-Gerät zu ermitteln, muss dort ein Terminal (wie z. B. Material Terminal) installiert werden. Nach Installation des Terminals, öffne das Terminal & tippe dort ifconfig, um sich alle Netzwerk Schnittstellen anzeigen zu lassen. Suche nach der Schnittstelle ‚rndis0'. Es zeigt eine IP-Adresse wie folgende:

```
rndis0    Link encap:UNSPEC
          inet addr:192.168.42.129  Bcast:192.168.42.255  Mask:255.255.255.0
```

Nun sollte es möglich sein das Android-Gerät von der OBU aus anzupingen:

- ping -I usb0 192.168.42.129

```
root@MK5:~# ping -I usb0 192.168.42.129
PING 192.168.42.129 (192.168.42.129) from 192.168.42.101 usb0: 56(84) bytes of data.
64 bytes from 192.168.42.129: icmp_seq=1 ttl=64 time=0.991 ms
64 bytes from 192.168.42.129: icmp_seq=2 ttl=64 time=0.618 ms
...
...
```

Damit die Beispielanwendung „`exampleETSI`“ Warnmeldungen & Warnungen auf dem Android-Gerät anzeigen kann, müssen folgende Parameter in der ‚`obu.conf`‘-Datei angepasst werden:

```
Cohda_HMI.Destination = <Android Device IP address>
Cohda_HMI.Port = 7100
Cohda_HMI.Interface = usb0
Cohda_HMI.Timeout_ms = 10
```

Die IP-Adresse des Android-Geräts kann mit der Terminal-Anwendung ermittelt werden (siehe oben).

Testen der App

- Bearbeite sowohl auf der OBU als auch RSU die ‚`*.conf`‘-Dateien mit Hilfe des ‚`vi`‘-Editors. Dort setze den Parameter ‚`ItsGnSecurity`‘ auf `0`. Standardmäßig ist dieser auf 1 gesetzt.
- Nun bearbeite nur auf der OBU die ‚obu.conf‘-Datei. Aktiviere dort EEBL & FCW, indem folgende Parameter gesetzt werden:
  - `Cohda_App_EEBL.ENABLE = 1;`
  - `Cohda_App_EEBL.SEND_ENABLE = 1;`
  - `Cohda_App_EEBL.REC_ENABLE = 1;`
  - `Cohda_App_FCW.ENABLE = 1;`
- Stelle sicher, dass beide MK5’s synchronisiert sind (Wenn die GPS-Antenne angeschlossen und zum Himmel gerichtet ist, werden die MK5’s synchronisiert). Führe den Befehl ‚`gpspipe -r`‘ aus, um zu überprüfen, ob NMEA-Datensätze angezeigt werden
- Verbinde das Android-Gerät mit der MK5 OBU via USB-Kabel. Überprüfe die Verbindungen mit Hilfe von pingen
- Sobald das Android-Gerät an der OBU angeschlossen & die DVI-App gestartet ist, erscheint folgender Bildschirm:

- Auf beiden MK5’s müssen nun die exampleETSI-Anwendungen gleichzeitig gestartet werden:
  - Befehl für die RSU: `./rc.exampleETSI start rsu`
  - Befehl für die OBU: `./rc.exampleETSI start obu`

Nun sollte folgender Bildschirm zu sehen sein:

- Die „20“ & „dBm“ impliziert, dass die OBU so konfiguriert ist, dass sie mit 20dBm sendet
- Das Auto-Icon & die „1“ daneben, implizieren, dass die OBU das Signal von einem Fahrzeug empfängt

Um die Kommunikation zu beenden, muss folgender Befehl auf beiden MK5’s ausgeführt werden:

- `./rc.exampleETSI stop`

Abschließend können die Log-Dateien in `/mnt/ubi/log/current/` oder `/mnt/src/log/current/` angesehen werden.

Automatisches Starten einer Anwendung nach dem Hochfahren der MK5

Um eine Anwendung nach dem Hochfahren der MK5 automatisch zu starten, muss eine Datei mit dem Name ‚`rc.local`‘ erstellt und im persistenten Speicher ‚`/mnt/ubi/`‘-Ordner (der mit dem /mnt/rw/ Ordner identisch ist) abgelegt werden.

- Navigiere zum ‚`/mnt/ubi/`‘-Ordner:
  - `cd /mnt/ubi/`
- Erstelle dort die Datei mit dem Namen ‚`rc.local`‘
  - `vi rc.local` (dann drücke Enter)
  - drücke die esc-Taste
  - tippe `:wq!` (das steht für write & quit)
- Diese Datei muss die Erlaubnis zur Ausführung erhalten (executable)
  - `chmod +x /mnt/ubi/rc.local`

- Füge in die ‚rc.local‘-Datei folgende Zeilen hinzu:

```
APPNAME=exampleETSI
APPDIR=/mnt/rw/$APPNAME/
MODE=obu

#AutoStart
$APPDIR/rc.$APPNAME $1 $MODE
```

