HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

BACHELORARBEIT

Frau
Pia Hanfeld

Realisierung eines forensischen Demonstrators für den CAN-Bus

2018

Fakultät **Angewandte Computer- und Biowissenschaften**

BACHELORARBEIT

Realisierung eines forensischen Demonstrators für den CAN-Bus

Autorin:
Pia Hanfeld

Studiengang:
Allgemeine und Digitale Forensik

Seminargruppe:
FO15w2-B

Erstprüfer:
Prof. Dr. rer. nat. Christian Hummert

Zweitprüfer:
Dipl.-Ing. (FH) Heiko Polster

Mittweida, August 2018

Faculty of **Applied Computer Sciences and Biosciences**

BACHELOR THESIS

Realization of a forensic demonstrator for the CAN bus

Author:
Pia Hanfeld

Course of Study:
General and Digital Forensics

Seminar Group:
FO15w2-B

First Examiner:
Prof. Dr. rer. nat. Christian Hummert

Second Examiner:
Dipl.-Ing. (FH) Heiko Polster

Mittweida, August 2018

Bibliografische Angaben

Hanfeld, Pia: Realisierung eines forensischen Demonstrators für den CAN-Bus, 73 Seiten, 48 Abbildungen, Hochschule Mittweida, University of Applied Sciences, Fakultät Angewandte Computer- und Biowissenschaften

Bachelorarbeit, 2018

Referat

In dieser Bachelorarbeit wird ein Demonstrator für das Controller Area Network (CAN) konzeptioniert, realisiert und getestet. Bis heute bietet das CAN-Protokoll kaum Möglichkeiten für eine forensische Auswertung und weist diverse Sicherheitslücken auf, die in der Vergangenheit erfolgreich ausgenutzt werden konnten. In dieser Arbeit werden die Grundlagen zu den Bussystemen, die aktuell Anwendung in der Automobilbranche finden, vermittelt und mögliche Angriffsszenarien auf das CAN vorgestellt. Der Demonstrator wird mit einem Detektionssystem, welches für reale Fahrzeuge entwickelt wurde, ausgestattet und den Angriffen unterzogen. Es kann gezeigt werden, dass dieses System für die digitale Forensik verwertbare Meldungen ausgeben kann.

Abstract

In this bachelor thesis, a demonstrator for the Controller Area Network (CAN) will be designed, realized and tested. Until today the CAN protocol offers nearly no possibilities for a forensic evaluation and several vulnerabilities have been exploited in the past. The principles of modern bus systems and possible exploitation scenarios of the CAN will be explained. The demonstrator was extended with an intrusion detection system originally developed for vehicles. The system generated messages applicable to digital forensic analysis.

I. Inhaltsverzeichnis

II. Abbildungsverzeichnis

III. Tabellenverzeichnis

IV. Abkürzungsverzeichnis

ACK ..... **Ack**nowledgement
AP ..... **A**ccess **P**oint
CAN ..... **C**ontroller **A**rea **N**etwork
CAN FD ..... **CAN** mit **f**lexibler **D**atenrate
CAPL ..... **C**AN oder **C**ommunication **A**ccess **P**rogramming **L**anguage
CBFF ..... **C**lassical **B**ase **F**rame **F**ormat
CEFF ..... **C**lassical **E**xtended **F**rame **F**ormat
CiA ..... **C**AN **i**n **A**utomation
CRC ..... **C**yclic **R**edundancy **C**heck
CSMA/CA ..... **C**arrier **S**ense **M**ultiple **A**ccess with **C**ollision **A**voidance
CTC ..... **C**lear **T**imer on **C**ompare Match
DLC ..... **D**ata **L**ength **C**ode
ECU ..... **E**lectronic **C**ontrol **U**nit
EoF ..... **E**nd **o**f **F**rame
ESP ..... **Esp**ressif (Firma)
EVB ..... **E**valuation**b**oard
GND ..... **G**rou**nd**
IDS ..... **I**ntrusion **D**etection **S**ystem
IG ..... **I**nteraktiver **G**enerator
ISO ..... Internationale Organisation für Normung
ISR ..... **I**nterrupt **S**ervice **R**outine
LAN ..... **L**ocal **A**rea **N**etwork
LED ..... **L**icht **e**mittierende **D**iode
LIN ..... **L**ocal **I**nterconnect **N**etwork
MOST ..... **M**edia **O**riented **S**ystems **T**ransport
OBD ..... **O**n-**B**oard-**D**iagnose
PWM ..... **P**uls**w**eiten**m**odulation
REC ..... **R**eceive **E**rror **C**ounter
RWTH ..... **R**heinisch-**W**estfälische **T**echnische **H**ochschule Aachen
SoF ..... **S**tart **o**f **F**rame
TCP ..... **T**ransmission **C**ontrol **P**rotocol
TEC ..... **T**ransmission **E**rror **C**ounter

TP ..... **T**wisted **P**air

UDP ..... **U**ser **D**atagram **P**rotocol

UTP ..... **U**nshielded **TP**

WLAN ..... **W**ireless **LAN**

1 Einleitung

Die Entwicklung in der Automobilbranche schreitet immer weiter voran. Neuwagen sind derzeit mit mehreren hundert Steuergeräten (Electronic Control Units (ECUs)) ausgestattet, die über verschiedene Bussysteme und Schnittstellen miteinander kommunizieren. Zu diesem Trend gehört auch, dass Autos über eine Anbindung zum Mobilfunknetz verfügen und so eine kabellose Kommunikation mit der Außenwelt stattfindet. Ebenso rückt das autonome Fahren in den Fokus der Forschung. Dabei wird über eine Vielzahl an Sensoren die Umwelt wahrgenommen und es entstehen große Mengen an Daten, die sicher über die Busse an die Steuergeräte verteilt werden sollen. Da jedoch, durch die Verbindung zum Mobilfunknetz, auch kabellose Kommunikation stattfindet, ergeben sich Angriffsszenarien in sicherheitskritischen Bereichen. Je mehr Entscheidungen über das Fahrverhalten autonom getroffen werden, desto wichtiger wird es, die Abläufe im Fahrzeug vor Angriffen aus der Ferne zu schützen.

1.1 Motivation

Das Controller Area Network (CAN) ist als Bussystem in nahezu jedem modernen Auto integriert, bietet aber keinerlei Absicherung vor kompromittierten Nachrichten, die auf den Bus gesendet wurden. Die Informationen, die zur Laufzeit im Auto zwischen den ECUs ausgetauscht werden, lassen sich unter anderem über spezielle Software, wie zum Beispiel dem CANalyzer der Firma Vector Informatik, auslesen. Steuergeräte können zwar durch solche Programme gezielt für die Diagnose angesprochen werden, der Aufbau des CAN-Protokolls lässt jedoch keinen Raum für die Wiederherstellung der Vorgänge, die zum Ist-Zustand geführt haben. Sollte es zu einem Angriff auf den CAN-Bus kommen, könnte der Fahrer eines Fahrzeuges durch Fehlfunktionen abgelenkt sein und Unfälle verursachen. Ein Angreifer kann ein Fahrzeug auch bewusst in gefährliche Situationen manövrieren und damit Menschenleben gefährden.
Aktuell lassen sich für die digitale Forensik in solchen Fällen nur wenige gerichtsverwertbare Spuren finden. Um in diesem Gebiet weiter forschen zu können, wird ein Auto und die Software benötigt. Die Kosten für forensische Analysen in Forschung und Lehre wären immens. Ein Testsystem, welches möglichst realitätsnah erstellt wird, könnte diesen Aufwand erheblich senken und bietet gleichzeitig eine anschauliche Möglichkeit zur Demonstration des CAN-Busses in Automobilen.

1.2 Ziel

Ziel dieser Arbeit soll es sein, einen Demonstrator zu gestalten und anzufertigen, der ein stark vereinfachtes Automobil darstellen soll. Er soll mit einem CAN-Bus ausgestattet sein und dadurch CAN-Botschaften erhalten können. Durch die Software CANalyzer Pro von Vector Informatik GmbH werden die Nachrichten generiert. Die Nachrichten sollen einfache Funktionen, wie das Beschleunigen und Stoppen der Motoren, Richtungsänderung und Ein- und Ausschalten von Leuchtdioden (LEDs) auslösen. Die Leuchtdioden sollen Scheinwerfern und Blinkern eines Autos nachempfunden sein. Das Joy-IT Roboter Fahrgestell Robot Car Kit 01 (siehe Abbildung 1.1) soll dazu mit einem Mikrocontroller (AT90CAN128) der Firma Atmel, welcher auf einem Evaluationsboard der Firma Olimex Ltd. verbaut ist, ausgestattet werden. Die Funktionen werden in C-Programmen realisiert. Die Ansteuerung der Motoren des Robot Car Kit soll über das Evaluationsboard EVAL6207N realisiert werden. Es soll getestet werden, ob Angriffe auf das System möglich sind und ob sich durch die Untersuchung der Ereignisse auch gerichtsverwertbare Ergebnisse ableiten lassen.

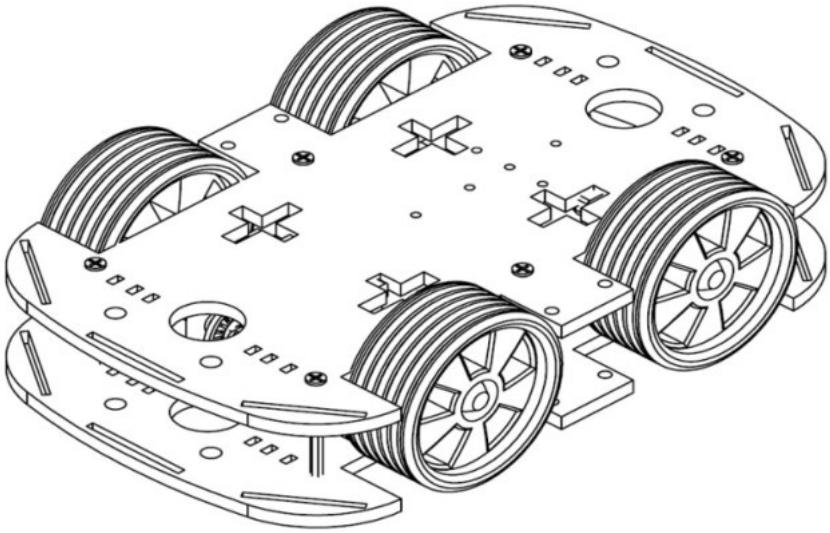

Abbildung 1.1: Robot Car Kit von Joy-IT. Quelle: [Joy-IT, 2017, S. 1]

2 Grundlagen

In diesem Kapitel werden die, für die Automobilbranche üblichen, Bussysteme vorgestellt, die Kommunikation innerhalb des CAN näher erläutert und die Grundlagen für den Aufbau des Demonstrators vermittelt. Alle Informationen stammen, wenn nicht anders gekennzeichnet, aus: [Etschberger, 2000, Kap. 1.6, 2], [Beierlein, 2011, S. 227 f.], [Zimmermann und Schmidgall, 2014, Kap. 2.1.1, 3], [Hering et al., 2017, Kap. 15.1, 17.2.1]

2.1 Netzwerktopologien

Um Steuergeräte in einem Netzwerk miteinander zu verknüpfen und den Austausch von Nachrichten möglichst effizient zu gestalten, gibt es verschiedene Arten des Aufbaus dieser Netzwerke (Topologie).

2.1.1 Linientopologie

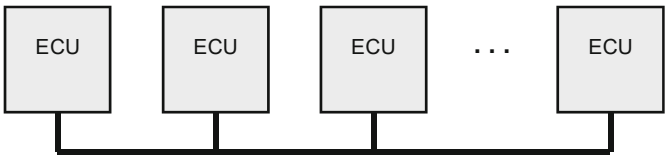

Abbildung 2.1: Linientopologie. Quelle: modifiziert nach [Zimmermann und Schmidgall, 2014, S. 17]

Bei diesem Aufbau des Netzwerkes (siehe Abbildung 2.1) werden alle Steuergeräte mit einer Datenleitung verbunden. Sendet ein Gerät eine Nachricht, so erhalten sie alle anderen Teilnehmer. Diese Topologie wird zum Beispiel beim Controller Area Network genutzt.

Vorteile

Der Verkabelungsaufwand wird gering gehalten. Neue Teilnehmer können, selbst im Betrieb, einfach hinzugefügt werden. Teilnehmer können sich ohne Auswirkungen auf das Netzwerk zurückziehen oder ausfallen.

Nachteile

Das Signal muss ab einer gewissen Länge des Busses und Anzahl an Teilnehmern verstärkt werden. Der Zugriff auf den Bus muss geregelt sein, sonst entstehen Kollisionen. Die Leitung muss durch einen Abschlusswiderstand terminiert werden.

2.1.2 Sterntopologie

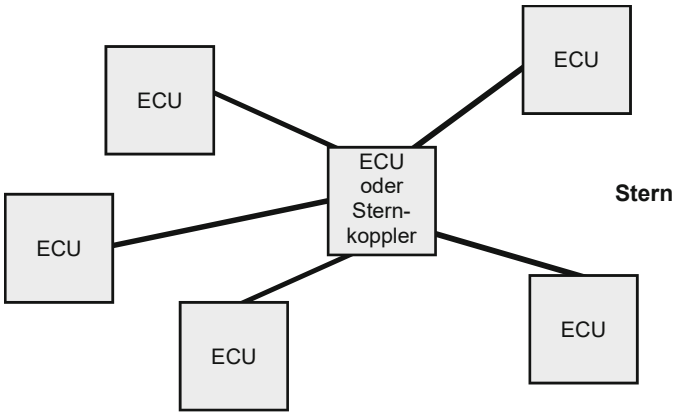

Abbildung 2.2: Sterntopologie. Quelle: [Zimmermann und Schmidgall, 2014, S. 18]

Alle Teilnehmer sind mit einem zentralen Steuergerät verbunden, über das alle Nachrichten verteilt werden (siehe Abbildung 2.2). So können einzelne Steuergeräte miteinander kommunizieren, oder ein Gerät mehrere Teilnehmer gleichzeitig ansprechen, ohne, dass alle Geräte die Nachricht erhalten. Die Sterntopologie findet sich hauptsächlich bei Ethernet-Netwerken.

Vorteile

Neue Teilnehmer können einfach eingebunden werden. Es werden keine Zugriffsregelungen benötigt.

Nachteile

Der Verkabelungsaufwand ist größer. Der zentrale Teilnehmer benötigt so viele Schnittstellen, wie Teilnehmer im Netz vorhanden sind. Fällt der zentrale Teilnehmer aus, ist keine Kommunikation mehr möglich.

2.1.3 Ringtopologie

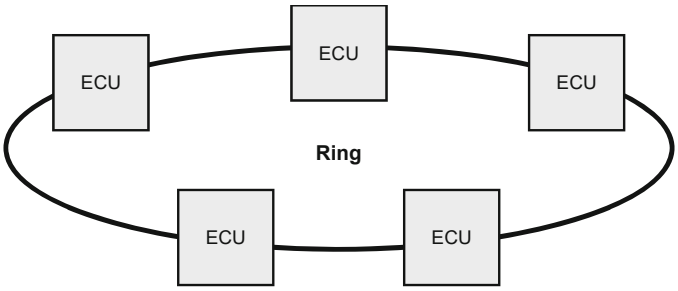

Abbildung 2.3: Ringtopologie. Quelle: [Zimmermann und Schmidgall, 2014, S. 18]

Alle Teilnehmer werden bei dieser Topologie durch eine gerichtete Verbindung untereinander verkettet und somit werden Nachrichten von jedem Steuergerät empfangen und weitergeleitet, bis der eigentliche Empfänger wieder erreicht wurde (siehe Abbildung 2.3).

Netzwerke mit Ringtopologie sind zum Beispiel lokale Netze (LAN) oder Weitverkehrsnetze (WAN).

Vorteile

Jeder Teilnehmer bewirkt eine Signalregeneration, weshalb die Netzwerke sehr groß sein dürfen.

Nachteile

Fällt ein Teilnehmer aus, fällt auch das Gesamtsystem aus, weshalb diese Topologie zusätzlich abgesichert werden muss. Das geschieht zum Beispiel durch Überbrückungen oder einen redundanten Ring, was den Verkabelungsaufwand erhöht. Wird ein neuer Teilnehmer hinzugefügt, muss der Betrieb unterbrochen werden.

2.2 Datenübertragung

Die Datenübertragung in Netzwerken erfolgt im Wesentlichen über drei Arten: Punkt-zu-Punkt, zum Beispiel bei der Kommunikation zweier Steuergeräte, Punkt-zu-Gruppe (Multicast), ein Steuergerät sendet Daten an mehrere Teilnehmer, und Punkt-zu-allen (Broadcast), das Steuergerät sendet an alle Teilnehmer im Netz.

Der Nachrichtenaustausch kann dabei in eine Richtung über eine Datenverbindung (Simplex), im Wechselverkehr über eine Datenverbindung (Halb-Duplex) oder bidirektional über zwei Datenverbindungen (Voll-Duplex) stattfinden. Es wird weiterhin unterschieden, ob die Daten seriell oder parallel übertragen werden. Bei der seriellen Datenübertragung müssen Sender und Empfänger synchronisiert werden, der Takt wird zum Beispiel aus Synchronisierungspulsen, die vor den eigentlichen Daten übertragen werden, abgeleitet. Nach der Synchronisation werden die Daten in einem Bitstrom bis zur Endsequenz übermittelt. Die Nachrichten sind häufig über Sicherungsmaßnahmen, wie zum Beispiel Checksummen, vor fehlerhaften Übertragungen gesichert.

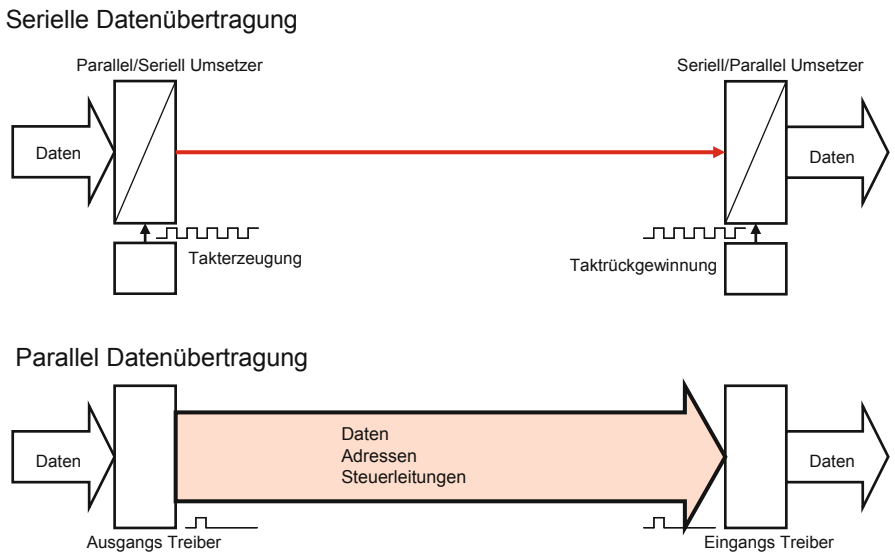

Abbildung 2.4: Serielle und parallele Datenübertragung. Quelle: [Hering et al., 2017, S. 689]

Bei der parallelen Datenübertragung werden Datengruppe, Adressgruppe und Steuergruppe über verschiedene Leitungen übertragen und diese Übertragungsart wurde vor allem zur Übermittlung von sehr großen Datenmengen bei sehr hohen Geschwindigkeiten verwendet. Auch hier muss ein Taktsignal für die Synchronisation übertragen werden. Mit steigender Bitbreite (Größe der Daten- und Adressregister) ist die parallele Übertragung jedoch nicht mehr sinnvoll umzusetzen, da die Kosten den Nutzen übersteigen und auf die serielle Übertragung zurückgegriffen werden kann. Beide Übertragungsarten sind schematisch in Abbildung 2.4 dargestellt.

2.3 Bussysteme in Automobilen

Durch die Weiterentwicklung der Technik in der Automobilbranche werden immer mehr Steuergeräte in einem Auto verbaut und der Verkabelungsaufwand steigt weiterhin an. Aus diesem Grund wurden neue Bussysteme für die effiziente Kommunikation von Steuergeräten in Fahrzeugen entwickelt.

2.3.1 Controller Area Network

Das Controller Area Network (CAN) wurde in den 1980er Jahren von der Firma Bosch entwickelt und ist bis heute eines der am weitesten verbreiteten Bussysteme in Automobilen. Im CAN können Daten zwischen Steuergeräten mit bis zu 1 Mbit/s übertragen werden und es findet sich unter anderem in Bereichen des Fahrwerkes oder des Komforts. Alle Steuergeräte sind mit einer Datenleitung verbunden und können gleichberechtigt Nachrichten auf den Bus senden. Das Netzwerk wird ausführlich in Kapitel 2.4 beschrieben.

2.3.2 Local Interconnect Network

Das Local Interconnect Network (LIN) wurde in den 1990er Jahren von der Firma Motorola (heute Freescale) und einigen Kraftfahrzeugherstellern als kostengünstigere Alternative zum CAN entwickelt und verknüpft Sensoren mit Aktoren. Die Übertragungsraten liegen bei 1-20 kbit/s und das Netzwerk ist häufig in den Türen oder Sitzen eines Fahrzeuges anzutreffen. Der Bus besteht aus einem Master, der gleichzeitig ein Gateway zum CAN-Bus darstellt, und mehreren Slaves. Der Master sendet Botschaftsheader mit einem LIN-Identifier auf den Bus. Alle Slaves prüfen nun nach dem Identifier, ob sie berechtigt sind, mit einer Datenbotschaft zu antworten. Genau ein Slave ist einem Identifier zugeordnet, womit ausgeschlossen ist, dass mehrere Steuergeräte versuchen Botschaften auf den Bus zu senden. Auch in diesem Protokoll gibt es Mechanismen, Fehler zu detektieren, jedoch werden diese nicht verfolgt. Bei LIN besteht, wie bei CAN, keinerlei Authentifizierung oder Verschlüsselung.

