HOCHSCHULE MITTWEIDA
UNIVERSITY OF APPLIED SCIENCES

Bachelorarbeit

Herr
Yuchao Xu

Entwurf und Simulation einer Lichtsignalanlage sowie deren Car-to-X-Kommunikation unter Verwendung des Simulationswerkzeugs CANoe

Mittweida, 2023

Fakultät Ingenieurwissenschaften

Bachelorarbeit

Entwurf und Simulation einer Lichtsignalanlage sowie deren Car-to-X-Kommunikation unter Verwendung des Simulationswerkzeugs CANoe

Autor:
Herr

Yuchao Xu

Studiengang:
Elektro- und Informationstechnik

Seminargruppe:
EI19sA-BC

Erstprüfer:
Prof. Dr.-Ing. Jan Thomanek

Zweitprüfer:
M. Sc. Hongwei Xu

Einreichung:
Mittweida, 01.05.2023

Verteidigung/Bewertung:
Mittweida, 2023

Faculty of Engineering

BACHLOR THESIS

Design and simulation of a traffic signal system and its Car-to-x communication using the simulation tool CANoe

author:
Mr.

Yuchao Xu

course of studies:
Electrical and Information Technology

seminar group:
EI19sA-BC

first examiner:
Prof. Dr.-Ing. Jan Thomanek

second examiner:
M. Sc. Hongwei Xu

submission:
Mittweida, 01.05.2023

defence/ evaluation:
Mittweida, 2023

Bibliografische Beschreibung:

Xu, Yuchao:
Entwurf und Simulation einer Lichtsignalanlage sowie deren Car-to-X-Kommunikation unter Verwendung des Simulationswerkzeugs CANoe
Design and simulation of a traffic signal system and its Car-to-x communication using the simulation tool CANoe
Mittweida, Hochschule Mittweida, Fakultät Ingenieurwissenschaften,
Bachelorarbeit, 2023

Referat:

Diese Arbeit befasst sich mit der Ausbildung von Studenten und der Erstellung von praktischen Versuchen auf Basis der Simulationssoftware CANoe im Modul "CAR-2-CAR".

Inhalt

Abbildungsverzeichnis

Tabellenverzeichnis

Listingverzeichnis

Abkürzungsverzeichnis

AU	Application Unit
CAM	Cooperative Awareness Message
CAN	Controller Area Network
CAPL	Communication Access Programming Language
DCF	Distributed Coordination Function
DENM	Decentralized Environmental Notification Message
DSRC	Dedicated Short Range Communication
ETSI	European Telecommunications Standards Institute
GNSS	Global Navigation Satellite System
IEEE	Institute of Electrical and Electronics Engineers
LIN	Local Interconnect Network
LTE	Long Term Evolution
MAC	Medium Access
OBU	On Board Unit
OFDM	Orthogonalen Frequenzmultiplexverfahrens

PCF	Policy Control Function
PTP	Precision Time Protocol
RSU	Road Side Unit
SAE	Society of Automotive Engineers
SPAT	Signal Phase and Timing
UTC	Coordinated Universal Time
VANET	Vehicular Ad Hoc Networks
V2I	Vehicle to Infrastructure
V2N	Vehicle to Network
V2P	Vehicle to Pedestrian
V2V	Vehicle to Vehicle
V2X	Vehicle-to-Everything

1 Einleitung

In der Einleitung werden die Motivation und die Zielsetzung der Arbeit für diese Bachelorarbeit erläutert.

1.1 Motivation

Autonomes Fahren ist eine fortschrittliche Technologie, die es ermöglicht, dass Autos unter bestimmten Bedingungen völlig unbemannt fahren können. Es nutzt eine Vielzahl von Sensoren, Ortungssystemen und Computer-Vision-Technologie, um Informationen zu sammeln, die es ihm ermöglichen, Straßen, Hindernisse und andere Fahrzeuge genau zu erkennen und entsprechende Maßnahmen und Entscheidungen zu treffen. Darüber hinaus eröffnet die Technologie des autonomen Fahrens viele neue Möglichkeiten für künftige Automobilanwendungen und kann die Zahl der Autounfälle verringern und damit die Sicherheit im Straßenverkehr erhöhen.

Autonomes Fahren wird von der SAE (Society of Automotive Engineers) in sechs Stufen eingeteilt, L0 bis L5. Je höher die Zahl, desto höher der Reifegrad des autonomen Fahrens.(Siehe Abb.1)

SAE Level	SAE Name	SAE Narrative Definition	Execution of Steering/ Acceleration/ Deceleration	Monitoring of Driving Environment	Fallback Performance of Dynamic Driving Task	System capability (driving modes)	BASt Level	NTHSA Level
Human Driver monitors the driving environment								
0	No Automation	the full-time performance by *the human driver* of all aspects of the *dynamic driving task*	Human Driver	Human Driver	Human Driver	N/A	Driver only	0
1	Driver Assistance	the *driving mode-specific* execution by a driver assistance system of either steering or acceleration/deceleration	Human Driver and Systems	Human Driver	Human Driver	Some Driving Modes	Assisted	1
2	Partial Automation	Part-time or driving mode-dependent execution by **one or more driver assistance systems** of both steering and acceleration/deceleration. Human driver performs all other aspects of the *dynamic driving task*	**System**	Human Driver	Human Driver	Some Driving Modes	Partially Automated	2
Automated driving system ("system") monitors the driving environment								
3	Conditional Automation	*driving mode*-specific performance by an *automated driving system* of all aspects of the *dynamic driving task* - *human driver* **does** respond appropriately to a *request to intervene*	System	**System**	Human Driver	Some Driving Modes	Highly Automated	3
4	High Automation	*driving mode*-specific performance by an *automated driving system* of all aspects of the *dynamic driving task* - *human driver* **does not** respond appropriately to a *request to intervene*	System	System	**System**	Some Driving Modes	Fully Automated	3/4
5	Full Automation	**full-time** performance by an *automated driving system* of all aspects of the *dynamic driving task* under all roadway and environmental conditions that can be managed by a *human driver*	System	System	**System**	Some Driving Modes		

Abbildung 1: Stufen der Fahrzeugautonomie [Quelle: SAE Federal Highway Research Institute]

Level 0: 100% manuelles Fahren. Alle Aufgaben zum Führen des Fahrzeugs werden von Menschen ausgeführt.

Level 1: Assistiertes Fahren. Das Fahrzeug kann die Geschwindigkeitseinstellungen des Fahrers ausführen, z. B. den Tempomat(Künstlich eingestellte Geschwindigkeiten). Aber für das Lenken, den Spurwechsel usw. muss der Fahrer dies manuell tun.

Level 2: Teilautomatisierung. Die Stufe L2 kann als eine Steigerung der Stufe L1 angesehen werden, bei der das Fahrzeug unter bestimmten Bedingungen autonom beschleunigen oder abbremsen kann.

Level 3: Autonomes Fahren mit Bedingungen. Das Fahrzeug kann die meisten Fahraufgaben auf Straßen mit guten Straßenverhältnissen erfüllen. Der Fahrer muss zum richtigen Zeitpunkt eingreifen und das Führen des Fahrzeugs übernehmen.

Level 4: Hochgradig automatisiertes Fahren. Das Fahrzeug kann auf den meisten Straßen mit guten Straßenverhältnissen alle Fahraufgaben erledigen, ohne dass der Fahrer eingreifen und das Führen des Fahrzeugs übernehmen muss.

Level 5: 100% autonomes Fahren. Das Fahrzeug ist in der Lage, alle Fahraufgaben auf allen Straßen und in allen Umgebungen vollständig zu übernehmen. Der Fahrer braucht keine Maßnahmen zu ergreifen.

Fahrzeugsensorkomponenten sind elektronische Bauteile, die zur Umweltüberwachung und Karosseriesensorik eingesetzt werden. Sensoren zur Umgebungserfassung spielen eine entscheidende Rolle bei der Umsetzung der Technologie des autonomen Fahrens, und die Ingenieure statten die Fahrzeuge mit immer mehr Sensorkomponenten aus, um ein höheres Niveau des autonomen Fahrens zu erreichen.

Abbildung 2 wird verwendet, um zu zeigen, wie V2X (Vehicle-to-Everything) als Erweiterung der Fahrzeugsensoren im Betrieb eines Fahrzeugs funktioniert.

Abbildung 2: Umfassenderer Informationsaustausch über V2X[Quelle: AUTOCAR professional]

Wenn ein Fahrzeug in Bewegung ist, erfassen die Fahrzeugsensorkomponenten möglicherweise nicht alle Informationen über die Umgebung, oder die

Fahrzeugsensorkomponenten erkennen die erfassten Informationen über die Umgebung nicht. Die V2X-Technologie (Vehicle-to-Everything) ist jedoch eine Erweiterung der Fahrzeugsensoren, die es den Fahrzeugen ermöglicht, umfassendere und genauere Informationen über ihre Umgebung zu erhalten. Mithilfe der V2X-Technologie können Fahrzeuge mit anderen Systemen auf der Straße, Menschen, Infrastrukturen und Fahrzeugen interagieren, um umfassendere Echtzeitinformationen über ihre Umgebung zu erhalten und ihre Routen effektiv zu planen und Risiken zu vermeiden.

1.2 Zielsetzung der Arbeit

Im Rahmen dieser Bachelorarbeit wurde das Konzept und die Umsetzung eines Praktikumsversuches zur Car2X-Simulation in der Simulationssoftware CANoe vorgestellt. V2X (Vehicle-2-X), eine der Technologien zur Unterstützung intelligenter Fahrzeuge und intelligenter Mobilität, wird zur Erkennung der Fahrzeugumgebung und zur Durchführung von Sicherheitsbewertungen in Echtzeit eingesetzt, wodurch Verkehrsunfälle verringert, Verkehrsstaus reduziert und die Verkehrseffizienz verbessert werden. Die V2X-Technologie ist ein wichtiger Garant für die Verwirklichung des autonomen Fahrens. Das Hauptziel dieses praktischen Experiments ist, toolgestützte Praktikumsversuche für V2X-Anwendungen zu entwickeln. Die Ampelanlage und die Logik zur Beurteilung der Ampelanlage durch das Auto wurden mit der Programmiersprache CAPL implementiert und mit der Software CANoe und dem Signalkommunikationsinterface VN4610 simuliert, getestet und umgesetzt.

2 Grundlagen der CAR-2-X-Kommunikation

Dieses Kapitel beginnt mit einer Einführung in die Ziele und Anwendungsfälle der Car-2-X-Kommunikation, gefolgt von einer Beschreibung der Architektur und des Stacks der Car-2-X-Kommunikation, dann wird auf die Art und Weise der Übertragung und die Struktur der Nachrichten eingegangen, und schließlich wird das Entwicklungs- und Simulationswerkzeug CANoe vorgestellt.

2.1 Ziele der Car-2-X-Kommunikation

Ziel der Car-2-X-Kommunikation ist es, die Zahl der Verkehrsunfälle im Straßenverkehr zu verringern, die Sicherheit im Straßenverkehr zu erhöhen, Verkehrsstaus zu verringern, die Umweltverschmutzung zu reduzieren und neue Anwendungen für Fahrzeuge zu schaffen. Ihr oberstes Ziel war jedoch immer, die Sicherheit der Menschen zu gewährleisten.

Abbildung 3: Ziele der Car-2-X-Kommunikation: Sicherheit [Quelle: MOBILITÄT von MORGEN]

Wie in Abbildung 3 dargestellt, handelt es sich bei der Car-2-X-Kommunikationstechnologie um eine fortschrittliche Technologie, die den Informationsaustausch in Echtzeit zwischen Fahrzeugen, zwischen Fahrzeugen und Straßeninfrastruktur und sogar zwischen Fahrzeugen und Menschen ermöglicht. Mit der Car-2-X-Kommunikationstechnologie sind Fahrzeuge in der Lage, zeitnah Informationen über ihre Umgebung zu erhalten, z. B. über den Verkehr und den Straßenzustand, und auf diese Informationen zu reagieren. Die Car-2-X-Kommunikationstechnologie erweitert auch die Anwendungsmöglichkeiten für Fahrzeuge, z. B. können Echtzeitinformationen mit der Navigation kombiniert werden, um dem Fahrer Verkehrsinformationen zu liefern.

2.2 Anwendungsfälle der Car-2-X-Kommunikation

Die Car-2-X-Kommunikation kann auch als V2X (Vehicle-to-Everything) bezeichnet werden. Beide Bezeichnungen deuten darauf hin, dass das Fahrzeug Informationen mit etwas anderem in der Außenwelt austauscht. Wie die Abbildung 4 zeigt, gibt es derzeit vier gemeinsame Hauptkategorien: V2V (Vehicle to Vehicle) für die Kommunikation von Fahrzeug zu Fahrzeug, V2I (Vehicle to Infrastructure) für die Kommunikation von Fahrzeug zu Straßeninfrastruktur (z. B. die Interaktion zwischen dem Fahrzeug und der RSU), V2N(Vehicle to network)Fahrzeug zu Cloud Netzwerk Kommunikation und V2P (Vehicle to Pedestrian) für die Interaktion zwischen Fahrzeugen und Personen.

Abbildung 4: Allgemeine Arten der Car-2-X-Kommunikation[Quelle: Cohda Wireless]

2.2.1 Kommunikation zwischen Fahrzeugen

Die Car-2-X-Kommunikationstechnologie ermöglicht es den Fahrzeugen, miteinander zu kommunizieren, Warnmeldungen oder Straßeninformationen zwischen den Fahrzeugen auszutauschen usw. und ermöglicht es den Fahrern, den aktuellen Straßenzustand genauer vorherzusagen.

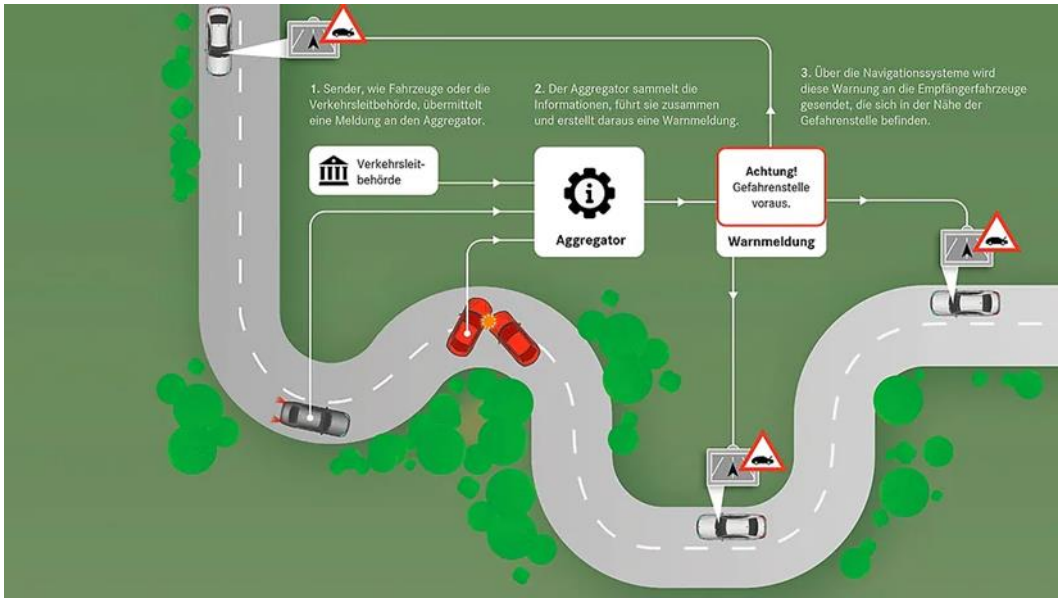

Abbildung 5: Anwendungsbeispiel Car-2-X-Kommunikation von Fahrzeug zu Fahrzeug[Quelle: Mercedes-Benz]

Im Falle einer Kollision mit einem Fahrzeug auf der mittleren Spur des Diagramms wird eine Warnmeldung gemeinsam von dem an der Kollision beteiligten Fahrzeug und dem Fahrzeug, das dem Vorfall ausgesetzt war, erstellt und anonym gesendet. (Siehe Abb.5)Die Nachricht enthält eine DENM- Nachricht mit der Art des Unfalls und eine CAM- Nachricht mit Zeit, Position, Richtung usw. Die Warnung wird über das Navigationssystem oder das Kommunikationssystem des Fahrzeugherstellers an das empfangende Fahrzeug in der Nähe des Gefahrenbereichs gesendet. Das Fahrzeug, das die Warnmeldung erhält, bestimmt die Relevanz des Ereignisses anhand seiner Position und Richtung und entscheidet, ob der Fahrer vor der Gefahr gewarnt werden muss. Gleichzeitig wird diese Warnmeldung über das empfangende Fahrzeug nach außen an Fahrzeuge weitergeleitet, die die Warnmeldung nicht erhalten haben. [MeBe]

Natürlich empfangen und senden auch die Fahrzeuge im normalen Verkehr ständig Nachrichten über die Car-2-X-Kommunikationstechnologie. Das Fahrzeug sammelt über seine eigene Sensorik kontinuierlich Informationen über seine Umgebung, integriert diese Informationen und sendet diese über die Car-2-X-Kommunikationstechnologie.

2.2.2 Kommunikation mit der Infrastruktur

Mit der zunehmenden Reife der Car-2-X-Kommunikationstechnologie wird diese Technologie auch in der Infrastruktur eingesetzt. Die Autofahrer werden im Voraus über Warnungen (z. B. aufgrund von Baustellen) informiert und können diese vermeiden, was zu ihrem Schutz und dem der im Warnbereich arbeitenden Personen beiträgt.

Wie in Abbildung 6 unten dargestellt, wird das "Traffic Light Information System" [AuMe] als V2I-Komponente auf dem Flachbildschirm angezeigt, und das System ist in der Lage,

anhand der Signalinformationen zu ermitteln, ob der Fahrer die Kreuzung überqueren kann, und die entsprechende Reaktion des Fahrers zu leiten. Dank der Informationen über die Ampeln können Autofahrer weitsichtiger sein. Dies wirkt sich insgesamt positiv auf den Verkehrsfluss aus.

Abbildung 6: Anwendungsbeispiel Car-2-X-Kommunikation zwischen Fahrzeug und Umgebung[Quelle:Audi]

2.3 Car2X-Systemarchitektur

In diesem Abschnitt werden die verschiedenen Komponenten der Car-2-X-Architektur kurz beschrieben. Für jede Komponente werden ihre wichtigsten Eigenschaften und Beziehungen zu anderen Komponenten beschrieben. Es folgt eine kurze Einführung in die EU-, US- und chinesischen Standards für den Car2X-Stack.

2.3.1 Komponenten und Domänen der Car2X-Architektur

Komponenten der Car-2-X-Architektur[HeMs 2007](Siehe Abb.7)

Application unit (AU):

Eine Application Unit (AU) ist eine interne Einheit im Fahrzeug, auf der Anwendungen laufen können, die auf die Kommunikationsfähigkeiten der OBU zugreifen. Die AU kann entweder fest in das Fahrzeug eingebaut (embedded) und dauerhaft mit der OBU verbunden sein oder dynamisch in das bordeigene Netz eingefügt und entfernt werden. Es ist ebenfalls möglich, dass mehrere AUs gleichzeitig an eine OBU angeschlossen werden und diese gemeinsam nutzen.

On-board unit (OBU):

Die On-Board-Unit (OBU) ist verantwortlich für die V2V-Kommunikation (Fahrzeug-zu-Fahrzeug) und V2I-Kommunikation (Fahrzeug-zu-Infrastruktur) und bietet zudem

Kommunikationsdienste für die Application Unit (AU) an. Eine OBU verfügt über mindestens ein Netzwerkgerät für drahtlose Kommunikation auf kurze Entfernungen, welches auf der IEEE 802.11p-Funktechnologie basiert. Dieses Netzwerkgerät ermöglicht das Senden, Empfangen und Weiterleiten von sicherheitsrelevanten Daten im Ad-hoc-Bereich.

Road-side units (RSU):

Eine Road Side Unit (RSU) ist ein physisches Gerät, welches sich an festen Standorten entlang von Straßen und Autobahnen befindet. Eine RSU verfügt über mindestens ein Netzwerkgerät für drahtlose Nahbereichskommunikation auf Basis der IEEE 802.11p-Funktechnologie. Es ist möglich, dass eine RSU auch mit anderen Netzwerkgeräten ausgestattet ist, um mit dem Infrastrukturnetz zu kommunizieren. In der Car2X-Architektur ist die Road Side Unit (RSU) für das Senden und Empfangen von Informationen zuständig und übernimmt ebenfalls die Weiterleitung von Informationen zwischen Fahrzeugen und dem übrigen System.

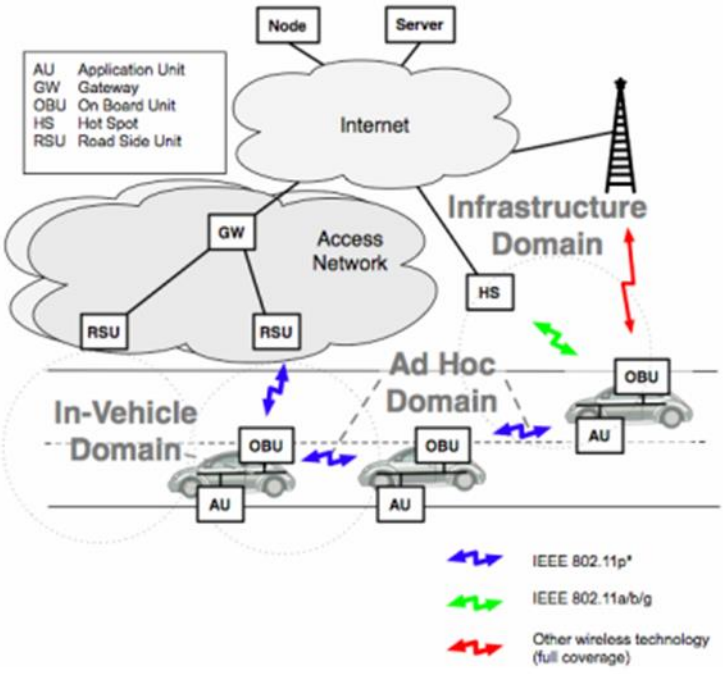

Abbildung 7: Architektur eines Car2X-Systems[HeMs 2007]

Im Car-2-X-Kommunikationssystem gibt es verschiedene **Domänen**:

Domäne: Vehicle

Das Fahrzeugdomänennetz setzt sich aus der On-Board-Unit (OBU) des Fahrzeugs und der Application Unit (AU) zusammen.

Domäne: Ad-Hoc[HeMs 2007]

Die Ad-Hoc-Domäne, auch bekannt als Vehicular Ad Hoc Network (VANET), besteht aus Fahrzeugen, die mit On-Board-Units (OBUs) und Road Side Units (RSUs) ausgestattet sind. Im VANET-Netz können Fahrzeuge direkt miteinander kommunizieren, ohne dass eine zentrale Koordinierungsinstanz erforderlich ist. Nachrichten können entweder direkt von einer OBU zu einer anderen gesendet werden oder durch Hopping zwischen mehreren OBU-ausgerüsteten Fahrzeugen übermittelt werden. Die RSU erhöht die Verkehrssicherheit, indem sie Daten im VANET-Netzwerk weiterleitet und empfängt und das Netzwerk während der Fahrt erweitert. Gleichzeitig kann die RSU Verkehrsmeldungen oder Unfallmeldungen empfangen. Wenn ein mit einer OBU ausgestattetes Fahrzeug in den Signalbereich der RSU gelangt, leitet die RSU die empfangene Nachricht an die OBU weiter.

Domäne: Infrastructure[PaSp 2022]

Dieser Bereich umfasst RSUs und andere Netzwerkeinrichtungen wie Hotspots, die es der OBU ermöglichen, auf verschiedene Weise auf die Infrastruktur zuzugreifen.

2.3.2 Car2X- Stack

Der Car-2X-Protokollstack ist eine Software-Protokollsammlung, die in der Car2X-Architektur für die Implementierung der V2X-Kommunikation verwendet wird. Er besteht aus verschiedenen Schichten, die für den reibungslosen Betrieb des Systems verantwortlich sind. Diese Schichten umfassen die physikalische Schicht, Netzwerkschicht, Anwendungsschicht und Sicherheitsschicht. Die physikalische Schicht ermöglicht die Übertragung und den Empfang von Informationen zwischen dem Fahrzeug und anderen Einheiten der Car2X-Architektur. Die Netzwerkschicht ist für die Weiterleitung von Informationen zwischen den Einheiten der Car2X-Architektur verantwortlich. Die Anwendungsschicht verarbeitet Informationen und bietet Funktionen wie Verkehrsinformationen, Navigationshilfe und Verkehrsführung an. Die Sicherheitsschicht gewährleistet, dass die Informationen sicher sind und nur für den vorgesehenen Empfänger zugänglich sind.

CANoe.Car2x unterstützt derzeit drei V2X-Protokollstandards:

- EU-V2X-Protokollstandards - ETSI ITS G5
- USA-V2X-Protokollstandards - IEEE 1609 WAVE
- China-V2X-Protokollstandards -V2X

2.3.2.1 EU-V2X-Protokollstandards - ETSI ITS G5[JaTh 2021/2022](Siehe Abb.8)

Der ETSI C-ITS G5 Protokollstapel ist eine Sammlung von Kommunikationsprotokollen, die vom Europäischen Institut für Telekommunikationsnormen (ETSI) entwickelt wurden. Es

basiert auf dem WLAN-Standard IEEE 802.11p und beinhaltet auch den Nachfolgestandard IEEE 802.11bd, der die V2X-Kommunikation standardisiert.

ETSI C-ITS G5 hat sechs Protokollschichten:

Access: Diese Schicht nutzt drahtlose Kommunikationstechnologien wie IEEE 802.11p, welches für europäische Regionen angepasst wurde, und agiert auf den OSI-Schichten 1 und 2.

Networking & Transport: Diese Schicht übermittelt CAM- und DENM-Nachrichten und agiert auf den OSI-Schichten 3 und 4.

Facilities: Diese Schicht definiert Informationen wie CAM und DENM und erstellt Nachrichten, die an andere Fahrzeuge und Infrastrukturkomponenten gesendet werden. Sie agiert auf den OSI-Schichten 5, 6 und 7.

Applications: Diese Schicht spezifiziert die Verwaltung von IVS-Anwendungen, die in Verkehrssicherheit, Verkehrseffizienz und andere Anwendungen unterteilt sind.

Management: Diese Ebene ist für die Verwaltung der verschiedenen Anwendungen und der Kanalnutzung zuständig.

Security: Es werden sicherheitsbezogene Dienste für die Kommunikation angeboten.