  - Dazu öffne die Datei: `vi rc.local`
  - Tippe ein `i` (steht für insert)
  - Tippe die oben aufgeführten Zeilen ab
  - Drücke die `esc-Taste` sowie `:wq!` (doppelpunkt, write & quit)
- Teste, ob die exampleETSI-App mit der soeben erstellten ‚`rc.local`‘-Datei startet & stopt
  - `/mnt/rw/rc.local start`
  - `/mnt/rw/rc.local stop`

LLC-Tool zur Erfassung von Rx-Paketen

Erfassen von Rx-Paketen

Die MK5-Einheiten besitzen ein LLC-Befehl, der ausgeführt werden kann, um Pakete von den Funkgeräten (Radios) zu erfassen. Dazu sind folgende Schritte notwendig:

- Einrichten der Kanäle, um Pakete von Radio-A und von Antenne-1 zu empfangen (unter Annahme, dass channel-nummer die 184 ist)
  - `chconfig -s -w CCH -i wave-raw -c 184 -r a -e 0x88B6 -a 1`
- Einrichten der Kanäle, um Pakete von Radio-B und Antenne-2 zu empfangen (unter Annahme, dass die channel-nummer die 182 ist)
  - `chconfig -s -w SCH -i wave-raw -c 182 -r b -e 0x88B6 -a 2`
- Aufrufen des „Monitoring-Interfaces“ für Radio-A
  - `ifconfig cw-mon-rxa up`
- Aufrufen des „Monitoring-Interfaces“ für Radio B
  - `ifconfig cw-mon-rxb up`
- Führe ‚`llc rcap`‘ aus, um Pakete von Radio-A zu erfassen
  - `llc rcap --HdrLen 52 --Interface cw-mon-rxa --OutputFilename llc_rcap_a.pcap –Meta`
- Führe ‚`llc rcap`‘ aus, um Pakete von Radio-B zu erfassen
  - `llc rcap --HdrLen 52 --Interface cw-mon-rxb --OutputFilename llc_rcap_b.pcap –Meta`

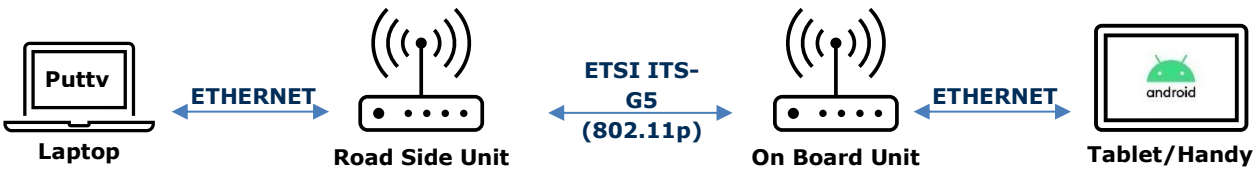
Kopiere die ‚*.pcap‘-Dateien auf den Computer unter Nutzung von SCP. Dort lassen sich die pcap-Dateien mit dem Tool Wireshark (ausschließlich mit Wireshark mit Cohda plugin möglich) ansehen.

Weiterleitung der Pakete an einen UDP Port

Der ‚`llc rcap`‘-Befehl kann auch ausgeführt werden, um die empfangenen Pakete an einen UDP-Port eines entfernten Hosts weiterzuleiten. Dazu müssen die folgenden Parameter in der ‚`Command line`‘-Option angegeben werden: `--RemoteHost` & `--RemotePort`. Der nachfolgende ermöglicht die Weiterleitung der Pakete an einen UDP Port, nachdem der Kanal eingerichtet & das „Monitoring-Interface“ aufgerufen wurde (siehe oben).