2.3.3 FlexRay

FlexRay wurde ab dem Jahr 2000 von einem Verbund mehrerer Firmen (BMW, Daimler, Motorola und Philips) entwickelt und ist seit 2010 in der ISO 17458-1 bis 17458-5 standardisiert. Durch eine höhere Datenrate soll FlexRay das CAN in zeitkritischen Bereichen, vor allem bei X-by-Wire-Anwendungen (Leitung elektronischer Steuersignale, zum Beispiel der Bremsen oder Lenkung) ersetzen. Der Bus kann sowohl eine Linien- als auch eine Sterntopologie aufweisen und erlaubt zweikanalige Systeme. Der Buszugriff erfolgt deterministisch (kontrolliert) und wird zentral und dezentral geregelt. Die Kommunikation verläuft zyklisch und jedes Steuergerät darf in seinem Zeitfenster (Slot) senden. Das Bussystem gilt als besonders ausfallsicher und kann Daten mit bis zu 10 Mbit/s übertragen. Für den fehlerfreien Ablauf müssen FlexRay Knoten wissen, wie alle Teile des Netzwerks konfiguriert sind, was den schnellen Einbau eines neuen, möglicherweise mit Schadsoftware infizierten Steuergerätes erschwert. Jedoch findet sich auch in diesem Protokoll keine Verschlüsselung. Erstmals wurde FlexRay serienmäßig 2006 im BMW X5 verbaut und kann das CAN derzeit noch nicht vollständig ersetzen.

2.3.4 Media Oriented Systems Transport

Das Media Oriented Systems Transport (MOST)-Bussystem wurde als reiner Infotainment-Bus für Telematik- und Multimediaanwendungen im Jahre 1997 entwickelt. Auch an diesem Bussystem waren BMW und Daimler beteiligt und es wird heute hauptsächlich in Modellen der Oberklasse eingesetzt. Im Gegensatz zu den bisher vorgestellten Bussystemen liegt bei MOST das Hauptaugenmerk nicht auf Übertragungssicherheit oder Echtzeitverhalten, sondern auf Übertragungsgeschwindigkeit. Derzeit können Daten mit bis zu 150 Mbit/s über Lichtwellenleiter übermittelt werden. Der Bus weist eine Ringtopologie auf. Bestimmte Teile der Spezifikation werden unter Verschluss gehalten und sind nur den Mitgliedern der MOST Cooperation zugänglich. Ein Steuergerät sendet als Timing Master in definierten Zeitabständen MOST-Botschaften auf den Bus um die Slaves zu synchronisieren. Die Frequenz, in der die Nachrichten auf den Bus gesendet werden, entspricht der Abtastfrequenz von modernen DVD-Audio-Playern. So wird gewährleistet, dass die Multimediadaten ausreichend schnell übermittelt werden. Bei diesem Bussystem lassen sich neue Komponenten über eine Programmierschnittstelle konfigurieren und später leicht in ein bestehendes Netzwerk einfügen (Plug-and-Play-Funktionalität).

2.3.5 Automotive Ethernet

Die Entwicklung der letzten Jahre hat gezeigt, dass selbst die neuen Bussysteme neben dem CAN den derzeitigen Anforderungen nicht ausreichend gerecht werden oder zu spezifisch sind, um mit einer Vielzahl an Systemen kompatibel zu sein. Deshalb wird immer häufiger auf Ethernet nach dem Standard IEEE 802.3 als Netzwerk im Automobil zurück-

gegriffen. Jedes Steuergerät ist dabei in einer Sterntopologie Punkt-zu-Punkt mit einem Switch verbunden und eine Voll-Duplex-Kommunikation ist möglich. Die Wegfindung der Botschaften findet über Ethernet-Adressen statt. Da über den Switch mehrere Nachrichten für verschiedene Geräte gleichzeitig übertragen werden können, ist das System kollisionsfrei und zu Verzögerungen kommt es nur, wenn mehrere Nachrichten an ein Steuergerät versendet werden sollen. Botschaften im Netzwerk können durch Störungen verloren gehen oder wenn der Puffer im Switch, in dem Nachrichten wie in einer Warteschlange gespeichert werden, keine ausreichende Größe aufweist. Daten können bei Automotive Ethernet mit ca. 100 Mbit/s übermittelt werden, was aktuell nicht ausgereizt wird.

2.4 Controller Area Network

Nach seiner Einführung verbreitete sich das CAN sehr schnell in neu gebauten Fahrzeugen und ist heute in jedem modernen Automobil anzutreffen. Seit 2008 ist die CAN-Schnittstelle die einzig zugelassene für die Diagnose abgasrelevanter Komponenten bei Neufahrzeugen. Das Netzwerk gilt bis heute als sehr zuverlässig und robust. Durch seine vielen Vorteile wird das CAN auch immer häufiger in der Automatisierungstechnik eingesetzt und derzeit hauptsächlich durch den, in Nürnberg ansässigen, gemeinnützigen Verbund *CAN in Automation* (CiA) standardisiert.

2.4.1 Nachrichtenübertragung im CAN

Das Netzwerk stellt sich als Bustopologie dar und die Datenübertragung findet seriell über Halb-Duplex statt. Alle Steuergeräte im Netzwerk sind dazu berechtigt, CAN-Nachrichten zu senden und zu empfangen (Multi-Master-System). Es wird zwischen High Speed CAN mit Übertragungsgeschwindigkeiten bis 1 Mbit/s, deklariert in der ISO 11898-2, und Low Speed CAN mit Übertragungsgeschwindigkeiten bis 125 kbit/s, deklariert in der ISO 11898-3, unterschieden. Die Datenleitung besteht aus einem Kupferkabel mit verdrillten Adernpaaren (Twisted Pair (TP)), die nicht zusätzlich abgeschirmt sind (unshielded TP (UTP)), und einer Masseleitung. [Mandl et al., 2008, S. 28 f.] Jeweils ein Adernpaar überträgt die Signale CAN High und CAN Low (siehe Kapitel 2.4.3). An dieser Datenleitung sind alle sogenannten CAN-Knoten angeschlossen. (siehe Abbildung 2.5)
Ein CAN-Knoten besteht aus drei Teilen:

- einer Software (auch Host), die Daten zur Verfügung stellt oder diese verarbeitet,
- einem Controller, der die Daten der Software in CAN-Botschaften (siehe Kapitel 2.4.5) umwandelt und auf den Bus sendet, oder Daten aus den Botschaften extrahiert und diese an die Software weiterleitet,
- einem Transceiver, der die Signale der Datenleitung aufnimmt oder Informationen des CAN-Controllers auf den Bus sendet.

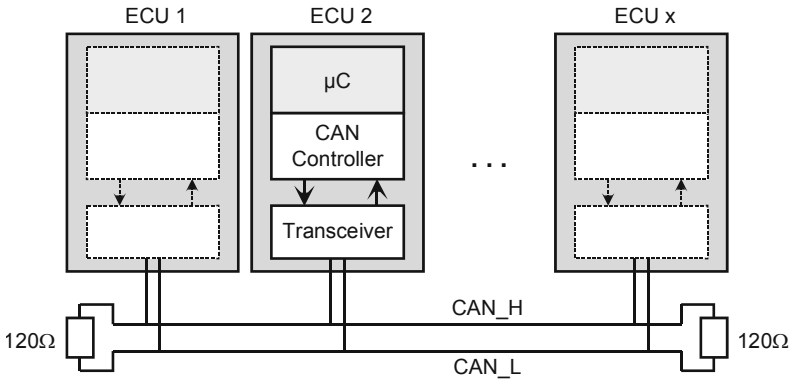

Abbildung 2.5: Komponenten des CAN. Quelle: [Zimmermann und Schmidgall, 2014, S. 59]

Der CAN-Knoten entspricht einer Electronic Control Unit. Controller und Software befinden sich heutzutage häufig auf einem Mikrocontroller. Der Aufbau eines CAN ist in Abbildung 2.5 dargestellt. Der Bus muss nur bei High Speed CAN mit einem Abschlusswiderstand von 120 Ω terminiert werden, was bei Low Speed CAN entfällt. μC steht in dieser Abbildung für den Mikrocontroller, der die Software beherbergt. Da alle Steuergeräte im CAN innerhalb einer Bitzeit synchronisiert werden und bei Fehlern reagieren müssen, kann der Bus nur eine bestimmte Länge haben. Für die Berechnung der Buslänge lässt sich folgende Faustformel verwenden:

$$\text{Buslänge} \leq 40...50\ \text{m} \cdot \frac{1\ \text{Mbit/s}}{\text{Bitrate}}$$

Alle Steuergeräte im Netzwerk müssen mit der gleichen Bitrate betrieben werden. Bei hohen Bitraten wirkt sich deshalb die Verzögerungszeit der CAN-Transceiver oder eines, bei hohen Buslängen eingesetzten, Repeaters auf alle Teilnehmer aus.

2.4.2 High Speed und Low Speed CAN

Für alle Übertragungsraten über 250 kbit/s gilt die ISO 11898-2, die High Speed CAN definiert. In Personenkraftwagen werden Daten bevorzugt mit 500 kbit/s übertragen. In diesem Netzwerk werden zeitkritische Informationen, gesendet von, zum Beispiel, Steuergeräten des Antriebs, über den Bus verteilt. Wird ein Adernpaar unterbrochen oder kommt es zu einem Kurzschluss fällt der Bus aus. Alle Steuerelemente dürfen einen maximalen Abstand von 30 cm zum Bus haben.
Bei Übertragungsraten kleiner als 125 kbit/s wird Low Speed CAN, definiert in der ISO 11898-3, eingesetzt. Über diese Netzwerke werden Informationen aus den Komfortbereichen, wie zum Beispiel Nachrichten der elektrischen Fensterheber, übertragen. Durch die langsameren Übertragungsraten kann die Buslänge höher sein. Bei Low Speed CAN muss der Bus nicht durch einen Abschlusswiderstand terminiert werden und es wird kein maximaler Abstand der Steuergeräte zur Datenleitung vorgegeben.

2.4.3 Signalpegel

Der CAN-Transceiver ist mit dem jeweiligen Adernpaar für CAN High und CAN Low des Busses verbunden. Über die Differenzspannung der beiden Signale wird entschieden, ob eine logische 0 oder 1 übertragen wird. Bei High Speed CAN beträgt die Differenzspannung für eine 1 (rezessives Bit) 0 V, wobei CAN High und CAN Low jeweils einen Signalpegel von 2,5 V haben. Wird eine 0 (dominantes Bit) übertragen, steigt die Differenzspannung für eine Bitzeit auf 2 V, CAN High weist dabei eine Spannung von 3,5 V und CAN Low eine von 1,5 V auf.
Bei Low Speed CAN ist die Differenzspannung für ein rezessives Bit -5 V, für ein dominantes Bit 2,2 V. Die Signalpegel für CAN High und Low können der Abbildung 2.6 entnommen werden.

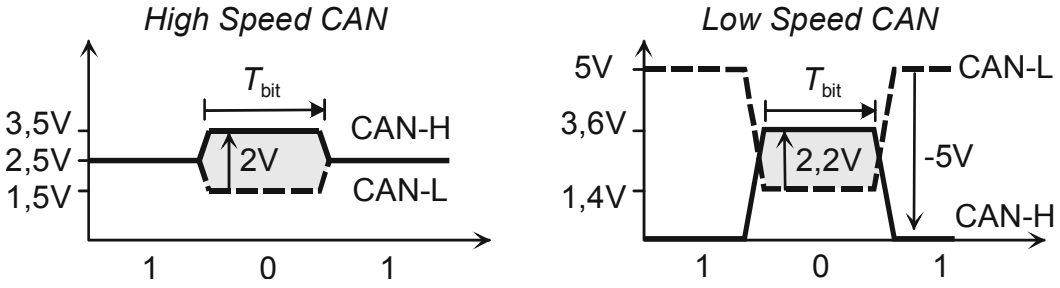

Abbildung 2.6: Signalpegel im CAN. Quelle: [Zimmermann und Schmidgall, 2014, S. 59]

2.4.4 Buszugriff und Sicherungsverfahren

Jede ECU ist zu jeder Zeit berechtigt Nachrichten auf den Bus zu senden, der Buszugriff ist damit zufällig, auch wahlfrei genannt. Um Kollisionen von Nachrichten zu vermeiden erfolgt eine bitweise Arbitrierung. Das Verfahren lautet Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). Der Ablauf der Arbitrierung wird in Abbildung 2.7 beschrieben. Zum Zeitpunkt (1) versuchen drei Teilnehmer zeitgleich eine Nachricht zu senden, auf dem Bus wird eine 0 übertragen. Teilnehmer Zwei sendet zum Zeitpunkt (2) ein rezessives Bit (eine 1), was von Teilnehmer Eins und Drei mit einem dominanten Bit (einer 0) überschrieben wird und wechselt daher zum Empfangen der Nachricht. Selbiges findet zum Zeitpunkt (3) mit Teilnehmer Eins statt, sodass zum Zeitpunkt (4) nur noch Teilnehmer Drei zum Senden berechtigt ist und die restliche Nachricht überträgt. Bei diesem Vorgang werden demnach Nachrichten priorisiert, deren Identifier eine geringe Wertigkeit haben. Das bedeutet, dass die Nachricht mit dem Identifier 0x00 die höchste Priorität hat. Nachdem Teilnehmer Drei den Sendevorgang beendet hat, können Teilnehmer Eins und Zwei erneut versuchen die Nachrichten nach dem gleichen Schema zu übertragen. Aus diesem Vorgang geht hervor, dass eine logische 0 immer eine 1 auf dem Bus überschreibt, weshalb die Bits als dominant oder rezessiv bezeichnet werden, und dass Nachrichten im CAN nicht verloren gehen.

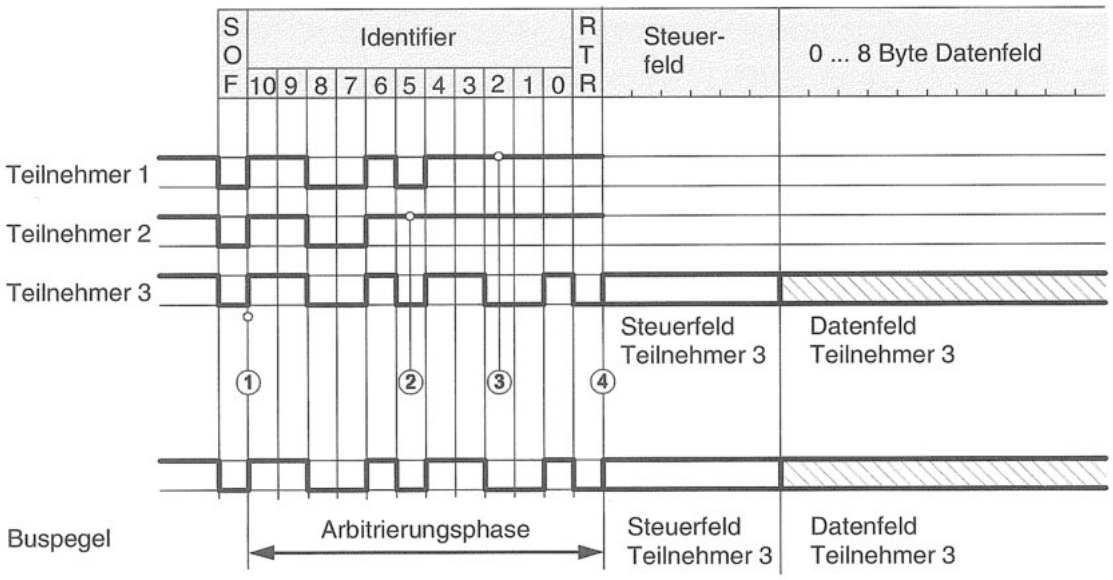

Abbildung 2.7: Beispiel für die bitweise Arbitrierung. Quelle: [Etschberger, 2000, S. 56]

Werden mehrere aufeinanderfolgende Bits gleicher Polarität übertragen, findet keine Pegeländerung statt. Dadurch kann die Synchronisation der Steuergeräte im Netzwerk verloren gehen. Nach fünf homogenen Bits wird deshalb ein komplementäres Bit in den Bitstrom eingefügt, was als Bitstuffing bezeichnet wird. Diese Stuffbits müssen nach Erhalt der vollständigen Nachricht von den Controllern herausgerechnet werden.

2.4.5 Aufbau einer CAN-Botschaft

Im Controller Area Network sind insgesamt vier Nachrichtentypen vorgesehen:

Data Frame
: Datentelegramme werden hauptsächlich für die reguläre Datenübertragung im Netzwerk genutzt.

Remote Frame
: Über Datenanforderungstelegramme werden Datentelegramme von Teilnehmern im Netz angefordert.

Error Frame
: Fehlertelegramme zeigen Fehler bei der Datenübertragung auf dem Bus an.

Overload Frame
: Überlasttelegramme können Verzögerungen zwischen Frames gezielt verursachen.

Da seit der Entwicklung des CAN die Menge an Identifiern, die insgesamt in einem Netzwerk benötigt werden, nicht mehr durch die, zunächst festgelegte, Größe des Feldes abgedeckt werden konnten, wurde das klassische Standard-Nachrichtenformat (Classical

Base Frame Format (CBFF)) um einige Felder zum klassischen erweiterten Nachrichtenformat (Classical Extended Frame Format (CEFF)) erweitert. Zum Verständnis wird in Tabelle 2.1 jedoch nur das CBFF erläutert und auf die wichtigsten Felder eingegangen.

Tabelle 2.1: Aufbau eines CAN-Frames

Feld	Länge in Bit	Beschreibung
Start of Frame	1	Im Feld SoF erfolgt ein Flankenwechsel von rezessiv nach dominant und eine netzwerkweite Synchronisation. Die Empfänger überprüfen das Bittiming und synchronisieren bei Abweichung um den Phasenfehler nach (Nachsynchronisation).
Identifier	11	Der Identifier legt die Priorität einer Nachricht fest. Sollten die Empfänger selbst versuchen zu senden und der eigene Identifier ist von niederer Priorität, muss gewartet werden, bis der Bus frei ist (siehe Kapitel 2.4.4).
Data Length Code	4	Im Feld DLC wird angegeben, wie viele Nutzbytes im Data Field enthalten sind.
Data Field	0 bis 64	In diesem Feld werden bis zu 8 Nutzbytes übertragen. Es entfällt bei Remote Frames.
CRC Sequence	15	Die zyklische Blockprüfung (Cyclic Redundancy Check (CRC)) sichert die Nutzbytes mit einer Prüfsumme ab. Über alle Bits (auch Stuffbits) vom SoF bis einschließlich des Data Frames wird mit einem Generatorpolynom ein Gesamtpolynom gebildet. [siehe Robert Bosch GmbH, 1991, S. 13] Dieses Gesamtpolynom ist ein Vielfaches der zu übertragenden Bits. Der Empfänger erkennt deshalb schnell einen Fehler, wenn die Division einen Rest ergibt.
Acknowledgement	1	Im Acknowledgement (ACK)-Slot quittiert der Empfänger den CRC mit positiv (dominant) oder negativ (rezessiv). Negativ quittierende CAN-Knoten müssen ein Error Flag senden. Bleibt der ACK-Slot negativ, bis die Übertragung schlussendlich den Sender erreicht, bricht er ebenfalls mit einem Error Flag ab.
Error Flag	6	Das Error Flag wird bei Fehlern direkt nach dem ACK-Slot übermittelt. Es werden 6 Bit gleicher Polarität eingefügt, was gegen die Regeln des Bitstuffings verstößt.
End of Frame	7	Im EoF-Feld werden sieben rezessive Bit übertragen und somit das Ende der Nachricht signalisiert.

2.4.6 Fehlerverfolgung

Im CAN-Protokoll kann nach Fehlern bei der Übertragung gescannt und auf diese reagiert werden, um die Integrität des Netzwerkes zu wahren. Der Bus kann nach Bitfehlern, Stuffingfehlern, Formfehlern und ACK-Fehlern untersucht werden. Bei Bitfehlern prüft der sendende Knoten, ob der Buspegel mit der gesendeten Nachricht übereinstimmt. Bei Formfehlern wird geprüft, ob an signifikanten Stellen einer CAN-Botschaft auch das Bit mit der vorgeschriebenen Wertigkeit übertragen wird. Es wird unterschieden, ob der Fehler von einem sendenden Knoten (Sendefehler) verursacht wurde oder beim Empfangen der Nachricht (Empfangsfehler) auftritt.
Jeder CAN-Controller zählt Sendefehler im Transmission Error Counter (TEC) und jeden Empfangsfehler im Receive Error Counter (REC) mit und bei erfolgreichen Übertragungen werden die Counter dekrementiert. Wenn ein Error Flag von einem Sender ausgeht, wird der TEC um acht und der REC um eins erhöht. Sendet der Empfänger ein Error Flag wird der REC um acht erhöht. Im Normalzustand (Error Active) besteht das Error Flag aus sechs dominanten Bits (siehe Tabelle 2.1).
Übersteigen TEC und/oder REC einen Wert von 127, wechselt der CAN-Controller in den Error Passive Mode. Hier wird das Error Flag durch sechs rezessive Bit repräsentiert und zwischen Data oder Remote Frames muss die Suspend Transmission Time von acht Bit abgewartet werden.
Bei Ausfall eines CAN-Controllers oder extremer Fehlerhäufigkeit (TEC > 255) erfolgt ein Zustandsübergang nach Bus Off, d.h. der CAN-Controller trennt sich vom Bus. Dieser Zustand kann nur durch Eingriff des Hosts (in dem dieser $128 \cdot 11 = 1408$ rezessive Bit sendet) oder durch einen Hardware-Reset verlassen werden.

2.5 On-Board-Diagnose

Die On-Board-Diagnose (OBD) dient der Analyse aller abgasbeeinflussender Systeme eines Fahrzeuges. Alle Informationen können über ein, im Normensatz ISO 15031 deklariertes, Nachrichtenprotokoll abgefragt werden. Da OBD auch Steuergräte im CAN überwacht, kann die Schnittstelle genutzt werden, um auf den Bus zuzugreifen. Es wird zwischen OBD-I und OBD-II unterschieden, wobei OBD-I lediglich alle, bis zur offiziellen Normierung verbauten, Systeme bezeichnet. OBD speichert Fehler innerhalb der Netzwerke und gibt diese über Kontrollleuchten im Cockpit an den Fahrer aus.
Der Port muss sich in jedem modernen Auto in den USA und Europa befinden, für Modelle mit Ottomotor seit 2001 und mit Dieselmotor seit 2004 [Beiter et al., 2010, S. 46]. Sie soll auch weltweit einheitlich standardisiert werden. Die OBD-II Buchse ist ein 16-poliges Stecksystem, für das Auslesen mit einer Diagnosesoftware werden jedoch nicht alle Pins benötigt. Der Pin 6 überträgt das Signal für CAN High, der Pin 14 überträgt das Signal für CAN Low. Ein OBD-II-Port ist in Abbildung 2.8 gezeigt.