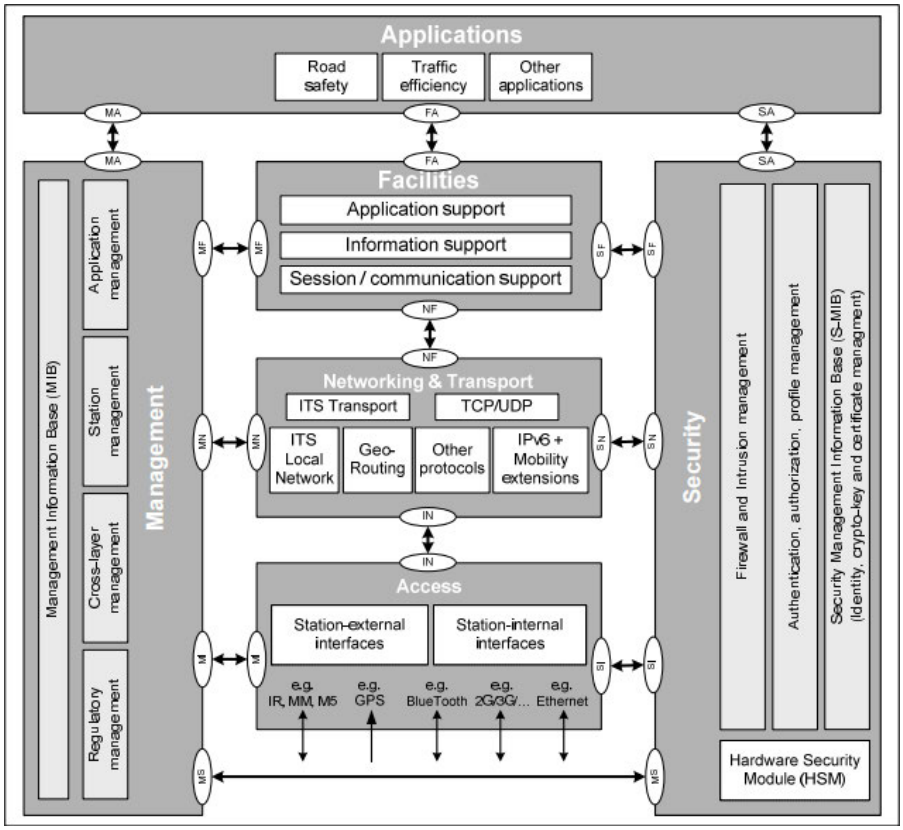

Abbildung 8: Protokollstack des ETSI ITS G5[Quelle: ResearchGate]

2.3.2.2 USA-V2X-Protokollstandards - IEEE 1609 WAVE[FuWi 2022]:

Der IEEE 1609 Protokollstack ist ein US-spezifischer Protokollstack, der auf dem von der IEEE entwickelten Kommunikationsprotokoll IEEE 802.11p basiert.(Siehe Abb.9)

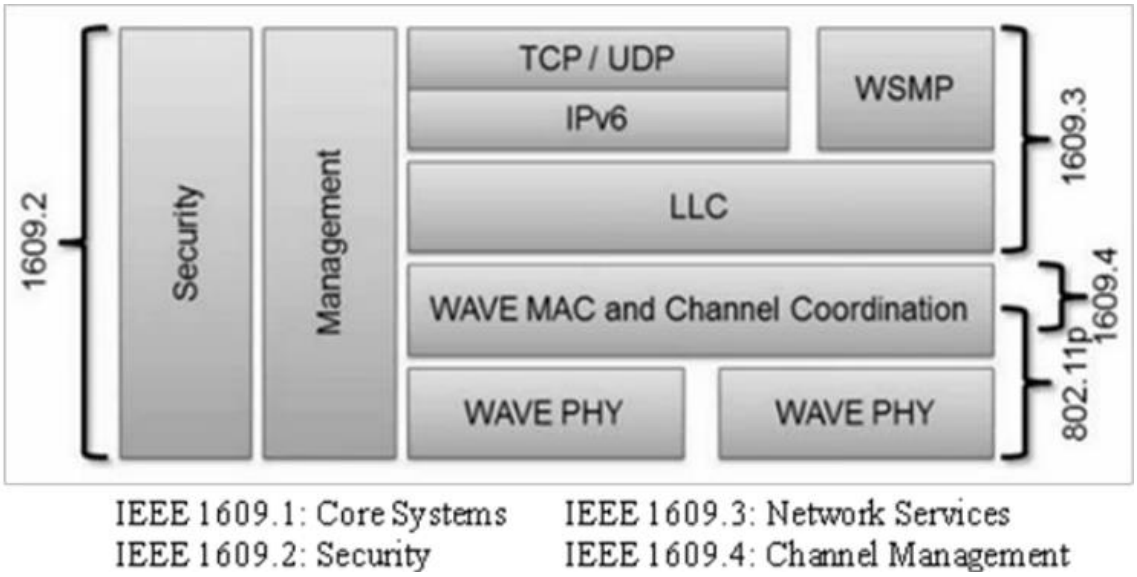

Abbildung 9: Protokollstack des IEEE 1609 WAVE[Quelle: ResearchGate]

Die IEEE 1609.x-Norm für WAVE standardisiert Kommunikationsprotokolle zwischen OBUs und RSUs sowie zwischen verschiedenen OBUs in einer vernetzten Fahrzeugumgebung. Mit einer Reihe genormter Dienstschnittstellen ist sie speziell auf Anwendungen wie Fahrzeugsicherheit, Verkehrsmanagement, dynamische Kartierung, Navigation und Positionierung ausgerichtet. Ihr Hauptzweck besteht darin, eine Kommunikationssystemarchitektur zu definieren und Kommunikationsstandards bereitzustellen, die in der vernetzten Fahrzeugumgebung von entscheidender Bedeutung sind.

IEEE 1609.1: IEEE 1609.1 ist eine Spezifikation im IEEE 1609-Standard für Wireless Access in Vehicular Environments (WAVE), die den Kontrollprozess zwischen Remote-Anwendungen und Ressourcenmanagement definiert.

IEEE 1609.2: IEEE 1609.2 ist eine Norm für die Sicherheitsdienste für WAVE (Wireless Access in Vehicular Environments), die spezifische Anforderungen für die Absicherung der drahtlosen Kommunikation in vernetzten Fahrzeugumgebungen definiert. Diese Norm beschreibt die notwendigen kryptografischen Techniken und Sicherheitsmechanismen, die zum Schutz der Vertraulichkeit, Integrität und Authentizität von Nachrichten in der vernetzten Fahrzeugumgebung eingesetzt werden müssen.

IEEE 1609.3: IEEE 1609.3 spezifiziert die Dienststandards für die Netztransportschicht, welche den Verbindungsaufbau und die Verwaltung von WAVE-Netzwerken regelt. Es werden zwei parallele Netztransportkanäle bereitgestellt: User Datagram Protocol (UDP) IPv6 und WSMP.

IEEE 1609.4: IEEE 1609.4 ist eine Norm des Institute of Electrical and Electronics Engineers (IEEE), die die Spezifikationen für die "Multichannel-Operation" in drahtlosen Kommunikationssystemen für vernetzte Fahrzeuge definiert. Es beschreibt die Kanalzugriffssteuerung (Channel Access Control, CAC) und die Multichannel-Operationen, um eine zuverlässige und effiziente Kommunikation in vernetzten Fahrzeugumgebungen zu ermöglichen. Die Norm definiert auch den Betrieb von parallelen Kanälen, um kritische Anwendungen von nicht-kritischen Anwendungen zu trennen und die Gesamtleistung des Systems zu verbessern.

2.3.2.3 China-V2X-Protokollstandards -V2X[YaLi 2019]:

Am 17. August 2020 präsentierte das National Automotive Standardization Technical Committee of China's Intelligent Networked Vehicle Branch das nationale Standardprojekt "Technische Anforderungen an das In-Vehicle Information Interaction System auf Basis der LTE-V2X Direct Connect Communication". Es zielt darauf ab, eine branchenübergreifende Koordination von C-V2X sicherzustellen. (Siehe Abb.10)

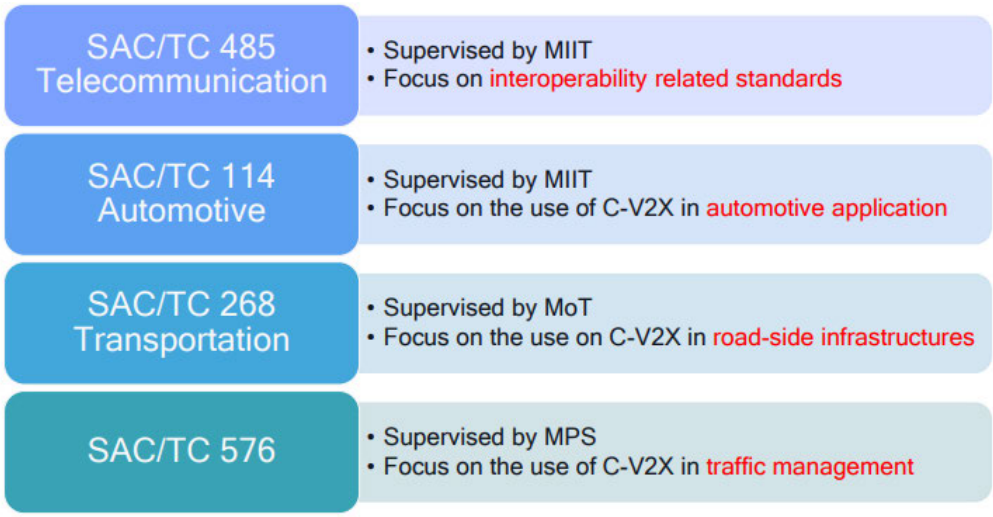

Abbildung 10: Nationales Normungsprojekt über "Technische Anforderungen an das bordeigene Informations-Interaktionssystem auf der Grundlage der Direktverbindungskommunikation LTE-V2X[YaLi 2019]

SAC/TC 485: SAC/TC 485 ist das Nationale Technische Komitee für die Normung der Kommunikation, das am 15. Mai 2009 gegründet wurde und hauptsächlich für die Überarbeitung von Protokollen und anderen damit verbundenen Normen zuständig ist.

SAC/TC 114: Nr. SAC/TC 114 ist ein untergeordnetes technisches Komitee des Nationalen Technischen Komitees für die Automobilnormung für intelligente vernetzte Fahrzeuge, das am 26. Dezember 2017 gegründet wurde und sich auf die Entwicklung von fahrzeuginternen Informationsdiensten im Zusammenhang mit C-V2X-Kommunikationsanwendungen konzentriert.

SAC/TC 268: Nr. SAC/TC 268 wird als Nationales Technisches Komitee für die Normung intelligenter Verkehrssysteme bezeichnet und wurde im September 2003 mit der Hauptverantwortung für die Entwicklung von Infrastrukturdiensten im Zusammenhang mit C-V2X-Kommunikationsanwendungen gegründet.

SAC/TC 576: Das Nationale Technische Komitee für die Standardisierung des Straßenverkehrsmanagements steht unter der Aufsicht des Ministeriums für öffentliche Sicherheit und ist hauptsächlich für die Anwendung von C-V2X im Verkehrsmanagement zuständig.

Der V2X-Protokollstapel besteht aus vier Schichten: Zugriffs-, Netzwerk-, Nachrichten- und Anwendungsschicht. Die Übertragung und Empfang von V2X-Nachrichten erfolgt in der Zugriffs- und Netzwerkschicht. Die Nachrichtenübertragungsschicht ist für die Codierung, Decodierung, Datenfusion und Verarbeitung von V2X-Nachrichten zuständig. Die Anwendungsschicht ermöglicht die logische Datenverarbeitung, gemeinsame Steuerung des autonomen Fahrens, Frühwarnübertragung und Mensch-Maschine-Interaktion in V2X-Szenarien.

2.4 Car2X-Übertragungstechnologie- IEEE 802.11p

In diesem Abschnitt werden eine Car2X-Übertragungstechnologie beschrieben: DSRC (Dedicated Short Range Communication, d.h. IEEE 802.11p). IEEE 802.11p ist eine Spezifikation für drahtlose Kommunikation, die vom Institute of Electrical and Electronics Engineers (IEEE) für die Kommunikation zwischen Fahrzeugen und Fahrzeugen sowie zwischen Fahrzeugen und Verkehrsinfrastruktur entwickelt wurde. IEEE 802.11p definiert die Physical Layer und die MAC Layer (Medium Access Layer) für die drahtlose Kommunikation zwischen Fahrzeugen und anderen Einheiten.

2.4.1 Physical Layer

Eine Bandbreite von 75 MHz im 5,9-GHz-Band ist als exklusives DSRC-Verkehrssicherheitsband ausgewiesen. 75 MHz der Bandbreite sind in 7 Kanäle unterteilt.(Siehe Abb.11) Der Kontrollkanal wird als CCH bezeichnet, die Servicekanäle als SCH.

Durch den IEEE 802.11p-Standard sind Fahrzeuge in der Lage:

- Die angestrebte Kommunikationsreichweite beträgt 1000 Meter.
- Im CH 172 interagieren die Fahrzeuge mit grundlegenden Sicherheitsinformationen. Notfallmeldungen werden mit höherer Priorität in CH 184 verbreitet.

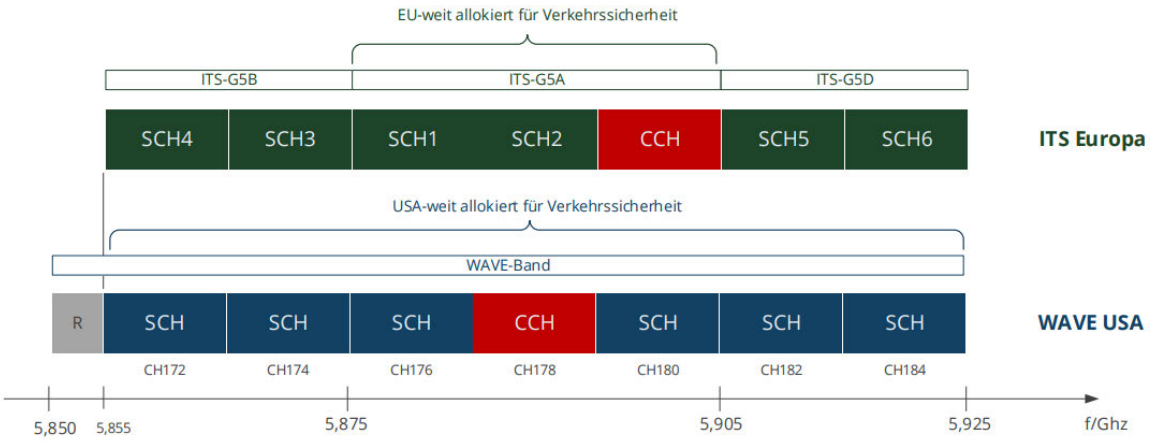

Abbildung 11: Frequenzbereiche für ITS-Applikation in Europa und USA[JaTh 2021/2022]

Diese Anforderungen führten zur entsprechenden Spezifikation IEEE 802.11p[JaTh 2021/2022]:

- Entsprechend der Frequenzbänder 7 x 10 MHz → Halbierung der Datenrate gegenüber IEEE 802.11a (20 MHz)
- Verwendung des Orthogonalen Frequenzmultiplexverfahrens (OFDM) (Siehe Abb. 12)
  - 48 Datenträger und 4 Pilotträger
  - mit je 156,25 kHz Bandbreite
  - Symboldauer 8 µs
  - Guard-Intervall 1,6 µs → Verdopplung gegenüber IEEE 802.11a
- Modulationsverfahren der Subträger: BPSK, QPSK, 16-QAM, 64-QAM
- Maximale Übetragungsleistung 33 dBm (Vergleich 2,4 GHz WLAN/Bluetooth: 20 dBm)

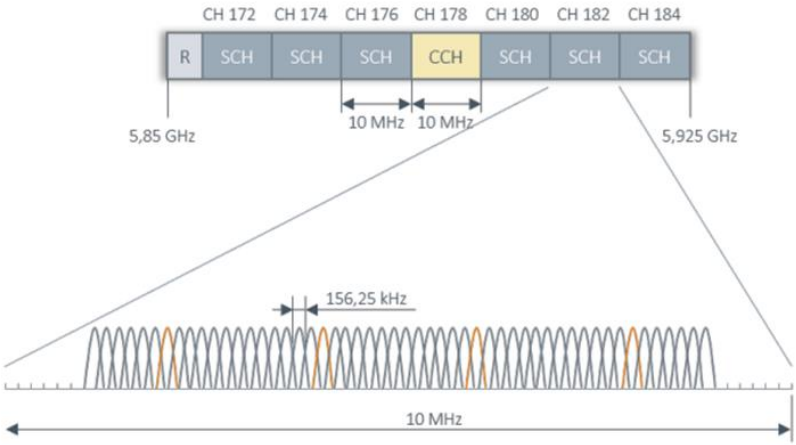

Abbildung 12: OFDM bei IEEE 802.11p[JaTh 2021/2022]

In Abhängigkeit der genutzten Konfiguration ergeben sich acht verschiedene Transferraten (Siehe Abb.13).

Mbit/s	Modulation	Coding	Bits/Subträger	Cod. Bits/Symbol	Datenbits/Symbol
3	BPSK	1/2	1	48	24
4,5	BPSK	3/4	1	48	36
6	QPSK	1/2	2	96	48
9	QPSK	3/4	2	96	72
12	16-QAM	1/2	4	192	96
18	16-QAM	3/4	4	192	144
24	64-QAM	2/3	6	288	192
27	64-QAM	3/4	6	288	216

Abbildung 13: Mögliche Transferraten bei IEEE 802.11p[JaTh 2021/2022]

2.4.2 MAC Layer

Der IEEE 802.11p-Standard spezifiziert eine gemeinsame Medienzugangsschicht.Die MAC Layer ist eine der sieben Schichten des OSI-Referenzmodells und bietet Dienste für die Anwendungen, die auf der darüber liegenden Schicht ausgeführt werden.

IEEE802.11p definiert zwei Formen des Medienzugriffs[LiMa 2013]:

DCF (Distributed Coordination Function) basiert auf dem CSMA/CA-Protokoll (Carrier Supervised Multiple Access/Collision Prevention). Im DFC-Modus darf die Station keine Informationen übertragen, während andere Stationen Funkrahmen senden. Wenn andere Stationen senden müssen, müssen sie warten, bis der Kanal frei wird.

DCF ist auch mit einem "Backoff"-Algorithmus ausgestattet, der dazu beiträgt, den Zugriff auf das Medium auf eine faire Weise zu gewährleisten. Wenn eine Station den Kanal benutzt, müssen andere Stationen eine zufällige Zeitspanne warten, bevor sie auf das

Medium zugreifen können. Dadurch wird sichergestellt, dass nicht mehrere Stationen, die Daten senden wollen, gleichzeitig eintreffen. Dadurch wird die Zahl der Konflikte bei drahtlosen Übertragungen erheblich reduziert, vor allem, wenn es mehr Nutzer gibt.

PCF (Point Coordination Function) ist ein optionaler Mechanismus im IEEE 802.11 Wi-Fi-Standard, der die Leistung und Effizienz des Zugriffs auf das Medium verbessert.

Im Gegensatz zur Distributed Coordination Function (DCF) bei der alle Stationen gleichberechtigt sind, ermöglicht die PCF-Technologie, dass eine bestimmte drahtlose Station, die als Point Coordinator (PC) bezeichnet wird, das Medium für eine bestimmte Zeitperiode kontrolliert und die Zugriffsrechte für andere Stationen verwaltet. Die PCF-Funktion wird normalerweise in Umgebungen mit hohem Datenverkehr oder in Netzwerken eingesetzt, bei denen zeitkritische Anwendungen priorisiert werden müssen.

2.5 Car2X-Botschaften

Zur Erleichterung der Informationsinteraktion zwischen Fahrzeugen sowie zwischen Fahrzeugen und Infrastruktur haben ETSI und SAE/IEEE Nachrichten wie CAM und SPAT definiert. Durch die Verwendung dieser Nachrichten wird eine effektive Kommunikation zwischen Fahrzeugen und Infrastruktur ermöglicht, was zur Verbesserung der Verkehrssicherheit beiträgt.

2.5.1 CAM

CAM (Cooperative Awareness Message) [AnFe 2014] ist eine zeitgesteuerte Nachricht. Sie wird zyklisch von der OUB oder RSU gesendet. Diese Art von Signalen enthält Informationen über die Position, Geschwindigkeit und Richtung des Fahrzeugs.

Eine CAM-Nachricht besteht aus einem ITS Protocol Data Unit (PDU) header und mehreren Containern.(Siehe Abb.14) Die ITS-PDU-Header enthalten die Protokollversion, den Nachrichtentyp und die Absenderadresse. Ein Basiscontainer enthält hauptsächlich den Typ der ITS-Station und Positioninformationen. Ein Hochfrequenz-Container enthält Informationen über den Kurs, die Geschwindigkeit und die Beschleunigung des Fahrzeugs. Der Niederfrequenz-Container enthält hauptsächlich Inhalte, die für die Fahrzeugsicherheit

weniger relevant sind. Der Spezialcontainer eignet sich für den Versand spezieller Fahrzeugtypen wie Busse, Rettungsfahrzeuge usw.

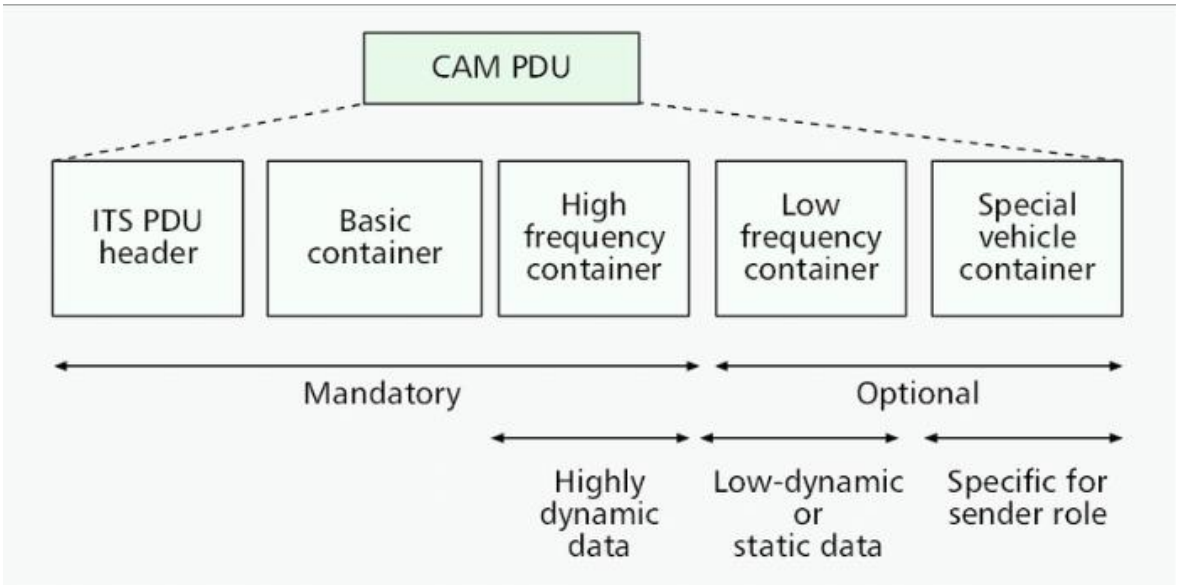

Abbildung 14: CAM-Nachrichtenstruktur[AnFe 2014]

2.5.2 DENM

Die DENM-Nachricht (Decentralised Environmental Notification Message) [AnFe 2014] wird in erster Linie zur Frühwarnung vor möglichen Gefahren im Straßenverkehr verwendet. Die Meldung wird von der OBU oder RSU ausgelöst und weitergeleitet, wenn das Fahrzeug ein gefährliches Ereignis auf der Straße feststellt.

Die DENM-Nachricht besteht aus einem ITS Protocol Data Unit (PDU) Header und mehreren Containern. (Siehe Abb.15) Es ist zu beachten, dass nur der ITS Protocol Data Unit (PDU) Header und der Management Container obligatorisch sind. Alle anderen sind optional. Der ITS Protocol Data Unit (PDU) Header enthält auch die Protokollversion, den Nachrichtentyp und Absenderinformationen. Der Management Container regelt den Inhalt der Nachrichten. Der Situation Container enthält die Ereignisinformationen. Der Location Container enthält den Ort, an dem das Ereignis stattgefunden hat. Der Menu Container enthält zusätzliche Informationen, die nicht in den ersten drei Containern enthalten sind.

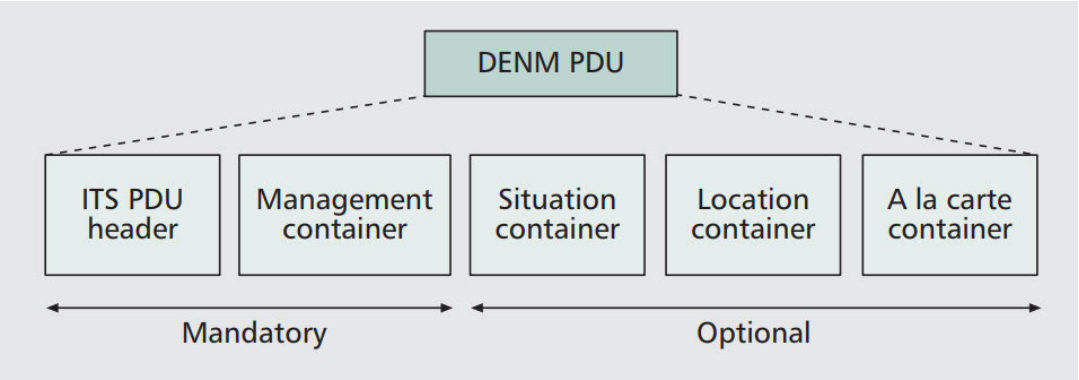

Abbildung 15: DENM-Nachrichtenstruktur[AnFe 2014]

2.5.3 SPAT

Die SPAT- Nachricht (Signal Phase and Timing Message) wird von der RSU in Bezug auf die Phase und Dauer der Ampelschaltung gesendet. Normalerweise werden SPAT-Nachrichten jede 100 ms gesendet.

SPAT-Nachrichten werden in einem schichtweisen Format verschachtelt.(Siehe Abb.16)

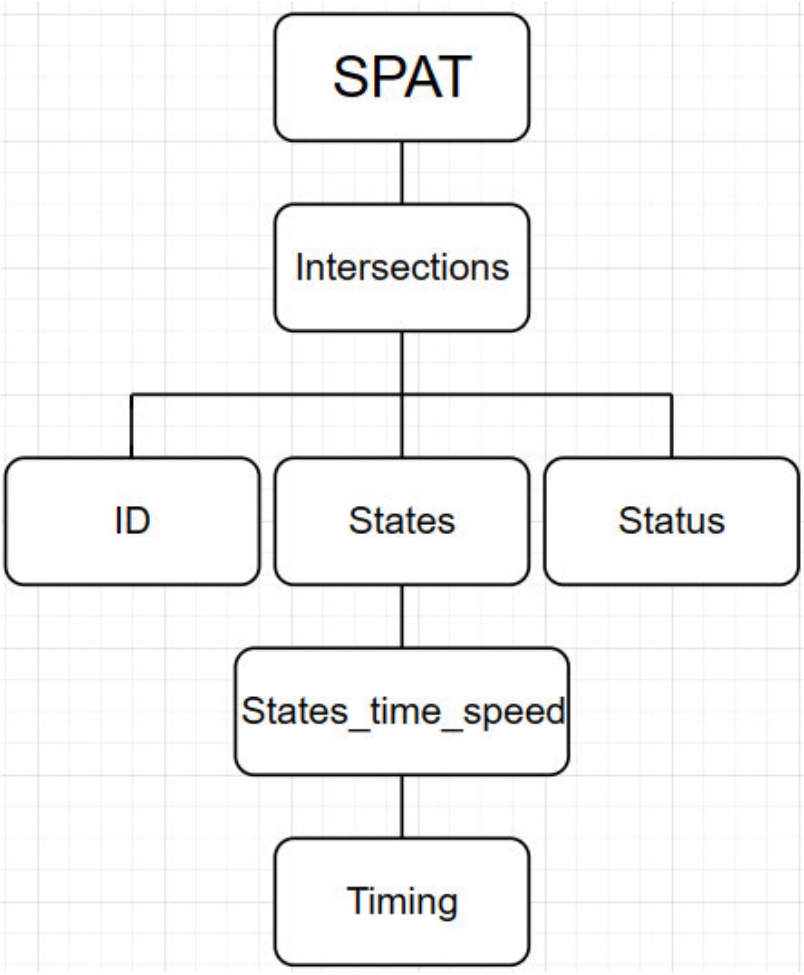

Abbildung 16: SPAT -Nachrichtenstruktur

Die Hauptstruktur der SPAT-Nachrichten besteht hauptsächlich aus[YiXi 2022]:

Intersections: Es befindet sich auf der ersten Ebene und kann so verstanden werden, dass es eine Reihe von Informationen über das Verkehrssignalsystem enthält.