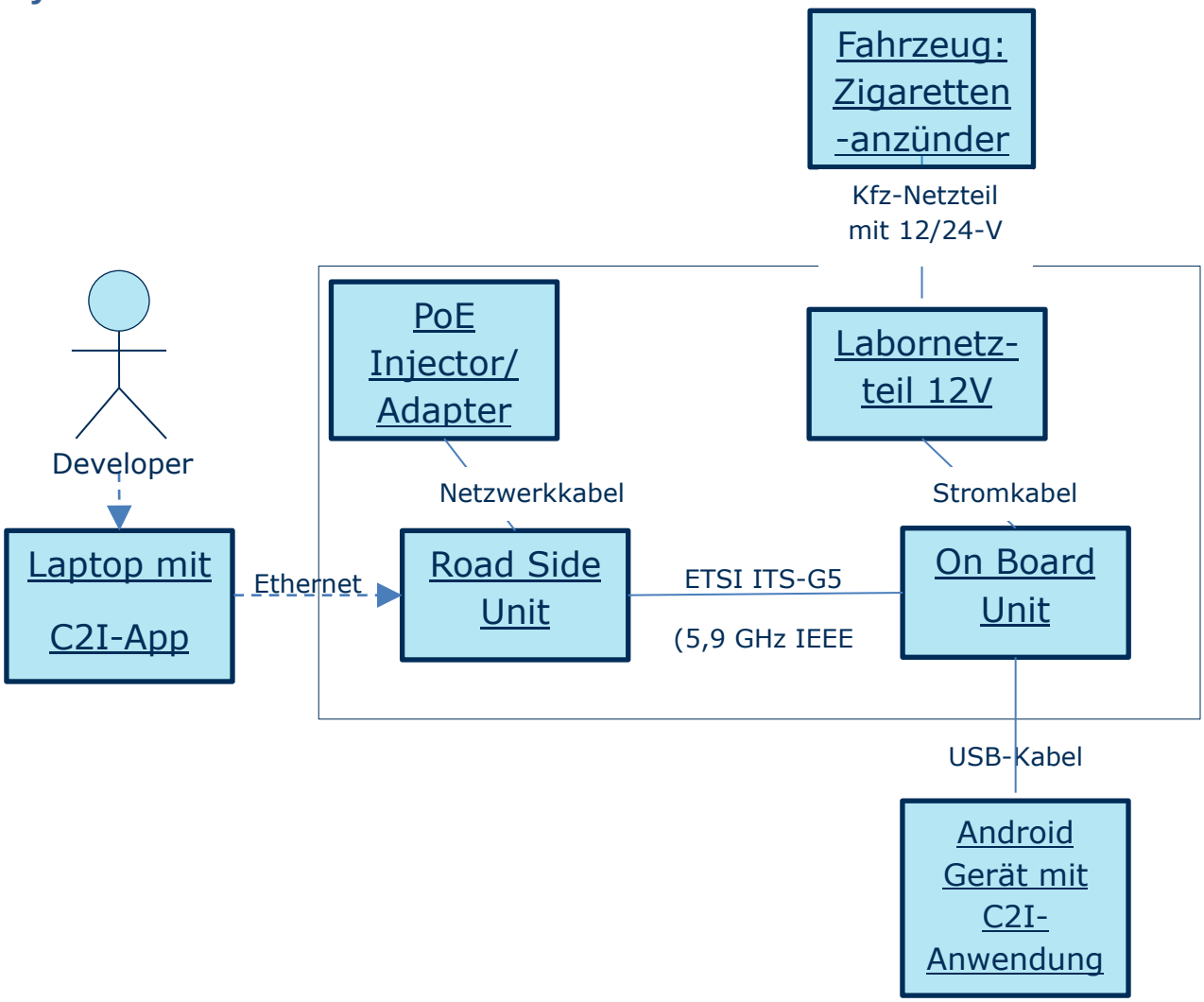
- `llc rcap --HdrLen 52 --Interface cw-mon-rxa --Meta --RemoteHost 192.168.52.114 --RemotePort 37008`

Use Case



Name des Use Cases	Gefahrenwarnung (Vorwarnanhänger/Anzeigetafel)
Kurzbeschreibung	Informationen über Gefahrenstelle werden via Ethernet an die RSU übermittelt und über das ETSI ITS-G5 Protokoll an die OBU ins Fahrzeug versendet.
Akteure	Laptop, Anzeigetafel, RSU, OBU, Fahrzeug, HMI-Lösung
Normaler Ablauf	1. Es liegen Infos (z. B. Ort) der Gefahrenstelle vor, die Fahrzeuge über ihre Umgebung erhalten sollen. 2. Ein Laptop übermittelt via Ethernet diese Infos an die Anzeigetafel. 3. Eine C2I-Anwendung (auf dem Laptop) generiert entspr. DENM-Nachricht & übermittelt diese via Ethernet an die RSU. 4. Die RSU versendet die DENM-Nachricht über das ETSI ITS-G5 Protokoll an die OBU im Fahrzeug. 5. Das Fahrzeug erhält die Information. 6. Diese Info kann auf dem HMI betrachtet werden.
Alternativer Ablauf	-
Annahme	-
Auslöser	Eine Gefahrenstelle, welche die Verkehrssicherheit/-effizienz betrifft, liegt vor.
Vorbedingungen	Daten müssen vom Laptop an die Anzeigetafel übermittelt werden.

Systemkontext



Troubleshooting

Inhaltsverzeichnis

Problem 1: Verbindungsprobleme mit Putty unter Windows

Die Verbindung via LAN (Ethernet) mit der **Link-local IPV6-Adressierungs-Methode** ergab immer Verbindungsprobleme. Es musste zigmal versucht werden, sich mit der MK5 zu verbinden, bis nach Versuch X endlich mal eine SSH-Verbindung über Putty ermöglicht werden konnte.