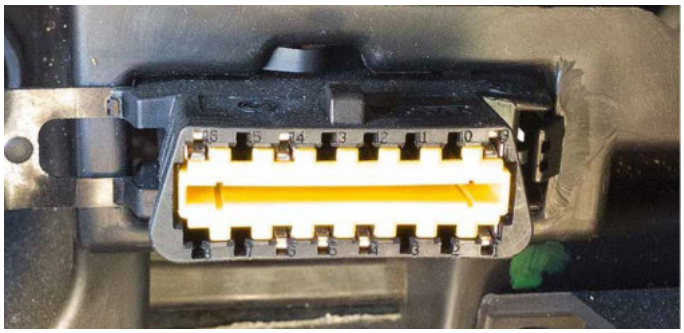

Abbildung 2.8: OBD-II-Port eines Alfa Romeo Gulietta, Baujahr 2012. Quelle: [Palanca et al., 2017, S. 47]

2.6 CANalyzer

Das Programm CANalyzer der Firma Vector Informatik GmbH wird eingesetzt, um den CAN-Bus eines Systems analysieren und manipulieren zu können. Die Software ist ein, nach der Norm für OBD Scan Tools, vollwertiger Diagnosetester. Der OBD-Port wird mit einem CAN-Interface der selben Firma verbunden und so die Signale für CAN Low und CAN High über USB einem Computer mit der Software zur Verfügung gestellt. Der Nachrichtenverkehr lässt sich im Netzwerk vollständig mitschneiden, die übertragenen Signale der Steuergeräte können grafisch in Paneln dargestellt und Nachrichten können effektiv in Datenbanken verwaltet werden. Es lassen sich CAN-Botschaften, wie zum Beispiel Remote Frames, als auch Diagnosenachrichten auf den Bus senden, um Informationen über den Zustand des Systems zu erhalten. Durch die beinhaltete Programmiersprache CAN oder Communication Access Programming Language (CAPL), die an C angelehnt ist, können Funktionen und Panel einfach programmiert werden. In dieser Arbeit wird die Professional-Variante der Software verwendet, die Zugang zu allen Funktionen erlaubt. Die grafische Oberfläche der Software ist in Abbildung 2.9 zu sehen.

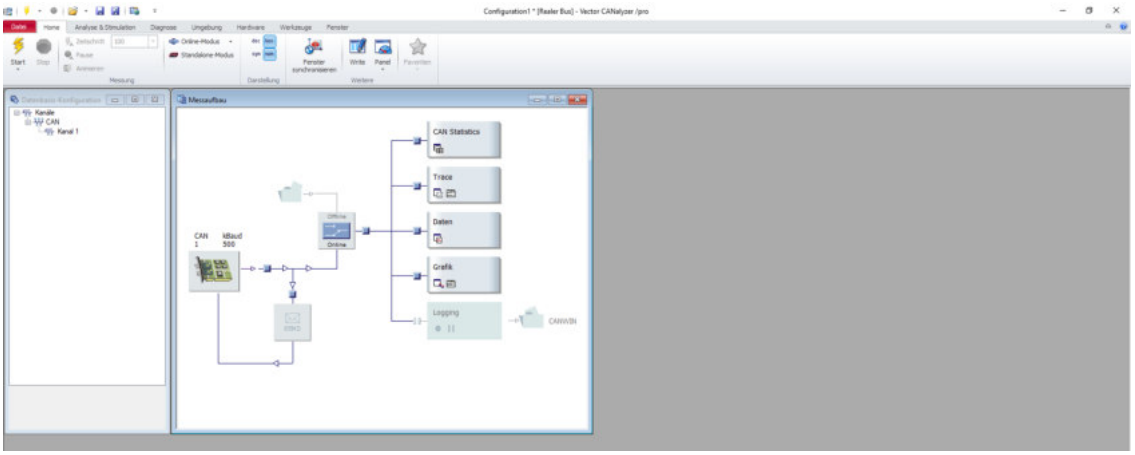

Abbildung 2.9: CANalyzer. Quelle: Bildschirmaufnahme

2.7 Pulsweitenmodulation

Ist die Eingangsspannung für ein im Netzwerk verbautes Gerät zu hoch, kann die Betriebsspannung durch die Pulsweitenmodulation (PWM) geregelt werden. Die Eingangsspannung wird dabei zyklisch unterbrochen (siehe Abbildung 2.10) und die Zeitspanne, in der die Spannung zugeschaltet wird, lautet Pulsbreite (auch Pulsweite). Als Duty-Cycle wird das Verhältnis zwischen Pulsbreite zu Pulspause in Prozent bezeichnet und in der Abbildung beträgt dieses 50 %. Der Flächeninhalt unter U_{Betrieb} entspricht dabei immer dem Flächeninhalt der Pulsbreite, es besteht demnach ein direkter Zusammenhang zwischen Pulsbreite und Betriebsspannung.

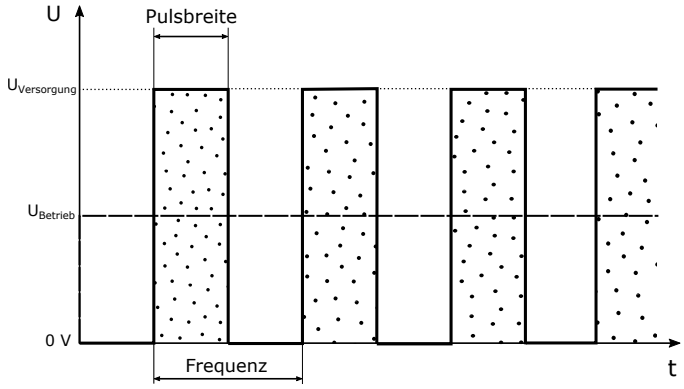

Abbildung 2.10: Skizze über das Prinzip der Pulsweitenmodulation. Quelle: eigene Arbeit

2.8 Ausgewählte Sicherheitslücken und Angriffsszenarien

Wie bereits aus Kapitel 2.4 hervorgeht, lässt das CAN-Protokoll jede Botschaft, solange sie keine Fehler aufweist, auf dem Bus zu. Nachfolgend wird erläutert, welche Sicherheitslücken aktuell in der Forschung bekannt sind und wie mögliche Angriffe verlaufen könnten.

2.8.1 Angriffe durch eine kompromittierte ECU und Entwicklung eines IDS

In der Master Thesis von Bresch und Salman wird beschrieben, wie eine ECU in einem simulierten Netzwerk Angriffe ausführt (siehe Kapitel 2.8). Später wird diese in einem vollwertigen Auto verbaut. Dadurch, dass im CAN-Protokoll keine Authentifizierung von Nöten ist, können alle CAN-Knoten Botschaften durch eine kompromittierte ECU erhalten. Ziel dieser Masterarbeit war es zunächst, das Verhalten des Netzwerkes bei

regulärem Betrieb zu erfassen und so Anomalien im Verhalten für ein Intrusion Detection System (IDS) zu detektieren. Dazu wurden insgesamt drei Szenarien getestet:

- Einschleusen von CAN-Botschaften mit, durch aufgestellte Regeln, unerlaubter Signatur
- Erhöhung der Geschwindigkeit mit zu hoher Differenz zwischen Ist- und Sollwert
- stetige Erhöhung der Geschwindigkeit bis zu gewünschtem Sollwert

Unerlaubte Nachrichten waren in diesem Versuch Nachrichten mit ID 0x00, Nachrichten, die über ein Gateway versendet wurden oder Nachrichten, die von keinem Knoten, durch die Konfiguration der Masken der Knoten, erhalten werden konnten. [Bresch und Salman, 2017, S. 43]
Da von außen auf den Bus eingeschleuste, bösartige Nachrichten die ursprünglich von den Steuergeräten gesendeten Informationen überschreiben könnten, wäre die Funktion des Netzwerkes gestört. Diese Fehlinformationen stören den normalen Betrieb und können den Fahrer ablenken, was gegebenenfalls Unfälle zur Folge hat. Im Versuch konnte gezeigt werden, dass sich die Geschwindigkeit des Autos manipulieren ließ, was eine sehr große Sicherheitslücke und Gefahr für alle Verkehrsteilnehmer darstellt.
Durch das IDS werden Textausgaben im CANalyzer generiert, die diese Verstöße melden. Eine Möglichkeit, diese Warnungen auch im Cockpit des Autos anzeigen zu lassen und so den Fahrer vor einem Angriff zu warnen bzw. auch die Beweissicherung nach einer Attacke ohne den CANalyzer zu gewährleisten, wurde jedoch nicht erarbeitet. Die Autoren weisen jedoch darauf hin, dass daran noch weiter geforscht werden sollte, was, wenn das IDS für den Demonstrator verwendet werden kann, auch mit dieser Arbeit möglich wäre. Die Funktionen, die die virtuellen ECUs für das IDS übernehmen, lassen sich auch auf real verbaute Steuergeräte anwenden und müssten per Softwareupdate nachgerüstet werden.

2.8.2 Denial-of-Service-Angriff

In der Arbeit von Palanca et al. [2017] wird ein Mikrocontroller so programmiert und modifiziert, dass, nach Anschluss an den OBD-II-Port des Autos, eine DoS-Attacke ausgeführt werden kann. Es wird nach bestimmten IDs oder Payloads gescannt und anschließend diese Art von Nachricht blockiert werden. Sobald die systemweite Synchronisation durch den SoF ausgeführt wird, wird durch eine Interrupt Service Routine (ISR) der modifizierte CAN-Knoten synchronisiert und ein Timer Interrupt erlaubt. Dieser Timer zählt genau eine Bitzeit des CAN-Bus ab und generiert den Interrupt. In dieser ISR wird geprüft, ob die Nachricht, die auf dem Bus gesendet wurde, dem Ziel entspricht und setzt dann das erste rezessive Bit auf dominant. Der sendende CAN-Knoten detektiert nun einen Bitfehler und sendet ein Error Flag. [Palanca et al., 2017, S. 41] Der Zielknoten empfängt deshalb kein Signal und der Sender geht nach mehrmaligen Versuchen die Nachricht erneut zu senden in den Bus Off Status über [Palanca et al., 2017, S. 90], das heißt

der CAN-Controller trennt sich vom Bus. Dieser Zustand kann nur durch Eingriff des Hosts (siehe Kapitel 2.4.6) oder durch einen Hardware-Reset verlassen werden. Durch diese Arbeit mussten keine vollständigen CAN-Botschaften, sondern lediglich ein Bit auf den Bus gesendet werden. Durch den Versuchsaufbau konnte der Parkassistent des Autos komplett abgeschaltet werden. Daraus ergibt sich die Möglichkeit, jede beliebige Funktion, deren Informationen über den CAN-Bus verteilt werden, auszuschalten und die Wiederaufnahme des regulären Betriebs solange zu unterbinden, bis der eingeschleuste Mikrocontroller vom Bus entfernt wurde. Sollte der Fahrer keine Kenntnis über diese Art von Angriffen haben und der Mikrocontroller wird nicht von ihm selbst entfernt, müsste das Fahrzeug in eine Werkstatt verbracht werden, das Auto kann in dieser Zeit nicht vom Besitzer genutzt werden und es fallen Kosten an.

2.8.3 Remoteangriff auf das CAN

2015 erarbeiteten die Autoren Miller und Valasek [2015] eine Möglichkeit ohne physische Anbindung an das CAN eines Jeep Cherokee aus dem Jahr 2014 den Bus mit Schadcode zu infizieren. Es gelang ihnen, eine Verbindung zum Infotainment System, welches mit dem High Speed und Low Speed CAN-Bus verbunden ist, über das Mobilfunknetz der Firma Sprint aufzubauen. Durch die Anbindung kann das System einen Wi-Fi Hotspot für mobile Endgeräte zur Verfügung stellen, sofern diese Option bezahlt wurde. [Miller und Valasek, 2015, S. 25, 45] Doch selbst wenn der Nutzer nicht für den Dienst bezahlt, lässt sich nach Autos mit dem offenen Port 6667 und einer IP-Adresse in den Bereichen 21.0.0.0/8 und 25.0.0.0/8 von einem anderen, mobilfunkfähigen Gerät im Sprint Netz scannen. [Miller und Valasek, 2015, S. 45 f.] Durch einen Exploit des D-Bus, dessen Service hinter dem offenem Port steckt, kann mit root-Rechten Code auf einer Shell ausgeführt werden. Es lässt sich Schadcode in das System einspielen und es wird möglich, CAN-Nachrichten zu einem Auto zu senden, dass sich nicht in unmittelbarer Nähe befinden muss. [Miller und Valasek, 2015, S. 64-71] Es ließen sich die Blinker, die Verriegelung und die Anzeige des Tachometers über CAN-Narichten im CBFF (siehe Kapitel 2.4.5) beeinflussen. Durch Diagnosenachrichten bei Geschwindigkeiten unter 5-10 mph ließen sich auch der Motor, die Bremsen und die Lenkung manipulieren. [Miller und Valasek, 2015, S. 83 ff.] Die getesteten Sicherheitslücken wurden von den Herstellern behoben. [Miller und Valasek, 2015, S. 87 f.]

3 Methoden

In diesem Kapitel wird der Prozess von der Kozeptionierung, der Programmierung und schlussendlich dem Zusammenbau der Komponenten des Demonstrators beschrieben.

3.1 Grundkonzeption

Für die Realisierung des Projektes wurden bestimmte Hardware- und Softwarekomponenten vorgegeben:

- Das Grundgerüst für diese Arbeit stellt das Robot Car Kit von Joy-It dar. Vier Motoren sind im Lieferumfang enthalten, die angesteuert werden müssen.
- Dazu wird das Evaluationsboard EVAL6207N von STMicroelectronics verwendet.
- Da der Demonstrator mit einem CAN-Bus ausgestattet sein soll, wird ein weiterer Mikrocontroller, der CAN beherrscht, benötigt. Mit dem AVR-CAN-Board von Olimex kann das EVAL6207N Signale für die Steuerung erhalten.
- Dem Aufbau sollen Scheinwerfer und Blinker, die denen eines Fahrzeuges nachempfunden sind, hinzugefügt werden.
- Die Steuerung des Demonstrators erfolgt durch die Software CANalyzer, über die geeignete CAN-Botschaften erzeugt werden.
- Um Nachrichten von einem Computer mit der Software an das Steuergerät (AVR-CAN-Board) des Demonstrators zu übertragen, wird das CAN-Interface VN1610 von Vector benötigt. Zwischen CAN-Interface und AVR-CAN-Board besteht der eigentliche CAN-Bus.

Der Versuchsaufbau ist schematisch in Abbildung 3.1 dargestellt.

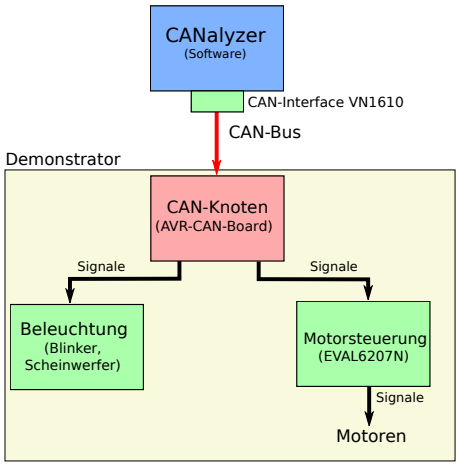

Abbildung 3.1: Blockschaltbild des Versuchsaufbaus. Quelle: eigene Arbeit

3.2 Präzisierung der Aufgabenstellung

Nach den Überlegungen des vorhergehenden Kapitels ergeben sich für die einzelnen Komponenten unterschiedliche Aufgaben. Diese werden im Folgenden vorgestellt und präzisiert.

CAN-Knoten (AVR-CAN-Board)
Das AVR-CAN-Board ist die Komponente, welche hauptsächlich die Steuerung innerhalb des Demonstrators übernimmt. Es muss ein Programm entwickelt werden, in dem Funktionen für die Motorsteuerung und die Beleuchtung enthalten sind. CAN-Nachrichten werden hier ausgewertet und weiter verarbeitet. Auch eine physische Verbindung über Kabel zu den weiteren Komponenten muss realisiert werden. Außerdem wird das Board als zentraler Stromversorger aller Bauteile genutzt.

Motorsteuerung (EVAL6207N)
Das Evaluationsboard erhält Informationen für die Ansteuerung der Motoren durch das AVR-CAN-Board. Dazu werden Signale des CAN-Knotens über die physische Verbindung empfangen und umgesetzt. An diesem Board müssen die Motoren angeschlossen sein.

Beleuchtung
Die Beleuchtung des Demonstrators erfolgt durch mehrere Leuchtdioden. Um die Scheinwerfer und Blinker eines Fahrzeuges zu imitieren, werden insgesamt zwei Blinker und Front-, Rückscheinwerfer und Bremsleuchten in entsprechenden Farben verbaut. Die Stromversorgung muss über Vorwiderstände reguliert werden. Des Weiteren muss eine Möglichkeit für die Befestigung der Dioden am Robot Car Kit entwickelt werden.

CANalyzer
AVR-CAN-Board und CANalyzer sind über das CAN-Interface VN1610 miteinander zu verbinden. In der Software müssen CAN-Botschaften definiert werden, die zu den Funktionen des CAN-Knotens kompatibel sind. Das bedeutet, dass die Identifier der Nachrichten übereinstimmen müssen (siehe Tabelle 2.1). Um die Steuerung des Demonstrators für den Nutzer intuitiver zu gestalten, wird eine grafische Oberfläche in Form eines Panels erstellt.

Damit digital-forensische Analysen mit dem Demonstrator möglich sind, müssen Angriffe auf ihn ausgeübt werden können. Diese Angriffe sollen möglichst realitätsnah sein, weshalb die Szenarien aus Kapitel 2.8 angewendet werden. Da das IDS von Bresch und Salman bereits Erfolge beim Detektieren von Angriffen in einem realen Automobil zeigte, wird dieses auch beim Demonstrator Anwendung finden.

Zusammenfassend ergeben sich diese Anforderungen an den Demonstrator:

- Steuerung über CAN-Botschaften
- Realisierung grundlegender Funktionen eines Fahrzeuges
- (daraus folgend) Mobilität
- Eintrittspforten für Angreifer
- Aufzeichnung von Attacken

Aus der bisherigen Aufgabenstellung ergibt sich jedoch das Problem der fehlenden Mobilität. Sowohl die Stromversorgung der Bauteile als auch die Übertragung der CAN-Botschaften sind kabelgebunden. Für eine bessere Anschaulichkeit der Funktionen, vor allem in der Lehre, muss der Aktionsradius erhöht werden. Dafür wird die, bis zu diesem Zeitpunkt geplante, Implementierung des CAN-Busses überdacht. Um die Verbindung drahtlos zu gestalten, müssen dem Versuchsaufbau weitere passende Module hinzugefügt werden. Eine Möglichkeit dafür sind Gateways, die Informationen der CAN-Botschaften in einem alternativen Übertragungsprotokoll umstrukturieren. Dabei muss jedoch weiterhin ein CAN-Bus im Demonstrator erhalten bleiben. Die Stromversorgung lässt sich durch eine Batterie oder einen Akkumulator realisieren.
Der finale Versuchsaufbau ist in Abbildung 3.2 gezeigt.

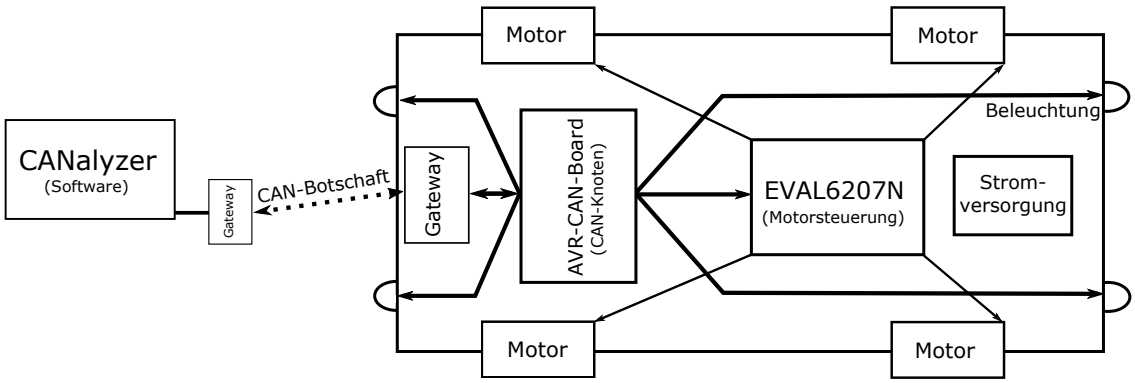

Abbildung 3.2: Skizze des Versuchsaufbaus. Quelle: eigene Arbeit

3.3 Auswahl weiterer Komponenten

Wie im vorhergehenden Kapitel erläutert, ist der Demonstrator durch die gegebenen Bauelemente und dem daraus resultierenden Versuchsaufbau nicht mobil. Eine größere Herausforderung stellt der kabelgebundene CAN-Bus dar. Eine mögliche Lösung ist die physische Verbindung zwischen Demonstrator und CANalyzer durch eine drahtlose zu ersetzen. Dabei soll jedoch weiterhin das CAN bestehen bleiben.

3.3.1 Drahtlose Verbindung

Ein CAN-Frame besteht aus maximal 131 Bit (CEFF mit acht Nutzbytes). Eine Botschaft kann deshalb leicht in den Payload diverser Übertragungsprotokolle eingebettet werden. Die eintreffenden Informationen müssen in einer Schnittstelle empfangen und in CAN-Botschaften umorganisiert werden, bevor sie über den Bus an das AVR-CAN-Board oder den CANalyzer übermittelt werden (siehe Abbildung 3.3).

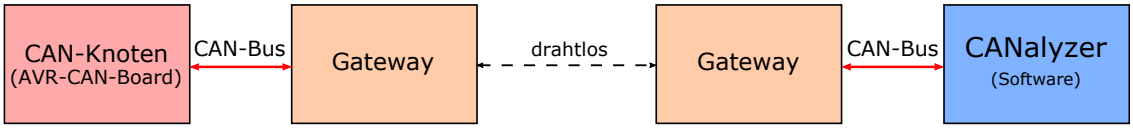

Abbildung 3.3: Schematische Darstellung der drahtlosen Kommunikation. Quelle: eigene Arbeit

Gängige drahtlose Übertragungsarten sind Bluetooth und das Wireless Local Area Network (WLAN). In lokalen Netzen wird Ethernet häufig als Standard genutzt und hält derzeit Einzug als Bussystem in Automobilen. WLAN basiert ebenfalls auf diesen Standards. [Sauter, 2015, S. 311 f.] Das Netzwerk im Demonstrator soll möglichst flexibel sein und den aktuellen Stand der Technik in Automobilen in diesem Bereich widerspiegeln. Deshalb wird in dieser Arbeit auf WLAN gesetzt.
Ein zusätzlicher Mikrocontroller, der die Funktion der Schnittstelle im Versuchsaufbau übernehmen wird, muss deshalb sowohl das WLAN- als auch das CAN-Protokoll beherrschen. Insgesamt wurden drei Varianten verglichen.