Status： Es befindet sich auf der zweiten Ebene und dient der Anzeige des Status der gesamten Lichtsignalanlage.

States： Es befindet sich auf der zweiten Ebene und enthält Informationen zu den einzelnen Signalgruppen. (z.B. Dauer, aktueller Status, etc.)

States_time_speed: Es befindet sich auf der dritten Ebene und in States. Es wird verwendet, um den aktuellen Status und die Dauer des Lichtsignals anzuzeigen.

Timing: Es befindet sich auf der vierten Ebene und in States_time_speed. Es wird verwendet, um die Dauer des Signals anzuzeigen. Es enthält Elemente wie startTime, minEndTime, maxEndTime.

StartTime: Es gibt den Zeitpunkt an, zu dem der Phasenzustand das nächste Mal beginnt. Befindet es sich in dieser Phase, ist der Wert 0.

MinEndTime: Es gibt die Mindestzeit bis zum Ende des Zustands zum aktuellen Zeitpunkt an.

MaxEndTime: Es gibt die maximale Zeit bis zum Ende des Zustands zum aktuellen Zeitpunkt an.

2.5.4 MAP

Die MAP-Nachricht (Map Data) wird ebenfalls von der RSU gesendet und begleitet normalerweise die SPAT-Nachricht. Diese Nachricht wird verwendet, um die Gesamtstruktur der Ampel zu beschreiben (z.B. Informationen zur Topologie und -geometrie von Straßenabschnitten und Kreuzungnen).

MAP-Nachrichten werden in einem schichtweisen Format verschachtelt.(Siehe Abb.17)

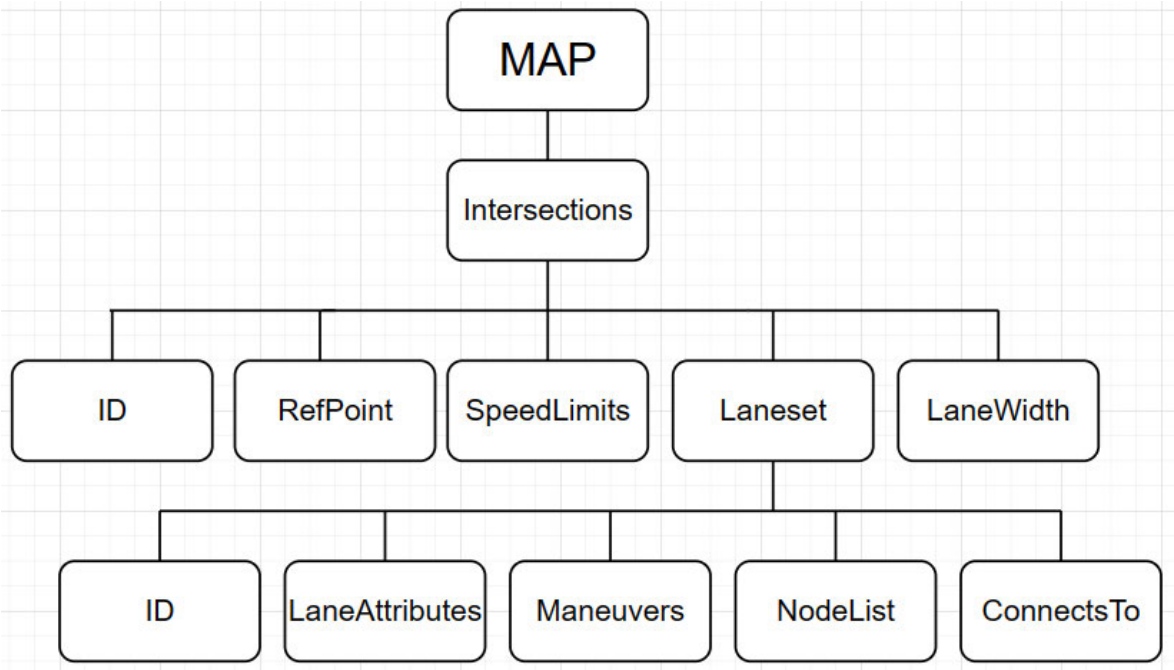

Abbildung 17: MAP-Nachrichtenstruktur

Die Hauptstruktur des MAP-Nachrichten besteht hauptsächlich aus:

Intersections: Es befindet sich auf der ersten Ebene und kann so verstanden werden, dass es eine Reihe von Informationen über das Verkehrssignalsystem enthält.

RefPoint: Es befindet sich auf der zweiten Ebene, die dazu dient, die Position dieser Lichtsignalanlage anzugeben.

LaneWidth: Es befindet sich auf der zweiten Ebene und dient dazu, die Breite dieser Straße zu bestimmen.

SpeedLimits: Es befindet sich auf der zweiten Ebene und dient zur Information über die Geschwindigkeitsbegrenzung auf dieser Straße.

Laneset: Es befindet sich auf der zweiten Ebene und dient dazu, Informationen über die Spuren zu enthalten.

LaneAttributes: Es befindet sich auf der dritten Ebene und ist in der LaneSet. In LaneAttributes können unterschieden werden, ob es sich um eine Einfahrtsspur oder eine Ausfahrtsspur handelt.

Maneuvers: Es befindet sich auf der dritten Ebene und ist in der LaneSet. Er wird verwendet, um Fahrspuren für das Geradeausfahren oder Abbiegen zu definieren.

NodeList: Es befindet sich auf der dritten Ebene und ist in der LaneSet. Es wird verwendet, um die beiden Punkte darzustellen, an denen die Fahrspur gezeichnet wird.

ConnectsTo: Es befindet sich auf der dritten Ebene und ist in der LaneSet. Es wird verwendet, um die mit dieser Einfahrspur verbundene Ausfahrspur anzugeben.

2.6 Entwicklungs- und Simulationswerkzeug CANoe mit Car2X-Option

Dieser Kapitel gibt eine kurze Einführung in das Entwicklungssimulationswerkzeug CANoe und das darin enthaltene CAR-2-X Modul.

2.6.1 CANoe-Software

CANoe ist ein äußerst vielseitiges Software-Tool, das den gesamten Prozess der Projektentwicklung von der Anforderungsanalyse bis zur Projektrealisierung unterstützt. Mit seiner umfangreichen Funktionalität und anforderungsgerechten Konfiguration ist es sowohl für die Entwicklung und den Test einzelner Steuergeräte als auch für die Analyse und den Test ganzer Netzwerke geeignet. Aus diesem Grund ist CANoe bei einer breiten Palette von Netzwerkdesignern, Entwicklungs- und Testingenieuren sehr beliebt. [Vector1]

2.6.2 CAR-2-X-Modul

Die V2X-Kommunikationstechnologie wird immer häufiger in der Fahrzeugkommunikation eingesetzt. Immer mehr Verkehrsteilnehmer wie Fahrzeug-OBUs und Straßeninfrastrukturen wie RSUs schließen sich dem auf der V2X-Kommunikationstechnologie basierenden Straßenkommunikationsnetz an. Das Modul CANoe.car2x spielt eine entscheidende Rolle bei der Entwicklung dieser Technologie. Das Car2x-Modul unterstützt eine Vielzahl von chinesischen, EU- und US-Regionalstandards sowie Protokollen, wodurch eine umfassende Analyse von Nachrichten ermöglicht wird. In Abbildung 18 werden im Trace-Fenster die Signaltypen angezeigt, die von den Fahrzeugen und der Straßeninfrastruktur gesendet werden, wobei auf der rechten Seite die in jedem Signal enthaltenen Informationen dargestellt werden.

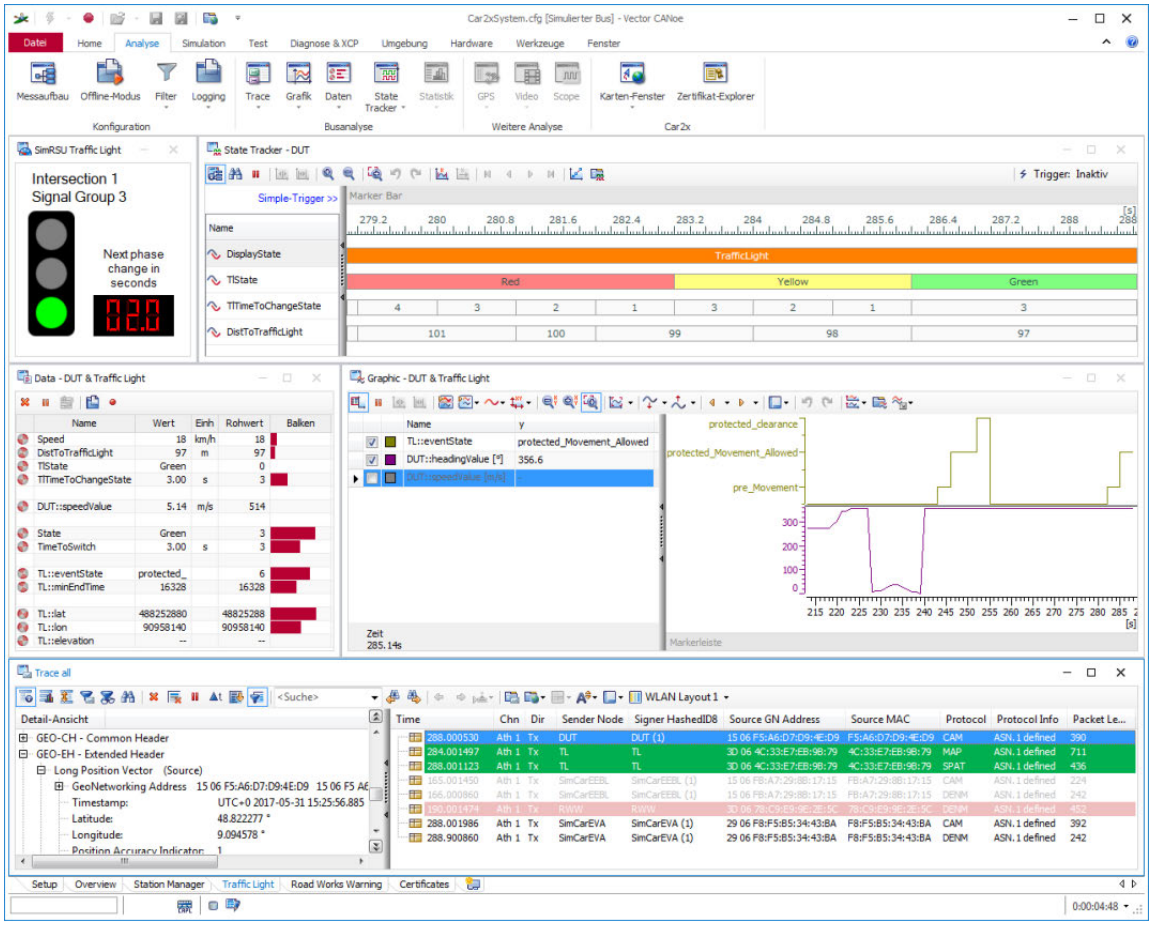

Abbildung 18: CANoe mit der Car2x option[Quelle:Vector]

2.6.3 Einführung in die Funktionsbereiche

Der Funktionsbereich des Moduls CANoe.car2x ist in 10 Hauptbereiche unterteilt：

File： Auf der Seite Datei können ein Projekt erstellt oder geladen oder die VECTOR-Website besucht werden.

Home: Der Funktionsbereich Home ist in drei Komponenten unterteilt: die Komponente „Measurement“, die Komponente „Appearance“ und die Komponente „More“. Die Hauptfunktionen der komponente „Measurement“ sind das Starten oder Stoppen des Projekts, das Umschalten zwischen der Umgebung, in der das Projekt läuft, dem Online- oder Offline-Modus und dem Real- oder Simulationsbus. （Die Hauptfunktion der Anzeigekomponente besteht darin, das Format der Zahlen und die Anzeige der Meldungen im Trace-Fenster einzustellen. Das Write-Fenster wird verwendet, um den Live-Status des Projekts anzuzeigen, und das Panel-Fenster wird hauptsächlich zum Hinzufügen oder Erstellen von Panels verwendet.

Analysis: Das Band Analyse ist in drei Komponenten unterteilt: die Konfigurationskomponente, die Busanalysekomponente und die weitere

Analysekomponente. Der wichtigste Teil dieser Multifunktionsleiste ist die Schaltfläche „Measurement Setup". (Siehe Abb.19)

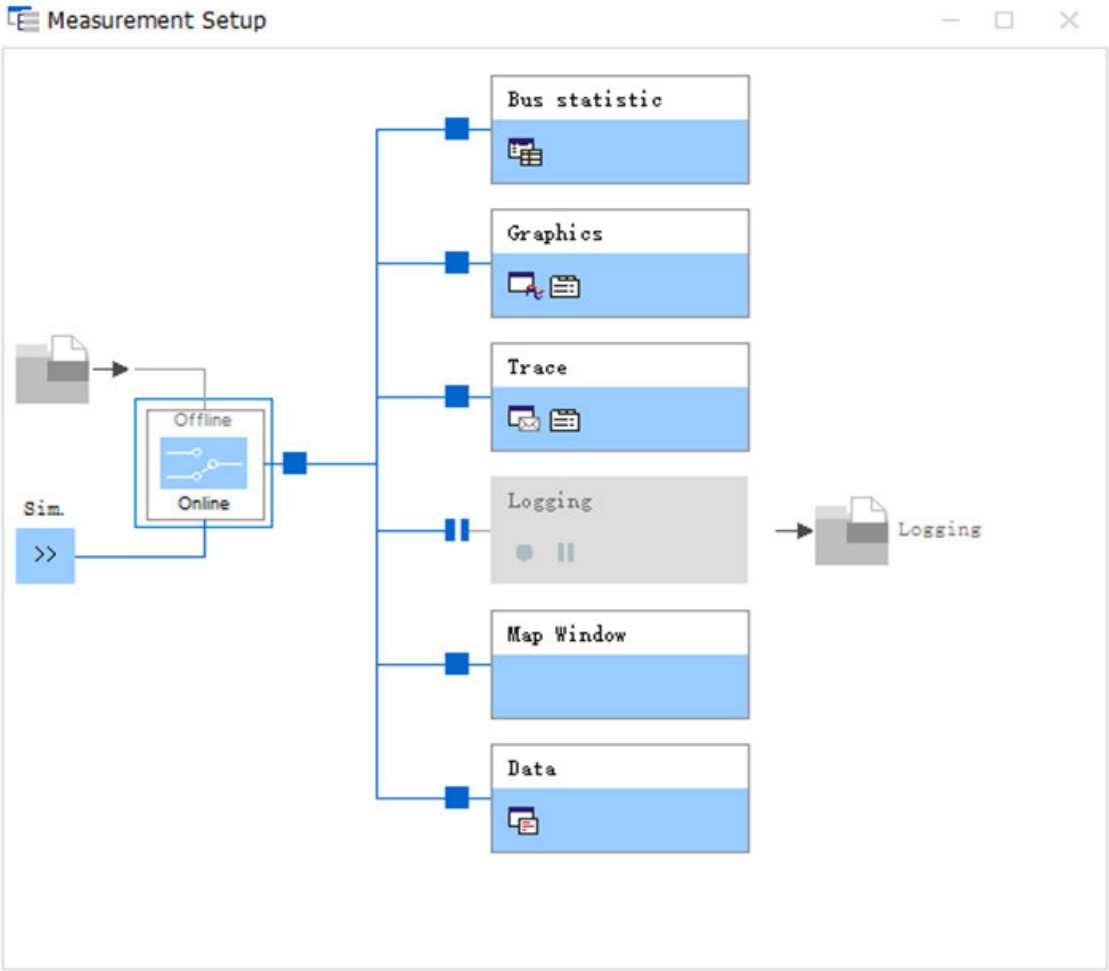

Abbildung 19: Measurement Setup

In diesem Funktionsfenster sind sowohl die Datenerfassung als auch alle Analysekomponenten integriert. (Trace, Daten, Grafiken, MAP-Fenster, etc.) Das Hauptfenster Trace wird für die Anzeige von Informationen in jedem Knoten verwendet.

Simulation: Das Band Simulation enthält zwei Komponenten: die Simulationskomponente und die Stimulationskomponente.Die wichtigste dieser Komponenten ist das Fenster „Simulation Setup". In diesem Fenster können die für das Projekt erforderlichen Netzknoten eingerichtet und angezeigt werden.(Siehe Abb.20)

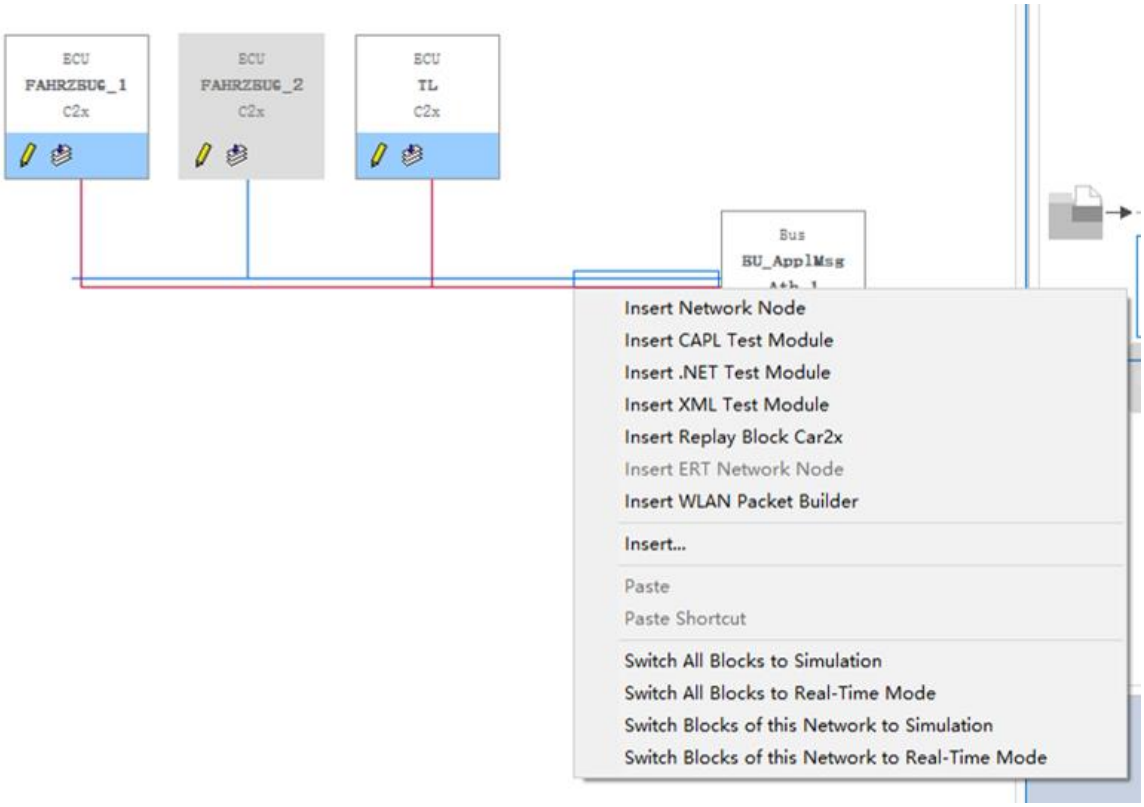

Abbildung 20: Simulation Setup

Insgesamt sind zwei Busse im Emulationsaufbau zu sehen. Eine rote horizontale Linie stellt den Simulationsbus und eine blaue horizontale Linie den physikalischen Bus dar. Die beiden Busse sind über Erweiterungskarten miteinander verbunden. Die Netzwerksimulationsknoten werden auf dem roten Simulationsbus hinzugefügt. Die Option "Netzwerkknoten einfügen" wird verwendet, um einen Emulationsknoten hinzuzufügen. Deaktivierte analoge Knoten werden mit dem physischen Bus verbunden.

Test: Die Funktionsbereich Test enthält die Komponente „Test Units“ und die Komponente Test „Modules“. Er dient dem Aufbau der Testumgebung und dem Testen und Beobachten von Leiterbahnen.

Diagnostics: Die Funktionsbereich Diagnostics besteht aus drei Komponenten, die sich auf die Diagnose beziehen: die Komponente „Konfiguration“, die Komponente „Control“ und die Komponente „Tools“. Dieser Bereich wird hauptsächlich für die Diagnose von Steuergeräten verwendet.

Environment: Die Multifunktionsleiste Environment besteht aus drei Hauptkomponenten, die Komponente „Symbols“, die Komponente „More“ und die Komponente „Car2x“. Eines der am häufigsten verwendeten Fenster ist die „System variables“. Es wird in der Regel zur Verwaltung von Systemvariablen verwendet, einschließlich des Hinzufügens oder Entfernens von Systemvariablen, und es wird auch zur Einstellung der Eigenschaften von Systemvariablen verwendet, wie z. B. ihren Namen, ihren Wert und ihren Datentyp Mit dem „Scenario-Manager“ der Komponente „More“ können die bearbeiteten Szenarien geladen

werden. Mit dem „Station Manager“ können die Namen geändert werden, die angezeigt werden, wenn die verschiedenen Netzwerkknoten in Betrieb sind.

Hardware: Die Multifunktionsleiste Hardware wird hauptsächlich für hardwarebezogene Kanaleinstellungen verwendet, während Sicherheitszertifikate über „Channel Usage“ konfiguriert werden können.

Tools: Die Multifunktionsleiste „Tools" besteht aus den Komponenten „Netzwerk-Tools" und „More". Das Fenster „CANdb++Editor" ermöglicht die Erstellung oder Änderung von Netzwerkdatenbanken. Das Fenster „Scenario Editor" ermöglicht die Bearbeitung der Route des Fahrzeugs.

Layout: Die Hauptfunktion der Multifunktionsleiste "Layout" besteht darin, den Anzeigemodus der einzelnen Fenster einzustellen.

2.6.4 CAPL-Programmierung

CAPL [Vector2] ist eine C-ähnliche Programmiersprache, die von Vector Informatik entwickelt wurde. Der vollständige Name ist Communication Access Programming Language und CAPL wird in CANoe verwendet. Die Ausführung von Blöcken in CAPL ist ereignisgesteuert und CAPL muss in einem professionellen Editor bearbeitet und entwickelt werden, damit auf alle in der Datenbank enthaltenen Objekte (Nachrichten, Signale, Umgebungsvariablen) und Systemvariablen zugegriffen werden kann.

Ein wesentlicher Unterschied zwischen CAPL und C/C++ liegt darin, wann und wie Programmelemente aufgerufen werden. In C/C++ beginnt jeder Verarbeitungsablauf mit der zentralen Startfunktion main(). Im Gegensatz dazu enthält ein CAPL-Programm eine Sammlung gleichberechtigter Programme, die jeweils auf externe Ereignisse reagieren.

Ausgelöst durch das System: Das sind zum einen die Ereignisse preStart, on start, on preStop und on stopMeasurement, die zur Initialisierung und Nachbearbeitung von Messläufen genutzt werden können, und zum anderen Zeit- und Tastaturereignisse auf Timern und Tasten.

Ausgelöst durch Buskommunikation: Die Ereignisprogramme, die aufgrund von Busereignissen (z.B. Kommunikation oder Fehlerbehandlung) auftreten, sind vielfältig und stark vom Bustyp abhängig. Beispiele hierfür sind on message und on busOff bei CAN oder frFrame und frStartCycle bei FlexRay.

2.6.5 Struktur eines CANoe-Car2x-Projekts[WiKf 2020]

Zu Beginn des Projekts sollte die Gesamtstruktur des Projekts festgelegt werden, damit Unklarheiten wirksam vermieden werden können und das Projekt reibungslos verläuft.

Die Schritte, die zur Erstellung eines vollständigen Car2x-Projekts erforderlich sind, lassen sich grob in folgende Bereiche unterteilen:Erstellung von Simulationsknoten, Hinzufügen von Systemvariablen, Konfiguration des Sicherheitszertifikats, Erstellung von Simulationsszenarien und Erstellung von Panel-Display.

2.6.5.1 Erstellung von Simulationsknoten

Die entsprechenden Simulationsknoten werden in der Car2x-Datenbank erstellt und die entsprechenden Anwendungsnachrichten zu den Knoten hinzugefügt; CAM-Nachrichten und DENM-Nachrichten können im Verkehrsteilnehmer-Simulationsknoten verwendet werden; SAPT-Nachrichten und MAP-Nachrichten können im Lichtsignalanlagen-Simulationsknoten verwendet werden. Simulationsknoten, die in der Car2x-Datenbank über "Node Synchronization" definiert wurden, werden dem Bus hinzugefügt.

Die Attribute der Knoten im Attributdialog können bearbeitet werden.

2.6.5.2 Hinzufügen von Systemvariablen

Systemvariablen werden in CANoe häufig verwendet.

Systemvariablen können zur Verwaltung von Parametern verwendet werden, die global sind und häufig geändert werden müssen.

Mit Hilfe von Systemvariablen können bestimmte Änderungszeitpunkte und Warnintervalle festgelegt werden.

Mit Hilfe von Systemvariablen kann ein dynamisches Echtzeit-Betriebssystem geschaffen werden, das die vielen Zustände, die in CANoe existieren können, erkennt und es dem System ermöglicht, entsprechend zu reagieren.

Das System kann Systemvariablen verwenden, um Benutzereingaben und durchgeführte Aktionen zu verfolgen, um festzustellen, was der Benutzer möglicherweise tut, und um sicherzustellen, dass CANoe korrekt auf die Anforderungen des Benutzers reagiert.

2.6.5.3 Konfiguration des Sicherheitszertifikats

Ein Sicherheitszertifikat ist ein wesentlicher Bestandteil der Entwicklung von Anwendungen mit Canoe-Software. Sie spielt eine wichtige Rolle bei der Verhinderung unbefugten Zugriffs, der Sicherung von Daten und der Überprüfung der Sicherheit der Anwendung.

Ein Sicherheitszertifikat trägt zu einem sicheren Informationsaustausch bei, da es gewährleistet, dass sowohl der Sender als auch der Empfänger vertrauenswürdig sind. Auf diese Weise wird auch die Identität des Absenders gewährleistet und sichergestellt, dass die Daten während der Übertragung nicht verfälscht oder verändert werden können.

Die Zertifikateigenschaften sind bekannt, bevor das Zertifikat konfiguriert wird.

Betreff Typ： Der Subject-Typ gibt den Verwendungszweck des Zertifikats an.Wurzelzertifikate können immer erzeugt werden. Alle anderen Zertifikatstypen können nur erstellt werden, wenn ein übergeordnetes Zertifikat mit passendem privaten Schlüssel vorhanden ist.