Normalerweise müssen folgende Schritte für die Verbindung über Ethernet & IPv6-Adresse erfolgen:

1. MK5 mit einem Ethernet-Kabel an den Ethernet-Port des PC’s anschließen
2. Terminal-Fenster öffnen
    a. Drücke die Windows-Taste,
    b. Tippe „`cmd`“ &
    c. Drücke ENTER
3. Im Terminal-Fenster muss folgendes getippt werden:
    a. `Ipconfig`
4. Im Terminal-Fenster sollte unter „Ethernet-Adapter Ethernet“ in der „Verbindungslokalen Ipv6-Adresse“ die Ethernet-Port-Nummer zu sehen sein. Diese folgt nach dem „%“-Zeichen.
    a. `Verbindungslokale IPv6-Adresse fe80::58a1:65fb:414e:1c80%11`
    b. Die 11 stellt die Ethernet-Port-Nummer dar
5. Nun sollte die Verbindung zur MK5 durch einen Ping getestet werden:
    a. `ping -6 fe80::6e5:48ff:fe20:1d48%11`
6. Nun wird Putty (oder TeraTerm) geöffnet und eine SSH-Verbindung zu der MK5 aufgebaut
    a. Connection type wird auf SSH gestellt

b. In das Feld unter „Host Name (or IP address)“ wird die IPv6-Adresse **mit eckigen Klammern** wie folgt eingetragen: [fe80::6e5:48ff:fe20:1d48%11]
c. Port Nummber bleibt die 22
d. Dann auf „Open“ klicken

Lösung zu Problem 1: Keine Lösung

Hierfür konnte keine Lösung gefunden werden! Der Hersteller konnte sich das auch nicht erklären.

Aber ein TIPP:

- Mit einem Mac-Book hat die Verbindung stets über SSH funktioniert
- Sobald man einmal Zugriff zur MK5 hatte, würde ich sofort die Adressierungs-Methode „Static IPv4 Adressierung“ (Beschreibung ist im Dokument „Cohda_Doku“ zu finden) verwenden. Das hat in 98% der Verbindungsversuchen funktioniert.

Problem 2: Die App zeigt nicht wie erwartet Warnhinweise

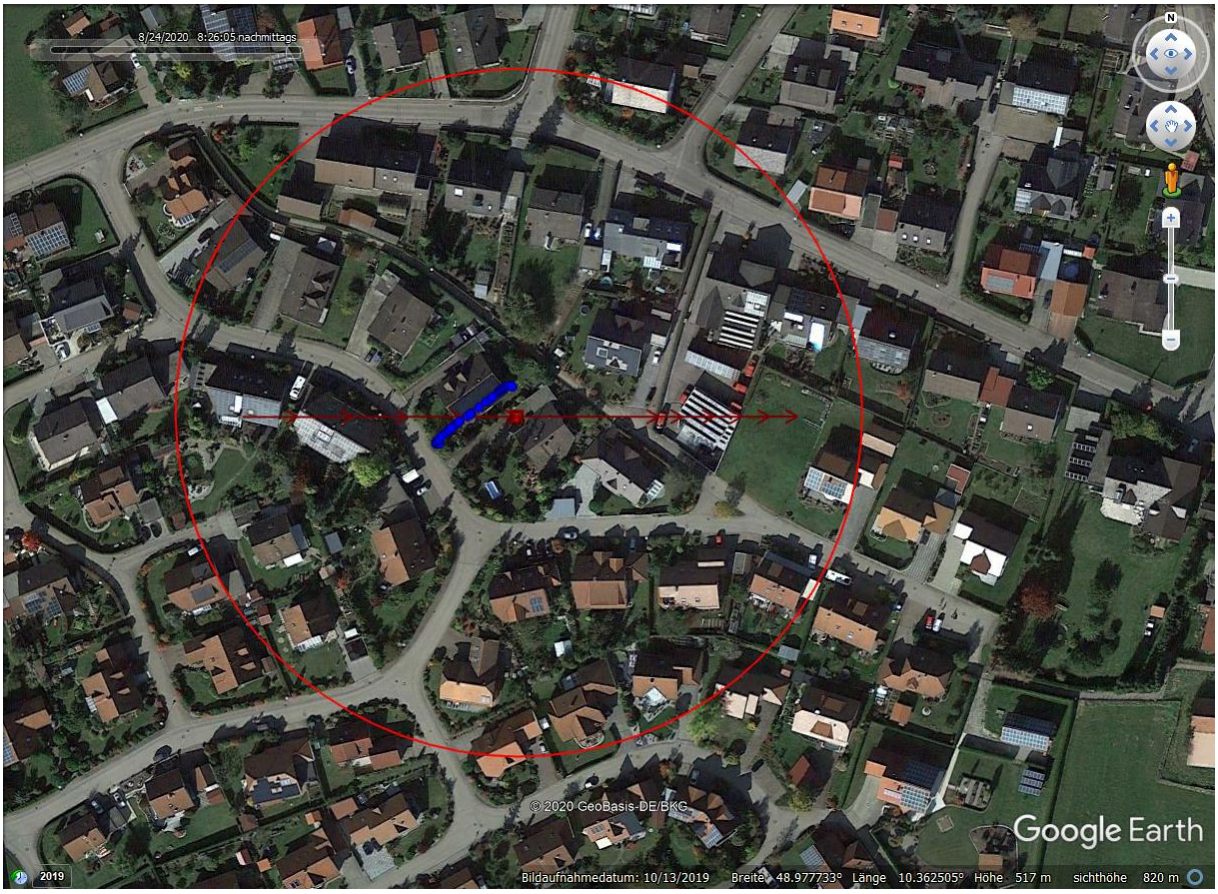
Gewünschtes (vom Hersteller versprochenes) Szenario:

- Die ExampleETSI-Anwendungen werden jeweils auf der RSU & der OBU installiert & gestartet
- Es müssen keine Änderungen an den Konfigurationsdateien vorgenommen werden
- Die OBU, die sich im Fahrzeug befindet, hat eine Verbindung zur Android-Cohda-App
- Die RSU wird aufgestellt & eingeschalten & die ExampleETSI-Anwendung gestartet
- Das Fahrzeug fährt (mit der OBU) auf die RSU zu & erhält die durch die RSU versendeten DENM-Nachrichten
- Die Cohda-App zeigt entsprechend Warnsignale

Das hat bisher nicht einmal funktioniert. In ***Vanetza_als_Alternative.docx*** wurde das Projekt „*Vanetza*“ als Alternative zur Visualisierung der Nachrichten ausprobiert. *Vanetza* ist eine Open-Source Implementierung des ETSI C-ITS Protokoll Standards.

Nachfolgend werden einige Probleme vorgestellt, die während dem Versuch „Problem Nummer 2“ zu lösen, aufgetreten sind.

Problem 2.1: Gefahrene Route stimmt nicht mit Ausbreitung der DENM-Msgs überein



- Die blauen Punkte :
  - Es wird für jede gesendete CAM-Nachricht (kommt von der OBU) ein blauer Punkt erzeugt
  - Somit zeigen die blauen Punkte quasi den Fahrtweg der OBU an
  - Scheint als ob das GNSS nicht sehr genau ist (der eigentliche Fahrtweg war nämlich rechts davon auf dem Weg)
  - Das Problem scheint die Position zu sein (zwischen den beiden Häusern)
- Roter Kasten in der Mitte
  - Das ist die Position der RSU (also da wo die RSU physisch abgelegt wurde)
  - Diese Position stellt aber NICHT die Event-Position dar (Events können z. B. Baustellen sein), diese sollte in der Konfigurationsdatei manuell erfasst werden (siehe Erklärung unten)
- Die rote Linie
  - Jede DENM-Nachricht (kommt von der RSU) besitzt eine traceHistory & eine eventHistory
  - Die traceHistory stellt den Annäherungsweg an das Event dar (horizontaler Weg links von der RSU)
  - Die eventHistory stellt die Ausdehnung des Events (z. B. Baustelle) dar (horizontaler Weg rechts von der RSU)

→ Wie man sieht, befindet sich der Fahrtweg (blaue Punkte) nicht auf dem Weg der DENM-Nachrichten (rote Linie), die von der RSU versendet werden.

Lösung zu 2.1: Anpassung der Konfigurationsdatei rsu.cfg

Die Konfigurationsdatei der ExampleETSI-Anwendung auf der RSU anpassen.

Hier ist Vorschlag, die Testfahrt im rot eingezeichneten Bereich durchzuführen, zu sehen:

Das heißt, hier müssen einige **GPS-Koordinaten** in der ***rsu.cfg***-Datei der RSU **manuell eingetragen** werden. Dabei handelt es sich um die Koordinaten der Event-Position, die Koordinaten der sogenannten traceHistory & die Koordinaten der sogenannten eventHistory (genaueres dazu folgt).

WICHTIG für das Eintragen: Die Koordinaten (Latitude & Longitude) müssen als 1/10 eines Mikrogrades angegeben werden. Das heißt:

- ➔ 10^7 * Latitude -Koordinate (ggbf. muss auf der rechten Seite aufgefüllt werden)
- ➔ 10^7 * Longitude -Koordinate (ggbf. muss auf der rechten Seite aufgefüllt werden)

WICHTIG: Bei den History-Koordinaten handelt es sich um Delta Latitude‘s & Delta Longitude’s eines 1/10 Mikrogrades. Der 1.te Punkt ist das Delta zur Event-Position, der 2.te Punkt das Delta zum 1.ten Punkt, der 3.te Punkt ist das Delta zum 2.ten Punkt und so weiter. Der Vorschlag im Bild hat nur einen traceHistory-Punkt & nur einen eventHistory-Punkt. Beide sind die Deltas zur Event-Position.

Manuelle Anpassung der Event-Position in der rsu.cfg-Datei auf der RSU

Die RSU wird zur Kreuzung im Süden verschoben. Das kann durch physisches Positionieren der RSU dort hin erreicht werden. **VORHER** ist aber ein **manuelles Eintragen der GPS-Koordinaten der EVENT-Position (also da wo z. B. die Baustelle beginnen würde)** in der Konfigurationsdatei notwendig. **WICHTIG: Die Event-Position sollte unbedingt NACH der physischen Position der RSU sein!** D.h. die Position der RSU muss vor der Event-Position sein. Hierfür werden mit Hilfe von Google Earth Latitude- & Longitude-Werte für das Event ermittelt, z. B. Latitude: 48.976733, Longitude: 10.362777. Diese werden in die Konfigurationsdatei ***rsu.cfg*** der RSU im folgenden Abschnitt

```
EventOffsetAbsolute = 1
EventPositionSource = 1; # 0=use current GPS position,1=use config position
EventLatitude       = 489767480;    # Lat in tenths of microdegrees
EventLongitude      = 103627640;    # Long in tenths of microdegrees
EventAltitude       = 800001;
StartingPointSpeedLimit = (
  {Latitude = 489767480, Longitude = 103627640, Altitude = 0}
);#position from the eventPosition, indicating where the speed limit starts
```