CAN to Wi-Fi Gateway
Texas Instruments stellt eine Anleitung zur Verfügung, in der Komponenten für ein Gateway beschrieben sind. [Knapp und Lam, 2015] Auf einem selbst gestalteten Board müssen dazu ein CAN-Transceiver und zwei Mikrocontroller verbaut werden, einer für das CAN- und einer für das WLAN-Protokoll. Außerdem werden weitere Bauteile benötigt, zum Beispiel für die Stromversorgung.

Vorteile

- flexible Gestaltung des Boards
- Software verfügbar

Nachteile

- Selbstanfertigung des Boards nötig
- zwei Mikrocontroller werden benötigt
- zusätzliche Bauteile werden benötigt
- hoher Aufwand

Kosten: ca. 17 Euro (allein für den CAN-Transceiver und die beiden Mikrocontroller)

ESP32-Pico-Kit
Der Mikrocontroller ESP32 der Firma Espressif erfüllt die Anforderung, sowohl WLAN als auch CAN zu beherrschen. Zusätzlich verfügt der ESP32 über Bluetooth. Er ist auf unterschiedlichen Entwicklungsboards bereits verbaut. Espressif stellt Softwarebibliotheken für den ESP32 bereit, darunter auch für CAN. Eine Firmware müsste jedoch mit diesen Bibliotheken erst programmiert werden. Eine platzsparende Ausführung eines Entwicklungsboards mit ESP32 ist das ESP32-Pico-Kit (siehe Abbildung 3.4).

Vorteile

- Mikrocontroller der den Anforderungen gerecht wird
- kleines Board für flexiblen Einsatz
- Der ESP32 beherrscht viele weitere nützliche Protokolle (zum Beispiel Bluetooth).

Nachteile

- zusätzlich wird ein CAN-Transceiver benötigt
- kaum weitere Ports
- Software muss selbst erarbeitet werden

Kosten: ca. 10 Euro (ohne CAN-Transceiver)

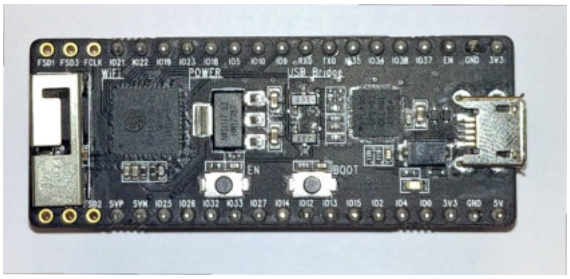

Abbildung 3.4: ESP32-Pico-Kit V4. Quelle: eigene Aufnahme

ESP32-EVB
Olimex bietet eine Komplettlösung als Evaluationsboard mit ESP32 und CAN-Transceiver an - das ESP32-EVB (siehe Abbildung 3.5).

Vorteile

- alle nötigen Komponenten auf einem Board
- Anforderungen werden vollständig erfüllt
- weitere nützliche Schnittstellen (unter anderen LAN, Micro-USB, SD-Karte)

Nachteile

- Software muss selbst erarbeitet werden

Kosten: 26 Euro

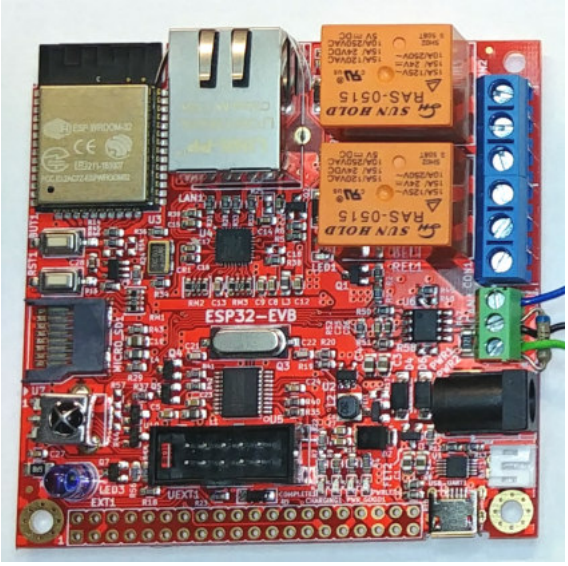

Abbildung 3.5: ESP32-EVB. Quelle: eigene Aufnahme

Nach diesen Überlegungen erweist sich das ESP32-EVB als die beste Lösung für diese Arbeit.

3.3.2 Stromversorgung

Der Demonstrator wird mit einer mobilen Stromversorgung ausgestattet. Dazu muss zunächst entschieden werden, ob eine Batterie oder ein Akkumulator zum Einsatz kommt. Ein Akkumulator lässt sich mehrmals aufladen und ist deshalb nachhaltiger als eine Batterie. Der Akku kann außerdem dauerhaft im Demonstrator verbaut werden, sofern die Anschlüsse für ein Ladegerät gut zugänglich platziert werden.
Die Kapazität des Akkus ergibt sich aus dem Verbrauch der Bauteile, diese sind in Tabelle 3.1 aufgelistet. Das AVR-CAN-Board versorgt alle weiteren Bauteile (bis auf die Motoren)

Tabelle 3.1: Details über die Stromversorgung der Bauteile des Demonstrators.

Bauteil	U in V	I in mA	Quelle
AVR-CAN-Board	7 - 12	50	[Olimex Ltd., 2010, S. 8]
EVAL6207N	5	100	Messung
ESP32-EVB	5	500	[Espressif Systems, 2018, S. 10]
Motoren	3 - 9	200	[Joy-IT, 2017, S. 2]
LEDs (10 Stück)	5	je 20	Messung

mit der nötigen Spannung, weshalb der Akku mindestens eine Spannung von 7 Volt liefern muss. Auch die Motoren lassen sich mit 7 Volt betreiben. Maximal werden 1050 mA verbraucht, um den Demonstrator für eine Stunde zu betreiben, muss der Akkumulator deshalb eine Kapazität von mindestens 1050 mAh aufweisen.
Im Folgenden werden drei Akkumulatoren mit diesen Daten verglichen. Eine weitere Anforderung ist ein möglichst geringer Einkaufspreis. Im Besonderen eignen sich Akkupacks aus dem Modellsportbereich für die Größe und die Anwendung des Demonstrators. In der Tabelle 3.2 werden drei Ausführungen solcher Packs vergleichend gegenübergestellt. Genaue Preise sind tagesabhängig. Die Akkupacks sind bei den gängigen Händlern zu

Tabelle 3.2: Vergleich verschiedener Ausführung geeigneter Akkumulatoren

	LRP	Gens ace	dfmodels
Ausgangsspannung	8,4 V	7,4 V	7,4 V
Kapazität	4600 mAh	2500 mAh	3000 mAh
Betriebsdauer	4,4 h	2,4 h	2,9 h
separater Anschluss für ein Ladegerät	nein	ja	ja
Preis	ca. 54 Euro	ca. 18 Euro	ca. 27 Euro

beziehen. Da der Demonstrator auch in der Lehre Anwendung finden soll, sollte die Betriebsdauer mindestens 90 Minuten (übliche Dauer einer Lehreinheit) betragen. Mit dem Akkupack von dfmodels könnten fast zwei Lehreinheiten ohne Ladeunterbrechung durchgeführt werden. Deshalb wurde sich letztendlich für diese Ausführung entschieden. Der Stromfluss kann durch einen Schalter manuell reguliert werden (siehe Abbildung A.1).

3.4 Realisierung der Hardwarekomponenten

Dieses Kapitel beschreibt den eigentlichen Prozess des Aufbaus des Demonstrators. Alle bereits vorgestellten Komponenten werden hier zusammengeführt und physisch über Kabel miteinander verbunden.

3.4.1 Motorsteuerung

Der Mikrocontroller des AVR-CAN-Boards ist der AT90CAN128 der Firma Atmel. Nähere Informationen können dem Datenblatt (siehe [Atmel Corporation, 2008]) entnommen werden. Das Evaluationsboard EVAL6207N wird zur Steuerung der Motoren genutzt. [STMicroelectronics, 2003, S. 49 ff.], [STMicroelectronics, 2006, S. 25-31], [STMicroelectronics, 2014].
Das EVAL6207N kann insgesamt über fünf mögliche Zustände der Motoren entscheiden (siehe Tabelle 3.3) und leitet die jeweiligen Signale an diese weiter. Dazu müssen Pins

des AVR-CAN-Boards mit den Pins des EVAL6207N verkabelt sein, deren Pegel für diesen Zweck eingelesen und ausgewertet werden. Drei Pins des EVAL6207N sind für die Steuerung eines Motors zuständig:

- Pin EN (Enable, Signal für „ein“ oder „aus“),
- Pin IN1 (Input 1, Steuerung der Zustände der Motoren),
- und Pin IN2 (Input 2, Steuerung der Zustände der Motoren).

Tabelle 3.3: Wahrheitstabelle über die Zustände der Motoren. Quelle: [STMicroelectronics, 2014, S. 13]

EN	IN1	IN2	Beschreibung
0	-	-	Motoren sind ausgeschaltet
1	0	0	Motoren stoppen
1	1	0	Fahrtrichtung vorwärts
1	0	1	Fahrtrichtung rückwärts
1	1	1	Motoren stoppen

Da das Board zwei Ausgänge für die Ansteuerung besitzt, werden die Pins für den jeweiligen Ausgang mit A und B gekennzeichnet. Die Verkabelung zwischen AVR-CAN-Board und Motorsteuerboard ist in Abbildung 3.6 gezeigt. Die Wahl der Pins auf der Seite des CAN-Knotens hängt von der Programmierung des Mikrocontrollers ab, die in Kapitel 3.5.4 beschrieben ist. Ein vollständiger Schaltplan findet sich im Anhang (Abbildung A.1).

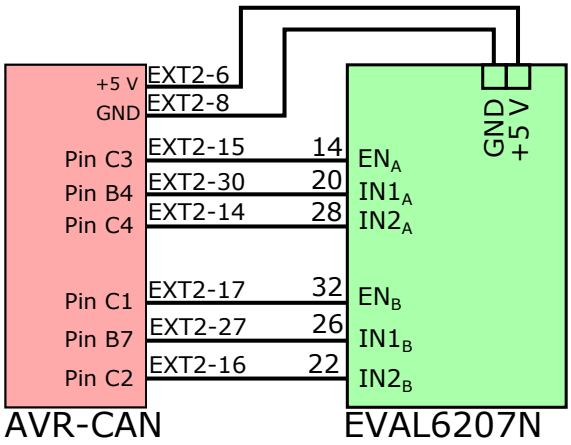

Abbildung 3.6: Skizze der Verkabelung für die Motoransteuerung. Quelle: eigene Arbeit

3.4.2 Beleuchtung

Um die LEDs anzubringen, wird eine Euroleiterplatte so zugeschnitten, dass sie auf dem hinteren Teil einer Grundplatte des Robot Car Kits befestigt werden kann. Da sich durch den bisherigen Aufbau und die Verkabelung der Bauteile das AVR-CAN-Board ebenfalls

im hinteren Teil des Modells befindet, muss die Platine eine Aussparung für dieses haben. Die zugeschnittene Euroleiterplatte ist in Abbildung 3.7 skizziert. Es werden insgesamt im hinteren Teil zwei Blinker, zwei Rückleuchten, zwei Rückscheinwerfer und zwei Bremsleuchten verbaut. Für den vorderen Teil, an den zwei Frontscheinwerfer angebracht werden, können die übrigen Teile der Leiterplatte genutzt werden. Diese Platte wird mittig am vorderen Bereich des Demonstrators ausgerichtet. Verwendet werden für die

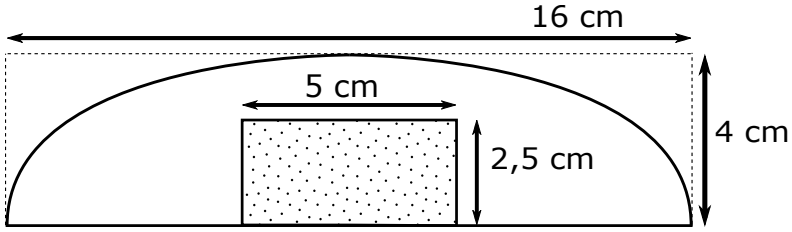

Abbildung 3.7: Skizze der zugeschnittenen Euroleiterplatte für die Befestigung der LEDs. Quelle: eigene Arbeit

Frontscheinwerfer LEDs der Firma HuiYuan in weiß, mit 5 mm Durchmesser und 20 mA Stromstärke. Für die restlichen LEDs wurden passende Dioden aus dem LED-Sortiment S036 von Kemo Electronic, die mit einer Stromstärke unter 20 mA betrieben werden sollen, verwendet.
Die eingesetzten LEDs werden mit einer Betriebsspannung von 5 V durch das AVR-CAN-Board versorgt, die Stromstärke wäre in diesem Fall für einen dauerhaften Betrieb zu hoch. Deshalb müssen Vorwiderstände für alle LEDs berechnet werden. Für die Berechnung wird die Durchlassspannung U_D und die Stromstärke benötigt. Zur Berechnung des Vorwiderstands werden folgende Formeln benötigt:

$$R = \frac{U_R}{I}$$
$$U_R = U_B - U_D$$

Die Betriebsspannung U_B beträgt 5 V und die Durchlassspannung U_D kann aus dem Datenblatt der LEDs oder experimentell durch eine Messung ermittelt werden. Alle Werte für die verwendeten LEDs können Tabelle 3.4 entnommen werden. Da nicht alle

Tabelle 3.4: Daten der LEDs

	Farbe	Pins auf Ext1	U_D in V	I in mA	R in Ω	$R_{gewählt}$ in Ω
Frontscheinwerfer	weiß	28, 29	3,3	13	85	100
Bremsleuchten	rot	24, 25	2,0	12	250	270
Rückleuchten	rot	26, 27	2,2	13	215	220
Rückscheinwerfer	weiß	30, 31	2,9	15	140	150
Blinker	gelb	12, 13	2,1	12	241	270

errechneten Vorwiderstände auch so als zu verbauende Widerstände existieren, sollte der nächst höhere Wert verwendet werden. Die LEDs werden an der Euroleiterplatte ausgerichtet und verlötet. Alle Widerstände werden mit der Anode der zugehörigen LED

und mit dem Ext1-6 Pin des AVR-CAN-Boards verbunden, der die Betriebsspannung von 5 Volt liefert. Die Kathode der LED wird mit dem Pin des AVR-CAN-Boards verkabelt, der den Stromfluss ein- oder ausschaltet. Die Verkabelung ist beispielhaft für eine LED in Abbildung 3.8 gezeigt, der vollständige Schaltplan befindet sich im Anhang (Abbildung A.1).

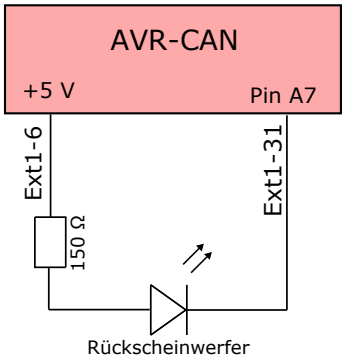

Abbildung 3.8: Skizze über die Verkabelung einer Leuchtdiode. Quelle: eigene Arbeit

3.4.3 CAN-Bus

Der CAN-Bus des Demonstrators überträgt Informationen zwischen einer ECU (AVR-CAN-Board) und dem Gateway (ESP32-EVB). Insgesamt werden für den Versuchsaufbau zwei Boards benötigt. Ein weiteres Gateway ist über das CAN-Interface VN1610 mit dem Computer verbunden, auf dem auch der CANalyzer installiert ist. Der CAN-Transceiver des ESP32-EVB wird mit drei Kabeln für CAN High (Pin CAN_T1), CAN Low und Masse verbunden. Zwischen den Ausgängen für CAN High und CAN Low wird ein 120 Ω Widerstand befestigt, da High Speed CAN-Botschaften übermittelt werden und der Bus auf der Seite des ESP32-EVB von Werk aus nicht mit einem Widerstand terminiert ist. Die Kabel werden mit einer D-Sub 9-Buchse verlötet, CAN High an Pol 7, CAN Low an Pol 3 und Masse an Pol 4. Jetzt können beide Boards sowohl mit dem CAN-Interface als auch mit dem AVR-CAN-Board verbunden werden (siehe Abbildungen 3.9 und 3.10).

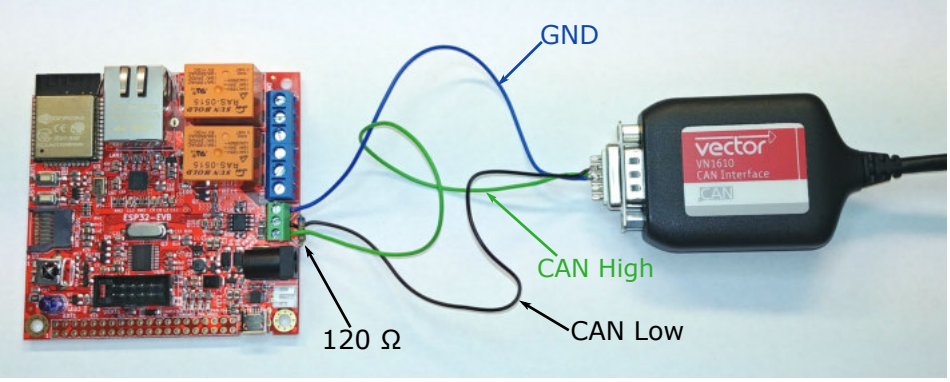

Abbildung 3.9: Verkabelung zwischen ESP32-EVB und CAN-Interface. Quelle: eigene Aufnahme

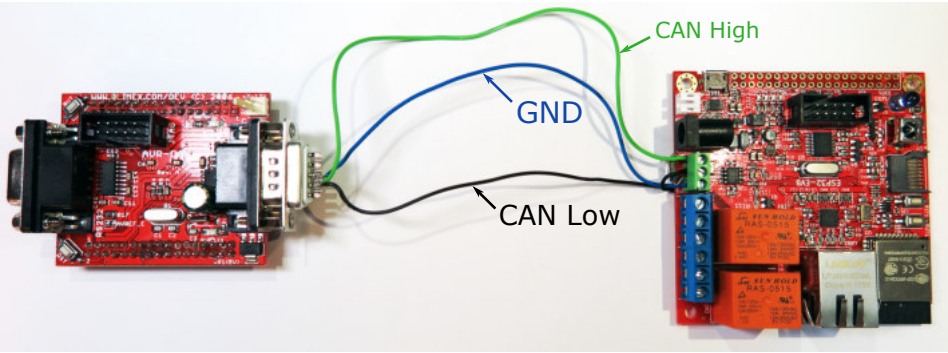

Abbildung 3.10: Verkabelung zwischen ESP32-EVB und AVR-CAN-Board. Quelle: eigene Aufnahme

Die Stromversorgung des ESP32-EVB, welches sich im Demonstrator befindet, wird über zwei weitere Kabel, die an der Rückseite an den jeweiligen Pins des Boards angelötet werden, umgesetzt. Da das Board nur mit 5 Volt Eingangsspannung versorgt werden darf, können diese Kabel mit einem entsprechenden freien Pin des AVR-CAN-Boards (zum Beispiel Ext1-3 für Masse und Ext1-5 für 5 Volt) verbunden werden. Das zweite ESP32-EVB kann über den Micro-USB-Port an den Computer, auf dem auch der CANalyzer installiert und das CAN-Interface angebracht ist, angeschlossen und somit mit Strom versorgt werden.

3.5 Realisierung der Softwarekomponenten

Im Folgenden wird die implementierte Software für dieses Projekt erläutert. Dabei wird grob auf die nötigen Funktionen eingegangen. Es wird hauptsächlich der Weg einer CAN-Botschaft von der Erstellung bis zur Verarbeitung im Demonstrator beschrieben. Die Übertragung vom Demonstrator hin zum CANalyzer gestaltet sich dazu analog.

3.5.1 Definition von CAN-Botschaften

Im CANalyzer wird für diese Arbeit ein neues Projekt angelegt. Im CANdb++ Editor wird jetzt eine neue Datenbank für CAN-Botschaften erstellt. Neu angelegte Nachrichten (Tabelle 2.1 entsprechend) benötigen einen Identifier, einen Namen und es muss festgelegt werden, ob es sich um eine Botschaft im CBFF oder CEFF handelt. Des Weiteren kann vereinbart werden, wie viele Byte im Datenfeld maximal zu übertragen sind. Zwischen definierten CAN-Knoten und den Nachrichten kann eine Relation festgelegt werden. Den Nachrichten können in der Datenbank Signale zugeteilt werden. Diese Signale liefern Informationen, zum Beispiel über den Status der LEDs oder die derzeitige Fahrtrichtung. Ein Signal kann bis zu acht Bit lang und durch ein Minimum und/oder Maximum begrenzt sein. Die Signale können mehreren Botschaften zugeordnet werden, wobei festgelegt werden muss, ab welchem Startbit das Signal im Datenfeld zu finden ist.

In der Tabelle 3.5 sind alle Nachrichten und die zugehörigen Signale des Demonstrators aufgelistet.

Tabelle 3.5: CAN-Botschaften des Versuchsaufbaus

Ziel	ID	Botschaft	Signale	Wertigkeit
Motor	0x02	Stopp		
	0x03	Fahrtrichtung vorwärts	Geschwindigkeit	0-255
			Fahrtrichtung (links/rechts)	0-2
	0x04	Fahrtrichtung rückwärts	Geschwindigkeit	0-255
			Fahrtrichtung (links/rechts)	0-2
LEDs	0x05	Blinker links	Status	0-1
	0x06	Blinker rechts	Status	0-1
	0x07	Warnblinker	Status	0-1
	0x08	Bremsleuchten	Status	0-1
	0x09	Frontscheinwerfer	Status	0-1
	0x0A	Rückleuchten	Status	0-1
	0x0B	Rückscheinwerfer	Status	0-1

3.5.2 Steuerung des Demonstrators

Um den Demonstrator steuern zu können, müssen die CAN-Telegramme über den Bus gesendet werden. Der CANalyzer bietet dazu zwei Möglichkeiten. Zum einen können die zuvor definierten Nachrichten aus der Datenbank einzeln und manuell versendet werden, zum anderen kann dazu mit dem Panel Designer eine grafische Oberfläche erstellt und mit Funktionen versehen werden.