Subject-Name： Der Subject-Name identifiziert im Falle von CA-Zertifikaten den Zertifikatsinhaber. Bei Pseudonym-Zertifikaten bleibt der Name leer.

Unterzeichner： Der Signierer gibt an, welches Zertifikat zum Signieren des neuen Zertifikats verwendet wird.Bei Root-CA-Zertifikaten ist dieses Feld nicht vorhanden, da diese selbstsigniert sind. Bei allen anderen Zertifikatstypen kann nur ein Zertifikat mit bekanntem privatem Schlüssel als Unterzeichner ausgewählt werden.

Beachtung:

Das Pseudonym-Zertifikat, das dem emulierten Knoten entspricht, muss unter demselben Root-Zertifikat stehen.

2.6.5.4 Erstellung von Simulationsszenarien

Der Scenario-Editor car2x scenario Editor wird geöffnet

Die folgenden Elemente werden zur Erstellung des Szenarios verwendet.

(1) Route.

Eine durch Wegpunkte definierte Route, auf der sich Stationen befinden können. Der Benutzer kann eine Route im Szenariofenster auf der linken Seite hinzufügen und mit der Schaltfläche "Edit" in der Symbolleiste bearbeiten

(2) Station.

Stationen sind Verkehrsteilnehmer und werden in der Regel Strecken zugeordnet. Der Benutzer kann im linken Szenariofenster Stationen in die Route einfügen.

Der Name der Station muss mit dem Namen des Knotens in der Datenbank übereinstimmen. Inkonsistente Knotennamen führen dazu, dass Szenarien nicht erfolgreich in den Szenariomanager geladen werden können.

2.6.5.5 Erstellung von Panel-Display

Das Panel ist ein Übersichtsdiagramm von CANoe, das Daten zu einer Vielzahl von Protokollen/Telegrammen/Signalen usw. anzeigt. Es ermöglicht auch die Überwachung und Fehlerdiagnose in Echtzeit und kann genaue und zeitnahe visuelle Informationen liefern, was die Effizienz der Ingenieure effektiv verbessert.

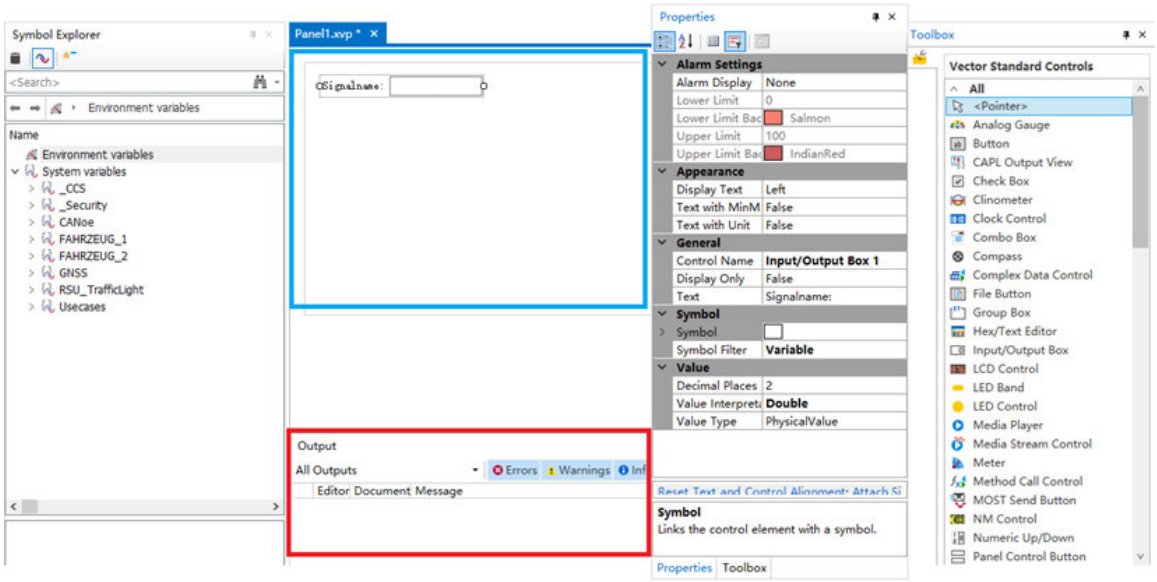

Abbildung 21: Panel

Wie in der Abbildung 21 zu sehen ist, sind die vorhandenen Systemvariablen auf der linken Seite des Bearbeitungsfeldes zu sehen, und auf der rechten Seite des Bearbeitungsfeldes befinden sich das Toolkit und das Bearbeitungsfeld Eigenschaften. Panels können durch Auswahl des entsprechenden Werkzeugs im Toolkit erstellt und an der durch das blaue Kästchen angezeigten Position angezeigt werden. In der "Input/Output Box" sind z.B. die Eigenschaften des Werkzeugs im Bereich "Eigenschaften" zu sehen. In der Statusleiste "Symbol" kann das Werkzeug mit Systemvariablen verknüpft werden, so dass Änderungen an den Systemvariablen als Bilder visualisiert werden können. Der mit einem roten Kästchen markierte Bereich kann zur Anzeige von Fehlern verwendet werden, die bei der Panel-Produktion auftreten, z. B. wenn die Eigenschaften einer Systemvariablen geändert werden, nachdem sie im Panel verknüpft wurde, führt dies zu einer Fehlanpassung der Systemvariablen, und der Fehler wird im Unter-OUTPUT angezeigt.

2.7 Car2x-Kommunikations-Hardware VN4610

Dieser Abschnitt beschreibt den Begriff, die Vorteile und die Anwendungsbereiche des VN4610.

2.7.1 Überblick

Das VN4610[Vector3] ist eine spezialisierte Lösung für Anwendungen, die auf IEEE 802.11p und CAN (FD) basieren. Es wurde von VECTOR entwickelt und ermöglicht den Zugriff auf IEEE 802.11p- und CAN (FD)-Netzwerke über USB- und Ethernet-PC-Anschlüsse. Die IEEE 802.11p-basierte Dedicated Short Range Communication (DSRC) arbeitet im 5,9 GHz-Bereich und das VN4610 unterstützt den ungefilterten Empfang und die Übertragung von IEEE 802.11p-Frames für die Implementierung von Car2x/V2x-

Anwendungen. Darüber hinaus verfügt das VN4610 über einen eingebauten GNSS-Empfänger, der die aktuelle GNSS-Position und GNSS-Zeit liefert.

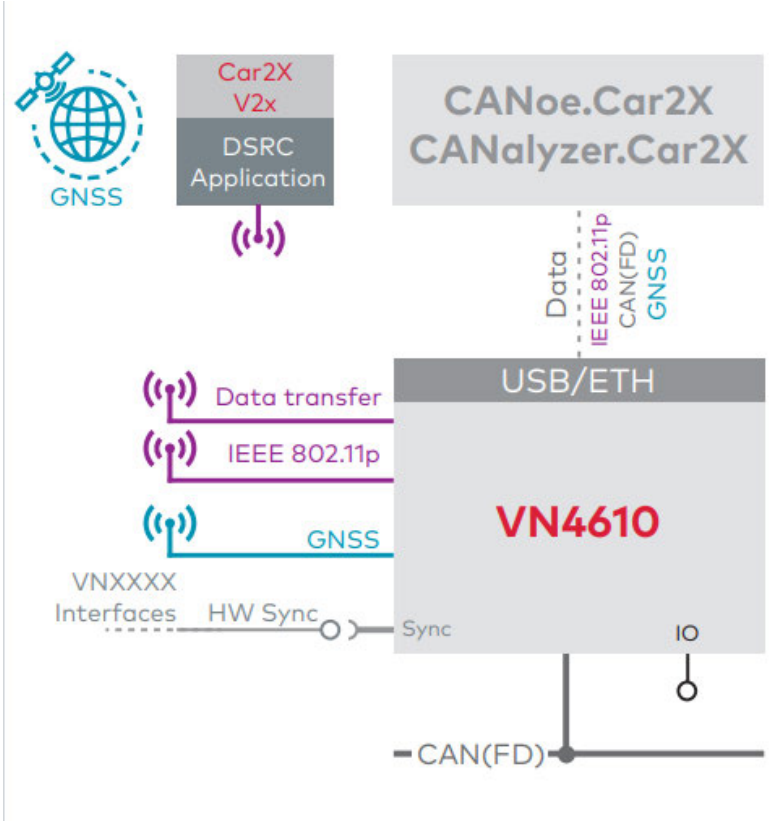

Abbildung 22: V4610: Beschaltungsmöglichkeiten und Anwendungsfälle[Quelle:Vector]

2.7.2 Vorteile

Das VN4610 bietet die folgenden Vorteile[Vector3]:

- Senden/Empfangen von Frames nach IEEE 802.11p
- Zwei konfigurierbare IEEE 802.11p WLAN Funkkanäle
- Ungefiltertes Weiterleiten von IEEE 802.11p Datenpaketen an die Applikation
- Einstellbare Kommunikationsparameter wie Funkkanalauswahl, Bandbreite, Sendeleistung, Modulationsart und Protokollformat LPD/EPD
- Zwei CAN-Highspeed Kanäle CAN (FD) fähig
- GNSS Empfänger liefert aktuelle Position und Zeit
- Präzise Zeitstempel (Genauigkeit 1µs) basierend auf GNSS Zeit
- VN4610 und CANoe .Car2x/CANalyzer .Car2x sind optimal aufeinander abgestimmt

- Synchronisation mit mehreren Interfaces und mit anderen Bussystemen (Ethernet, CAN, LIN, FlexRay, ...)
- Analog/Digital IO-Funktionalität
- Robustes Gehäuse
- Stromversorgung und Temperaturbereich ideal für Automotive- sowie industrielle Anwendungen
- Zeitsynchronisation mit PTP nach IEEE 1588 Standard

2.7.3 Anwendungsbereiche

Das VN4610 erfüllt alle Hardware-Anforderungen, die die Grundlage für das Testen von DSRC-Anwendungen über den IEEE 802.11p-Funkkanal bilden.(Siehe Abb.23)

Analyse:

Das VN4610 leitet alle empfangenen Funktelegramme von beiden Kanälen ungefiltert an das Testtool zur Analyse weiter. Dadurch ist es möglich, auch Frames zu analysieren, die vom Steuergerät aufgrund von Fehlern in Zeit, geografischen Informationen oder durch Car2x/V2x verursachten Protokollen zurückgewiesen wurden. Die Zeitstempel der Nachrichten auf den Buskanälen sind zeitlich synchronisiert, was auch Verzögerungsmessungen ermöglicht.

Simulation/Emulation:

CANoe Car2x bietet in Kombination mit dem VN4610 eine perfekt aufeinander abgestimmte Lösung, um eine Testumgebung für Car2x/V2x-Anwendungen zu schaffen. Das VN4610 sendet die übertragenen Frames, so dass die Kommunikationsparameter für verschiedene Tests individuell konfiguriert werden können, was die Durchführung von Tests erleichtert.

GNSS-Empfänger:

Der VN4610 liefert präzise Positions-, Zeit- und Geschwindigkeitsinformationen, die als Teststimuli oder zur Protokollierung genutzt werden können. Außerdem können absolute GNSS-Zeitstempel zur Synchronisation von verteilten Messaufzeichnungen für spätere Analysen verwendet werden.

Zeitsynchronisation:

Das VN4610 kann mit PTP nach dem IEEE1588-Standard präzise zeitsynchronisiert werden. Das Gerät kann z.B. als PTP-Master mit einer UTC-Zeitbasis konfiguriert werden, die vom eingebauten GNSS-Empfänger bereitgestellt wird.

	VN4610
802.11p channels/transceiver	2 channels with NXP SAF5100
GNSS channel/transceiver	uBlox NEO-M8U supports GPS, GLONASS, Beidou, Galileo; up to 3 systems simultaneously
CAN/CAN FD channels/transceiver/ physical layer	2 x NXP TJA1057G CAN Highspeed (CAN FD capable)
Analog and digital I/O	1 x analog in, 2 x digital in, 1 x digital out
Time stamp accuracy within one device sync. of multiple devices with sync cable	 1 µs typ. 50 µs typ. 1 µs
Time synchronization	PTP according to IEEE1588-2008 standard
Connectors	2 x SMA for 802.11p; 1 x SMA for GNSS; 2 x DSUB9 for CAN/CAN-FD (single channel); 1 x DSUB9 for I/O
Baudrates	CAN up to 2 Mbit/s. CAN FD up to 8 Mbit/s. 802.11p depending on modulation type up to 27 Mbit/s
PC interface	USB 2.0 highspeed / Ethernet IEEE 100Base-TX/1000Base-T
Average response time	250 µs
External power supply	6 V...50 V DC
Power consumption	typically 7 W
Operating system requirements	Windows 10 (64 bit)
Driver library	XL Driver Library for CAN
Temperature range operating: storage:	-40...+60°C -40...+85°C
Dimensions (WxHxD)	111mm x 45mm x 157mm (without antennas)
Weight	ca. 610g
Housing	Robust aluminium housing

Abbildung 23: Technische Daten[Quelle:Vector]

3 Implementierung und Simulation einer Lichtsignalanlage über CANoe

Verkehrssignale sind ein fester Bestandteil des menschlichen Verkehrs und eine wichtige Einrichtung, um für Ordnung unter den Verkehrsteilnehmern zu sorgen. Dieses Abschnitt beschreibt den Prozess der Simulation einer Lichtsignalanlage an einer realen Kreuzung in der Stadt Mittweida mit der Software CANoe und dieser Lichtsignalanlage.

3.1 Allgemeine Informationen zur simulierten Lichtsignalanlage

Die Lichtsignalanlage befindet sich an der Kreuzung von Tzschirnerstraße und Poststraße, Tzschirnerplatz in Mittweida.

Abbildung 24 zeigt die Straßeninformationen für die Kreuzung.

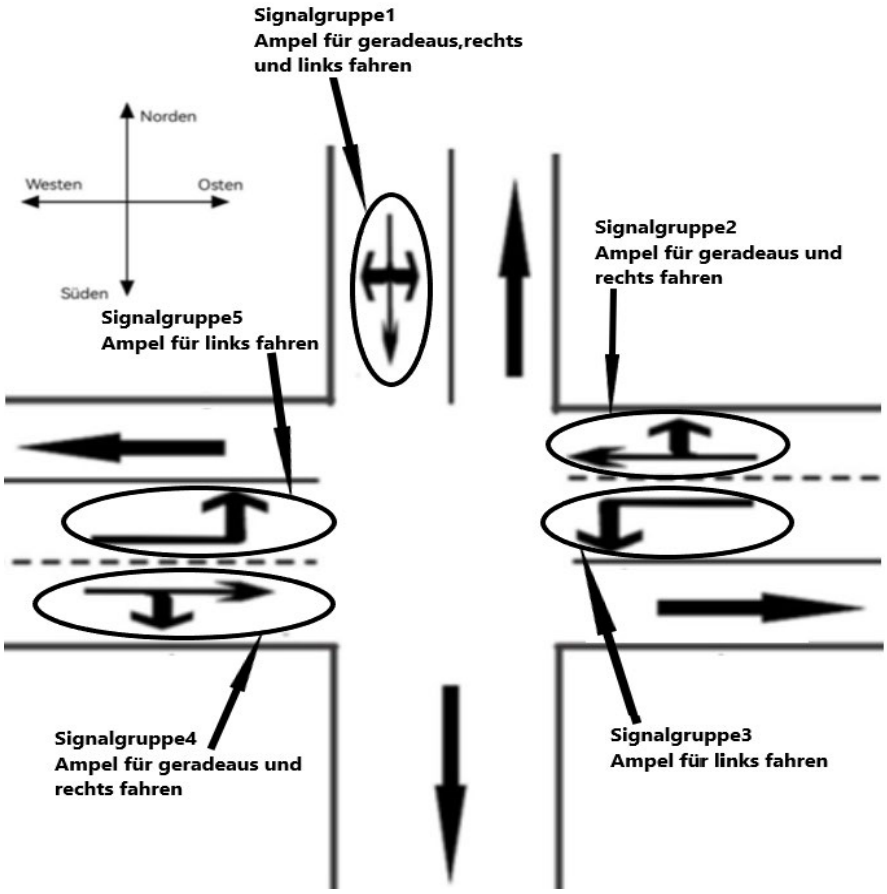

Abbildung 24: die Straßeninformationen für die Kreuzung

Spezifische Informationen sind in Tabelle 1 aufgeführt.

Straße in Richtung Norden	Die Straße enthält eine Einfahrts- und eine Ausfahrtsstraße, und die Signalgruppe, die mit dieser Einfahrtsstraße verbunden ist, hat drei Richtungen: geradeaus, links abbiegen und rechts abbiegen. Diese Signalgruppe wird als Signalgruppe 1 bezeichnet.
Straße in Richtung Süden	Diese Straße ist eine Einbahnstraße und enthält nur eine Ausfahrt. Für diese Fahrbahn gibt es keine Signalgruppe
Straße in Richtung Osten	Der Fahrstreifen besteht aus zwei Einfahrspuren und einer Ausfahrspur. Eine der Einfahrspuren ist eine Geradeaus- und Rechtsabbiegespur und ist der Signalgruppe 2 zugeordnet. Die andere Einfahrspur ist eine Linksabbiegespur und ist der Signalgruppe 3 zugeordnet.
Straße in Richtung Westen	Der Fahrstreifen besteht aus zwei Einfahrspuren und einer Ausfahrspur. Eine der Einfahrspuren ist eine Geradeaus- und Rechtsabbiegespur und ist der Signalgruppe 4 zugeordnet. Die andere Einfahrspur ist eine Linksabbiegespur und ist der Signalgruppe 5 zugeordnet.

Tabelle 1: Spezifische Informationen über Straßenkreuzungen

In Tabelle 2 wird die Dauer der einzelnen Signalgruppen angegeben.

Signalgruppe 1	Insgesamt:80s,Rot: 60s, Rot-gelb: 2s, Grün: 16s, Gelb:2s
Signalgruppe 2	Insgesamt:80s,Rot: 40s, Rot-gelb: 2s, Grün: 36s, Gelb:2s
Signalgruppe 3	Insgesamt:80s,Rot:70s, Rot-gelb: 2s, Grün: 6s, Gelb:2s
Signalgruppe 4	Insgesamt:80s,Rot: 40s, Rot-gelb: 2s, Grün: 36s, Gelb:2s
Signalgruppe 5	Insgesamt:80s,Rot: 70s, Rot-gelb: 2s, Grün: 6s, Gelb:2s

Tabelle 2: Dauer der Lichtsignalanlagen

3.2 Implementierung in CANoe

In diesem Abschnitt wird die Simulation von Verkehrssignalanlagen mit Hilfe von CANoe beschrieben.

3.2.1 Projekte erstellen

Die für die Aufgabe benötigten Knoten werden entsprechend den Schritten der car2x-Projekterstellung angelegt.

Zunächst wird ein neues car2x-Konfigurationsprojekt erstellt. Eine Datenbank namens "EU_ApplMsg_20221103.xml" wird zu den Datenbanken im Simulations-Setup hinzugefügt. Die Datenbank "EU_ApplMsg_20221103.xml" wird dann geöffnet, ein neuer Knoten mit dem Namen "TL" wird in der Datenbank erstellt und die Signale MAP und SPAT werden hinzugefügt und ihre Eigenschaften bearbeitet. (wie in der Abbildung 25 unten dargestellt)

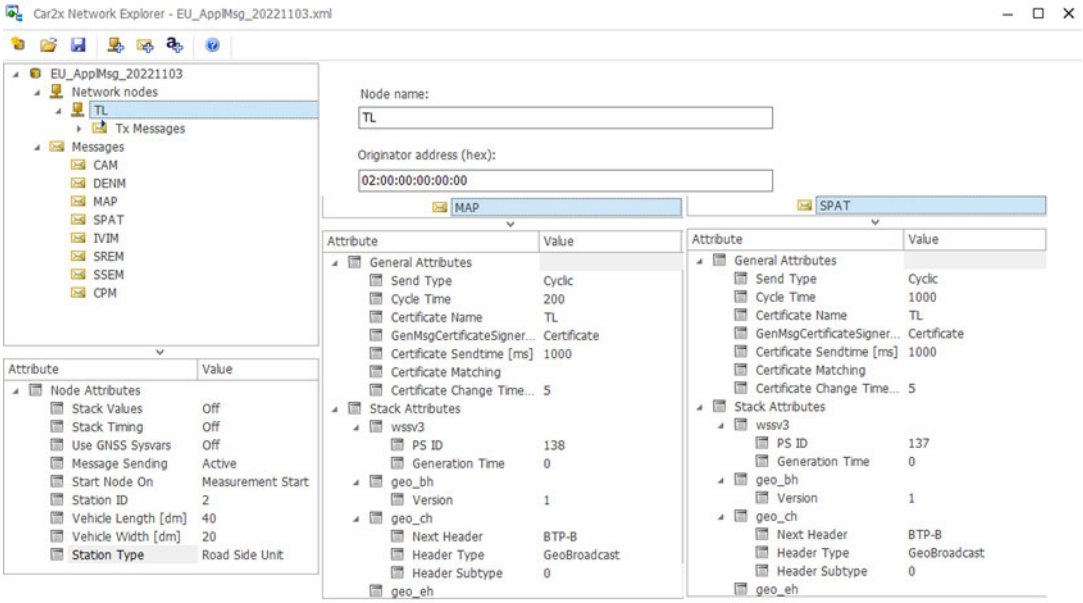

Abbildung 25: Eigenschaften von TL-Knoten

Mit der Funktion "Node Sync" wird ein Knoten zum Bus hinzugefügt und die CAN-Datei erstellt. (Siehe Abb.26)

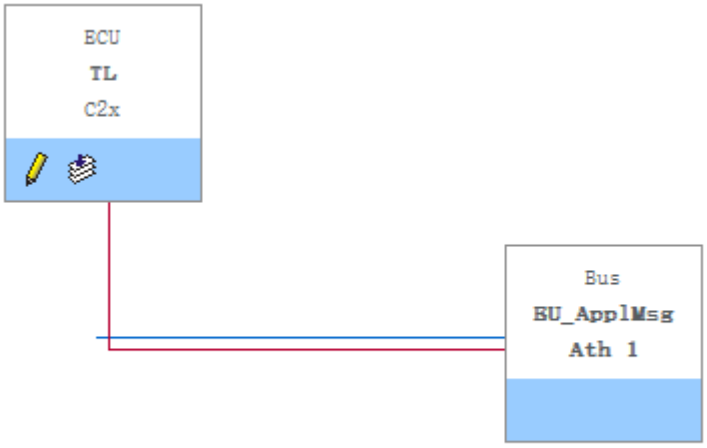

Abbildung 26: TL-Knoten-Zusatz

3.2.2 Hinzufügen von Systemvariablen

Die für die Aufgabe erforderlichen Systemvariablen werden zu den "Systemvariablen" hinzugefügt. Wie in der nachstehenden Abbildung 27 dargestellt, werden die Systemvariablen entsprechend den Anforderungen der Aufgabe hinzugefügt. Die Bedeutung der Systemvariablen wird in der Tabelle angegeben.

Variable	Datatype	Initial Value	Min	Max	Location
RSU_TrafficLight					
button	Int32	1	0	1	Configuration
Hauptstrasse_gerade	Int32	0	0	20	Configuration
Hauptstrasse_gerade_1_0	Int32	0	0	20	Configuration
Hauptstrasse_gerade_1_1	Int32	0	0	20	Configuration
Hauptstrasse_gerade_1_2	Int32	0	0	20	Configuration
Hauptstrasse_gerade_1_3	Int32	0	0	20	Configuration
Reamin	Int32	0	0	20	Configuration
State_SG1	Int32	0	0	4	Configuration
State_SG2	Int32	0	0	4	Configuration
State_SG3	Int32	0	0	4	Configuration
State_SG4	Int32	0	0	4	Configuration
State_SG5	Int32	0	0	4	Configuration
TimeToSwitch_SG1	Double	-	-	-	Configuration
TimeToSwitch_SG2	Double	-	-	-	Configuration
TimeToSwitch_SG3	Double	-	-	-	Configuration
TimeToSwitch_SG4	Double	-	-	-	Configuration
TimeToSwitch_SG5	Double	-	-	-	Configuration

Abbildung 27: Systemvariablen für TL-Knoten

Die Systemvariablen sind wie folgt definiert:

RSU_TrafficLight::button	Es wird verwendet, um die Ampeln zu aktivieren und auszuschalten.
RSU_TrafficLight::Hauptstrasse_gerade RSU_TrafficLight::Hauptstrasse_gerade_1_0, _1_1, _1_2, _1_3	Sie werden verwendet, um die Dauer des grünen Lichts auf Hauptstraßen für Geradeaus- und Rechtsabbiegersignale zu verlängern.
RSU_TrafficLight:: State_SG1/2/3/4/5	Es zeigt die vier Zustände der Ampel an: rot, rot-gelb, grün und gelb.
RSU_TrafficLight::TimeToSwitch_SG1/2/3/4/5	Es wird verwendet, um die Dauer der einzelnen Zustände der Ampel anzuzeigen.
RSU_TrafficLight::Reamin	Es wird verwendet, um den Wert der Systemvariablen "Hauptstraße_gerade" auszugleichen,

	wenn "Hauptstraße_gerade" ungleich 0 ist.

Tabelle 3: Bedeutungen von Systemvariablen für Lichtsignalanlage

3.2.3 Sicherheitszertifikat

Die Konfiguration des Sicherheitszertifikats kann im Abschnitt "Channel Usage" der Komponente "Hardware" geöffnet werden oder im Abschnitt "Optionen" der Komponente "File".(Siehe Abb.28)

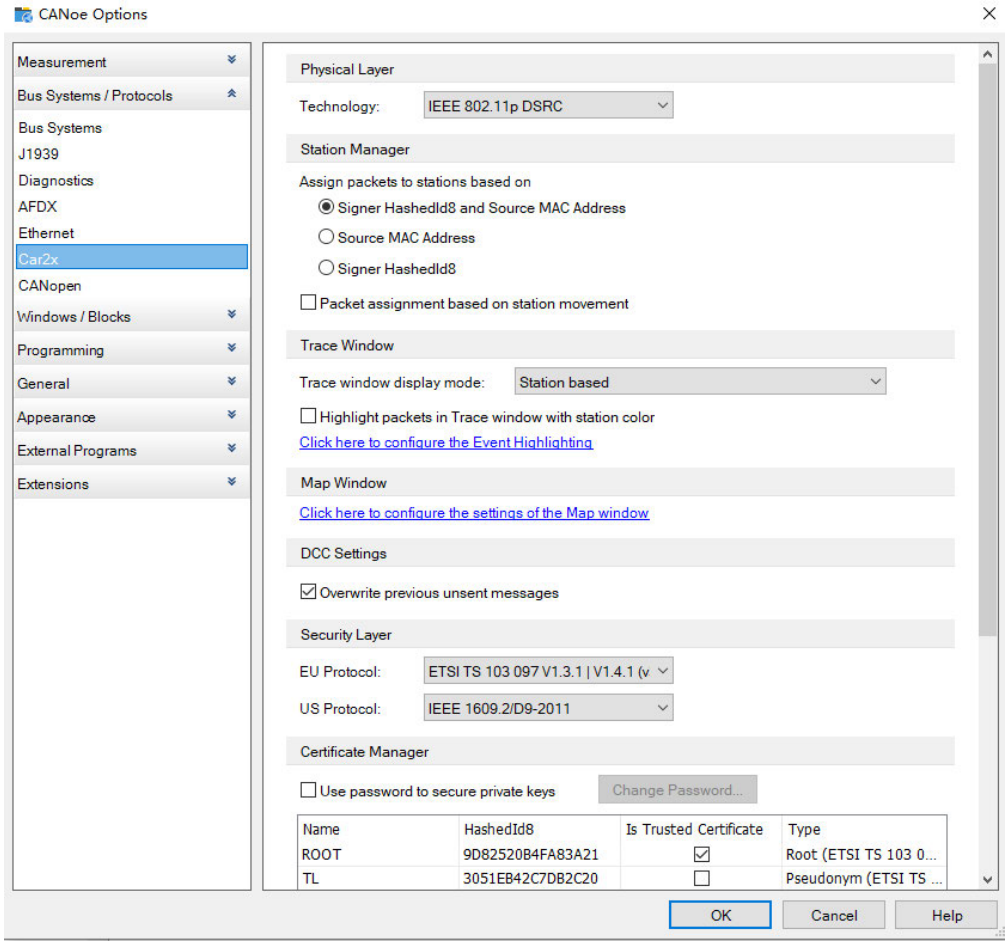

Abbildung 28: Zertifikate für TL-Knoten

3.2.4 Die Struktur der can-Datei

Die Programmierung des Ampelsystems erfolgt mit CAPL, das eine can-Datei und fünf cin-Dateien enthält:

- **SimRsuTL.can:** Die Datei besteht aus insgesamt fünf Abschnitten

- Includes: In diesem Abschnitt werden die für die Programmierung benötigten cin-Dateien hinzugefügt.