Zudem wird

- EventOffsetAbsolute = 1 (≙traces, eventHistory, startingPointSpeedLimit are absolute values)
- EventPositionSource = 1 (≙ use config position)
- EventAltitude = 800001 (≙ nicht verfügbar),
- StartingPointSpeedLimit mit den Koordinaten der Event-Position

gesetzt.

Manuelle Anpassung der traceHistory in der rsu.cfg-Datei auf der RSU

💭 Die traceHistory ist der Annäherungsweg an das Event (horizontaler Weg links von der RSU).

Hierfür wird mit Hilfe von Google Earth ein GPS-Punkt auf der roten Linie (links von der RSU, siehe **Fehler! Verweisquelle konnte nicht gefunden werden.**) ausgewählt, z. B: Latitude: 48.976421, Longitude: 10.362513. Diese werden in die Konfigurationsdatei ***rsu.cfg*** der RSU im folgenden

```
# Path History points from event position
# Latitude and Longitude specified in 1/10th of a degree (degrees x 1e7)
# Altitude specified in centimetres

Traces = (
 { Latitude = 489764210, Longitude =  103625130, Altitude = 0},
 { Latitude = 0, Longitude =  -4386, Altitude = 0},
 { Latitude = 0, Longitude =  -6579, Altitude = 0},
 { Latitude = 0, Longitude =  -8772, Altitude = 0},
 { Latitude = 0, Longitude = -10966, Altitude = 0}
);
```

Es reicht ein einzelner GPS-Punkt, die anderen Zeilen können einfach gelöscht werden.

Manuelle Anpassung der eventHistory in der rsu.cfg-Datei auf der RSU

💭 Die eventHistory ist die Ausdehnung des Events (z. B. Baustelle) (horizontaler Weg rechts von der RSU).

Hierfür wird mit Hilfe von Google Earth ein GPS-Punkt auf der roten Linie (rechts von der RSU, siehe **Fehler! Verweisquelle konnte nicht gefunden werden.**) ausgewählt, z. B. Latitude: 48.976752, Longitude: 10.363562. Diese werden in die Konfigurationsdatei ***rsu.cfg*** der RSU im folgenden Abschnitt eingetragen:

```
# Event points from event position
# InformationQuality 0 = unavailable,1 = lowest, 7 = highest

EventHistory = (
{ Latitude = 489767520, Longitude = 103635620,Altitude=0,InformationQuality=7},
 { Latitude = 0, Longitude =  6579, Altitude = 0, InformationQuality = 7},
 { Latitude = 0, Longitude =  7833, Altitude = 0, InformationQuality = 7},
 { Latitude = 0, Longitude =  8929, Altitude = 0, InformationQuality = 7},
 { Latitude = 0, Longitude = 10026, Altitude = 0, InformationQuality = 7},
 { Latitude = 0, Longitude = 11123, Altitude = 0, InformationQuality = 7}
);
```

Es reicht ein einzelner GPS-Punkt, die anderen Zeilen können einfach gelöscht werden.

Problem 2.2: startingPointSpeedLimit deltaLatitude (488291310 >= 131072)

Im Log-Ordner auf der RSU unter `/mnt/src/` befindet sich die *`stderr`*-Datei. Dort kann folgender Fehler auftreten: *`DENMTx_FillDENMRoadWorks ERROR: startingPointSpeedLimit deltaLatitude (488291310 >= 131072).`*

Lösung zu 2.2: Anpassen des Attributes „StartingPointSpeedLimit“

Die Latitude- und Longitude-Werte des Attributes „`StartingPointSpeedLimit`“ in der ***rsu.cfg***-Datei verändern.
Hierfür werden die Werte der Attribute „`EventLatitude`“ & „`EventLongitude`“ genommen & in die Latitude- und Longitude-Werte des Attributs „`StartingPointSpeedLimit`“ eingetragen:

```
StartingPointSpeedLimit = ({
   Latitude = hier der Wert des EventLatitude's,
   Longitude = Hier der Wert des EventLongitude's,
   Altitude = 0
});
```

Problem 2.3: EventPoint index 0 deltaLatitude (488291330 >= 131072)

Im Log-Ordner auf der RSU unter `/mnt/src/` befindet sich die `stderr`-Datei. Dort kann folgender Fehler auftreten: `DENMTx_FillDENMRoadWorks ERROR: EventPoint index 0 deltaLatitude (488291330 >= 131072).`

Lösung zu 2.3: Anpassen des Attributes „EventOffsetAbsolute“

Wenn absolute Werte für die Attribute „`eventHistory`“ & „`traceHistory`“ verwendet werden, muss auch das Attribut „`EventOffsetAbsolut`“ auf 1 (≙traces, eventHistory, startingPointSpeedLimit are absolute values) gesetzt werden:

```
EventOffsetAbsolute = 1
```

Ansonsten wird angenommen, dass die Offsets relativ zur Event-Position sind und daher viel zu groß sind.