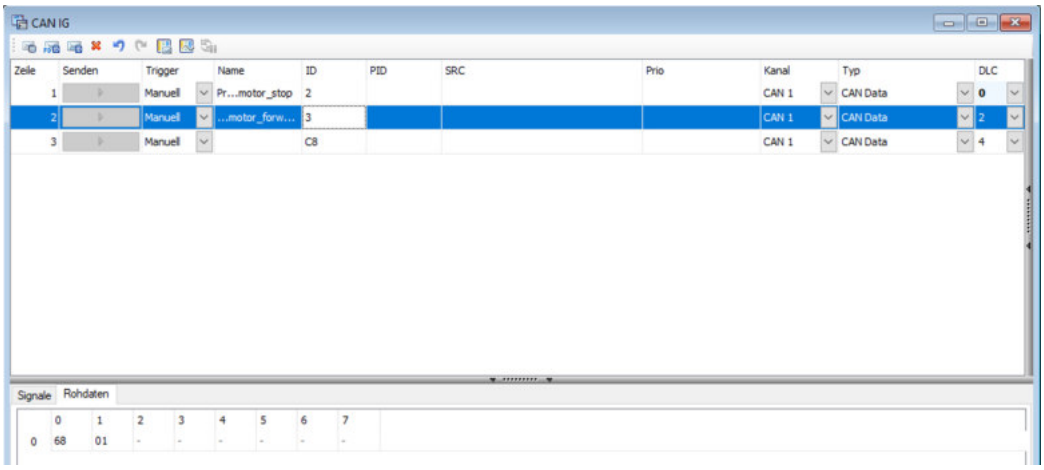

Abbildung 3.11: Interaktiver Generator mit CAN-Botschaften. Quelle: Bildschirmaufnahme

manuelles Versenden von CAN-Botschaften

Fügt man dem Versuchsaufbau im CANalyzer einen Interaktiven Generator (CAN IG) hinzu, lassen sich CAN-Botschaften aus der CANdb++ Datenbank manuell versenden. In der Abbildung 3.11 ist das Fenster für den Interaktiven Generator zu sehen.
Ihm wurden bereits zwei Nachrichten aus der Datenbank und eine weitere Nachricht, welche manuell erstellt wurde, hinzugefügt. Die Nutzbytes lassen sich editieren, in diesem Beispiel wird dem Board das Signal „Geschwindigkeit“ mit dem Wert 0x68 und das Signal „Fahrtrichtung (links/rechts)“ mit dem Wert 0x01 für „links“ übertragen.
Wird die Aufzeichnung der Nachrichten auf dem Bus gestartet, können die Nachrichten durch den Button im Fenster manuell gesendet werden. Es ist ebenfalls möglich die Nachrichten periodisch in zuvor definierten Zeitabständen in Millisekunden zu versenden oder den Wert der Signale nach bestimmten Mustern zu erzeugen. Diese Werte können zum Beispiel zufällig, durch eine Sinusfunktion oder innerhalb eines Wertebereiches generiert werden.
Im Fenster Trace kann jetzt die Kommunikation zwischen Demonstrator und CANalyzer verfolgt werden, was in Abbildung 3.12 dargestellt ist.

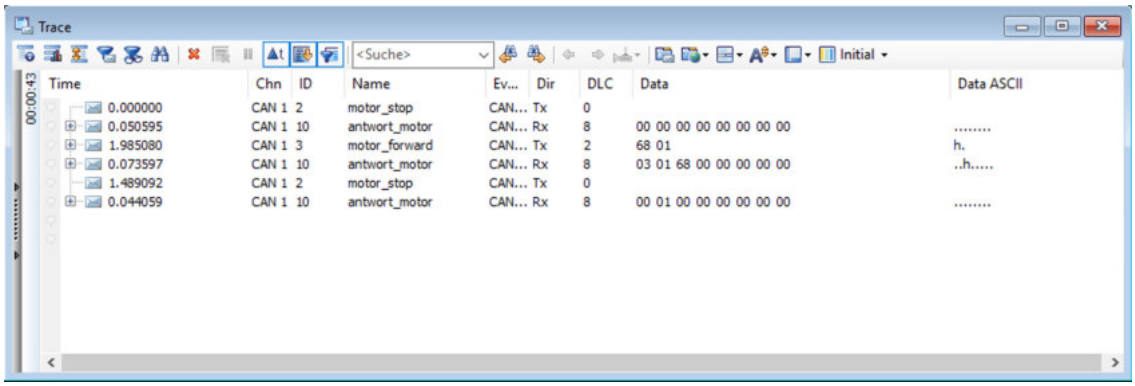

Abbildung 3.12: Aufzeichnung der Kommunikation im CAN. Quelle: Bildschirmaufnahme

Erstellung eines Panels

Der Panel Designer liefert diverse Funktionen, um grafische Oberflächen für den Versuchsaufbau zu erstellen. Per Drag-and-Drop lassen sich vorgefertigte Elemente in eine Arbeitsfläche integrieren und der Hintergrund kann durch eine Bilddatei ersetzt werden. Jedes Element kann mit einer zuvor angelegten Systemvariable verknüpft werden, damit automatisierte Funktionen auf diese zugreifen können oder auf ausgelöste Ereignisse dieser Elemente reagiert werden kann. Zusätzlich können den Bausteinen Signale aus der Datenbank zugeordnet und deren Werte angezeigt werden. Die für diese Arbeit benötigten Komponenten werden im Folgenden erläutert, ein Panel ist in Abbildung 3.13 dargestellt.

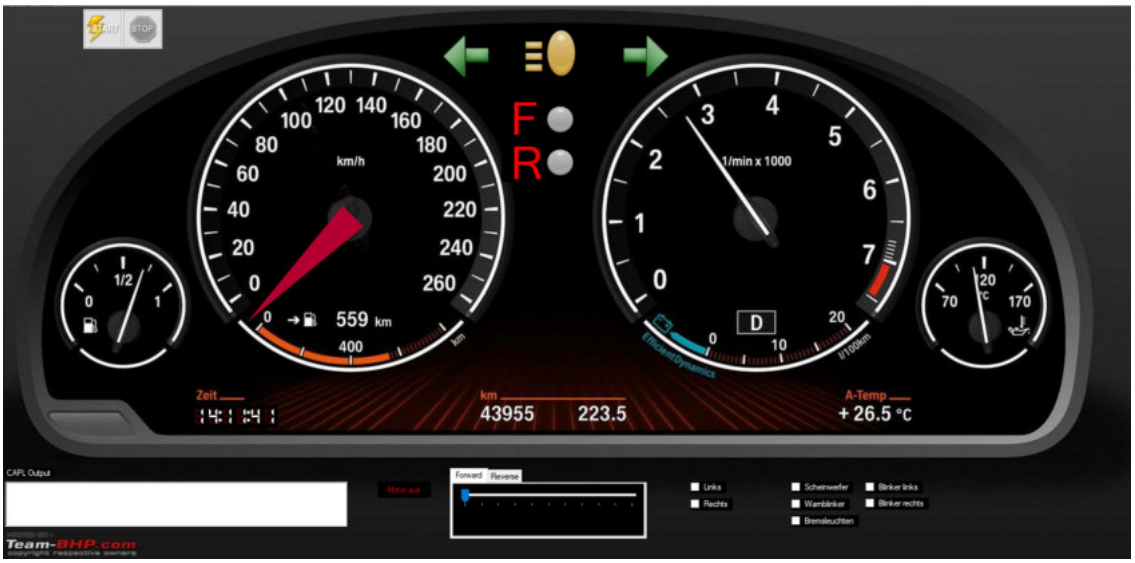

Abbildung 3.13: Panel für den Versuchsaufbau. Quelle: Bildschirmaufnahme, [Team BHP, 2014]

Tab Control

Dieses Element wird verwendet, um zwischen den Fahrtrichtungen vorwärts und rückwärts zu unterscheiden. Innerhalb eines Tabs soll sich eine Track Bar befinden.

Track Bar

Zwei Track Bars mit jeweils zehn Stufen sollen die Geschwindigkeit für die jeweilige Richtung festlegen.

Check Box

Check Boxen können den Zustand „ein“ (1) oder „aus“ (0) speichern, durch einen Klick auf die Box kann zwischen diesen Werten gewechselt werden. Durch sie soll die Fahrtrichtung (links, rechts oder keine Änderung) gewählt und die LEDs angesprochen werden können.

Button

Durch den Klick auf einen Button sollen die Motoren gestoppt werden können.

Meter

Ein Meter soll die derzeitige Geschwindigkeit, ähnlich einem Tachometer im Cockpit eines Autos, anzeigen.

LED Control

Diverse Bausteine, die LEDs nachempfunden sind, können durch Signale ein- bzw. ausgeschaltet werden und so den Status dieser wiedergeben.

Control Panel

In einem Control Panel können Konsolenausgaben des CANalyzers angezeigt werden.

Start Stop Control

Mit diesem Baustein kann die Aufzeichnung der Kommunikation auf dem Bus gestartet oder gestoppt werden.

Die Programmiersprache CAN Access Programming Language (CAPL, auch Communication Access Programming Language) ist an C angelehnt und wird innerhalb des CANalyzers genutzt, um Abläufe zu automatisieren. Eine umfassende Beschreibung von CAPL findet sich in [Vector CANtech, 2004].
Dem Projekt muss ein Programmknoten hinzugefügt werden, danach kann ein Programm im CAPL Browser editiert werden, zu sehen in Abbildung 3.14.

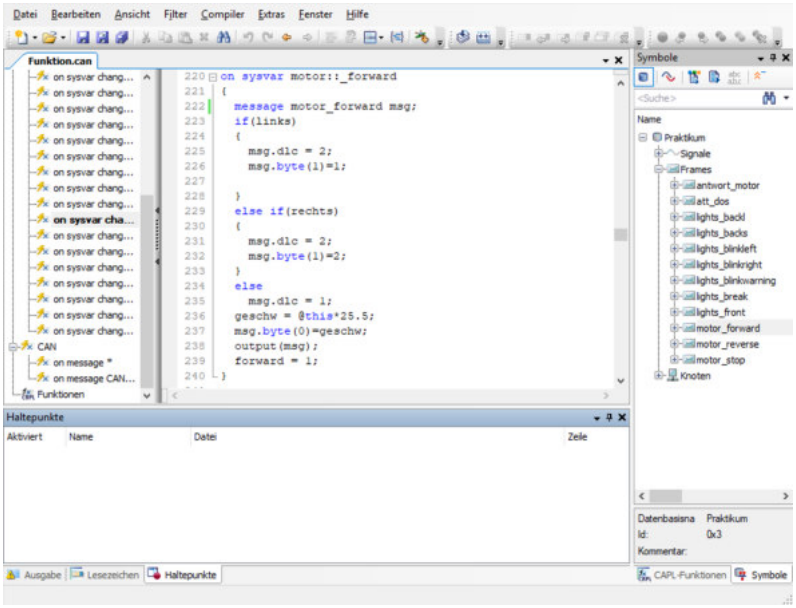

Abbildung 3.14: Der CAPL Browser von Vector mit Beispielprogramm. Quelle: Bildschirmaufnahme

Im Allgemeinen besteht ein CAPL-Programm aus drei Abschnitten: Inkludierte Bibliotheken, Definitionen globaler Variablen und die Funktionen. Funktionen werden durch Ereignisse (Events) aufgerufen und man unterscheidet zwischen Ereignissen, die durch Systemvariablen, durch bestimmte Nachrichten auf dem Bus oder durch den Ablauf von Timern ausgelöst werden.
Um die Elemente des Panels mit einer Funktionalität zu versehen, müssen diese Systemvariablen zugeordnet sein. Mit der Systemvariable lässt sich eine Funktion aufrufen, die zum Beispiel das Datenfeld einer CAN-Botschaft ändert und diese dann auf den Bus sendet. Die Werte, die in dieser Botschaft übergeben werden sollen, lassen sich durch die Elemente festlegen. Bei der Track Bar im Tab für die Vorwärtsbewegung können Werte stufenweise angepasst werden. Jede Änderung löst dabei einen erneuten Aufruf der Funktion aus, die die passende CAN-Botschaft versendet. Da hier nur zehn Stufen vorhanden sind, muss der gewünschte Wert noch mit 25 multipliziert werden, um einen sinnvollen Wert für die PWM zu wählen. Diese Funktion kann der Abbildung 3.14 entnommen werden.
Beim Betätigen einer Check Box wird das jeweilige Telegramm mit dem Wert für „ein“ oder „aus“, je nachdem, ob sich ein Häkchen in der Box befindet oder nicht, versandt.
Mit dem CAPL Browser können auch Timer gesetzt werden, die, nach dem Ablauf der vordefinierten Zeit, ein Event generieren. Dadurch können Telegramme zum Beispiel periodisch versendet werden.

3.5.3 WirelessCAN

Für die drahtlose Kommunikation zwischen einem Computer, auf dem der CANalyzer installiert ist, und dem Demonstrator wurden dem Versuchsaufbau zwei WLAN-fähige Module hinzugefügt. Der CAN-Bus wird dadurch streckenweise, aber nicht vollständig ersetzt. Als Übertragungsprotokoll wird in dieser Ausarbeitung das Übertragungssteuerungsprotokoll (Transmission Control Protocol (TCP)) verwendet. TCP ermöglicht eine Punkt-zu-Punkt-Verbindung zwischen den Teilnehmern. Es bedarf eines TCP-Servers, der auf Anfragen eines TCP-Clients antwortet. [Mandl et al., 2008, Kap. 5.2]
Im Wesentlichen werden für die Erstellung eines CAN-Frames nur drei wichtige Informationen benötigt:

Identifier
: Durch den Identifier werden Nachrichten priorisiert, jede Funktion des Demonstrators erhält genau eine ID.

DLC
: Im Feld DLC wird angegeben, wie viele Nutzbytes übertragen werden. In dieser Arbeit ist diese Information mit der Anzahl der zu übermittelnden Signale pro Nachricht gleichzusetzen.

Daten
: Ein Nutzbyte entspricht immer genau einem Signal.

Sind diese drei Informationen inhaltlich identisch, ist auch jedes damit erstellte CAN-Telegramm gleich. Das Gateway A muss eine eintreffende Nachricht interpretieren und diese drei Bestandteile extrahieren. Sie werden in einem TCP-Paket eingebettet und per WLAN übertragen. Das empfangende Gateway B muss in der Lage sein, das TCP-Paket zu interpretieren und die enthaltenen Daten in ein CAN-Telegramm umzustrukturieren. Dieser Ablauf ist in Abbildung 3.15 skizziert. Die erstellten CAN-Botschaften dieser Arbeit finden sich in Tabelle 3.5.

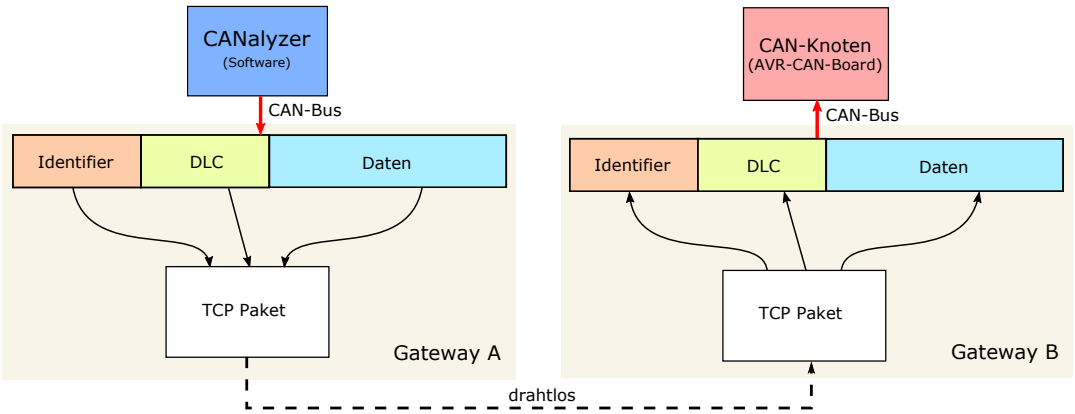

Abbildung 3.15: Schematische Darstellung der Kommunikation zwischen den Gateways. Quelle: eigene Arbeit

Das Gateway im Demonstrator (Gateway B) soll als Server dienen und das WLAN als sogenannter Access Point (AP) zur Verfügung stellen. Nachdem diese Dienste gestartet sind, kann sich der Client (Gateway A) mit dem WLAN verbinden und sich beim Server anmelden. Wird eine Nachricht vom CANalyzer versendet, leitet das CAN-Interface sie an den CAN-Transceiver des Boards weiter. Der ESP32 registriert die eintreffende Nachricht und generiert einen Interrupt. Die Interrupt Service Routine wertet das CAN-Frame aus und alle Informationen werden in einem globalen Objekt gespeichert und die Nachricht in einer Warteschlange abgelegt. Befindet sich eine Nachricht in der Warteschlange, werden die Inhalte daraus extrahiert, in ein TCP-Paket eingebettet und danach zum Server gesendet. Sobald der Server ein TCP-Paket des Clients erhält, wird die Nachricht geprüft und, sollte es sich um Informationen eines legitimen CAN-Telegrams handeln, in ein CAN-Frame umorganisiert. Dieses Frame wird auf den Bus zum AVR-CAN-Board übertragen. Analog kann der Server CAN-Botschaften vom AVR-CAN-Board erhalten, diese für das Versenden über TCP vorbereiten und an den Client übertragen, welcher die Inhalte für den CANalyzer zur Verfügung stellt. Der Ablauf des Programms ist in Abbildung 3.16 dargestellt.

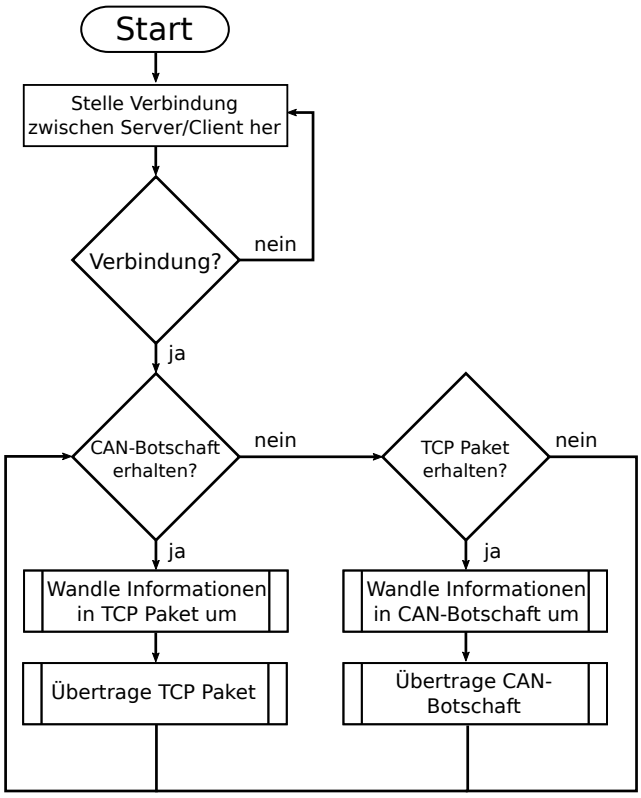

Abbildung 3.16: Skizze über den Ablauf der Kommunikation mit den Gateways. Quelle: eigene Arbeit

3.5.4 Verarbeitung der Botschaften durch das AVR-CAN-Board

Trifft eine CAN-Botschaft beim AVR-CAN-Board ein, muss der integrierte Mikrocontroller AT90CAN128 die Inhalte interpretieren und gewünschte Aktionen durchführen. Für die wichtigsten Funktionen des Mikrocontrollers wurde bereits eine Bibliothek von Glietsch [2008] angefertigt. Wenn eine Nachricht vom CAN-Knoten empfangen wird, wird ein Interrupt generiert. In der ISR werden die ankommenden Daten gesichert, in einem globalen Objekt zur Verfügung gestellt und der Durchlauf der Routine in einer globalen Variable mit einer 1 quittiert. Im Hauptprogramm kann jetzt in einer Schleife geprüft werden, ob dieser Interrupt stattgefunden hat. Ein vollständiger Programmablauf befindet sich im Anhang (Abbildungen B.1 und B.2).

Motorsteuerung

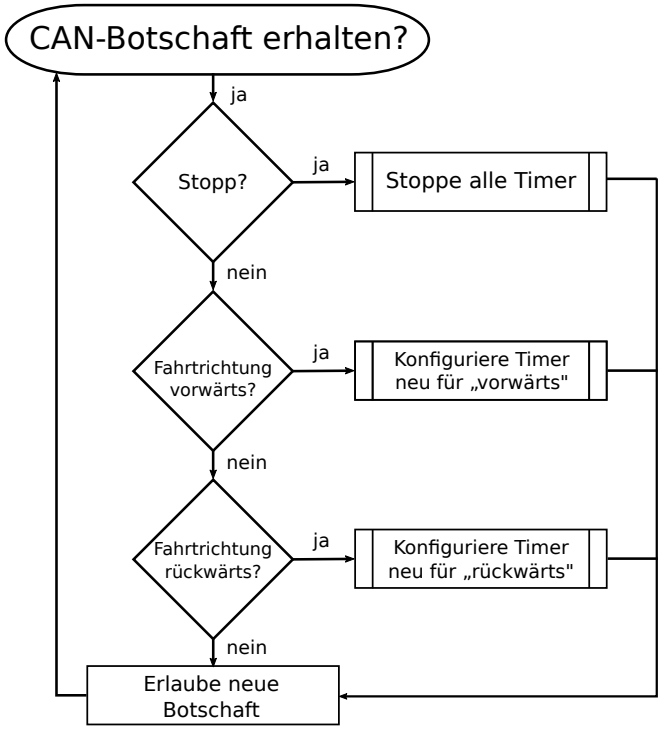

Abbildung 3.17: Skizze über den Ablauf des Programms für die Motoransteuerung. Quelle: eigene Arbeit

In der Abbildung 3.17 ist der allgemeine Ablauf für die Motoransteuerung dargestellt, im Folgenden wird näher auf die nötigen Konfigurationen eingegangen.
Damit die Geschwindigkeit der Motoren angepasst werden kann, muss eine Pulsweitenmodulation stattfinden (siehe Kapitel 2.7).