SimRsuTL.cin
ASNV_Template_BaseDatatypes.cin
ASNV_Template_SPAT.cin
ASNV_Template_SPAT.cin
Car2xAPI.cin

Tabelle 4: cin-Dateien in SimRsuTL.can

- Variables: Es wird verwendet, um die in dieser Programmierdatei benötigten Variablen zu setzen.
- System: Es enthält Ereignisbehandler für Systemereignisse.
- Value Objects: Es enthält Ereignishandler für Systemvariablen.
- Function: Es enthält die für die Implementierung des Programms erforderlichen Funktionen.

- **SimRsuTL.cin:** Es dient dazu, den Text der Lichtsignalanlage so zu bearbeiten, dass er auf dem Bildschirm dargestellt werden kann.
- **ASNV_Template_BaseDatatypes.cin:** Diese Datei wurde direkt aus dem "Car2x-system"-Beispiel extrahiert, ohne jegliche Änderungen. Diese Datei zeigt die Datentypen, die im gesamten Programmierprojekt benötigt werden.
- **ASNV_Template_SPAT.cin:** Diese Datei wurde direkt aus dem "Car2x-system"-Beispiel extrahiert, ohne jegliche Änderungen. Die Datei kompiliert das konkrete Format der SPAT-Nachricht.
- **ASNV_Template_MAP.cin:** Diese Datei wurde direkt aus dem "Car2x-system"-Beispiel extrahiert, ohne jegliche Änderungen. Die Datei kompiliert das konkrete Format der MAP-Nachricht.
- **Car2xAPI.cin:** Diese Datei wurde direkt aus dem "Car2x-system"-Beispiel extrahiert, ohne jegliche Änderungen. Diese Datei wird verwendet, um APIs zu schreiben, die für das gesamte Projekt gemeinsam ist.

3.2.5 Programmierung

In diesem Abschnitt wird der Code in den Dateien SimRsuTL.can und SimRsuTL.cin beschrieben.

3.2.5.1 Code in den Dateien SimRsuTL.cin

Die Datei SimRsuTL.cin enthält 3 Hauptfunktionen (Siehe Tabelle 5):

Funktion: `MapInit`	Es wird zum Bearbeiten von MAP-Nachrichten verwendet.
Funktion: `SpatInit`	Es wird zum Bearbeiten von SPAT-Nachrichten verwendet.
Funktion: `UpdateSpat`	Es wird verwendet, um den Code für den Übergang der Dauer und des Status der Ampelanlage zu bearbeiten.

Tabelle 5: Hauptfunktionen in den Dateien SimRsuTL.cin

In der Funktion `MapInit` wird die MAP-Nachricht bearbeitet.

Wie in der Abbildung 29 unten dargestellt, besteht die Lichtsignalanlage aus der RoadSideUnit TL und der Lichtsignal-Kreuzung ID 1. Der grüne Kreis mit dem schwarzen Kasten stellt die RoadSideUnit TL dar, die für die Übertragung und den Empfang von Signalen verwendet wird. Der andere Teil des Systems ist der Lichtsignal Intersection ID 1, der hauptsächlich aus 9 Fahrspuren besteht.

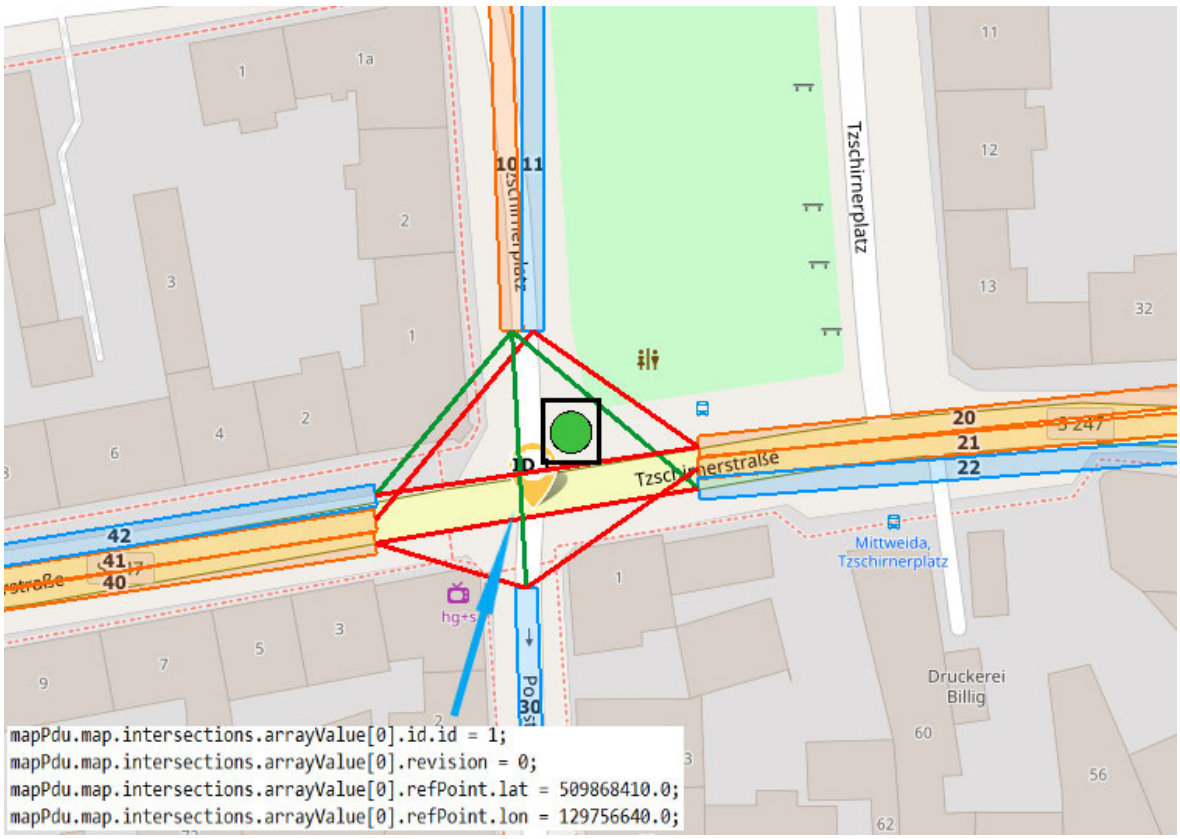

Abbildung 29: Präsentation der Lichtsignalanlage

Der blaue Pfeil zeigt die Standortinformation des Lichtsignals Intersection ID 1 an. Am Verkehrsknotenpunkt wird eine geeignete Breiten- und Längenangabe ausgewählt und die Straßeninformation mit dieser Breiten- und Längenangabe gezeichnet.

Der Code in der Listing 1 wird verwendet, um die Anzahl der Fahrspuren in der Lichtsignalanlage anzugeben, die gemäß der tatsächlichen Situation aus insgesamt 9 Fahrspuren besteht. Daher wird diesem Code der Wert 9 zugewiesen. Der Pfad zum Code in der Datenbank entspricht dem Format, in dem der Code geschrieben ist. laneSet befindet sich in den Intersections der MAP-Nachricht, der Wert von arrayValue wird von 0 berechnet und stellt die ID seines Präfixes in der Datenbank dar, z. B. `laneSet.length = 9`, `laneSet.ArrayValue[X]`, der Wert von X ist 0-8.

```
mapPdu.map.intersections.arrayValue[0].laneSet.length = 9;
```

Listing 1: Bestimmung der Anzahl der Lane

Die XY-Koordinatenachse wird mit dem Breiten- und Längengrad der Lichtsignalanlage als Mittelpunkt festgelegt. Die Breiten- und Längengrade der diagonalen Koordinaten der Punkte, an denen die Fahrspuren über Google Maps eingezeichnet sind, werden bestimmt. Die Koordinaten der gezeichneten Fahrspur ergeben sich durch Subtraktion des Ursprungs

und des gemessenen Punktes. Da die Kartenkoordinaten große Fehler enthalten, müssen die Daten verbessert werden, um die endgültigen Fahrspuren zu erhalten.

In der nachstehenden Abbildung 30 ist einer der Punkte der Lane 11 als Beispiel dargestellt.Der Breitengrad und der Längengrad der Lichtsignalanlage sind auf den Längengrad X1 = 129756640.0 und den Breitengrad Y1 = 509868410.0 festgelegt. Der Breitengrad und der Längengrad von Punkt 1 sind auf den Längengrad X2 = 129756640.0 und den Breitengrad Y2 = 509870410.0 festgelegt. Die Koordinaten von Punkt 1 auf der Achse sind (X2-X1=0,Y2-Y1= 2000). Die für Breiten- und Längengrade verwendeten Einheiten sind Bruchteile von Grad(DD). Und die Einheit wurde gegenüber dem Original um den Faktor 10 hoch 7 vergrößert.

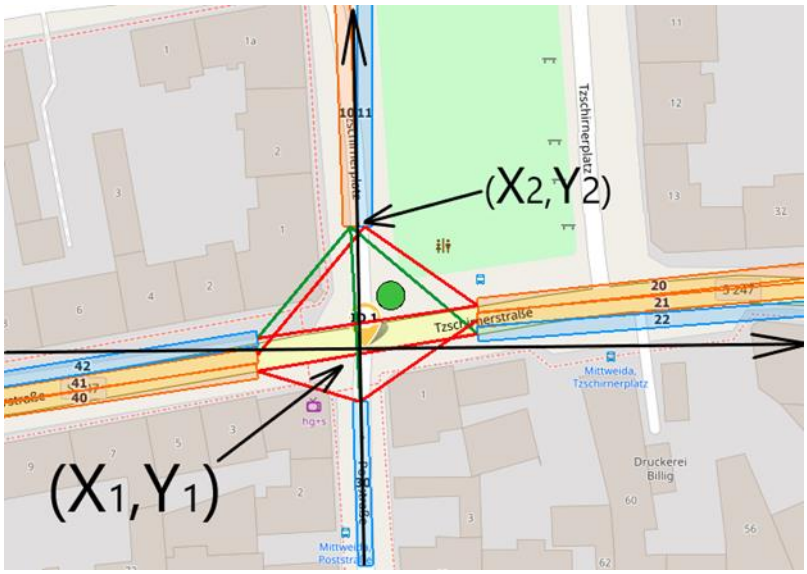

Abbildung 30: die Herstellung der Lane der Lichtsignalanlage

In der nachstehenden Tabelle sind die Punkte aufgeführt, die für die Darstellung der einzelnen Fahrspuren verwendet wurden.

laneID 10	Punkt 1：x = -250 y = 2000 Punkt 2：x = -140 y = 4000
laneID 11	Punkt 1：x = 0 y = 2000 Punkt 2：x = 5 y = 4000
laneID 20	Punkt 1：x = 2000 y = 600 Punkt 2：x = 6400 y = 680
laneID 21	Punkt 1：x = 2000 y = 350

	Punkt 2：x = 6500 y = 600
laneID 22	Punkt 1：x = 2000 y = 100 Punkt 2：x = 6500 y = 500
laneID 30	Punkt 1：x = -80 y = -1100 Punkt 2：x = 100 y = -2900
laneID 40	Punkt 1：x = -1900 y = -560 Punkt 2：x = -6300 y = -950
laneID 41	Punkt 1：x = -1900 y = -280 Punkt 2：x = -6300 y = -1000
laneID 42	Punkt 1：x = -1900 y = 20 Punkt 2：x = -6200 y = -1000

Tabelle 6: Koordinaten für das Einzeichnen von Lichtsignalanlagen

```
mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].laneID = 10;
mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].laneAttributes.directionalUse.stringLength = 2;
strncpy(mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].laneAttributes.directionalUse.string, "10", 3);
mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].maneuvers.isValidFlag = 1;
mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].maneuvers.stringLength = 3;
strncpy(mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].maneuvers.string, "111", 13);
mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].nodeList.nodes.length = 2;
mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].nodeList.nodes.arrayValue[0].delta.choice = 4;
mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].nodeList.nodes.arrayValue[0].delta.node_XY5.x = -250;
mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].nodeList.nodes.arrayValue[0].delta.node_XY5.y = 2000;
mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].nodeList.nodes.arrayValue[1].delta.choice = 4;
mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].nodeList.nodes.arrayValue[1].delta.node_XY5.x = -140;
mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].nodeList.nodes.arrayValue[1].delta.node_XY5.y = 4000;
mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].connectsTo.isValidFlag = 1;
mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].connectsTo.length = 3;
mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].connectsTo.arrayValue[0].connectingLane.lane = 22;
mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].connectsTo.arrayValue[0].signalGroup.isValidFlag = 1;
mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].connectsTo.arrayValue[0].signalGroup.value = 1;
mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].connectsTo.arrayValue[1].connectingLane.lane = 30;
mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].connectsTo.arrayValue[1].signalGroup.isValidFlag = 1;
mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].connectsTo.arrayValue[1].signalGroup.value = 1;
mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].connectsTo.arrayValue[2].connectingLane.lane = 42;
mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].connectsTo.arrayValue[2].signalGroup.isValidFlag = 1;
mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].connectsTo.arrayValue[2].signalGroup.value = 1;
```

Listing 2: MAP

In der Listing 2 ist die Lane 10 als Beispiel dargestellt. Die Codes werden in Anlagen, Teil1 angezeigt.

Anhand des Codes in der Blackbox wird die Art der Fahrspur bestimmt. Die Fahrspur wird in eine einfahrende und eine ausfahrende Fahrspur unterteilt. Die Bestimmung erfolgt in der Reihenfolge von links nach rechts und unter Verwendung boolescher Werte. Die Zahl "10" enthält insgesamt zwei Urteile, wobei die Zahl "1" bedeutet, dass das Urteil für die

einfahrende Fahrspur gültig ist, und die Zahl "0", dass das Urteil für die ausfahrende Fahrspur ungültig ist.

Der Code im roten Feld wird verwendet, um die zulässige Fahrtrichtung für diese Fahrspur zu bestimmen. In der xml-Datenbank können alle Fahrspurtypen angezeigt werden, in dieser Aufgabe werden nur die ersten drei Fälle verwendet: geradeaus erlaubt, links abbiegen erlaubt, rechts abbiegen erlaubt. Dies geschieht ebenfalls in der Reihenfolge von links nach rechts und unter Verwendung boolescher Werte. Die Zahl "111" bedeutet, dass das Geradeausfahren erlaubt ist, und die Beurteilungen für das Links- und Rechtsabbiegen sind alle zutreffend, so dass die lane 10 alle Fahrtrichtungen erlaubt.(Siehe Listing 3)

```
0 : maneuverStraightAllowed

1 : maneuverLeftAllowed

2 : maneuverRightAllowed
```

Listing 3: Eigenschaften der Eingangsspur

Der Code im grünen Feld wird zum Zeichnen der Fahrspur verwendet. Der Code "`intersections.arrayValue[0].laneSet.arrayValue[0]`" steht für die erste Straße im ersten Lichtsignalsystem. Der Wert der length gibt die Anzahl der Punkte an. Zwei Punkte werden durch die XY-Achse identifiziert und bilden so ein Rechteck. Jeder Punkt hat einen Wertebereich, in dieser Aufgabe wird XY5 verwendet, mit einem Bereich von (-8192, 8191). Diese Daten können in der xml-Datenbank abgefragt werden.

Der Code im blauen Feld wird zum Zeichnen der Verbindungslinien verwendet, Der Code "`intersections.arrayValue[0].laneSet.arrayValue[0].connectsTo.arrayValue[0]`" bezeichnet den ersten Fahrstreifen, der mit dem ersten Fahrstreifen der ersten Lichtsignalanlage verbunden ist, und der Code "`connectsTo.arrayValue[1]`" bezeichnet den zweiten Fahrstreifen, der damit verbunden ist. Und der Wert der length gibt die Anzahl der mit den ausfahrenden Fahrspuren verbundenen Fahrspuren an. Wie in der nachstehenden Tabelle 7 dargestellt, haben nur die einfahrenden Fahrspuren diese Codes, die ausfahrenden Fahrspuren sind die, die verbunden werden.

laneID 10	Verbindung Lane：laneID 22，laneID 30，laneID 42
laneID 20	Verbindung Lane：laneID 11，laneID 42
laneID 21	Verbindung Lane：laneID 30
laneID 40	Verbindung Lane：laneID 22，laneID 30

laneID 41	Verbindung Lane：laneID 11

Tabelle 7: Die Beziehungen zwischen den Regionen

In der Funktion `SpatInit` wird die SPAT-Nachricht bearbeitet.

In der Listing 4 ist die Signalgruppe 1 als Beispiel dargestellt. Die globalen Variablen in capl geben die Eigenschaften der Signalgruppe vor.

StateTime gibt die Dauer der Signalgruppe an, 60 Sekunden für rot, 2 Sekunden für rot-gelb, 16 Sekunden für grün und 2 Sekunden für gelb.

CurrentState ist rot-gelb und gibt an, dass der Zustand der Signalgruppe zu Beginn des Laufs rot-gelb ist.

NextStateChange gibt den Zeitpunkt an, zu dem der Lauf gestartet wurde.

PermissiveMovement ist ein boolescher Wert, der verwendet wird, um zu bestimmen, ob es sich bei dem Signal um eine gerade Hauptstraße oder eine Linksabbiegespur handelt.

```
/* id */ 1,

/* stateTime */ { 60,  2,  16,  2 },

 /* currentState */ eRedYellow,

 /* nextStateChange */ 2,

 /* permissiveMovement */ 0
```

Listing 4: Eigenschaften von Signalgruppen

In der Funktion `SpatInit` wird im Folgenden ein Beispiel für die Signalgruppe 1 dargestellt.(Siehe Listing 5) Die Codes werden in Anlagen, Teil 2 angezeigt.

Die Funktion wird hauptsächlich zum Einstellen und Zurücksetzen der Dauer des Ampelsignals verwendet. Der Code "`intersections.arrayValue[0].states.arrayValue[0].state_time_speed.arrayValue[0]`" kennzeichnet die erste "state_time_speed" Attribut der ersten Signalgruppe im ersten Lichtsignalsystem.

```
signalGroup_1[0].id = 1;
signalGroup_1[0].stateTime[0] = 60;
signalGroup_1[0].stateTime[1] = 2;
signalGroup_1[0].stateTime[2] = 16;
signalGroup_1[0].stateTime[3] = 2;
signalGroup_1[0].currentState = eRedYellow;
signalGroup_1[0].nextStateChange = 2;
signalGroup_1[0].permissiveMovement = 0;

spatPdu.spat.intersections.arrayValue[0].states.arrayValue[0].signalGroup = 1;
spatPdu.spat.intersections.arrayValue[0].states.arrayValue[0].state_time_speed.length = 1 ;
spatPdu.spat.intersections.arrayValue[0].states.arrayValue[0].state_time_speed.arrayValue[0].eventState = 3; // stop_And_Remain
spatPdu.spat.intersections.arrayValue[0].states.arrayValue[0].state_time_speed.arrayValue[0].timing.isValidFlag = 1;
```

Listing 5: SPAT

Der Code für die Zustandsänderung der Signalgruppen wird in der Funktion `UpdateSpat` geschrieben.

Wie in der Listing 6 gezeigt, bedeutet der Code "`signalGroup[iSignalGroup].nextStateChange -= 1`" die Verringerung der Dauer an, die ausgedrückt wird als "`signalGroup[ nextStateChange = signalGroup[iSignalGroup].nextStateChange - 1`". Der Wert von nextStateChange für jeden Zustand wird in der globalen Variablen "`struct SignalGroup signalGroup_1[5]`" festgelegt. Wobei der Code `[iSignalGroup]` für die Signalgruppe steht und durch eine For-Schleife auf alle Signalgruppen verwiesen werden kann.

Der Rest ist für den Übergang zwischen den Zuständen Rot, Rot-Gelb, Grün und Gelb geschrieben. Der rote Zustand wird als Beginn des Zyklus und der gelbe als Ende des Zyklus verwendet. Der Zyklus wird neu gestartet, indem festgelegt wird, dass der nächste Zustand rot ist, wenn die Ampel gelb ist. Der Übergang der Zustände innerhalb des Zyklus wird durch die Inkrementierung des Codes "`signalGroup[iSignalGroup].currentState += 1`" erreicht, was zusammenfassend als "`signalGroup[iSignalGroup] .currentState = signalGroup[iSignalGroup].currentState + 1`".Der Zustand der Ampel wird durch Inkrementierung in der Reihenfolge rot, rot-gelb, grün, gelb geändert.

```
for (iSignalGroup = 0 ; iSignalGroup < signalGroupCnt ;
++iSignalGroup)

  {

    signalGroup[iSignalGroup].nextStateChange -= 1;

    if (signalGroup[iSignalGroup].nextStateChange == 0)

    {

      if (signalGroup[iSignalGroup].currentState == eYellow)

        signalGroup[iSignalGroup].currentState = eRed;

      else

        signalGroup[iSignalGroup].currentState += 1;



      signalGroup[iSignalGroup].nextStateChange = sig
 nalGroup[iSignalGroup].stateTime[signalGroup[iSignalGroup].cur-
 rentState];

    }

  }
```

Listing 6: Code zum Umschalten des Status einer Signalgruppe im Normalzustand

Listing 6 zeigt den Übergang von Zeit und Zustand des Lichtsignals in seinem normalen Betriebszustand. Der beim Ausschalten des Signals verwendete Code wird im gleichen Format geschrieben (siehe Listing 7). Wenn der Lichtzustand gelb ist, wird der nächste Zustand auf grau gesetzt, und wenn das Licht grau ist, wird der nächste Zustand auf gelb gesetzt, so dass das Licht blinken kann.

```
if (signalGroup_2[iSignalGroup].nextStateChange == 0)

    {

      if (signalGroup_2[iSignalGroup].currentState == eYellow )

      {

        signalGroup_2[iSignalGroup].currentState = dark ;

      }

      else if(signalGroup_2[iSignalGroup].currentState == dark )

      {

         signalGroup_2[iSignalGroup].currentState = eYellow;

      }
```

Listing 7: Code zum Umschalten des Zustands einer Signalgruppe in den Aus-Zustand

Während des Betriebs einer Lichtsignalanlage treten Spitzenverkehrszeiten auf, in denen es notwendig ist, die Grünzeit der Hauptstraße zu verlängern, um die Verkehrsdichte auf der Hauptstraße zu verringern und den Hauptstraßenverkehr zu entlasten. Hier wird eine Methode zur Änderung des Timings vorgeschlagen: Addieren oder subtrahieren vom ursprünglichen Wert.

Neben der Methode des Addierens oder Subtrahierens vom ursprünglichen Wert ist es auch möglich, eine Zuweisungsmethode zu verwenden, bei der die Dauer direkt der Systemvariablen entspricht. Bei der zweiten Methode können die Werte, die geändert werden sollen, visuell eingegeben werden. Im Vergleich zur zweiten Methode ist es mit der hier verwendeten Methode jedoch einfacher, die Beziehungen zwischen Gruppen von Signalen zu ordnen und Konflikte zwischen Gruppen von Signalen zu vermeiden.Gleichzeitig ist der Wertebereich der Systemvariablen kleiner, da die Werte zu den ursprünglichen Werten addiert oder von ihnen subtrahiert werden, anstatt sie direkt zu verändern. Das System ist einfacher zu verwalten.

Zunächst wird jede Gruppe von Lichtsignalen in Tabelle 8 nochmals dargestellt. Die "`stateTime`" gibt die Dauer der einzelnen Zustände an, in der Reihenfolge rot, rot-gelb, grün, gelb. Der "`currentState`" gibt den Zustand zu Beginn des Betriebs an. "`nextStateChange`" gibt die verbleibende Zeit zu Beginn des Laufs an.

Signalgruppe 1	`stateTime: { 60, 2, 16, 2 },`
	`currentState: eRedYellow,`
	`nextStateChange: 2,`

Signalgruppe 2	`stateTime: { 40, 2, 36, 2 },`
	`currentState: eRed,`
	`nextStateChange: 30,`

Signalgruppe 3	`stateTime: { 70, 2, 6, 2 },`
	`currentState: eRed,`
	`nextStateChange: 30,`

Signalgruppe 4	`stateTime: { 40, 2, 36, 2 },`
	`currentState: eRed,`
	`nextStateChange: 40,`

Signalgruppe 5	`stateTime: { 70, 2, 6, 2 },`
	`currentState: eRed,`
	`nextStateChange: 70,`

Tabelle 8: Status jeder Signalgruppe

Die folgende Abbildung 31 zeigt den Übergang der Lichtsignale in einem Zyklus im Normalbetrieb. Ein Zyklus hat eine Gesamtdauer von 80 Sekunden.

Wert 1 bedeutet Rot, Wert 2 bedeutet rot-gelbes Licht, Wert 3 bedeutet grünes Licht und Wert 4 bedeutet gelbes Licht. Und die Signalgruppe 2 und die Signalgruppe 3 enden gleichzeitig rot und wechseln gleichzeitig den Status von rot zu rot-gelb, und die Signalgruppe 4 und die Signalgruppe 5 enden gleichzeitig gelb und wechseln gleichzeitig den Status von gelb zu rot.