Problem 2.4: Wenn CAM-, MAP-, SPAT- and IVIM-Msgs vorhanden, aber keine DENM-Msgs in tx.pcap (RSU) sind

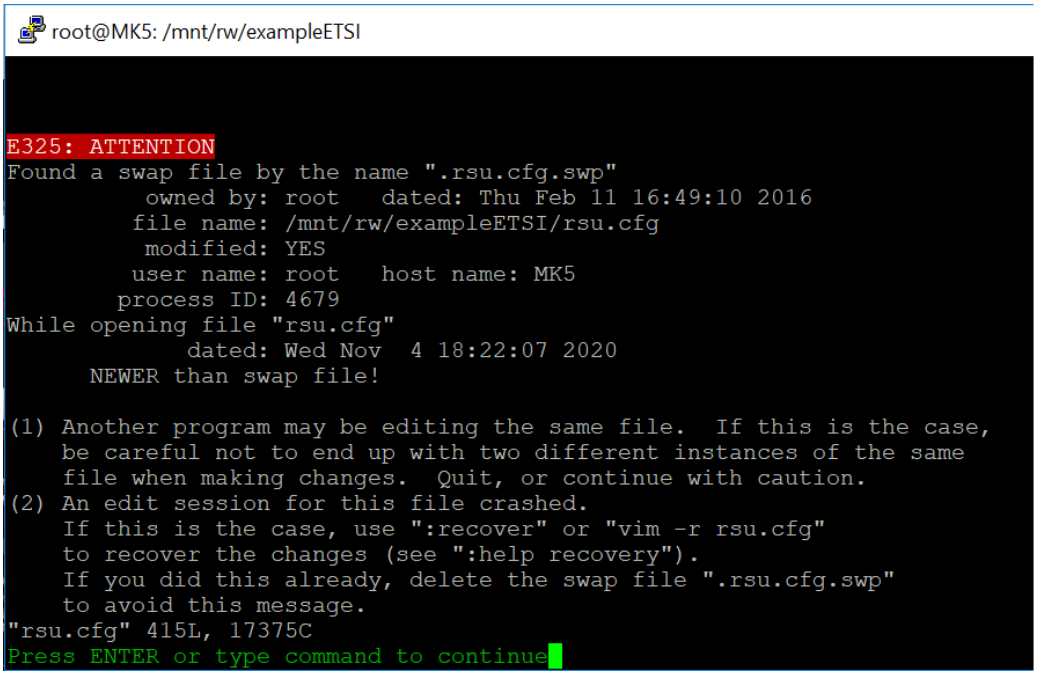
Eventuell hat sich beim Verändern der Attribute „`traceHistory`“, „`eventHistory`“, „`Event-Position`“ und anderer Attribute ein Syntax-Fehler eingeschlichen.

Lösung zu 2.4: Eventuellen Syntax-Fehler finden und beseitigen

Um eine lange Suche nach einem Syntax-Fehler abzukürzen, wird empfohlen eine Kopie der „unberührten“ ***rsu.cfg***-Datei von `/opt/cohda/application/` nach `/mnt/rw/exampleETSI/` zu kopieren & die Bearbeitung erneut gewissenhaft vornehmen. Eine Suche nach dem Syntax-Fehler kann, wenn gewünscht, aber auch erfolgen.

Problem 2.5: Found a swap file by the name „.rsu.cfg.swp“

Beim Bearbeiten einer Datei (z. B. die Konfigurationsdatei) kann es zu Problemen (z. B. Abstürzen des PC’s) kommen und es kann passieren, dass diese Datei nicht richtig abgespeichert werden konnte. Beim Versuch diese Datei erneut bearbeiten zu wollen, könnte folgende Meldung erscheinen:



Lösung zu 2.5: Swap-File finden & löschen

Einfach im Terminal auf der MK5 folgenden Befehl eingeben:

- `find . -type f -name "*.sw[klmnop]" -delete`

3. RSU ExampleETSI Start Errors

Beim Starten kann es manchmal zu folgendem Fehler kommen:

```
root@MK5:/mnt/rw/exampleETSI# ./rc.exampleETSI start rsu

Starting Example ETSI Application...
Reading app configuration from rsu.cfg...
Reading stack configuration from rsu.conf...
Reading extra stack configuration from /mnt/rw/exampleETSI/rsu.conf...
Overriding limits in configuration files...
[1615998034.367666337] App_Init FNSTART: (0x54b66e00)
[1615998034.373122337] App_Init: CANVSC3: 0
[1615998034.373262670] App_Init: CANVState: 0
[1615998034.373322337] App_Init: ITS: 1
[1615998034.373374337] App_Init: PosExt: 0
[1615998034.373664337] App_Init FNEND: () = 0
Initialising and starting stack components...
Conf: rsu.conf opened
Conf: ./board.conf opened
Conf: /mnt/rw/exampleETSI/rsu.conf opened
Conf: ./board.conf opened
Log directory is /mnt/src/log/2021.0317.1620_C04E54810BF84-0000_2298
/mnt/rw/exampleETSI
root@MK5:/mnt/rw/exampleETSI# libPlat Init...
libITSNet Init...
Security Test ERROR
libITSNet ERROR
libITSNet_Init() failed, returned -1
libITSNet Exit...
libPlat Exit...

root@MK5:/mnt/rw/exampleETSI#
```

Das bedeutet, dass die Sicherheit nicht richtig eingerichtet ist:

Lösung zu 3.:

Begehe dich zum /mnt/rw/exampleETSI Ordner & tippe folgende Befehle:

- ./rc.exampleETSI aerolink clear
- Danach starte die Anwendung neu: ./rc.exampleETSI start rsu

Literaturverzeichnis

AMEISE, Kommunikationsbüro Ulmer GmbH, Herausgeber (2021). *AMEISE Ganzheitliche Forschung zu den Potenzialen des autonomen Fahrens im ÖPNV*. Zuletzt aufgerufen: 04.06.2021. URL: https://ameise.wandelgesellschaft.de/.

Baldessari, Roberto u. a. (2007). "Car-2-car Communication Consortium Manifesto". In.

BMVI, Bundesministerium für Verkehr und digitale Infrastruktur, Herausgeber (2013). *Intelligente Verkehrssysteme im Straßenverkehr*. Zuletzt aufgerufen: 30.06.2021. URL: https://www.bmvi.de/SharedDocs/DE/Artikel/DG/ivs-im-strassenverkehr.htm.

Crow, Brian P u. a. (1997). "IEEE 802.11 wireless local area networks". In: *IEEE Communications magazine* 35.9, Seiten 116–126.

ETSI, ETSI EN (2010). "Intelligent Transport Systems (ITS); Communications Architecture". In: *European Standard (Telecommunications Series)*.

Kagermann Henning, Wolf-Dieter Lukas und Wolfgang Wahlster (2011). "Industrie 4.0: Mit dem Internet der Dinge auf dem Weg zur 4. industriellen Revolution". In: *VDI nachrichten* 13 (2011). Zuletzt aufgerufen: 30.06.2021, Seite 2. URL: https://pdfs.semanticscholar.org/5fbc/8571686c74516b4da38f6780c30fb31da2f0.pdf.

KG, Rohde & Schwarz GmbH & Co., Herausgeber (2021). *WLAN IEEE 802.11p testing*. Zuletzt aufgerufen: 04.06.2021. URL: https://www.rohde-schwarz.com/de/loesungen/test-and-measurement/wireless-communication/wireless-connectivity/wlan-wifi/wlan-ieee-802.11p-testing/wlan-ieee-802.11p-testing_250905.html.

Kleuker, Stephan (2013). *Grundkurs Software-Engineering mit UML: der pragmatische Weg zu erfolgreichen Softwareprojekten*. 3., korr. und erw. Aufl. Wiesbaden: Springer Vieweg, 2013. ISBN: 978-3-658-00641-9.

Kommission, Europäische (2016). *Eine europäische Strategie für Kooperative Intelligente Verkehrssysteme - ein Meilenstein auf dem Weg zu einer kooperativen, vernetzten und automatisierten Mobilität*. COM(2016) 766 final.

Malleck Helmut und Mecklenbräuker, Christoph (2015). "Die Digitalisierung des Verkehrs – Mobilität 4.0". In: *e & i Elektrotechnik und Informationstechnik* 132.7. zuletzt aufgerufen: 22.06.2021, Seiten 371–373. URL: https://link.springer.com/article/10.1007/s00502-015-0347-9.

Protzmann, Robert u. a. (2018).
"INTEGRIERTE BETRACHTUNG FAHRZEUGKOMMUNIKATION".
In: zuletzt aufgerufen: 03.07.2021.

Schneider, Jochen und Aynur Arslan (2007).
"Das Internet der Dinge unter dem Aspekt der Selbststeuerung - ein Überblick".
In: *Logistics Journals* 2007. Zuletzt aufgerufen: 30.06.2021, Seiten 1–2. URL:
http://www.logistics-journal.de/not-reviewed/2007/5/1067/schneider.pdf.

Wireless, Cohda, Herausgeber (2021). *V2X*. Zuletzt aufgerufen: 04.06.2021.
URL: https://www.cohdawireless.com/sectors/v2x/.

Wolter, Stefan (2012). *Zukünftige Entwicklungen in der Mobilität: betriebswirtschaftliche und technische Aspekte*. Herausgegeben von Proff Heike.
Wiesbaden: Springer/Gabler, 2012. ISBN: 978-3-8349-7117-3.