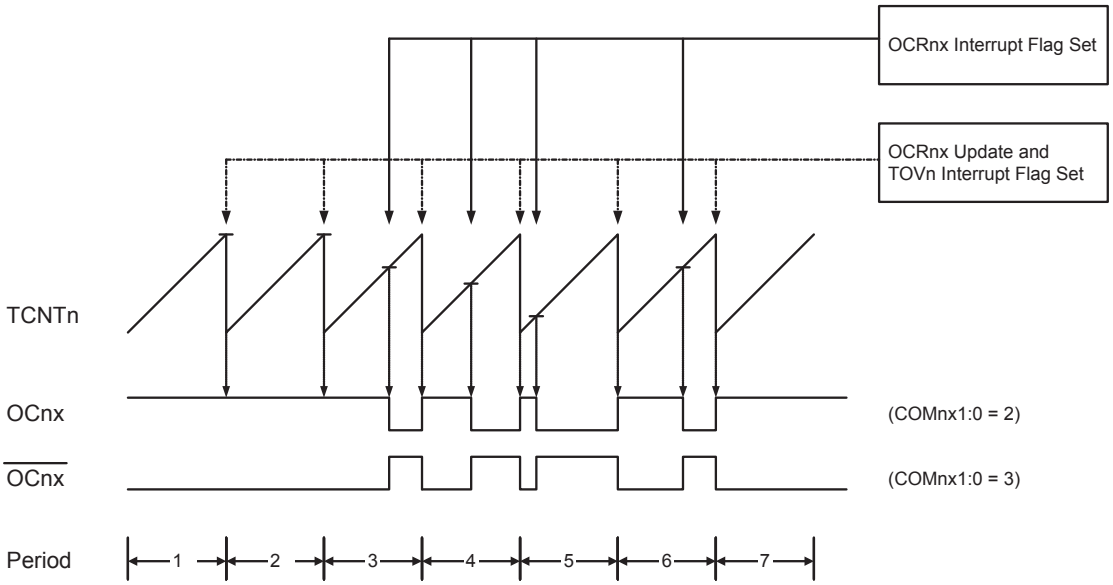

Abbildung 3.18: Timing Diagram des PWM-Modus. Quelle: [Atmel Corporation, 2008, S. 152]

Der Mikrocontroller AT90CAN128 verfügt über Timer, die PWM beherrschen. In der Abbildung 3.18 ist die Funktion dieser Timer in Form eines Timing- Diagramms skizziert. Für diese Arbeit sind die Perioden drei bis sieben ausschlaggebend. Der Timer zählt bis zu einem bestimmten Wert. Ist dieser Wert erreicht, wird ein Pin auf Low (0) geschaltet. Es wird jedoch weiterhin gezählt, bis ein Übertrag stattfinden müsste. In diesem Moment wird der Pin wieder auf High (1) gesetzt und der Timer beginnt von vorn zu zählen. Durch unterschiedliche Vergleichswerte wird die Pulsweite reguliert.
Der Takt, mit dem der Timer das Zählregister inkrementiert, lässt sich anpassen. Dazu wird folgende Formel verwendet:

$$\frac{\text{Frequenz der CPU}}{\text{Vorteiler} \cdot \text{maximaler Wert des Zählregister}} = \text{Zählfrequenz des Timers}$$

$$\frac{16.000.000 \text{ Hz}}{64 \cdot 2^8} = 976,6 \text{ Hz}$$

Aus der Wahrheitstabelle 3.3 aus Kapitel 3.4.1 geht hervor, dass die Motoren stoppen, sobald IN1 und IN2 des EVAL6207N das selbe Spannungspotential haben. Aus diesem Grund werden $IN1_A$ und $IN1_B$ mit den Pins eines Timers verbunden. Die PWM kann so die Geschwindigkeit der beiden Motoren regulieren, während $IN2_A$ und $IN2_B$ die Fahrtrichtung festlegen.

Beleuchtung

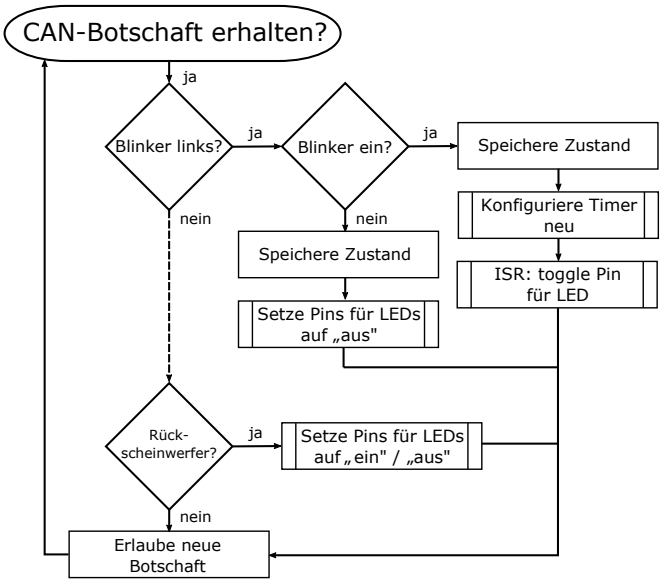

Abbildung 3.19: Skizze über den Ablauf des Programms für die Steuerung der LEDs. Hier nur zwei Beispiele (Blinker links und Rückscheinwerfer). Quelle: eigene Arbeit

In der Abbildung 3.19 ist der allgemeine Ablauf für die Steuerung der LEDs gezeigt. Die in der Abbildung verwendeten Identifier lassen sich der Tabelle 3.5 entnehmen.
Die Blinker des Demonstrators sollen durch den Erhalt eines CAN-Telegramms ein- und ausgeschaltet werden können. Sobald eine passende Botschaft eintrifft, soll ein Timer einen Interrupt generieren, der die Pins, an denen die Blinker angeschlossen sind, anspricht.
Der Timer/Counter 3 des AT90CAN128 ist ein 16-Bit Timer/Counter. Um mit diesem Timer/Counter die Leuchtdioden (LEDs) mit einer Frequenz von 2 Hz blinken zu lassen, muss dieser immer bis zu einem gewissen Wert zählen. Das entspricht dem Clear Timer on Compare Match (CTC)-Modus. In einem 16-Bit breiten Register kann maximal bis $2^{16} - 1 = 65.535$ gezählt werden. Um zu berechnen, welcher Wert sich im Vergleichsregister befinden muss, wird folgende Formel verwendet:

$$\frac{\text{Frequenz der CPU}}{(\text{Vorteiler} \cdot \text{Frequenz der Blinker})} \leq 65.535$$

$$\frac{16.000.000 \text{ Hz}}{(64 \cdot 4 \text{ Hz})} = 62.500$$

Der Timer zählt nun mit einer Frequenz von 250.000 Hz von 0 bis 62.500 und in der ISR werden die Pins für die Blinker auf „ein“ oder „aus“ geschaltet. Dieser Vorgang sollte nun viermal pro Sekunde stattfinden, die Blinker leuchten deshalb zweimal pro Sekunde auf.

3.6 Angriffe auf den Demonstrator

In dieser Arbeit lassen sich ausschließlich CAN-Nachrichten über den CANalyzer auf den Bus bringen. Um Angriffe wie im Kapitel 2.8 beschrieben auf den CAN-Bus auszuüben, können die bereits erstellten Nachrichten (siehe Tabelle 3.5) genutzt werden. Folgende Angriffsszenarien sind für den Demonstrator umsetzbar:

abrupte Änderung der Geschwindigkeit
: Setzt ein Angreifer die Geschwindigkeit eines Fahrzeuges auf ein Maximum (zum Beispiel durch volle Beschleunigung) oder Minimum (zum Beispiel durch das Blockieren der Räder) ergeben sich für die Insassen und alle weiteren Verkehrsteilnehmer erhebliche Gefahren. Ein solcher Angriff kann zu schweren Unfällen mit Personenschaden führen.

schleichende Änderung der Geschwindigkeit
: Wird die Geschwindigkeit eines Fahrzeuges schleichend manipuliert, wird der Angriff zu spät durch den Fahrer registriert. Außerdem dauert der Angriff länger an, der Fahrer kann in dieser Zeit nicht eingreifen. Dadurch wird er abgelenkt und auch dieses Szenario kann zu Unfällen führen.

periodisches Versenden hoch priorisierter Nachrichten
: Wird der CAN-Bus mit hoch priorisierten Nachrichten geflutet, werden wichtige Informationen nicht mehr an die jeweiligen Steuergeräte übermittelt. Das kann zum kompletten Ausfall mehrerer oder sogar aller Funktionen eines Fahrzeuges führen. Auch hier kann es zum Unfall mit Personenschaden kommen.

Im Folgenden wird die Umsetzung für diese Arbeit beschrieben.

3.6.1 Abrupte Änderung der Geschwindigkeit

Um eine solche Attacke durchführen zu können, wird eine CAN-Nachricht für die Fahrtrichtung verwendet. Im Datenfeld wird ein maximaler (0xFF) oder minimaler (0x00) Wert übertragen.
In einem realen Angriffsszenario auf ein Automobil müssten die Nachrichten zunächst ermittelt werden, die für die Beschleunigung zuständig sind und welchen Inhalt diese übertragen. Dieser Prozess wird von Miller und Valasek [2013] beschrieben.
Da bei dem Demonstrator bekannt ist, welche Identifier für die Nachrichten zum Ändern der Geschwindigkeit zuständig sind (Nachrichten 0x03 und 0x04), entfällt dieser Vorgang.

3.6.2 Schleichende Änderung der Geschwindigkeit

Um die Geschwindigkeit schleichend zu ändern, werden Funktionen benötigt, die periodisch CAN-Botschaften in sehr kurzen Zeitabständen generieren. Durch das kleine Zeitfenster zwischen den Botschaften wird der Bus nicht überladen und der Nutzer des Demonstrators kann dennoch nicht manuell eingreifen. Um dieses Szenario zu realisieren, werden Timer mit einer Dauer von 80 ms gesetzt, die die Geschwindigkeit vom aktuellen Wert aus inkrementieren oder dekrementieren. Es wird geprüft, in welche Richtung sich der Demonstrator im Moment des Angriffs bewegt und falls keine Fahrtrichtung festgelegt wurde, werden die Nachrichten für die Vorwärtsbewegung gewählt.

3.6.3 Periodisches Versenden von hoch priorisierten Nachrichten

Im CAN-Protokoll werden Botschaften ereignisorientiert auf den Bus gesendet und um Kollisionen zu vermeiden, werden Nachrichten mit niedrigen Identifiern priorisiert (siehe Kapitel 2.4.4). Ist der Bus mit einer Nachricht belegt, müssen alle sendenden CAN-Knoten warten, bis vorhergehende Nachrichten abgearbeitet sind. Um eine simple DoS-Attacke auszuführen, wird der Bus durch periodisches Senden von hoch priorisierten Nachrichten (Identifier 0x00) vollständig blockiert. Sobald der Schalter für diesen Angriff betätigt wird, wird in einer Endlosschleife diese Nachricht ohne Inhalt verschickt. Der CANalyzer stürzt ab und der Demonstartor lässt sich selbst nach einem Neustart der Anwendung nicht steuern. Erst wenn die Stromversorgung vom Demonstrator unterbrochen und das CAN-Interface getrennt wurde, lässt sich die reguläre Funktion wiederaufnehmen.

3.7 Entwurf eines Intrusion Detection Systems

Um einen Angriff auf ein System detektieren und melden zu können, wird häufig auf den Einsatz von IDS gesetzt. In Fahrzeugen besteht aktuell jedoch keine Möglichkeit, einen Angriff auf den CAN-Bus digital-forensisch nachvollziehen und nachweisen zu können. Bresch und Salman entwickelten ein IDS, das im CANalyzer integriert und deshalb auch für den Demonstrator getestet werden kann. Da die Botschaften für die Beschleunigung des Demonstrators ähnlich zu denen im Versuch von Bresch und Salman sind, müssen die Funktionen des IDS nur minimal angepasst werden. Bei den Angriffen werden Nachrichten gesendet, in deren Payload entweder zu hohe Änderungen der Geschwindigkeit zum Ist-Zustand übermittelt werden oder deren Frequenz von der im Normalbetrieb abweicht. Genau diese beiden Szenarien deckt das IDS ab. Der Programmablauf ist in Abbildung 3.20 skizziert.

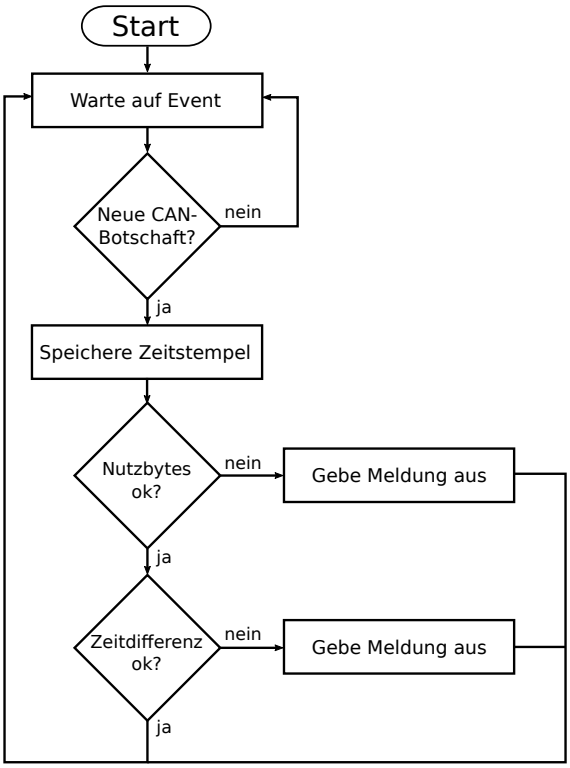

Abbildung 3.20: Skizze über den Programmablauf des IDS. Quelle: eigene Arbeit

Um zu wissen, wie hoch der Schwellenwert für die Nachrichtenfrequenz bei dieser Arbeit ist, werden Botschaften über das Panel aus Kapitel 3.5.2 versendet, die eine ungestörte Fahrt simulieren sollen. Dabei folgen die Botschaften nie in unter 300 ms aufeinander, was deshalb einen guten Schwellenwert für die Zeit zwischen aufeinanderfolgenden Nachrichten darstellt. Das IDS gibt Warnmeldungen in der Konsole des CANalyzers aus, sie lässt sich über das Control Panel auch im Dashboard anzeigen.

4 Ergebnis

Alle Hardware- und Softwarekomponenten wurden, wie im vorhergehenden Kapitel beschrieben, implementiert und in Betrieb genommen. Der final umgesetzte Versuchsaufbau ist in Abbildung 4.1 zu sehen. Im Folgenden werden die Ergebnisse anhand der zu Beginn gestellten Anforderungen erläutert. Das allgemeine Ziel war es, einen Demonstrator mit den gegebenen Hard- und Softwarekomponenten zu realisieren und forensische Analysen durchzuführen.

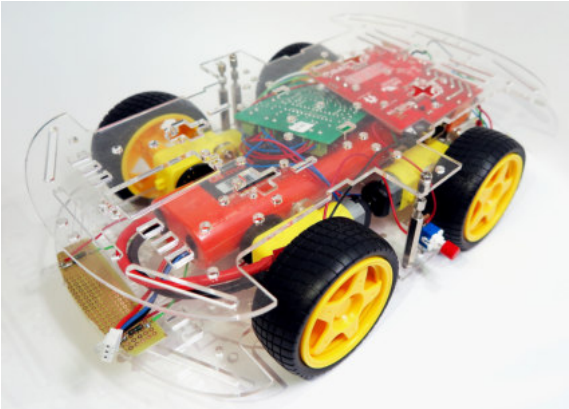

Abbildung 4.1: Finaler Aufbau des Demonstrators. Quelle: eigene Aufnahme

4.1 Steuerung durch CAN-Botschaften

Die Steuerung des Demonstrators soll über CAN-Botschaften erfolgen. Das AVR-CAN-Board empfängt dazu als CAN-Knoten die Nachrichten auf dem Bus. Erstellt wurden die CAN-Nachrichten mit Hilfe des CANalyzers. Dazu wurden alle verwendeten Botschaften in einer Datenbank abgelegt. Diese entspricht der Tabelle 3.5.
Aus den Differenzspannung zwischen CAN High und CAN Low lässt sich der Bitstrom der Nachricht rekonstruieren. In Kapitel 2.4.3 sind die Signalpegel erläutert. Ausschlaggebend für jedes CAN-Telegramm sind die Felder Identifier, DLC und Daten. Lassen sich diese Daten aus dem Bitstrom auslesen, lässt sich auf den Nutzen mit Hilfe der Datenbank schließen. Diese Anforderung an den Demonstrator ist vollständig erfüllt. Ein Oszillogramm einer CAN-Botschaft findet sich im Anhang (siehe Abbildung C.1).
Im Betrieb fällt jedoch auf, dass ein Einblick in den aktuellen Status des AVR-CAN-Boards nur über die Programmierschnittstelle möglich ist. Hier müssen weitere Anpassungen vorgenommen werden (siehe Kapitel 4.6).

4.2 Realisierung grundlegender Funktionen eines Fahrzeuges

Eine grundlegende Funktion eines Automobils ist das Fahren an sich. Die Motoren des Robot Car Kits von Joy-It können erfolgreich über das Evaluationsboard EVAL6207N und den CAN-Knoten AVR-CAN durch CAN-Botschaften gesteuert werden. Um die Geschwindigkeit der Motoren zu regulieren, wurde eine PWM programmiert. Die Signalpegel lassen sich aus dem Oszillogramm aus Abbildung 4.2 entnehmen. Es können sowohl die Richtungen vorwärts und rückwärts als auch links und rechts gewählt werden. Für die Fahrtrichtungen links oder rechts wird die PWM für die Motoren der jeweiligen Seite gedrosselt.

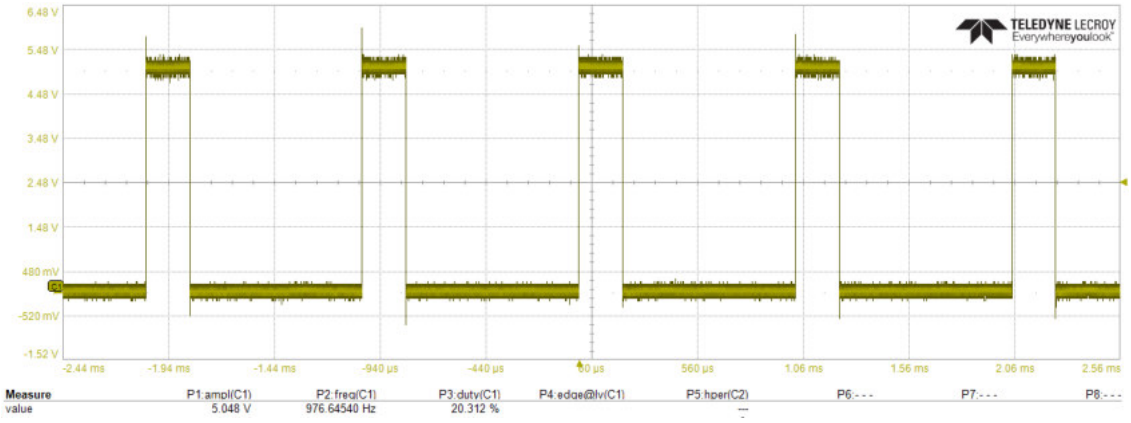

Abbildung 4.2: Oszillogramm der Pulsweitenmodulation. Quelle: Bildschirmaufnahme

Des Weiteren sollte der Demonstrator über eine, für Fahrzeuge typische, Beleuchtung verfügen. Neben diversen Leuchten und Scheinwerfern sind auch die Funktionen eines Blinkers nachempfunden. Dafür wurde ein Timer programmiert, der die LEDs in gesetzten Zeitintervallen ein- oder ausschaltet. Auch diese Signale können durch ein Oszillogramm dargestellt werden (Abbildung 4.3).

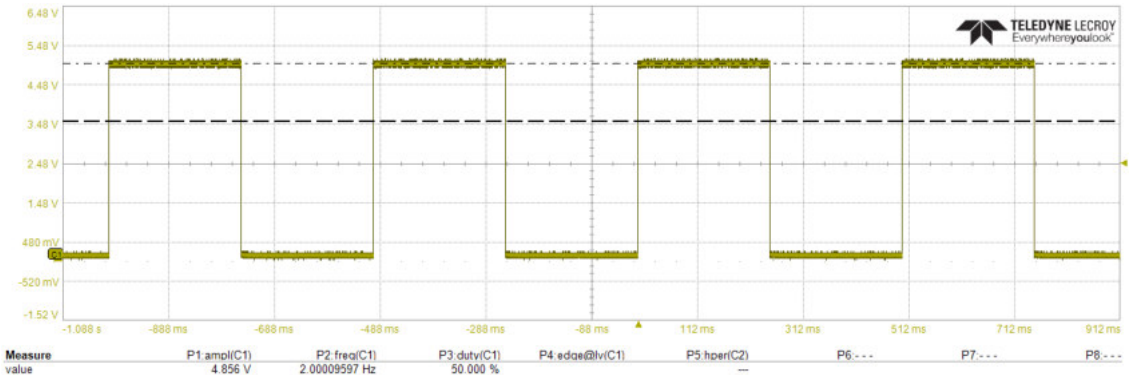

Abbildung 4.3: Oszillogramm der Funktionsweise der Blinker. Quelle: Bildschirmaufnahme

4.3 Mobilität

Die mobile Stromversorgung des Demonstrators ist durch den Einsatz des Akkumulators gegeben. Die Auswahl einer geeigneten Ausführung mit ausreichender Ausgangsspannung und Kapazität erfolgte im Kapitel 3.3. Der Demonstrator lässt sich dadurch ca. drei Stunden betreiben.
Durch eine Messung konnte ermittelt werden, dass der reale Stromverbrauch unter Volllast ca. 760 mA beträgt. Die Differenz zum theoretischen Wert von 1050 mA ergibt sich daraus, dass für die Berechnung immer der maximale, im Datenblatt angegebene, Verbrauch des jeweiligen Bauteils verwendet wurde. Real wird dieser Wert jedoch selten erreicht. Daraus ergibt sich eine Betriebsdauer von ca. vier Stunden.
Bei der Inbetriebnahme verlief die Kommunikation über CAN-Botschaften mit den WLAN-fähigen Gateways wie vorgesehen.
Innerhalb eines Raumes kann der Demonstrator problemlos über die drahtlose Kommunikation gesteuert werden. Im Freien beträgt der Aktionsradius ca. 100 m.