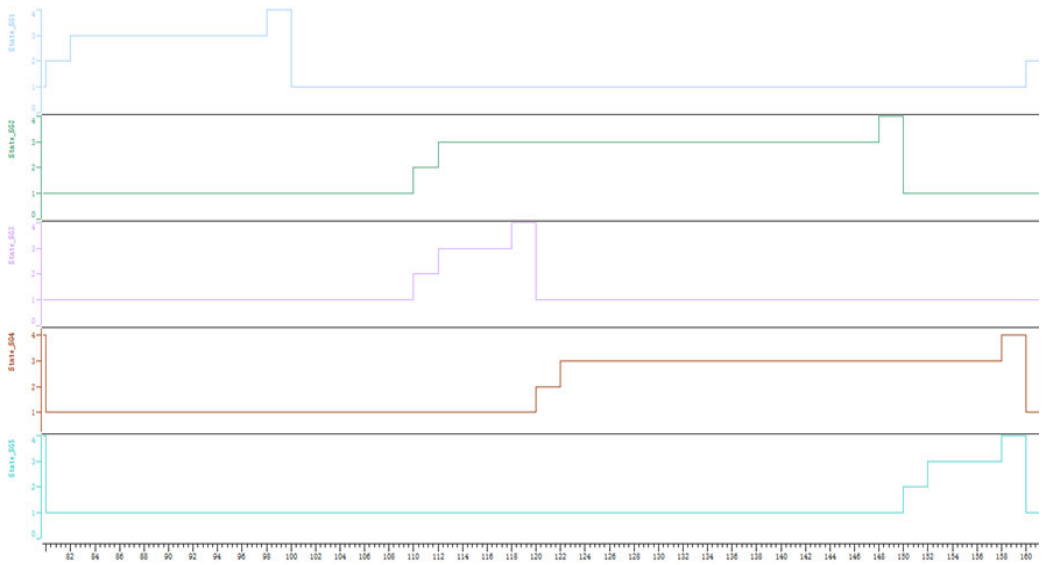

Abbildung 31: Impulsdiagramm der Signalgruppe

Für diese Arbeit wurde die Methode der Verlängerung der Dauer des grünen Lichts für die Signalgruppe 2 und die Signalgruppe 4 sowie die Dauer des grünen Lichts für das Signal auf der Hauptstraße geradeaus gewählt. Insgesamt gibt es zwei Szenarien, eines, bei dem die Grünlichtdauer um 1 bis 10 Sekunden verlängert wird, und das andere, bei dem die Grünlichtdauer um 11 bis 20 Sekunden verlängert wird.

Ein Teil des Codes für die Änderung der Dauer ist ebenfalls in der Funktion `UpdateSpat` enthalten.

Erstes Szenario: Das grüne Licht wird um 1 bis 10 Sekunden verlängert.

Diese Zeitverlängerung funktioniert, indem die eingegebene Zahl zur Dauer des grünen Lichts addiert wird und die eingegebene Zahl von der Dauer des roten Lichts in den Signalgruppen 2 und 4 subtrahiert wird, nachdem die Anzahl der Sekunden (1-10), um die das grüne Licht verlängert werden soll, in die Komponente Hauptstraße_gerade eingegeben worden ist. Die Auslösebedingung wird wirksam, wenn der aktuelle Status der Signalgruppe rot-gelb ist. Die Auslösung der Codes für Signalgruppe 2 und Signalgruppe 4 unterliegt der Signalgruppe 3. Konkret bedeutet dies, dass die Codes der Signalgruppe 2 und der Signalgruppe 4 ausgeführt werden müssen, nachdem der Code der Signalgruppe 3 wirksam geworden ist. Wenn der Code für die Signalgruppe 3 nicht ausgeführt wird, werden auch die Codes für die Signalgruppe 2 und die Signalgruppe 4 nicht ausgeführt. Wenn der Code in Kraft ist und die grüne Zeit wieder geändert werden soll, müssen der Zustand der Signalgruppen 2, 3 und 4 alle rot sein. (Die Codes in der Listing 8 zeigt die Signalgruppe 2 als Beispiel)

```
signalGroup_1[1].stateTime[0] = 40-@sysvar::RSU_Traf-
ficLight::Hauptstrasse_gerade_1_1;

signalGroup_1[1].stateTime[2] = 36+@sysvar::RSU_Traf-
ficLight::Hauptstrasse_gerade_1_1;
```

Listing 8: Code für die Änderung der Dauer Szenario 1

Der Code "`signalGroup_1[1].stateTime[0]`" zeigt den Rotlichtstatus der Signalgruppe 2 an, der Code "`stateTime[1]`" zeigt den Rot-Gelblichtstatus an, der Code "`stateTime[2]`" gibt den grünen Lichtstatus an, der Code "`stateTime[3]`" den gelben Lichtstatus.

Die verlängerte Grünzeit der Signalgruppe 2 im Intervall (1, 10) kollidiert nicht mit der Signalgruppe 1, so dass es nicht notwendig ist, die Dauer der Signalgruppe 1 an dieser Stelle zu ändern.

Das grüne Statusintervall der Signalgruppe 3 muss im Impulsdiagramm nach links verschoben werden, um es mit der Signalgruppe 2 synchron zu halten und um einen Konflikt mit dem grünen Intervall der Signalgruppe 4 zu vermeiden. Dies wird erreicht, indem die Dauer des roten Lichts im ersten Zyklus, in dem die Änderung der Dauer auftritt, von der Eingangsperiode subtrahiert wird und die ursprüngliche Dauer in den nachfolgenden Zyklen beibehalten wird.

Das Flussdiagramm (Siehe Abb.32) für den Bereich 1 bis 10 Sekunden ist unten dargestellt.

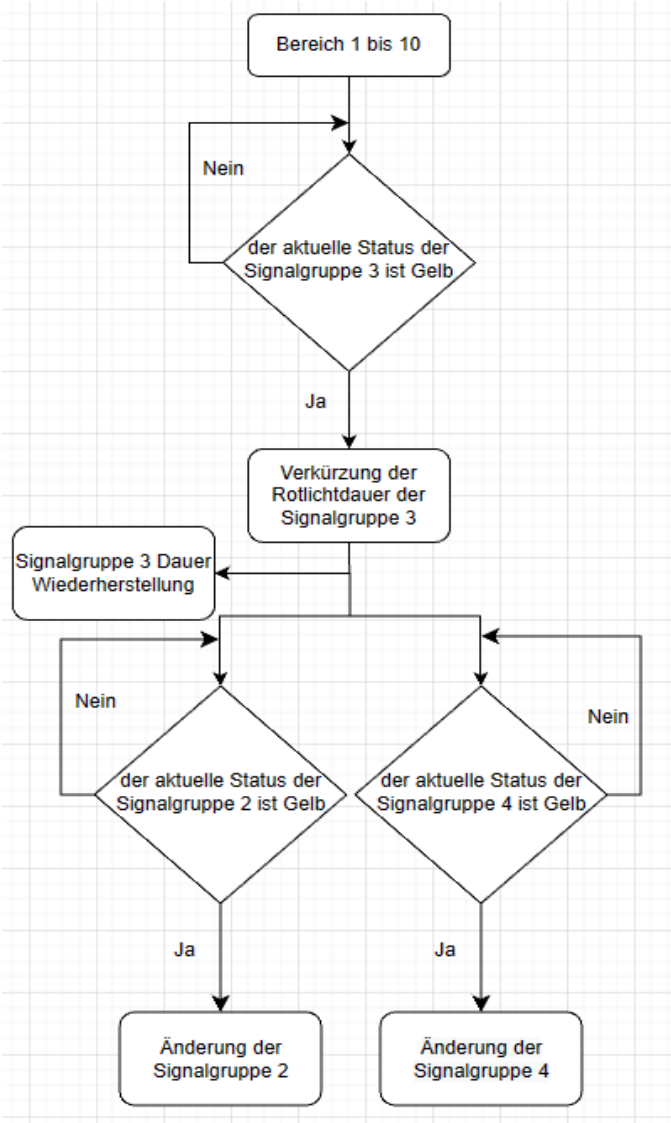

Abbildung 32: Flussdiagramm für den Variationsbereich (1, 10)

Zweites Szenario: Das grüne Licht wird um 11 bis 20 Sekunden verlängert.Das System erholt.

Im Eingangsintervall 11 bis 20 werden die Signalgruppe 2, die Signalgruppe 3 und die Signalgruppe 4 auf die gleiche Weise beurteilt wie im Intervall 1 bis 10. In den Zonen 11 bis 20 kollidiert jedoch die Grünzeit der Signalgruppe 2 mit der Grünzeit der Signalgruppe 1, so dass die Grünzeit der Signalgruppe 1 reduziert werden muss, um den Konflikt zu vermeiden. Der Code wird ausgeführt, wenn die Signalgruppe 1 rot-gelb ist. Auch hier muss der Code der Signalgruppe 1 ausgeführt werden, nachdem der Code der Signalgruppe 3 wirksam geworden ist. Wenn die grüne Dauer von Signalgruppe 2 und Signalgruppe 4 in Szenario 2 ein zweites Mal geändert werden soll, müssen die Signalgruppen 2, 3 und 4 erst auf Rot und die Signalgruppe 1 auf Grün geschaltet werden.

Der Code für die Signalgruppe 1 funktioniert wie folgt:Der Abstand zwischen den grünen Intervallen von Signalgruppe 1 und Signalgruppe 2 beträgt 10 Sekunden. Die Dauer des

grünen Lichts für Signalgruppe 1 ist also der Wert, der sich ergibt, wenn die ursprünglichen 16 Sekunden um den eingegebenen Wert reduziert und 10 Sekunden hinzugefügt werden.(Siehe Listing 9)

```
signalGroup_1[0].stateTime[0] = 60-10+@sysvar::RSU_Traffic-
Light::Hauptstrasse_gerade_1_0;

signalGroup_1[0].stateTime[2] = 16+10-@sysvar::RSU_Traffic-
Light::Hauptstrasse_gerade_1_0;
```

Listing 9: Code für die Änderung der Dauer Szenario 2

Es müssen Lösungen gefunden werden, um das Lichtsignalsystem nach dem Ende der Hauptverkehrszeit wieder auf seine ursprünglichen Werte zurückzusetzen. Für die Signalgruppe 2 und die Signalgruppe 4 ist es nur erforderlich, dass der im roten Zustand reduzierte Wert und der entsprechende im grünen Zustand erhöhte Wert auf Null gesetzt werden, um die Signalgruppe 2 und die Signalgruppe 4 in den ursprünglichen Zustand zurückzuführen, während das Grünintervall der Signalgruppe 3 um den eingegebenen Wert nach rechts verschoben werden muss. Wenn der eingegebene Wert größer als 10 ist, wird gleichzeitig die Signalgruppe 1 wiederhergestellt.

Die folgende Abbildung 39 zeigt das Flussdiagramm, wenn der Eingangswertbereich 11 bis 20 beträgt und wenn er wiederhergestellt wird.

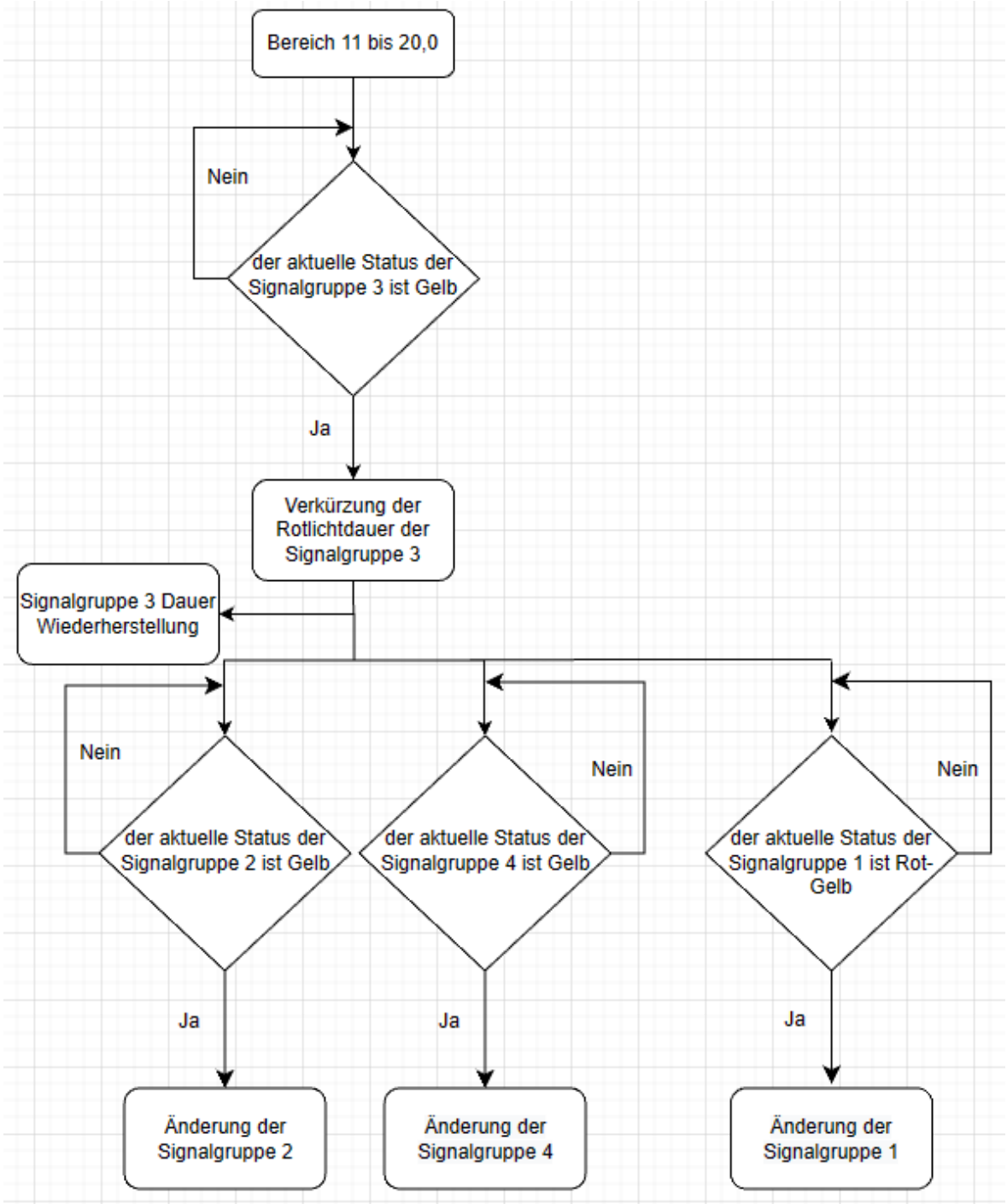

Abbildung 33: Flussdiagramm für den Variationsbereich (11, 20) und 0

3.2.5.2 Code in den Dateien SimRsuTL.can

Der in SimRsuTL.can zu bearbeitende Code ist für das Senden von MAP- und SPAT-Nachrichten, den Empfang von CAM-Nachrichten und die Anwendung von Systemvariablen.

Der mit den Systemvariablen verbundene Code ist in Anlagen, Teil 3 dargestellt.

Die Systemvariable RSU_TrafficLight::State_SG3 wird für die Wiederherstellung der Signalgruppe 3 nach dem Panning verwendet.

Die Systemvariable RSU_TrafficLight::Hauptstrasse_gerade wird verwendet, um den Systemvariablen Werte zuzuweisen, die den einzelnen Signalgruppen entsprechen. Die Umwandlung der Dauer der grünen Ampel erfolgt durch Addition der entsprechenden Systemvariablen zum Startwert. Die Rotlichtdauer wird transformiert, indem die entsprechende Systemvariable vom Startwert subtrahiert wird.

Insgesamt gibt es drei Funktionen in der Datei SimRsuTL.can:

- Die Funktion `OnRx` dient zum Empfang des CAM-Nachrichts vom Fahrzeug.
- Die Funktion `OnTxMAP` wird zum Senden von MAP-Nachrichten verwendet, die die Funktion MapInit in der Datei SinRsuTL.cin enthalten.
- Die Funktion `OnTxSPAT` wird zum Senden von SPAT-Nachrichten verwendet, die die Funktion `SpatInit` und Funktion `UpdateSpat` in der Datei SinRsuTL.cin enthalten. Die Anzeige des Ampelstatus und Ein- und Ausschalten des Ampelstatus ist ebenfalls in der Funktion `OnTxSPAT` zusammengefasst.

Die Systemvariable "button" dient zur Steuerung der Umschaltung zwischen dem Normalbetrieb des Lichtsignals und dem ausgeschalteten Zustand. Wenn die Systemvariable "button" auf 1 gesetzt ist, arbeitet das Lichtsignal normal. Wenn "button" auf 0 gesetzt ist, sind die Lichtsignale ausgeschaltet, was dadurch angezeigt wird, dass alle Lichtsignalgruppen abwechselnd in grau und gelb blinken.

Am Beispiel der Signalgruppe 1 (Siehe Listing 10) wird die if-Anweisung verwendet, um den normalen Laufzeitcode auszuführen, wenn die bedingte Systemvariable "button" den Wert 1 hat.

```
 if(@sysvar::RSU_TrafficLight::button  == 1 )
{

   switch (signalGroup_1[0].currentState)
   {
     case eRed:       @sysvar::RSU_TrafficLight::State_SG1 = 1; break;
     case eRedYellow: @sysvar::RSU_TrafficLight::State_SG1 = 2; break;
     case eGreen:     @sysvar::RSU_TrafficLight::State_SG1 = 3; break;
     case eYellow:    @sysvar::RSU_TrafficLight::State_SG1 = 4; break;
   }

   @sysvar::RSU_TrafficLight::TimeToSwitch_SG1 = (double)signalGroup_1[0].nextStateChange;

}
```

Listing 10: Normaler Laufzeitcode

Der Code "`signalGroup_1[0]`" bedeutet Signalgruppe 1, 0 kann durch eine Zahl zwischen 0 und 4 ersetzt werden, die jeweils die Signalgruppen 1-5 repräsentiert. Und verwenden die

switch case-Anweisung, um den Zustand der Ampel an die Systemvariable RSU_Trafficlight::State_SG1 zu binden.

Der Code für das Schließen wird auf der Grundlage des Codes für den Normalbetrieb geschrieben. Wenn die bedingte Systemvariable "button" 0 ist, wird der Code für das Schließen ausgeführt (siehe Listing 11).

```
if(@sysvar::RSU_TrafficLight::button  == 0)
{

   switch (signalGroup_2[0].currentState)
    {
      case dark:       @sysvar::RSU_TrafficLight::State_SG1 = 0; break ; // dark
      case eYellow:    @sysvar::RSU_TrafficLight::State_SG1 = 4; break ; // pre_Movement
    }
    @sysvar::RSU_TrafficLight::TimeToSwitch_SG1 = (double)signalGroup_2[0].nextStateChange;
}
```

Listing 11: Nicht-normaler Laufzeitcode

3.2.6 Panel mit Lichtsignalen

Unter diesem Titel wird eine Tafel für die Lichtsignalanlage an der Kreuzung Tschirnerplatz in Mittweida erstellt.

Die Anforderungen an die Aufgabe sind wie folgt:

- Signalanzeige für fünf Signalgruppen, Richtungsanzeige und Daueranzeige
- Ampeln können manuell ausgeschaltet werden, wobei gelbe und graue Lichter in Intervallen blinken
- Verlängerung der Grünlichtdauer für die Signalgruppe Hauptstraße geradeaus

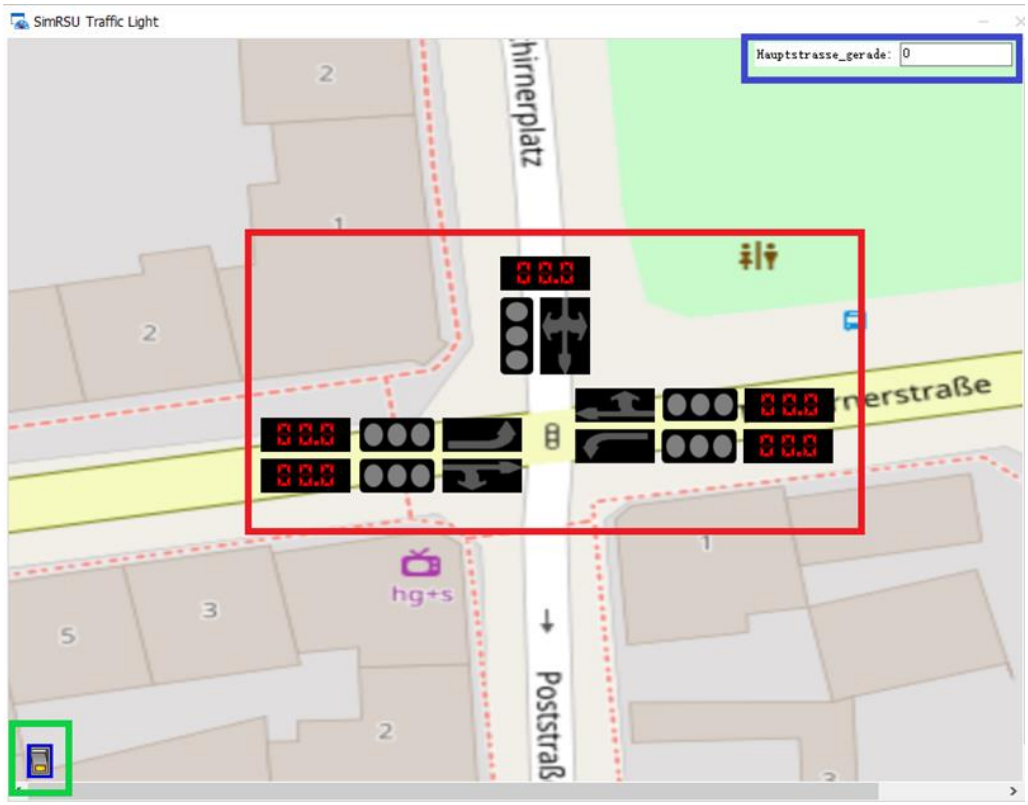

Abbildung 34: Panel der Lichtsignalanlage

Wie in der Abbildung 34 oben dargestellt:

Die Komponenten in der roten Box sind die Lichtsignalanzeigen, die aus insgesamt fünf Signalen bestehen, die jeweils aus drei Komponenten zusammengesetzt sind, im Fall der Signalgruppe 1, die aus der " LCD Control und zwei Switch/Indicator " aus dem Bausatz besteht. Die mit der " LCD Control " der Signalgruppe 1 verbundene Systemvariable ist "TimeToSwitch_SG1" in "RSU_TrafficLight", die zur Anzeige der Dauer der roten, grünen und gelben Lichter der Signalgruppe 1 im Normalzustand verwendet wird. Einer der "Schalter/Anzeiger" wird zur Erstellung der Ampelanzeige verwendet. Die damit verbundene Systemvariable ist "State_SG1" in "RSU_TrafficLight".Es ist in fünf Teile unterteilt, die in dieser Reihenfolge grau, rot, rot-gelb, grün und gelb darstellen.Dies ist ein direkter Verweis auf das Ampeldiagramm in der europäischen Norm "Car2x configuration". Ein weiterer "Schalter/Anzeiger" wird verwendet, um Richtungsanzeigen zu erstellen. Damit verbunden ist auch der "State_SG1" in der Systemvariablen "RSU_TrafficLight", der auf die gleiche Weise wie eine Ampelanzeige erstellt und angezeigt wird.

Der Knopf im grünen Kasten ist der Ampelschalter, der aus dem " Switch/Indicator " im Bausatz hergestellt wird. Damit verbunden ist der "Knopf" in der Systemvariablen "RSU_TrafficLight", der immer auf 1 gesetzt ist und anzeigt, dass die Ampelanlage im Normalbetrieb ist. Wenn Ampelschalter angeklickt wird, wird die Systemvariable auf 0 gesetzt, was bedeutet, dass die Ampelanlage blinkt (die Ampelanlage kann als ausgeschaltet betrachtet werden).

Die Komponente in der blauen Box wird verwendet, um die Dauer des grünen Lichts auf der Hauptstraße geradeaus zu ändern. Diese Komponente wird aus der 'Input/Output Box' im Toolkit erstellt. Mit ihr verbunden ist die Systemvariable "Hauptstraße_gerade" in der "RSU_TrafficLight", die immer auf 0 gesetzt ist, was bedeutet, dass sich die Dauer des grünen Lichts für die Hauptstraße geradeaus nicht ändert. In dieser Komponente können Werte im Bereich von 0 bis 20 eingegeben werden. Die Dauer des grünen Lichts auf der Hauptstraße wird beim nächsten Zyklus zur ursprünglichen Dauer addiert.

4 Evaluierung durch Laborversuche mit simulierten Fahrzeugen

Das Fahrzeug als häufigster Verkehrsteilnehmer spielt eine wichtige Rolle im Verkehr. In diesem Kapitel werden neue Fahrzeugknoten verwendet, um mit den grundlegenden Funktionen von Fahrzeugen im Straßenverkehr und ihrer Interaktion mit Ampeln zu experimentieren. Das Fahrzeug muss eine bestimmte Strecke befahren und mit den Ampeln interagieren.

4.1 Projekte erstellen

Die für die Aufgabe benötigten Knoten werden entsprechend den Schritten der car2x-Projekterstellung angelegt.

Das bestehende Dateiprojekt für den Ampelanlageneditor wird geöffnet. Die Netzknoten für die Fahrzeuge werden in der Datenbank "EU_ApplMsg_20221103.xml" angelegt. Bei diesem Praktikumsversuch wurden zwei den Wagen entsprechende Knoten mit den Namen "FAHRZEUG_1" und "FAHRZEUG_2" erstellt. Den beiden Fahrzeugknoten wurden CAM-Signale hinzugefügt und die Attribute wurden bearbeitet. Die folgende Abbildung 35 zeigt den Knoten "FAHRZEUG_1" als Beispiel.

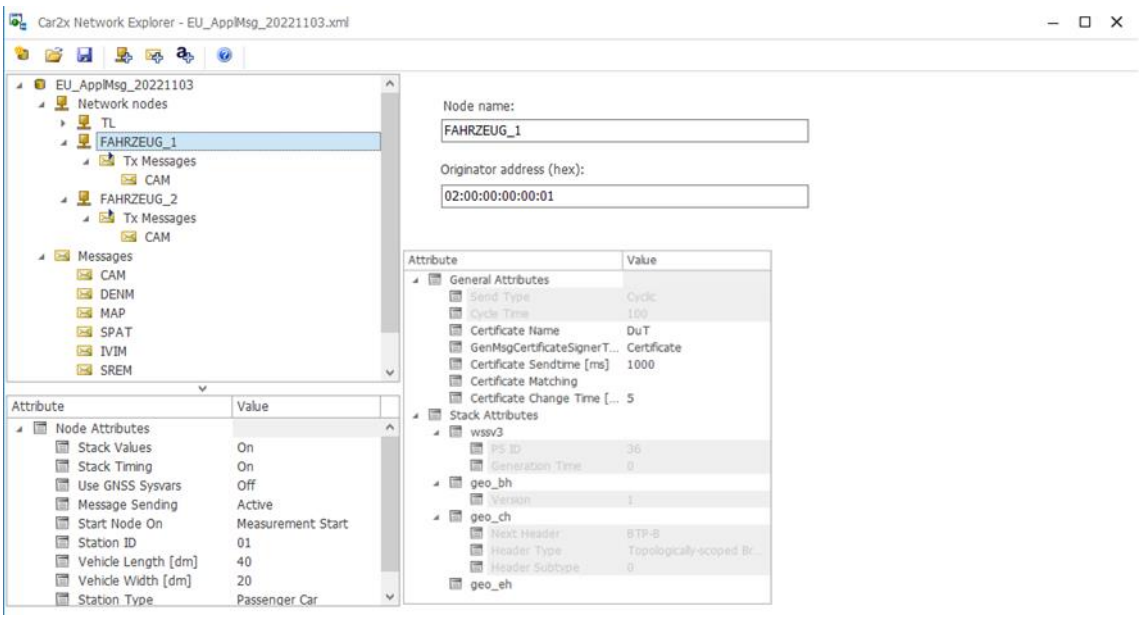

Abbildung 35: Eigenschaften des simulierten Fahrzeugs

Mit der Funktion "Node Sync" wird ein Knoten zum Bus hinzugefügt und die CAN-Datei erstellt. (Siehe Abb.36)

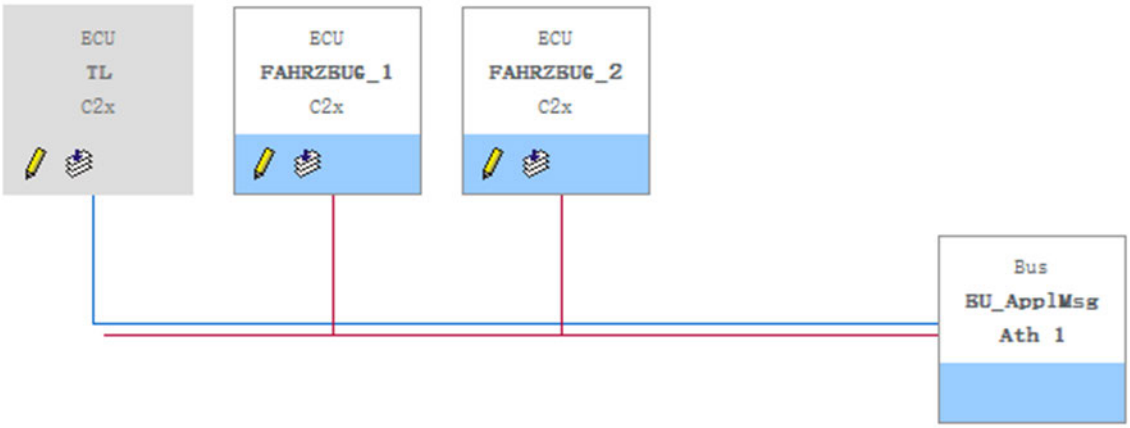

Abbildung 36: Knoten für simulierte Fahrzeuge

4.2 Hinzufügen von Systemvariablen

Wie in der folgenden Abbildung 37 dargestellt, nehmen das Fahrzeug 1 als Beispiel und fügen die für die Fahrzeugsimulation benötigten Systemvariablen in den "Systemvariablen". Die im Fahrzeug enthaltenen Systemvariablen lassen sich grob in zwei Teile unterteilen: die Variablen, die die Eigenschaften des Fahrzeugs selbst betreffen, und die Variablen, die die Anzeige der Ampelanlage im Fahrzeug betreffen.

FAHRZEUG_1

Acceleration	Double	0	-10	10	Configuration
Current_head	Double	0	-	-	Configuration
Current_lat	Double	0	-	-	Configuration
Current_lon	Double	0	-	-	Configuration
DisplayState	Int32	0	0	4	Configuration
Distance	Int32	0	0	300	Configuration
DistancePB	Double	0	0	150	Configuration
DistToTrafficLight	Int32	0	0	-	Configuration
Horizont	Double	100	50	300	Configuration
SendPathPoints	Int32	0	0	1	Configuration
Speed	Double	0	0	200	Configuration
timeReadDuTSpeed	Double	0	0	180	Configuration
TlState	Int32	1	0	4	Configuration
Tlstate_1	Int32	1	0	4	Configuration
Tlstate_2	Int32	1	0	4	Configuration
Tlstate_3	Int32	1	0	4	Configuration
Tlstate_4	Int32	1	0	4	Configuration
Tlstate_5	Int32	1	0	4	Configuration
Tltimeall_1	Double	0	0	10	Configuration
Tltimeall_2	Double	0	0	10	Configuration
Tltimeall_3	Double	0	0	10	Configuration
Tltimeall_4	Double	0	0	10	Configuration
Tltimeall_5	Double	0	0	10	Configuration
TlTimeToChangeState	Double	0	0	10	Configuration

Eigenschaften des Fahrzeugs selbst

Kartierung von Ampel-Eigenschaften

Abbildung 37: Systemvariablen für simulierte Fahrzeuge

Die Systemvariablen sind wie folgt definiert:

FAHRZEUG_1::TlState FAHRZEUG_1::TlTimeToChangeState	Diese beiden Systemvariablen werden im Ampelsystem für die Anzeige der Signalgruppen einer bestimmten Strecke im Fahrzeug und für die Interaktion mit dem Fahrzeug verwendet.
FAHRZEUG_1:: TlState_1, _2, _3, _4, _5 FAHRZEUG_1:: Tltimeall_1, _2, _3, _4, _5	Diese Systemvariablen werden für die Anzeige der Ampelanlage in Fahrzeugen verwendet.
FAHRZEUG_1::Acceleration	Diese Systemvariable wird verwendet, um anzuzeigen, ob die Route neu geladen wird oder nicht.
FAHRZEUG_1::Current_lat FAHRZEUG_1::Current_lon FAHRZEUG_1::Current_head	Diese drei Systemvariablen werden verwendet, um die Position des Fahrzeugs in Echtzeit anzuzeigen.
FAHRZEUG_1::Distance FAHRZEUG_1::DistancePB FAHRZEUG_1::DistToTrafficLight FAHRZEUG_1::Horizont	Diese vier Systemvariablen werden verwendet, um den Abstand zwischen einem Fahrzeug und einem Lichtsignal in Echtzeit sowohl numerisch als auch grafisch darzustellen.
FAHRZEUG_1::DisplayState	Diese Systemvariable wird für die Anzeige der verschiedenen Interaktionszustände verwendet.
FAHRZEUG_1::Speed FAHRZEUG_1::timeReadDuTSpeed	Diese beiden Systemvariablen werden für die Anzeige der Fahrzeuggeschwindigkeit verwendet.

Tabelle 9: Systemvariablen für simulierte Fahrzeuge

4.3 Sicherheitszertifikat

Die car2x-Option in File-Options-Bus System/Protocol wird eingeschaltet und ein neues pseudonymes Zertifikat wird zum Root-Zertifikat hinzugefügt: ROOT, das bearbeitet und benannt wird (Siehe Abb.38). Das pseudonyme Zertifikat wird dann den neu angelegten Knoten "FAHRZEUG_1" und "FAHRZEUG_2" zugewiesen.

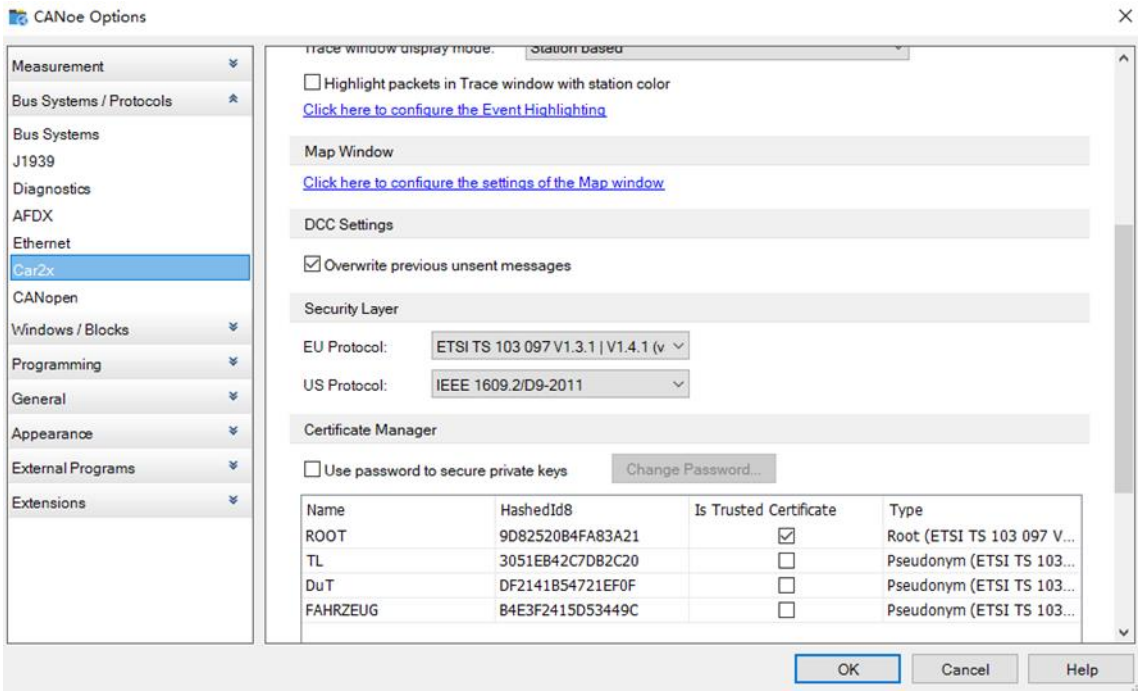

Abbildung 38: Zertifikat für simulierte Fahrzeuge

Es ist zu beachten, dass die Lichtsignalanlage und das simulierte Fahrzeug in der Sicherheitsschicht die gleiche Protokollversion haben müssen. Unterschiedliche Protokollversionen führen dazu, dass der analoge Knoten keine Signale empfängt und das Programm nicht richtig funktioniert.

4.4 Programmierung

Die Datei DuT.can besteht ebenfalls aus fünf Abschnitten und hat die gleichen Funktionen wie die Datei SimRsuTL.can:

- **Includes**: In diesem Abschnitt werden die für die Programmierung benötigten cin-Dateien hinzugefügt.

- **Variables**: Es wird verwendet, um die in dieser Programmierdatei benötigten Variablen zu setzen.

- **System**: Es enthält Ereignisbehandler für Systemereignisse.

- **Value Objects**: Es enthält Ereignishandler für Systemvariablen.

- **Function**: Es enthält die für die Implementierung des Programms erforderlichen Funktionen.

Die zu Includes hinzugefügte cin-Datei lautet:

- **ASNV_Template_BaseDatatypes.cin:** Diese Datei wurde direkt aus dem "Car2x-system"-Beispiel extrahiert, ohne jegliche Änderungen. Diese Datei zeigt die Datentypen, die im gesamten Programmierprojekt benötigt werden.

- **ASNV_Template_CAM.cin:** Diese Datei wurde direkt aus dem "Car2x-system"-Beispiel extrahiert, ohne jegliche Änderungen. Die Datei kompiliert das konkrete Format der CAM-Nachricht.

- **Car2xAPI.cin:** Diese Datei wurde direkt aus dem "Car2x-system"-Beispiel extrahiert, ohne jegliche Änderungen. Diese Datei wird verwendet, um APIs zu schreiben, die für das gesamte Projekt gemeinsam ist.

Der Code in der Datei DuT.cin besteht aus drei Hauptabschnitten: der Ermittlung der Fahrzeuggeschwindigkeit, der Anzeige der Lichtsignalanlage im Fahrzeug und Übermittlung von Fahrzeuginformationen.

4.4.1.1 Der Ermittlung der Fahrzeuggeschwindigkeit

Das Fahrzeug muss die SPAT-Nachricht anzeigen, die es von der Lichtsignalanlage erhält, und auf der Grundlage dieser Nachricht eine Geschwindigkeitsbeurteilung vornehmen.

Das folgende Flussdiagramm 39 zeigt die Geschwindigkeitsermittlung eines Fahrzeugs auf der Grundlage des Status einer Lichtsignalgruppe. Die erste Geschwindigkeitsbewertung basiert auf dem Status der Ampel, die 25 bis 100 Meter vor der Ampel aufgestellt ist. Das Fahrzeug reduziert seine Geschwindigkeit auf 30 km/h, wenn die Ampel rot, gelb oder rot-gelb ist, und hält 50 km/h, wenn die Ampel grün ist. In einem Abstand von 0-25 Metern zwischen dem simulierten Fahrzeug und der Ampelanlage nimmt das simulierte Fahrzeug eine zweite Geschwindigkeitsbewertung auf der Grundlage des Ampelstatus vor. Wenn die Ampel gelb, rot oder rot-gelb ist, wird das Fahrzeug an der Kreuzung anhalten. Wenn die Ampel grün ist und die Dauer des Signals weniger als 2 Sekunden beträgt, hält das Fahrzeug an der Kreuzung an, wenn es nicht genügend Zeit hat, die Kreuzung zu überqueren. Wenn die Ampel grün ist und die Dauer des Signals mehr als 2 Sekunden beträgt, passiert das Fahrzeug die Kreuzung.

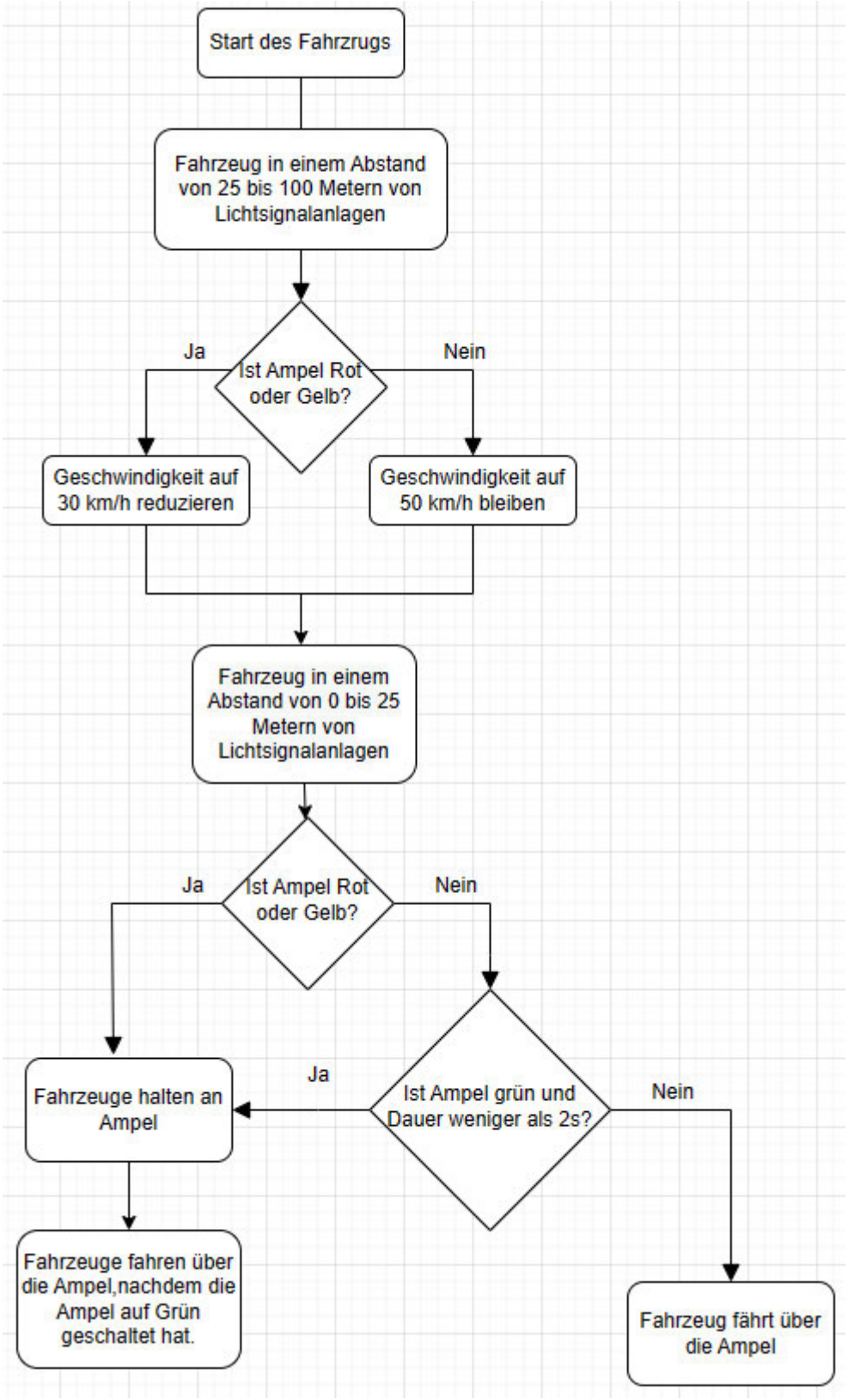

Abbildung 39: Flussdiagramm des simulierten Fahrzeugs

Im “Scenario editor” wird die Geschwindigkeit des simulierten Fahrzeugs durch Hinzufügen von KEYPOINT in der Zeitleiste festgelegt, und die meisten Anfangspunktgeschwindigkeiten sind auf 50 km/h eingestellt. In der Capl-Programmierung

kann mit der Funktion "`C2xSetStationAttributeDouble`" die Geschwindigkeit im zugewiesenen KEYPOINT geändert werden, so dass das Fahrzeug mit der Ampelanlage interagieren kann. Der Code "`(@sysvar::FAHRZEUG_1:: Distance <= 25.0)&&(gTL.state == 1|2|3 )`" bedeutet, dass, wenn das simulierte Fahrzeug weniger als 25 Meter von der Ampel entfernt ist und die Ampel gelb oder rot ist, wie in Listing 12 unten gezeigt. Der entsprechende KEYPOINT wird auf Null gesetzt, um ein Fahrzeug zu simulieren, das an einer Kreuzung anhält.

```
if   ((@sysvar::FAHRZEUG_1::Distance   <=25.0)&&(gTL.state   ==
1|2|3 ))

  {

    C2xSetStationAttributeDouble("FAHRZEUG_1", "Speed", 3, 0);

    C2xSetStationAttributeDouble("FAHRZEUG_1", "Speed", 4, 0);

    C2xSetStationAttributeDouble("FAHRZEUG_1", "Speed", 5, 0);

  }
```

Listing 12: Beispiele für Geschwindigkeitsbeurteilungen

Nach Ablauf der Wartezeit an einer roten Ampel muss das Fahrzeug vom Warten zum Fahren übergehen. Mit der CAPL-Programmierung kann das Fahrzeug am Ende der Rotlichtphase selbst starten. Der Code wird ausgelöst, wenn der Abstand zwischen dem Fahrzeug und dem Lichtsignal 0 ist, die Geschwindigkeit des Fahrzeugs 0 ist und das Signal rot ist.

```
for(idx = 3;idx < ((gTL.timeToSwitch/10) + 3);idx++)

    {

      C2xSetStationAttributeDouble("FAHRZEUG_1", "Speed", idx,
0);

    }
```

Listing 13: Fahrzeug-Warteschlangen-Code

Listing 13 zeigt den Code, während das Fahrzeug wartet. `(gTL.timeToSwitch/10)` gibt die verbleibende Zeit bei Rot an, und idx gibt den dieser Zeit entsprechenden Keypoint an. Allen Keypoints während dieser Zeit wird der Wert 0 zugewiesen.

Der Code in Listing 14 zeigt an, dass die verbleibenden Keypoints nacheinander einen inkrementellen Wert der gleichen Beschleunigung erhalten.

```
for(idx = ((gTL.timeToSwitch/10) + 3);idx < (42+2);idx++)

    {

      speedToSet = 0.0 /*initial speed*/ + (idx - ((gTL.time-
ToSwitch/10) + 3)) * (50/*target speed*/ - 0.0/*initial
speed*/) / (42 - gTL.timeToSwitch/10);

      C2xSetStationAttributeDouble("FAHRZEUG_1", "Speed",
idx, speedToSet);

    }
```

Listing 14: Beschleunigung des Fahrzeugs

Je nach den Anforderungen der Aufgabe muss das Fahrzeug in Bewegung gehalten werden, wenn es zu seinem Ausgangspunkt zurückkehrt. Dies kann mit den Funktionen C2xStopScenario und C2xStartScenario in den CAPL-Programmierfunktionen erreicht werden. Die Auslösebedingung ist, dass das Fahrzeug an der angegebenen Position ankommt. Der detaillierte Breiten- und Längengrad muss über "Graphics" im Ribbon "Analysis" ermittelt werden. Gleichzeitig werden alle Geschwindigkeiten für den nächsten Fahrzyklus auf die im Umgebungseditor eingestellten Originalwerte zurückgesetzt.(Siehe Listing 15)

```
//Callback function when state of scenario meets condtions,called at the end of scenario
on sysvar FAHRZEUG_1 ::Current_lat
{
  OnScenarioStateChange();
}

void OnScenarioStateChange()
{
  if((@sysvar::FAHRZEUG_1::Current_lat==50.986664022664)&&(@sysvar::FAHRZEUG_1::Current_lon==12.975654437856))
  {
      C2xSetStationAttributeDouble("FAHRZEUG_1", "Speed", 0, 50);
      C2xSetStationAttributeDouble("FAHRZEUG_1", "Speed", 1, 50);
      C2xSetStationAttributeDouble("FAHRZEUG_1", "Speed", 2, 50);
      C2xSetStationAttributeDouble("FAHRZEUG_1", "Speed", 3, 50);

    C2xStopScenario();
    C2xStartScenario ();
  }
  else
  {

  }
}
```

Listing 15: Codes für das Radfahren von Fahrzeugen

Der Code im schwarzen Feld wird verwendet, um die Funktion `OnScenarioStateChange()` auszulösen, die ausgelöst wird, wenn sich der Wert der Systemvariablen FAHRZEUG_1 ::Current_lat ändert.

Der Code im roten Feld gibt den genauen Breiten- und Längengrad des simulierten Fahrzeugs am Ende des Laufs an.

Der Code im blauen Kästchen gibt die Geschwindigkeit des simulierten Fahrzeugs an jedem Punkt des Szenario-Editors an, der wiederhergestellt wird. Insgesamt gibt es 48 KEYPOINTs, die zurückgesetzt werden müssen.

Der Code im grünen Kästchen dient dazu, den Umgebungseditor ein- und auszuschalten, so dass das Fahrzeug schließlich weiterfahren kann, sobald es den Startpunkt erreicht hat.

4.4.1.2 Übermittlung von Fahrzeuginformationen

Das Senden von Fahrzeuginformationen erfolgt mit der Funktion `OnPreCAM` und der Funktion `UpdatePosition`. Die Funktion `UpdatePosition` dient zum Sammeln und Aktualisieren der Positionsdaten des Fahrzeugs. (Längengrad, Breitengrad, Geschwindigkeit, Fahrtrichtung) Es ist in der Funktion `OnPreCAM` enthalten und wird gemeinsam gesendet.

4.4.1.3 Der Anzeige der Lichtsignalanlage

Die Darstellung der Lichtsignalanlage im Fahrzeug wird durch drei Funktionen realisiert: `OnRx`, `RxTlMAP` und `RxTlSPAT`. Die Funktion `OnRx` dient dazu, MAP-Nachrichten und SPAT-Nachrichten von der Lichtsignalanlage zu sammeln, und die Funktion `RxTlMAP` ist für das Parsen der MAP-Nachrichten zuständig, die aus der Identifizierung der Lichtsignalanlage und ihrer geografischen Lage bestehen. Die Funktion `RxTlSPAT` ist für die SPAT-

Nachrichten zuständig, die den Echtzeitstatus und die Dauer der einzelnen Signalgruppen enthalten.

Am Beispiel der Lichtsignalgruppe 4 auf der aktuellen Straße, auf der das simulierte Fahrzeug fährt (Siehe Tabelle.10) Die Funktion "Switch Case" wird verwendet, um den Status der Ampel mit dem Auto zu korrelieren. Der Status der Ampel wird korrekt und rechtzeitig im Wagen angezeigt.

	Ample	**Fahrzeug**
Rotes Licht	Case3	gTL.State = 2
Gelbe und rot-gelbe Lichter	Case4, Case7, Case8	gTL.State =1, gTL.State =3
Grünes Licht	Case5 und Case 6	gTL.State = 4
Graue Lampe	Case1	gTL.State = 0

Tabelle 10: Status Verbände zwischen Ample und Fahrzeug

Die Ermittlung der Entfernung zwischen Fahrzeug und Lichtsignalanlage erfolgt durch die Funktion DistanceUpdate. Die Funktion DistanceUpdate ermittelt, ob sich das Fahrzeug im relevanten Bereich der ITS-Station befindet und ob sich das Fahrzeug der Station nähert oder von ihr entfernt.

```
if ((0.0 <= distanceNew) && (distanceNew <= DistanceInfinite))

  {

    station.distanceOld = station.distance;

    station.distance = (distanceNew > station.relevanceArea) ?
distanceNew - station.relevanceArea : 0.0 ;

  }
```

Listing 16: Entfernungsbestimmung

Wie in Listing 16 dargestellt, befindet sich das Fahrzeug außerhalb des Relevanzbereichs des ITS, wenn distanceNew größer ist als station.relevanceArea. Wenn distanceNew kleiner oder gleich station.relevanceArea ist, befindet sich das Fahrzeug innerhalb des relevanten Bereichs des IVS und der Wert von station.distance wird 0.

Die Annäherung und Abfahrt eines Fahrzeugs wird durch "approaching" bestimmt, wobei "approaching" gleich 1 ist, um anzuzeigen, dass sich das Fahrzeug der ITS-Station nähert, und "approaching" gleich 0 ist, um anzuzeigen, dass sich das Fahrzeug von der ITS-Station entfernt, wodurch bestimmt wird, ob die Lichtsignalinformationen auf der Tafel angezeigt werden oder nicht.(Siehe Listing 17)

```
if ((station.distanceOld - station.distance ) / station.distance >
0.04) station.approaching = 1;

  else if ((station.distance - station.distanceOld) / station.dis-
tanceOld > 0.01) station.approaching = 0;
```

Listing 17: Beurteilung der Nähe oder Entfernung

4.5 Panel mit Fahrzeug

Dieser Abschnitt beschreibt die Erstellung des Panels für das simulierte Fahrzeug,und verwendet das simulierte Fahrzeug 1 als Beispiel.

Das Panel der simulierten Fahrzeuge benötigt eine Anzeige, die alle Signalgruppen der Lichtsignalanlage darstellt und den Abstand des Fahrzeugs zum Signal anzeigt.

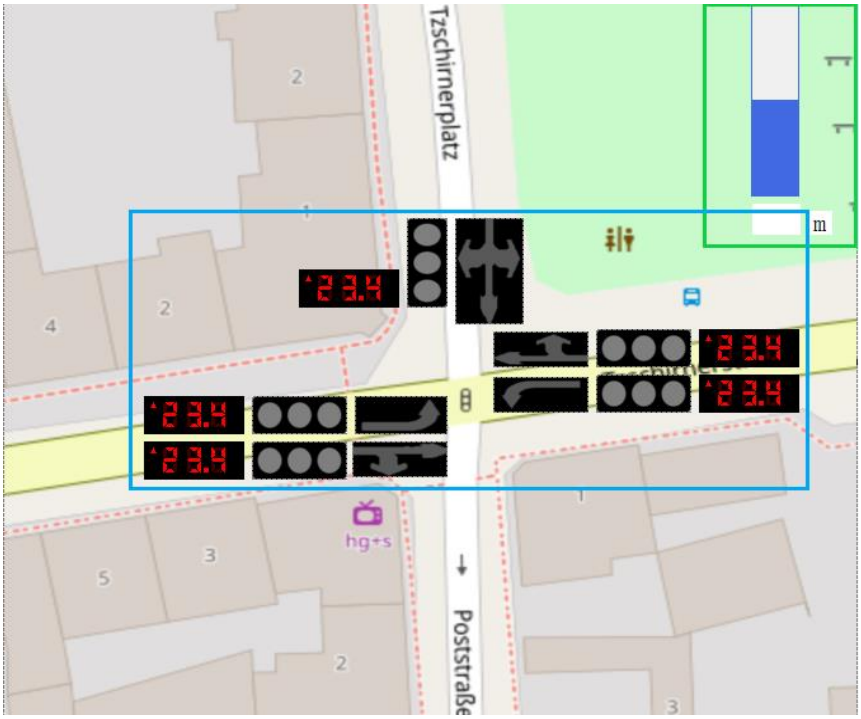

Abbildung 40: Panel für simulierte Fahrzeuge

Wie oben(Siehe Abb.40)gezeigt, wird die Komponente im grünen Kasten verwendet, um die Entfernung zwischen dem Fahrzeug und dem Lichtsignal anzuzeigen.