4.4 Eintrittspforten für Angreifer

Durch die programmierten Angriffsszenarien konnte der Demonstrator erfolgreich kompromittiert werden. Sowohl die schleichende als auch die abrupte Änderung der Geschwindigkeit führte zum gewünschten Ergebnis. Die DoS-Attacke führt zum Absturz des CANalyzers, eine Steuerung des Demonstrators durch den Nutzer ist ab diesem Zeitpunkt nicht mehr möglich. Auch nach einem Neustart der Software werden keine CAN-Botschaften übermittelt. Erst nach der Unterbrechung der Stromversorgung und einer erneuten Verbindung zwischen Computer und CAN-Interface ist die normale Funktionalität wiederhergestellt.
Auch die drahtlose Verbindung über WLAN kann als eine weitere Eintrittspforte für Angreifer angesehen werden. Ein Test für mögliche Angriffsszenarien lag jedoch nicht im Fokus dieser Arbeit. Bei der derzeitigen Programmierung lässt sich außerdem das Passwort für den Access Point im Klartext im Programmcode finden.
Eine Anbindung des CAN-Busses an das Internet über ein Gateway spiegelt den aktuellen Stand der Technik in der Automobilbranche wider (siehe Kapitel 2.8.3).

4.5 Aufzeichnung von Attacken

Das IDS registriert die abrupte und schleichende Änderung der Geschwindigkeit wie gewünscht. Die Meldungen werden in der Konsole ausgegeben und lassen sich selbst im Panel anzeigen (siehe Abbildung 4.4). Wird eine DoS-Attacke auf das System ausgeübt, stürzt der CANalyzer und damit auch das IDS ab. Die Attacke kann deshalb in der aktuellen Umsetzung nicht registriert werden. Die Zielstellung ist für zwei von drei Szenarien erfüllt.

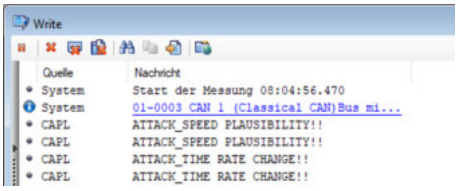

Abbildung 4.4: Ausgaben des IDS. Quelle: Bildschirmaufnahme

4.6 Weitere Anpassungen

Bei der Inbetriebnahme ergaben sich zwei Probleme, die sich bei den Vorüberlegungen und der Realisierung nicht stellten:

- keine Statusinformationen des Demonstrators im mobilen Einsatz
- keine gesicherte Verbindung zwischen den Gateways

Das Debugging des Demonstrators ist bis zu diesem Zeitpunkt nur über die Programmierschnittstelle des AVR-CAN-Boards möglich. Um Informationen über den Betriebsstatus des Demonstrators zu erhalten, wird eine neue CAN-Botschaft erstellt. Diese wird vom CAN-Knoten an den CANalyzer übermittelt. In der Nachricht sind Informationen über die aktuelle Fahrtrichtung, der Vergleichswert für die PWM und dem Status der LEDs angegeben. Diese Nachricht wird als Antwort auf jede neue Eingabe des Benutzers versendet. Die übermittelten Informationen können auch für Komponenten der grafischen Oberfläche genutzt werden. Sie steuern den Status der LEDs des Panels.
Der Verbindungsaufbau zwischen den Gateways ist bisher nicht gesichert. Auch hier wird eine neue CAN-Botschaft dafür genutzt, den aktuellen Status anzuzeigen. Besteht eine Verbindung zwischen den beiden Gateways, übermittelt das Modul, welches mit dem CAN-Interface verbunden ist, eine Nachricht, in dessen Payload eine 1 enthalten ist. Besteht sie nicht oder bricht die Verbindung ab, wird das Signal 0 übertragen. In Tabelle 4.1 sind alle neu erstellten CAN-Botschaften für die Datenbank enthalten.
Um den Abbruch der Verbindung in einem angemessenen Zeitraum festzustellen, wird ein Timer in den Programmablauf beider Module integriert. Zwischen ihnen werden zyklisch sogenannte Watchdog-Nachrichten übermittelt. Hat ein Teilnehmer nach dem

Ablauf dieses Timers keine Watchdog-Nachricht erhalten, wird der Verbindungsaufbau wiederholt. Das ESP32-EVB, welches mit den CAN-Interface verbunden ist, übermittelt die CAN-Botschaft über den Status an den CANalyzer. Der aktualisierte Programmablauf ist in Abbildung 4.5 skizziert.

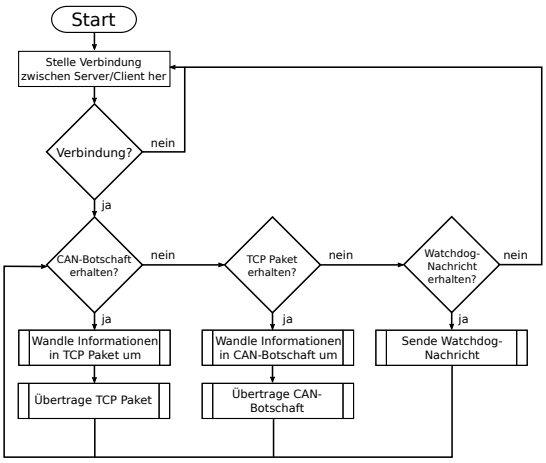

Abbildung 4.5: Aktualisierte Skizze des Ablaufs der Kommunikation über das Gateway. Quelle: eigene Arbeit

Das System befindet sich jetzt in einem sicheren Zustand. Mit einer weiteren neuen CAN-Botschaft mit der ID 0x200 (siehe Tabelle 4.1) kann das Zeitintervall für den Timer manuell festgelegt werden.

Tabelle 4.1: Hinzugefügte CAN-Botschaften für den Versuchsaufbau

Ziel	ID	Botschaft	Signale	Wertigkeit
CANalyzer	0x10	aktueller Status	Fahrtrichtung (vorwärts/rückwärts)	0, 3, 4
			Fahrtrichtung (links/rechts)	0-2
			Geschwindigkeit	0-255
			Frontscheinwerfer	0-1
			Bremsleuchten	0-1
			Rückscheinwerfer	0-1
			Blinker links	0-1
			Blinker rechts	0-1
	0x100	Verbindungsaufbau	Status	0-1
ESP32-EVB	0x200	Änderung Watchdog	Timerintervall (oberes Byte)	0-255
			Timerintervall (unteres Byte)	0-255

5 Diskussion

In dieser Arbeit wurde ein Demonstrator für den CAN-Bus erstellt. Er soll hauptsächlich in der Forschung und Lehre eingesetzt werden. Aus der Zielstellung ergaben sich mehrere Anforderungen, die im Verlauf dieser Arbeit erfüllt wurden.

Steuerung über CAN-Botschaften
Im Programm CANalyzer werden CAN-Botschaften erstellt und in einer Datenbank verwaltet. Durch ein Panel kann die Steuerung des Demonstrators durch den Nutzer erfolgen. Der CANalyzer wandelt alle Informationen in eine vollständige CAN-Nachricht um, welche über den CAN-Bus an den Demonstrator gesendet wird. Der CAN-Knoten (AVR-CAN-Board) ist so programmiert, dass er eintreffende Nachrichten verarbeiten und gewünschte Funktionen aufrufen kann.

Realisierung grundlegender Funktionen eines Fahrzeuges
Der Demonstrator soll ein abstraktes Automobil darstellen. Es können die Fahrtrichtung, die Geschwindigkeit und die Beleuchtung durch den Nutzer gesteuert werden. Scheinwerfer und Blinker eines Fahrzeuges sind durch Leuchtdioden realisiert.

Mobilität
Der Demonstrator ist innerhalb eines Raumes frei beweglich. Die Stromversorgung erfolgt über einen ausreichend starken Akkumulator und der Demonstrator kann ca. vier Stunden unter voller Auslastung genutzt werden. Durch den Einsatz geeigneter Gateways werden CAN-Botschaften innerhalb eines drahtlosen Übertragungsprotokolls übertragen. Der Demonstrator stoppt automatisch bei Verbindungsverlust zwischen den Gateways und entfernt sich deshalb nicht zu weit vom Nutzer. So ist er vor Schaden durch Dritte oder den Verbindungsabbruch weitestgehend geschützt.

Eintrittspforten für Angreifer
Es konnte gezeigt werden, dass der Demonstrator erfolgreich angegriffen werden kann. Dazu fanden Szenarien Anwendung, die bereits an echten Fahrzeugen erprobt wurden. Der Einbau der WLAN-fähigen Gateways spiegelt darüber hinaus die Realität wider, in der der CAN-Bus über das Infotainmentsystem für die Außenwelt angreifbar ist.

Aufzeichnung von Attacken
Das von Bresch und Salman übernommene IDS kann mit nur wenigen Anpassungen für den Demonstrator genutzt werden. Sowohl die abrupte als auch die schleichende Änderung der Geschwindigkeit werden erkannt. Da das IDS nur als Funktion im CANalyzer programmiert und nicht innerhalb der Hardware realisiert ist, stürzt es bei einer DoS-Attacke mit dem CANalyzer ab.

Das CAN ist dafür bekannt, zuverlässig Nachrichten zu übertragen. Durch Schutzmechanismen werden Informationen nach Relevanz übertragen und auf ihre Vollständigkeit geprüft. Die Kommunikation gilt damit grundlegend als sicher. Jedoch besteht im Kontext

der IT-Sicherheit kein Schutz vor unautorisierten Nachrichten durch ein, mit Schadsoftware verseuchtes oder neu in das Netzwerk eingebrachtes, Steuergerät. Kann das CAN eines Fahrzeuges kompromittiert werden, lässt es sich so manipulieren, dass eine Gefahr für alle Verkehrsteilnehmer besteht. Hat der Angreifer böswillige Absichten, kann das Fahrzeug genutzt werden, um Unfälle zu provozieren oder zu verursachen. Durch die Verbindung über Gateways zum Mobilfunknetz sind diese Angriffe sogar über weite Distanzen realisierbar.
Immer aktueller wird das autonome Fahren. Der Fokus liegt derzeit noch dabei, auf Hindernisse, Verkehrsschilder und -situationen angemessen zu reagieren. Diese Manöver sollen in Zukunft vollkommen selbstständig durch Steuergeräte ausgeführt werden.
Sollten die Bussysteme eines Fahrzeuges bis dahin nicht ausreichend vor Attacken durch Cyberkriminelle geschützt sein, könnten Angreifer die vollständige Kontrolle über ein Fahrzeug erhalten. Im Falle eines Unfalls kann darüber hinaus ein solcher Angriff nicht digital-forensisch nachgewiesen werden, da bisher kein IDS in Automobilen vorgesehen ist. Es ist deshalb von großer Wichtigkeit, nicht nur die Fahrsysteme, sondern auch die Busse abzusichern und für den Fall eines Angriffs Logdateien zu erstellen.
Der Demonstrator soll dazu dienen, die Möglichkeiten der forensischen Auswertung des CAN-Busses weiter zu erforschen und die Technologie in diesem Feld weiterzuentwickeln. Er erfüllt alle Anforderungen, die sich in dieser Arbeit an ihn gestellt haben, ist kostengünstig und mit wenig Aufwand reproduzierbar. Das Prinzip des CAN-Busses wird durch ihn für die Lehre anschaulich. Der Demonstrator kann deshalb eine Grundlage für weitere Forschungsprojekte sein.

5.1 Alternativen zum aktuellen Versuchsaufbau

Nach der vollständigen Implementierung aller Komponenten und der Vorstellung der Ergebnisse in Kapitel 4, ergeben sich Überlegungen für einen alternativen Versuchsaufbau.

5.1.1 Entwurf des Intrusion Detection Systems

Aktuell ist das IDS dieser Arbeit als Funktion des CANalyzers programmiert. Es werden zwar Meldungen über die Konsole ausgeben, aber nicht gespeichert.
Bresch und Salman schlagen vor, das IDS über ein Firmwareupdate nachträglich in ein bereits verbautes Steuergerät zu integrieren. [Bresch und Salman, 2017, S. 59] Dafür müsste das Programm in eine andere Programmiersprache überführt werden (zum Beispiel C). Außerdem muss das Steuergerät über genügend Speicherplatz verfügen, um anfallende Logdatein über einen gewissen Zeitraum ablegen zu können.
Nilsson und Larson erörtern in ihrer Arbeit ebenfalls die Frage, wie Ereignisse auf den Bussystemen eines Fahrzeuges für eine forensische Auswertung mitgeschnitten und ausgelesen werden können.

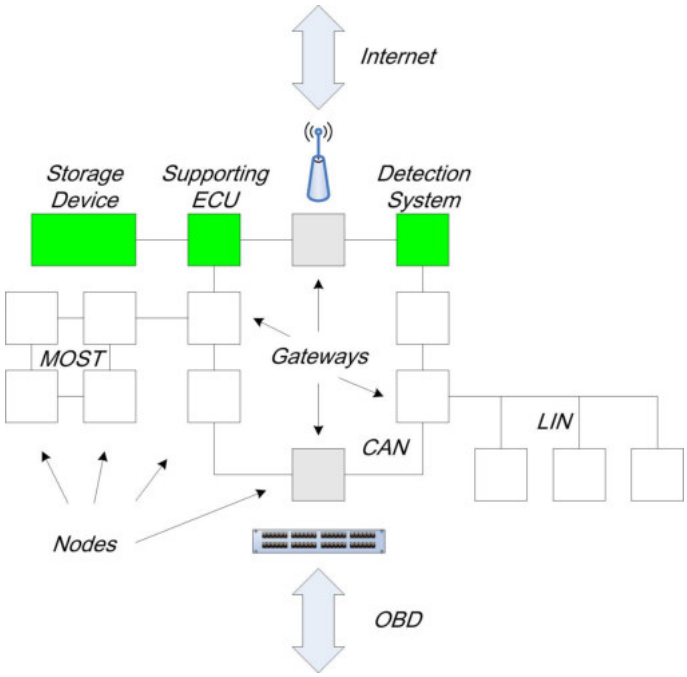

Abbildung 5.1: Vorschlag für ein Detektionssystem in einem Fahrzeug. Quelle: [Nilsson und Larson, 2008, S. 4]

In Abbildung 5.1 sind Steuergeräte in grün markiert, welche in das Netzwerk eingebaut sein müssten, um dieses Modell umzusetzen. Eine unterstützende ECU (Supporting ECU) zeichnet den Netzwerkverkehr vollständig auf, ein weiteres Steuergerät fungiert als IDS und auf ein Speichermedium (Storage Device) werden alle vom IDS produzierten Ausgaben abgelegt.
Das ESP32-EVB könnte im bisher vorgestellten Versuchsaufbau alle Funktionen des Systems von Nilsson und Larson übernehmen. Es verfügt mit einem Modul für eine Micro SD-Karte über ausreichenden Speicherplatz (heute gängig bis 256 GB) für Logdateien. Des Weiteren findet die gesamte CAN-Bus-Kommunikation über das Gateway statt. Meldungen müssten jedoch sowohl auf der Seite des Demonstrators als auch auf der des CANalyzers mitgeschnitten werden, da bei einer Unterbrechung der Verbindung trotzdem Angriffe von Außen stattfinden können. Außerdem stellt sich die Frage, ob ein Programm für das ESP32-EVB effizient genug ist, sodass keine CAN-Botschaften überschrieben, alle Ereignisse ausgewertet und Anomalien geloggt werden können.

5.1.2 Design einer Leiterplatte

Alle Hardwarekomponenten finden Platz im Robot Car Kit von Joy-It, was als Grundgerüst des Demonstrators genutzt wird. Alternativ hätte jedoch auch eine Leiterplatte, auf der alle Komponenten fest verbaut sind, designet werden können. Die Mikrocontroller AT90CAN128 und ESP32, die Vorwiderstände der LEDs und die LEDs selbst hätten so direkt miteinander verbunden werden können. Der CAN-Bus wäre trotzdem in Form von Leiterbahnen zwischen den Mikrocontrollern vorhanden. Ebenfalls muss ein CAN-

Transceiver auf der Leiterplatte verbaut sein. Lediglich die Motoransteuerung durch das EVAL6207N muss überdacht werden. Eventuell würden sich Komponenten dieses Boards auch für die Leiterplatte eignen.
Die Entwicklung und Herstellung einer Leiterplatte gestaltet sich aber als sehr zeit- und kostenaufwendig und lag nicht im Fokus dieser Arbeit.

5.1.3 Verwendete Software

Auf dem Markt werden diverse Softwarelösungen zur Anzeige und Manipulation des CAN-Busses angeboten. Der CANalyzer weist als vollwertiges Diagnosetool eine Vielzahl an Funktionen auf, die für diese Arbeit jedoch nicht unbedingt genutzt werden. Die Beschaffung der Software und des CAN-Interface ist mit hohen Kosten verbunden. Im Folgenden werden einige Alternativen vorgestellt.

PCAN-View
Für das schlichte Versenden und Erhalten von CAN-Botschaften stellt PEAK-System das Programm PCAN-View kostenlos zur Verfügung, zu sehen in Abbildung 5.2.

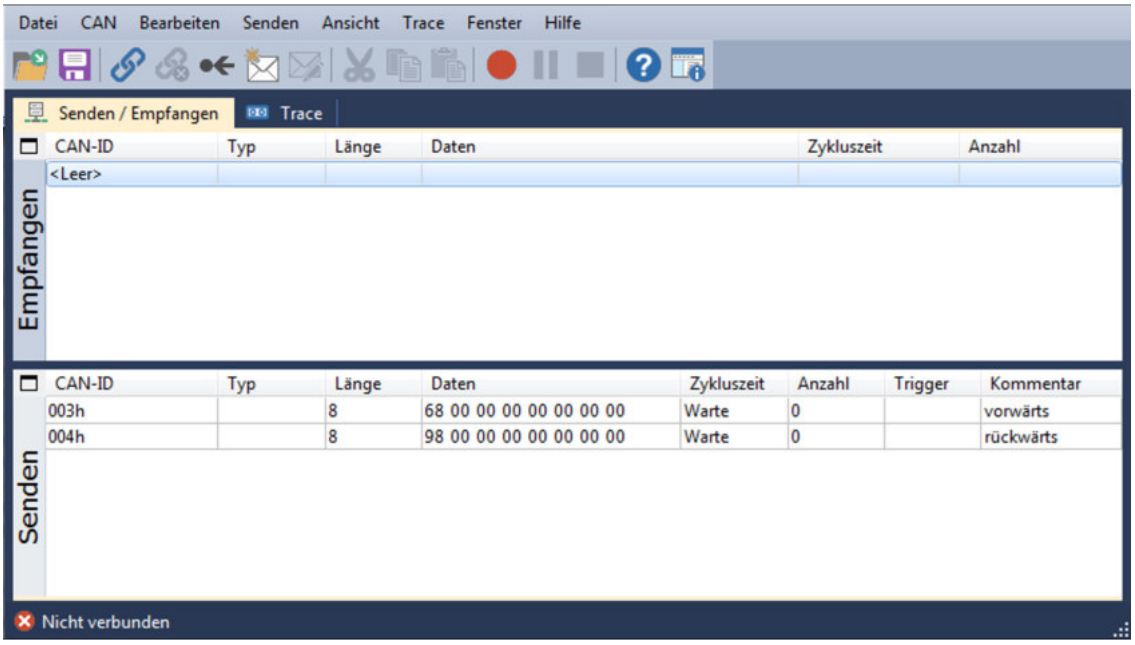

Abbildung 5.2: PCAN-View von PEAK-System Technik GmbH. Quelle: Bildschirmaufnahme

Für PCAN-View wird das CAN-Interface von PEAK-System benötigt, das bisher verwendete VN1610 von Vector wird nicht erkannt. Neue CAN-Botschaften können hier unkompliziert angelegt und im Fenster „Senden“ verwaltet werden. Eine Datenbank, in der Nachrichten erstellt und verwaltet werden können, bietet PCAN-View nicht. CAN-Nachrichten müssen manuell oder zyklisch versendet werden und generell lassen sich keine Funktionen oder grafische Oberflächen programmieren.

BUSMASTER

Die Firma ETAS GmbH entwickelte in Zusammenarbeit mit Bosch eine ebenfalls kostenlose Variante, um CAN-Botschaften aufzuzeichnen oder durch den Anwender auf den Bus zu senden. Im Gegensatz zu PCAN-View können die Nachrichten im BUSMASTER mit einer Datenbank organisiert werden, einzelne Signale überwacht und sogar CAPL-Funktionen für die Automatisierung der Abläufe importiert werden. Dabei wird neben dem CAN auch das LIN unterstützt. Dieses Programm erkannte das CAN-Interface von Vector und der Demonstrator konnte gesteuert werden, die Kommunikation zwischen Demonstrator und BUSMASTER ist in Abbildung 5.3 dargestellt. Eine Liste der unterstützten Hardware findet sich auf der Internetpräsenz des Herstellers. [ETAS GmbH, 2017] Der BUSMASTER kann die erstellten Datenbanken in verschiedene Formate konvertieren und sie so anderen Programmen zur Verfügung stellen, oder die gespeicherten Nachrichten aus, zum Beispiel, dem CANalyzer verwenden.

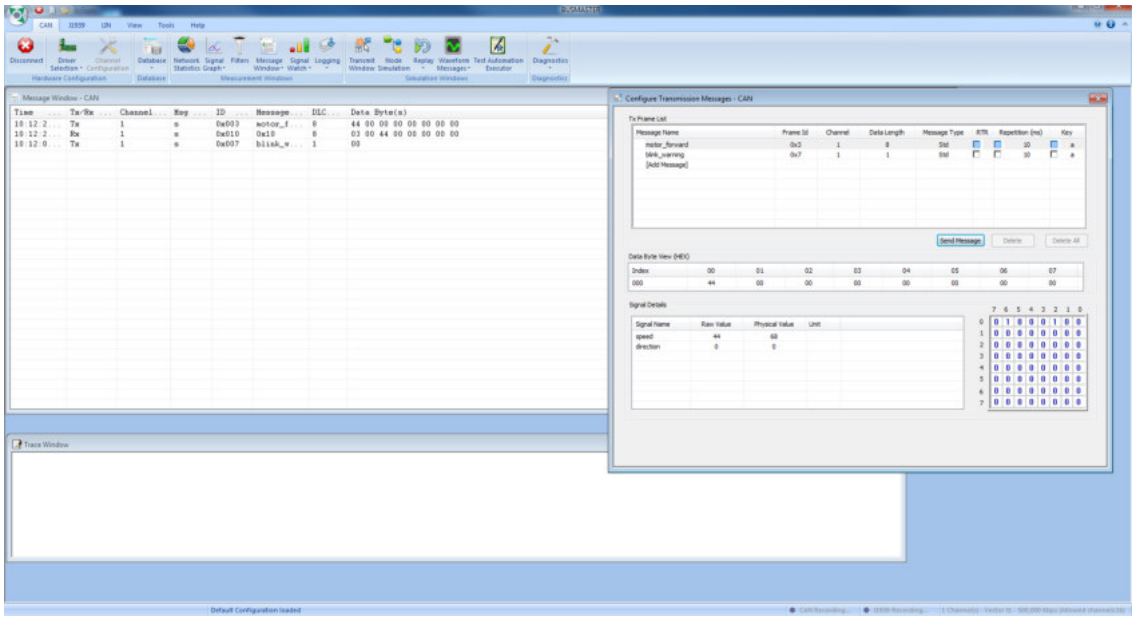

Abbildung 5.3: Aufzeichnen und Versenden von CAN-Botschaften mit dem BUSMASTER. Quelle: Bildschirmaufnahme

Die vorgefertigten Nachrichten manuell zu versenden gestaltet sich im Gegensatz zur Bedienung des, mit dem CANalyzer erstellten, Dashboards als sehr unflexibel.