Die Komponente im blauen Kasten dient zur Darstellung der Abbildung der Lichtsignalanlage im simulierten Fahrzeug. Wie bei der Tafel für die Lichtsignalanlage hat jede Signalgruppe drei Komponenten: den Zähler, die Signalzustandsanzeige und die Richtungsanzeige. Anders als bei der Lichtsignalanlage werden bei der Anzeige des Signalzustandes im simulierten Fahrzeug drei Zustände verwendet: rot, gelb und grün.

4.6 Routen-Editor

Die Schaltfläche "Scenario-Editor" befindet sich in der Multifunktionsleiste "Tool". Die Route für Fahrzeug 1 und Fahrzeug 2 wird in diesem Menüband erstellt.

Im "Scenario" auf der linken Seite des Fensters wird eine Route mit einer Station hinzugefügt. Der Name der Strecke muss so bearbeitet werden, dass er mit dem Namen des Knotens in der Datenbank übereinstimmt, sonst wird das Szenario des Fahrzeugknotens im Menüband "Scenario-Manager" nicht erkannt. Die Farbe der Fahrzeuge und Routen sowie die Größe der Fahrzeuge in der Bildrealität können im Eigenschaftsbearbeitungsfenster „Properties“ geändert werden.

KEEYPOING kann über die Zeitleiste "Timeline" unterhalb des Anzeigebereichs zur Route hinzugefügt werden. An dieser Stelle kann die Anfangsgeschwindigkeit des simulierten Fahrzeugs eingestellt werden.

Beispiel für die Route von Fahrzeug 1:

Das simulierte Fahrzeug fährt von der Tzschirnerstraße in Mittweida ab. Unterwegs passiert es die Ampelanlage an der Kreuzung Tzschirnerstraße/Poststraße und fährt dann geradeaus über die Kreuzung. Das Fahrzeug kehrt dann über die vorgesehene Strecke (wie Abb.41 dargestellt) zum Ausgangspunkt zurück und fährt auf der vorgesehenen Straße weiter.

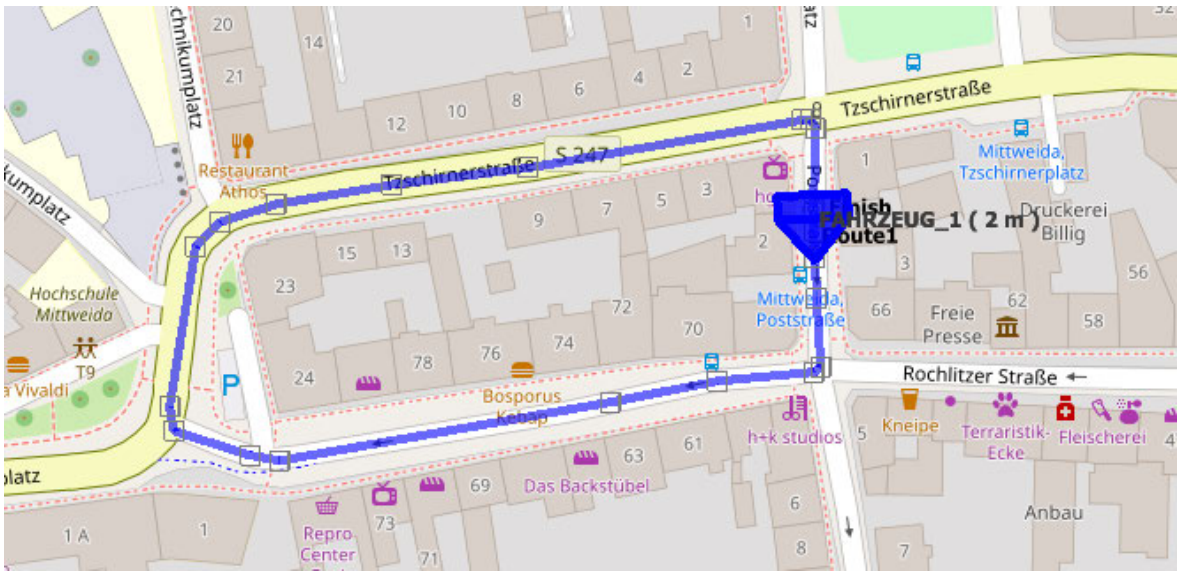

Abbildung 41: Scenario Editor für simulierte Fahrzeuge 1

Beispiel für die Route von Fahrzeug 2:

Die Route für Fahrzeug 2 ist eine Kurzstrecke. Das simulierte Fahrzeug startet am Tzschirnerplatz in Mittweida, fährt ca. 50 m geradeaus und biegt dann rechts ab, am Ende des Tzschirnerplatzes biegt es rechts ab auf die Tzschirnerstraße, fährt dann geradeaus bis zur Ampel und biegt rechts ab zurück zum Ausgangspunkt. Und weiter geht es zum nächsten Radweg. (Siehe Abb.42)

Abbildung 42: Scenario Editor für simulierte Fahrzeuge 2

4.7 Betrieb des Projekts

In diesem Abschnitt werden die Ergebnisse der gesamten Projektarbeit sowie die Mängel des Projekts und Verbesserungsvorschläge vorgestellt. Das Projekt besteht aus einem

Ampelknoten "TL" und zwei simulierten Fahrzeugknoten "FAHRZRUG_1" und "FAHRZEUG_2".

4.7.1 Ampelknoten "TL"

Auf PC1 läuft nur der Knoten "TL", und wenn alles ordnungsgemäß funktioniert, kann man Abbildung 43 auf PC1 sehen.

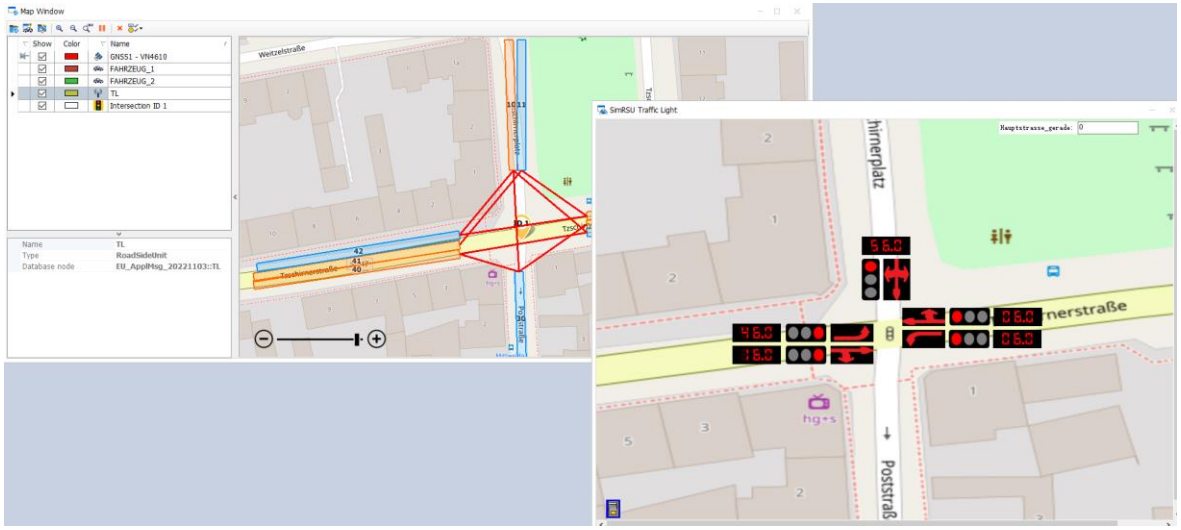

Abbildung 43: Betrieb des Knotens "TL"

Der Status jeder Gruppe von Signalen wird im Panel angezeigt. Wenn die Schaltfläche in der linken unteren Ecke angeklickt wird, blinkt die Ampel in gelben und grauen Intervallen und es wird kein Wert angezeigt. (Siehe Abb.44)

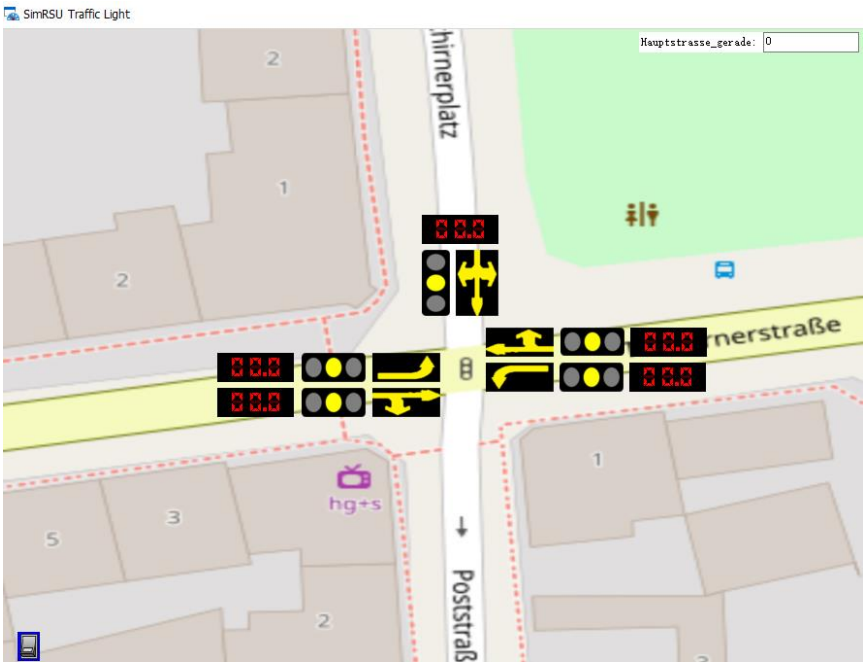

Abbildung 44: Abschaltung der Lichtsignalanlage

Die Signale können normal funktionieren, aber ihre Dauer muss immer noch durch menschliche Kontrolle geändert werden, was fortschrittlicher oder intelligenter wäre, wenn die Dauer der Verkehrssignale entsprechend dem aktuellen Verkehrsfluss geändert werden könnte.

4.7.2 Simulierte Fahrzeugknoten "FAHRZRUG_1" und "FAHRZRUG_2"

PC2 und PC3 werden die Knoten "FAHRZRUG_1" bzw. "FAHRZEUG_2" ausführen. Wenn das Programm ordnungsgemäß läuft, kann man Abbildung 45 und Abbildung 46 sehen.

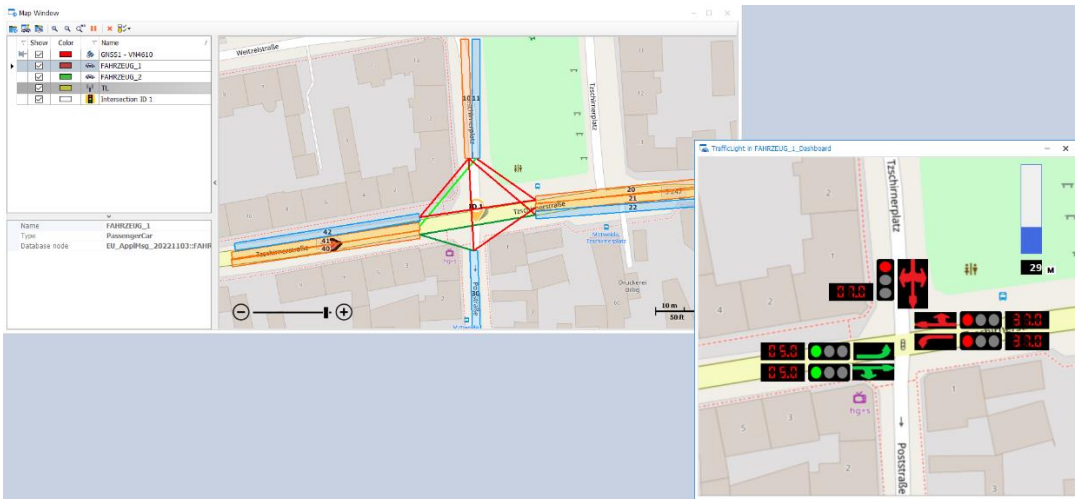

Abbildung 45: Betrieb des Knotens "FAHRZRUG_1" auf PC2

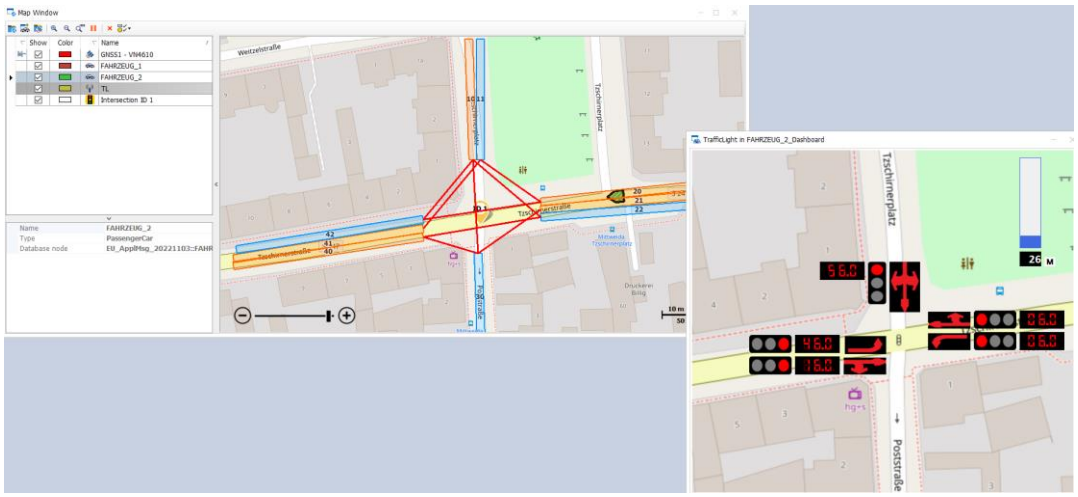

Abbildung 46: Betrieb des Knotens "FAHRZRUG_2" auf PC3

Wenn sich ein simuliertes Fahrzeug einer Ampel nähert, werden Informationen über das Signal und den Abstand zwischen dem Fahrzeug und dem Signal im Panel angezeigt. Diese Anzeige blinkt auch zwischen gelb und grau, wenn das Signal ausgeschaltet ist. Leider können analoge Fahrzeuge nur mit einer festen Anzahl von Signalen interagieren und können nur auf einer festen Straße eingesetzt werden. Es wäre besser und möglich,

dies zu implementieren, wenn die simulierten Fahrzeuge selbst die Grünlichtroute zwischen roten und grünen Ampeln priorisieren können.

5 Zusammenfassung und Ausblick

In diesem Kapitel wird zunächst eine Zusammenfassung des praktischen Projekts vorgestellt. Im Ausblick wird ein Konzept für die Erweiterung dieses praktischen Experiments und die Möglichkeiten für zukünftige Forschung gegeben.

5.1 Zusammenfassung

Im ersten Teil der Arbeit geht es um die Einrichtung eines Verkehrssignalsystems (Knoten TL) und die Erweiterung seiner Funktionalität.Die Lichtsignalanlage sendet ununterbrochen MAP- und SPAT-Nachrichten mit dem geografischen Standort der Lichtsignalanlage (Längengrad, Breitengrad, Höhe) sowie dem aktuellen Status und der Dauer der Lichtsignale für jede Kreuzung.

Auf der Grundlage des Beispiels "Car2x System" wurde der "TL-Knoten" in Teil 1 erstellt. Durch dieses Praktikum 1 konnte ein erstes Verständnis für die verschiedenen Schritte der Projektbearbeitung des Moduls Canoe Car2x und seine wichtigsten Punkte erreicht werden. Durch das Erlernen der Programmierung von "RSU_TrafficLight.can" im "Car2x System"-Beispiel kann die Programmierung dann geändert und erweitert werden, z.B. um die Anzeige von Ampeln im Nicht-Normalbetrieb einzubeziehen. Dies ist eine große Hilfe beim Erlernen der CAPL-Programmierung.

Der zweite Teil der Arbeit bestand darin, das gesamte Projekt auf der Grundlage von Teil 1 um ein Testfahrzeug zu erweitern. Es wurden zwei Fahrzeuge gebaut und in der Arbeit hinzugefügt. Die Fahrzeuge empfingen SPAT- und MAP-Nachrichten von der Lichtsignalanlage und sendeten regelmäßig CAM-Nachrichten. Die CAM-Nachricht enthält grundlegende Informationen über das Fahrzeug, wie Breitengrad, Längengrad, Geschwindigkeit, Fahrtrichtung usw.

Der zweite Teil der Arbeit fügt eine zusätzliche Anzeige der Fahrzeugroute und die Bestimmung der Fahrzeuggeschwindigkeit hinzu. Sowohl die Fahrzeugroute als auch die Fahrzeuggeschwindigkeit können über die Funktion des Moduls "Car2x Scenario Editor" oder direkt über die CAPL-Programmierung eingestellt werden. Für Einsteiger ist es mit dem Modul "Car2x Scenario Editor" jedoch einfacher und intuitiver, die Route des Fahrzeugs einzustellen und die Geschwindigkeit des Fahrzeugs kann über die "Time Line" eingestellt werden. Das Modul "Car2x Scenario Editor" reicht jedoch nicht aus, um die Geschwindigkeit des Fahrzeugs selbst zu bestimmen. Daher wird in diesem praktischen Projekt eine Kombination aus dem Modul "Car2x Scenario Editor" und der CAPL-Programmierung verwendet.

5.2 Ausblick

Die Anwendung des Car2x-Moduls und der CAPL-Programmierung in der Canoe-Software ermöglicht die Simulation des Fahrzeugbetriebs und des Betriebs der Lichtsignalanlage. Die Lichtsignalanlage kann sowohl den Normalbetrieb als auch die Abschaltung simulieren und die Dauer der Signalsätze kann während des Betriebs künstlich verändert werden. Für die Lichtsignalanlage gibt es weitere Möglichkeiten, z.B. kann durch Hinzufügen weiterer Testfahrzeuge die entsprechende Ampeldauer entsprechend dem aktuellen Verkehrsfluss verändert werden. Für die simulierten Fahrzeuge ist zunächst die Funktion der Geschwindigkeitsermittlung nach Erhalt einer bestimmten Signalgruppe in der Software CANoe implementiert. In zukünftigen praktischen Arbeiten werden die Funktionen des simulierten Fahrzeugs mit Hilfe der CANoe Software erweitert. Das Fahrzeug kann die Straße und Ampeln erkennen und beurteilen. Das Fahrzeug kann die Eigenschaften der aktuellen Straße (geradeaus, links oder rechts abbiegen) erkennen. Das simulierte Fahrzeug kann auch die Fahrtroute auswählen, indem es den Status des aktuellen Signalsatzes kombiniert. Wenn das Signal beispielsweise rot für geradeaus und grün für rechts abbiegen ist, kann das Fahrzeug die rechte Abbiegespur wählen, um sein Ziel zu erreichen.Durch kontinuierliche Simulation und Übung wird das Verkehrssignalsystem vielseitiger und das Fahrzeug intelligenter werden und die Stufe des vollständig autonomen Fahrens (Stufe 5) erreichen.

Literatur

[MeBe] Mercedes-Benz: Mercedes-Benz startet europaweites Kooperationsprojekt, URL: <https://group.mercedes-benz.com/innovation/case/connectivity/europaweite-kooperation-car-to-x.html>

[AuMe] Audi MediaCenter: Audi networks with traffic lights in the USA, URL:< https://www.audi-mediacenter.com/en/press-releases/audi-networks-with-traffic-lights-in-the-usa-7147>

[HeMs 2007] Heyms: CAR 2 CAR Communication Consortium: Overview of the C2C-CC System, Version 1.1, 08, 28, 2007

[PaSp 2022] Patrizia Spinola: Anforderungen an eine robuste Car-2-Infrastructure Kommunikation für einen autonomen Fahrbetrieb zur erweiterten Umfelderfassung, Stuttgart. 02. 01. 2022

[JaTh 2021/2022] Prof.Dr.-Ing.J.Thomanek: Vorlesungsskript Modul V2X WS 2021/2022,S.10-15

[FuWi 2022] FUXI_Willard von CSDN: Study Note 12 - Dedizierte Kurzstrecken-Kommunikation auf Basis von Telematik,URL< https://blog.csdn.net/qq_39032096/article/details/125708869>, verfügbar am 10.07.2022

[YaLi 2019] Yan Li: China Readiness for LTE-V2X Commercialization, Seoul Korea, 03.12.2019

[LiMa 2013] LinuxMan: Einführung in die IEEE 802.11 MAC-Schicht, URL< https://blog.csdn.net/stoneliul/article/details/8824382>, verfügbar am 19.04.2013

[AnFe 2014] Andreas Festag: Cooperative Intelligent Transport Systems Standards in Europe,12.2014

[YiXi 2022] YingXiu: Interpretation von SPAT-Nachrichten für V2X-Nachrichten,URL< https://blog.csdn.net/fantasyYXQ/article/details/123056306?spm=1001.2101.3001.6650.3&utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7ERate-3-123056306-blog-114434931.235%5Ev29%5Epc_relevant_default_base3&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7ERate-3-123056306-blog-114434931.235%5Ev29%5Epc_relevant_default_base3>, verfügbar am 02.04.2022

[Vector1] VECTOR:Testen von Steuergeräten und Netzwerken mit CANoe,URL< https://www.vector.com/de/de/produkte/produkte-a-z/software/canoe/#>

[Vector2] VECTOR:The use of CAPL,URL< https://www.vector.com/int/en/know-how/capl/#>

[Vector3] VECTOR: VN4610 Powerful Interface for Car2x/V2x Communication.PDF

[WiKf 2020] Wissenswertes über Kfz-Netzwerke: CANoe Option Car2X Demo（Car2xSystem）Eingehende Analyse, URL< https://blog.csdn.net/miracle8510/article/details/109348167?spm=1001.2014.3001.5502>, verfügbar am 28.10.2020

Anlagen

Anlagen, Teil 1

Ein Teil des Codes in MapInit, um die Herstellung einer Straße zu zeigen.

```
mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].laneID = 10;
mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].laneAttrib-
utes.directionalUse.stringLength = 2;
strncpy(mapPdu.map.intersections.arrayValue[0].laneSet.ar-
rayValue[0].laneAttributes.directionalUse.string, "10", 3);
mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].maneu-
vers.isValidFlag = 1;
mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].maneu-
vers.stringLength = 3;
strncpy(mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].ma-
neuvers.string, "111", 13);
mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].nodeL-
ist.nodes.length = 2;
mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].nodeL-
ist.nodes.arrayValue[0].delta.choice = 4;
mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].nodeL-
ist.nodes.arrayValue[0].delta.node_XY5.x = -250;
mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].nodeL-
ist.nodes.arrayValue[0].delta.node_XY5.y = 2000;
mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].nodeL-
ist.nodes.arrayValue[1].delta.choice = 4;
mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].nodeL-
ist.nodes.arrayValue[1].delta.node_XY5.x = -140;
mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].nodeL-
ist.nodes.arrayValue[1].delta.node_XY5.y = 4000;
mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].con-
nectsTo.isValidFlag = 1;
mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].con-
nectsTo.length = 3;
mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].con-
nectsTo.arrayValue[0].connectingLane.lane = 22;
mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].con-
nectsTo.arrayValue[0].signalGroup.isValidFlag = 1;
mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].con-
nectsTo.arrayValue[0].signalGroup.value = 1;
mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].con-
nectsTo.arrayValue[1].connectingLane.lane = 30;
mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].con-
nectsTo.arrayValue[1].signalGroup.isValidFlag = 1;
mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].con-
nectsTo.arrayValue[1].signalGroup.value = 1;
mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].con-
nectsTo.arrayValue[2].connectingLane.lane = 42;
mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].con-
nectsTo.arrayValue[2].signalGroup.isValidFlag = 1;
mapPdu.map.intersections.arrayValue[0].laneSet.arrayValue[0].con-
nectsTo.arrayValue[2].signalGroup.value = 1;
```

Anlagen, Teil 2

Die Funktion wird hauptsächlich zum Einstellen und Zurücksetzen der Dauer des Ampelsignals verwendet.

```
spatPdu.spat.intersections.arrayValue[0].states.arrayValue[0].signalGroup
= 1;
spatPdu.spat.intersections.arrayValue[0].states.ar-
rayValue[0].state_time_speed.length = 1 ;
spatPdu.spat.intersections.arrayValue[0].states.ar-
rayValue[0].state_time_speed.arrayValue[0].eventState = 3; //
stop_And_Remain
spatPdu.spat.intersections.arrayValue[0].states.ar-
rayValue[0].state_time_speed.arrayValue[0].timing.isValidFlag = 1;
signalGroup_1[0].id = 1;
signalGroup_1[0].stateTime[0] = 60;
signalGroup_1[0].stateTime[1] = 2;
signalGroup_1[0].stateTime[2] = 16;
signalGroup_1[0].stateTime[3] = 2;
signalGroup_1[0].currentState = eRedYellow;
signalGroup_1[0].nextStateChange = 2;
signalGroup_1[0].permissiveMovement = 0;
```

Anlagen, Teil 3

Der Code wird für die Zuweisung von Werten verwendet, die für die Verringerung der Anzahl der roten Lichter und die Erhöhung der Anzahl der grünen Lichter für jede Signalgruppe verwendet werden.

```
on sysvar RSU_TrafficLight::State_SG3
{
    if(signalGroup_1[2].currentState == eRed)
      {
        @sysvar::RSU_TrafficLight::Hauptstrasse_gerade_1_2=0;
        signalGroup_1[2].stateTime[0] = 70;
      }
}
on sysvar RSU_TrafficLight::Hauptstrasse_gerade
{
   if(@sysvar::RSU_TrafficLight::Hauptstrasse_gerade!=0)
   {
     @sysvar::RSU_TrafficLight::Hauptstrasse_gerade_1_0=@sys-
var::RSU_TrafficLight::Hauptstrasse_gerade;
     @sysvar::RSU_TrafficLight::Hauptstrasse_gerade_1_1=@sys-
var::RSU_TrafficLight::Hauptstrasse_gerade;
     @sysvar::RSU_TrafficLight::Hauptstrasse_gerade_1_2=@sys-
var::RSU_TrafficLight::Hauptstrasse_gerade;
     @sysvar::RSU_TrafficLight::Hauptstrasse_gerade_1_3=@sys-
var::RSU_TrafficLight::Hauptstrasse_gerade;
   }
   else if (@sysvar::RSU_TrafficLight::Hauptstrasse_gerade==0)
   {
     @sysvar::RSU_TrafficLight::Hauptstrasse_gerade_1_0 = 10;//variable
for signalGroup_1[0]
     @sysvar::RSU_TrafficLight::Hauptstrasse_gerade_1_1 = 0;//variable
for signalGroup_1[1]
     @sysvar::RSU_TrafficLight::Hauptstrasse_gerade_1_2 = @sys-
var::RSU_TrafficLight::Reamin;//variable for signalGroup_1[2]
     @sysvar::RSU_TrafficLight::Hauptstrasse_gerade_1_3 = 0;//variable
for signalGroup_1[3]
   }
}
on sysvar_update RSU_TrafficLight::Hauptstrasse_gerade
{
  if(@sysvar::RSU_TrafficLight::Hauptstrasse_gerade!=0)
  {
   @sysvar::RSU_TrafficLight::Reamin=@sysvar::RSU_Traffic-
Light::Hauptstrasse_gerade;
  }
}
```

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Chemnitz, den 01.05.2023

[redacted]

Yuchao Xu