CAN-Manipulator

An der Hochschule Mittweida wurde 2017 eine Software zur Manipulation des CAN-Busses erstellt und erfolgreich in Betrieb genommen. [Stebner, 2017] Mit Hilfe eines Single-Board-Computers (Red Pitaya) wurde die Verbindung zum CAN-Bus eines realen Fahrzeuges hergestellt. Der Computer sendet die physischen Signale einer Botschaft auf den CAN-Bus. Der CAN-Manipulator kann über eine Weboberfläche gestartet und so unabhängig von der Plattform und vom Standort eingesetzt werden. Durch die Software könnte sich der Demonstrator auch von einem Smartphone aus steuern lassen. Der CAN-Manipulator ist keine Analysesoftware, die Kommunikation über den Bus lässt sich

weder mitschneiden noch aufzeichnen. Eine weitere Entwicklung des Programms könnte mit Hilfe des Demonstrators geplant werden. Fraglich ist jedoch, wie bei PCAN-View oder dem BUSMASTER, ob eine Bedienung rein über das manuelle Versenden von CAN-Botschaften für den potentiellen Anwender intuitiv und schnell genug ist. Die Oberfläche ist in Abbildung 5.4 gezeigt.

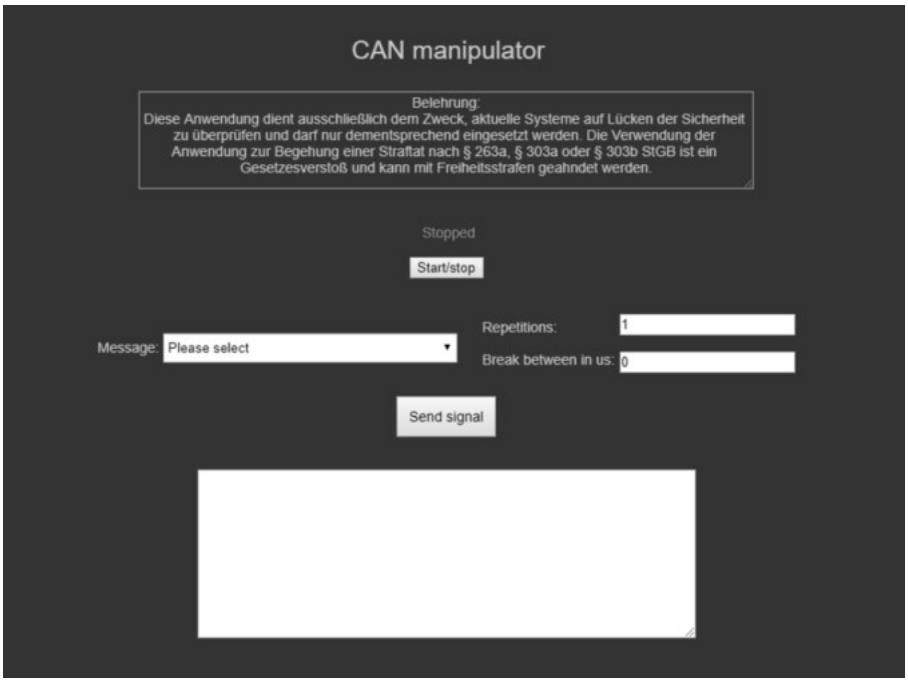

Abbildung 5.4: CAN-Manipulator. Quelle: [Stebner, 2017, S. 27]

5.1.4 Mobilität

Der Demonstrator ist durch die Vorüberlegungen und die Implementierung der Hardwarekomponenten kabellos verwendbar. Dadurch kann er mobil und ähnlich zu einem Modellauto eingesetzt werden und die Funktionen eines Automobils stark vereinfacht nachstellen.
Dennoch besteht weiterhin die Möglichkeit, den CANalyzer und CAN-Knoten direkt über das CAN-Interface zu verbinden. Für einfache Tests oder Vorführungen besteht keine Notwendigkeit für den Einbau WLAN-fähiger Gateways. Wird der CANalyzer auf einem Laptop installiert und liegt eine direkte Verbindung zum Demonstrator über das CAN-Interface vor, ist ein mobiler Einsatz prinzipiell möglich. Dazu muss die Stromversorgung der Hardware weiterhin über eine Batterie oder einen Akkumulator erfolgen. Verfügt der verwendete Computer darüber hinaus über einen Touchscreen, könnte das erstellte Panel instinktiver bedient werden.

drahtlose Verbindung

In dieser Arbeit werden CAN-Botschaften teilweise über eine drahtlose Verbindung übertragen. In Kapitel 3.3.1 wurde WLAN gewählt und sich damit gegen andere gängige Verbindungsarten, wie zum Beispiel Bluetooth, entschieden. Bluetooth wird in Automobilen bisher hauptsächlich für Infotainment-Systeme genutzt. Über das Smartphone lassen sich so Multimediaanwendungen bedienen. [Zimmermann und Schmidgall, 2014, S. 4]
Ein weiteres Anwendungsgebiet ist die drahtlose Übertragung von Informationen zwischen dem OBD-Port eines Fahrzeuges und einem Diagnosewerkzeug (zum Beispiel CANalyzer). [Zimmermann und Schmidgall, 2014, S. 459 f.]
In *Experimental Validation of CAN to Bluetooth Gateway for In-Vehicle Wireless Networks* [Durga Ganesh Reddy et al., 2013] wird zwischen Steuergeräten eines Fahrzeuges ein Netzwerk, das sehr ähnlich zu dem der vorliegenden Arbeit ist, aufgebaut (siehe Abbildung 5.5).

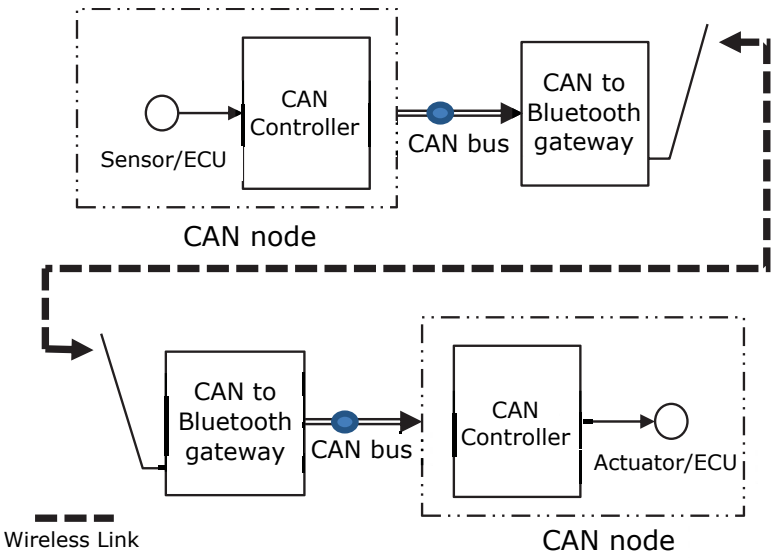

Abbildung 5.5: Skizze über die Kommunikation mit einem CAN-to-Bluetooth-Gateway. Quelle: modifiziert nach [Durga Ganesh Reddy et al., 2013, S. 3]

Durga Ganesh Reddy et al. nutzen für den Versuchsaufbau bereits auf dem Markt erhältliche CAN-to-Bluetooth-Gateways, um CAN-Botschaften zwischen Steuergeräten zu übertragen. Diese Gateways werden regulär für die Verwendung im Werkstattbetrieb mit einem Diagnosewerkzeug genutzt. Ziel der Arbeit von Durga Ganesh Reddy et al. ist jedoch die Verringerung der Kosten durch den sonst hohen Verkabelungsaufwand bei Automobilen.
Der ESP32 beherrscht neben WLAN auch Bluetooth. Die Kommunikation könnte auf Wunsch des Nutzers bei gleichbleibenden Versuchsaufbau umprogrammiert werden.

Übertragungsprotokoll
Um CAN-Botschaften drahtlos zu übertragen, werden sie in das Datenfeld eines Transportprotokolls eingebettet. In dieser Arbeit wurde dafür das Transmission Control Protocol (TCP) gewählt. Ein weiteres, sehr verbreitetes Transportprotokoll ist das User Datagram Protocol (UDP). In Tabelle 5.1 sind TCP und UDP anhand einiger Kriterien vergleichend gegenübergestellt. TCP ließ sich darüber hinaus unkompliziert durch die Bibliotheken des ESP32 in die Software einbinden. Sollte ein UDP-Paket mit einer CAN-Botschaft im

Tabelle 5.1: Vergleich von TCP und UDP. Quelle: [Mandl et al., 2008, Kap. 5]

	TCP	UDP
Verbindungsart	verbindungsorientiert	verbindungslos
Empfangsbestätigung	ja	nein
Paketreihenfolge	gesichert	nicht gesichert
Paketverlust	wird registriert und kompensiert	ja
Empfangspuffer	ja	nein
gilt als	sicher	unzuverlässig

Netzwerk verloren gehen, sind Fehlfunktionen des Demonstrators nicht auszuschließen. Deshalb ist vor allem die gesicherte Verbindung, die einen Paketverlust registriert und kompensiert, das überzeugende Argument für den Einsatz von TCP in dieser Arbeit.

5.2 Vergleich mit weiteren CAN-Bus-Demonstratoren

In der Literatur lassen sich folgende relevante Demonstratoren für den CAN-Bus finden:

A *Development Method of Simulation and Test System for Vehicle Body CAN Bus based on CANoe* [Zhou et al., 2008]
B *YellowCar Automotive Multi-ECU Demonstrator Platform* [Englisch et al., 2017]
C *Demonstrator der RWTH Aachen* [Gerhards, 2005], [Kowalewski, 2011]

Im Folgenden werden diese Demonstratoren vorgestellt und mit der vorliegenden Arbeit verglichen.

Demonstrator A
Die Autoren Zhou et al. erstellten in ihrer Arbeit einen virtuellen Demonstrator. Dazu wurde ein vollständiges Fahrzeug im Programm CANoe von Vector Informatik simuliert. Mit CANoe lässt sich, im Gegensatz zum CANalyzer, ein Netzwerk aus mehreren Steuergeräten simulieren. Der Versuchsaufbau ist in Abbildung 5.6 skizziert.
Die Komponenten kommunizieren über CAN-Botschaften und simulieren Funktionen eines Fahrzeuges. Da der Aufbau nur virtuell erfolgt, können Steuergeräte einfach hinzugefügt oder entfernt werden. Der Fokus der Arbeit von Zhou et al. liegt jedoch nicht bei der digitalen Forensik, sondern bei der Demonstration der Kommunikation mit dem

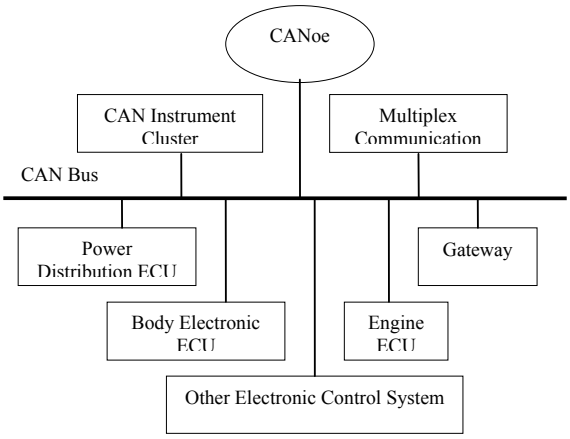

Abbildung 5.6: Versuchsaufbau des virtuellen Demonstrators. Quelle: [Zhou et al., 2008, S.7516]

CAN-Protokoll. Dennoch könnten sich Steuergeräte hinzufügen lassen, die die Funktionen der gezeigten Angriffsszenarien und des IDS übernehmen.

Demonstrator B

Der Demonstrator „YellowCar“ der Technischen Universität Chemnitz verfügt über drei Steuergeräte, die über einen CAN-Bus miteinander kommunizieren. Die Steuerung des gesamten Modells erfolgt ebenfalls drahtlos. Als Grundgerüst dient hier ein elektrisches Fahrzeug für Kinder. Dem Aufbau wurden diverse Sensoren hinzugefügt, die die Umwelt wahrnehmen und die Informationen an ein verarbeitendes Steuergerät leiten. Die restlichen ECUs steuern die Beleuchtung, die Lenkung und die Motoren an. Hier erfolgt die Stromversorgung über eine Batterie. Der Versuchsaufbau ist in Abbildung 5.7 dargestellt.

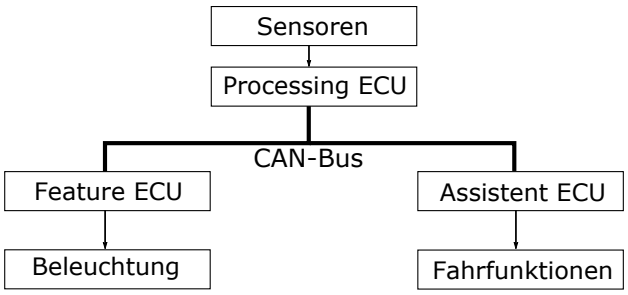

Abbildung 5.7: Versuchsaufbau des „YellowCars“. Quelle: erstellt nach [Englisch et al., 2017]

Ziel dieser Arbeit ist es, einen Demonstrator für Forschung und Lehre zu gestalten, der Softwareentwicklung für reale Fahrzeuge ermöglicht. Außerdem lassen sich Lösungen für das autonome Fahren implementieren und testen.

Demonstrator C

Der Demonstrator der Rheinisch-Westfälische Technische Hochschule Aachen ist auf einem Chassis aus dem Modellsportbereich aufgebaut. Wie Demonstrator B verfügt auch er über Sensoren, die durch Steuergeräte überwacht werden. Ein *Embedded PC* übernimmt Aufgaben mit hohen Anforderungen an die Rechenleistung. Er verfügt über eine drahtlose

Verbindung mit einem Computer, auf dem die Funktionen überwacht werden und über den die Fernsteuerung möglich ist. CAN-Botschaften werden jedoch nur zwischen den verbauten Steuergeräten ausgetauscht. Demonstrator C verfügt darüber hinaus über die Funktion eines Parkassistenten. In Abbildung 5.8 ist der Aufbau schematisch gezeigt. Hier liegt der Fokus vor allem bei der Auswertung der Daten der Sensoren für das autonome Fahren.

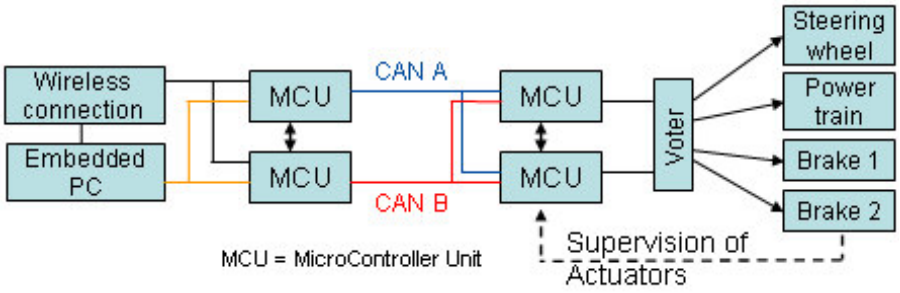

Abbildung 5.8: Skizze des Demonstrators der RWTH Aachen. Quelle: [Kowalewski, 2011]

5.3 Zusammenfassung

Der aktuelle Versuchsaufbau des Demonstrators muss mit den vorhergehenden Überlegungen neu bewertet werden. Die Anforderungen, die sich aus der Zielstellung ergeben, wurden erfüllt. Jedoch konnten im Kapitel 5.1 Alternativen vorgestellt und diskutiert werden. Zusammenfassend ergeben sich daher folgende Konsequenzen:

IDS

Die Umsetzung des IDS als Funktion des CANalyzers ist nicht optimal. Es muss getestet werden, ob eine Implementierung über das Gateway (ESP32-EVB) möglich ist und damit den aktuellen Anforderungen aus der Forschung entspricht.

verwendete Software

Es konnte gezeigt werden, dass der CANalyzer nicht unbedingt bei dieser Arbeit zum Einsatz kommen muss. Kostenlose Alternativen können trotzdem zum gewünschten Ergebnis führen und demonstrieren ebenfalls die Funktionsweise der CAN-Kommunikation. Dennoch bietet der CANalyzer, vor allem mit dem Panel Designer, Funktionen die kein anderes Programm beinhaltet.

drahtlose Kommunikation

Die Kommunikation über WLAN kann komplett durch Bluetooth ersetzt werden, ohne dass der Versuchsaufbau geändert werden muss. Der Einsatz von WLAN stellt allerdings eine größere Sicherheitlücke dar und spiegelt die Realität in der Automobilbranche wider. Vor allem in der Lehre kann die WLAN-Kommunikation anschaulich und effektiv angegriffen werden. So ergeben sich weitere Forschungsprojekte für die digitale Forensik.

Tabelle 5.2: Direkter Vergleich mit weiteren CAN-Bus-Demonstratoren

	forensischer Demonstrator	Demonstrator A	Demonstrator B	Demonstrator C
Chassis aus dem Bereich	Modellbau	-	Fahrzeug für Kinder	Modellsport
Anzahl Steuergeräte	1	> 6	3	4
Sensoren	nein	nein	ja	ja
Kommunikation über das CAN	ja	ja	ja	ja
drahtlose Kommunikation	ja	-	ja	ja
Funktionen eines Automobils	ja	simuliert	ja	ja
Mobilität	ja	-	ja	ja
Fokus	forensische Auswertung	Simulation des CAN	Software-entwicklung für Automobile	Test für autonomes Fahren

Bei allen Arbeiten, die zum Vergleich herangezogen wurden, liegt das Hauptaugenmerk auf der realitätsnahen Umsetzung eines Fahrzeuges in Modellform und nicht bei der digital-forensischen Auswertung der CAN-Kommunikation. Die Anforderungen an alle Demonstratoren decken sich deshalb nur in bestimmten Punkten (Steuerung über CAN-Botschaften, Realisierung einfacher Funktionen eines Fahrzeuges, Mobilität, siehe Tabelle 5.2). Die Sicherheitslücken des CAN-Protokolls, die in der vorliegenden Arbeit aufgezeigt und ausgenutzt wurden, sind jedoch ein allgemeines Problem. Es tritt auf, sobald das CAN zum Einsatz kommt.
Der Vergleich mit weiteren CAN-Bus-Demonstratoren hat ergeben, dass es derzeit keine ähnliche Arbeit gibt, die sich auf die digital-forensische Auswertung des CAN-Protokolls fokussiert. Der hier erarbeitete Demonstrator stellt eine Grundlage für die weitere Forschung auf diesem Gebiet dar.

6 Fazit

Mit dieser Arbeit konnte ein funktionsfähiger Demonstrator mit den gegebenen Komponenten erstellt und durch CAN-Nachrichten gesteuert werden. Durch die drahtlose Kommunikation ist es sogar möglich, grundlegende Funktionen eines Fahrzeuges ähnlich zu einem Modellauto nachzustellen. Dadurch kann der CAN-Bus vor allem in der Lehre anschaulich demonstriert werden. Es können Angriffe auf ihn ausgeübt werden, die bereits bei realen Fahrzeugen Anwendung fanden. Das eingebaute IDS kann vor Angriffen durch Meldungen warnen, eine forensische Analyse ist daher möglich. Jedoch werden die Ausgaben nicht in den Modulen des Demonstrators gespeichert.
Alle gestellten Anforderungen, die sich aus der Zielstellung ergaben, wurden erfüllt. Der Demonstrator kann deshalb eine Grundlage für weitere Forschungsprojekte auf dem Gebiet der digitalen Forensik sein und eventuell als Alternative zu einem realen Fahrzeug genutzt werden. Die programmierten Funktionen müssen dennoch weiter auf unerwünschtes Verhalten oder Fehler getestet werden.

Während der Umsetzung dieser Arbeit ergaben sich bereits weitere Ansätze für den Einsatz des Demonstrators. Es könnte eine Applikation für Smartphones entwickelt werden, die Nachrichten über TCP-Pakete an den Demonstrator übermittelt. Durch den Touchscreen könnte eine grafische Oberfläche schneller und intuitiver bedient werden als das Dashboard im CANalyzer. Außerdem kann so auf den Einsatz teurer Software verzichtet und die Oberfläche noch besser personalisiert werden.
Das ESP32-EVB beherrscht weitere drahtlose Übertragungsprotokolle. Die CAN-Botschaften könnten demnach auch per Bluetooth übermittelt werden. Außerdem lässt sich testen, ob sich das IDS als Software im Gateway realisieren lässt. Dabei könnten ebenfalls die aufgestellten Regeln für die Klassifikation überdacht werden.
In einem bereits begonnenen Projekt wird ein weiteres Steuergerät aus einem realen Fahrzeug mit dem Demonstrator verbunden. Es handelt sich hierbei um eine Wegfahrsperre, die mit dem zugehörigen Autoschlüssel deaktiviert werden kann. Dadurch soll die Steuerung durch den Benutzer erst zugelassen werden. Auch weitere Steuergeräte können dem verbauten CAN-Bus für Testzwecke hinzugefügt werden. In Anlehnung an das „Yellow Car“ aus Chemnitz oder den Demonstrator der RWTH Aachen könnten so mit Hilfe von Sensoren Funktionen für das autonome Fahren implementiert werden.

Die Sicherheitslücken in Automobilen sind aus digital-forensischer Sicht betrachtet schwerwiegend. Ein Angriff durch Cyberkriminelle auf ein Fahrzeug ist möglich und wird zudem nicht gemeldet oder dokumentiert. Obwohl in der Forschung häufiger Vorschläge für eine Abänderung (LiBrA-CAN [Groza et al., 2012]) oder eine Erweiterung (VeCure [Wang und Sawhney, 2014]) des CAN-Protokolls vorgeschlagen werden, wurde das CAN in den letzten Jahren offiziell nur um die Möglichkeit einer höheren Datenübertragungsrate ausgebaut (CAN mit flexibler Datenrate (CAN FD)). Eine Authentifizierung bei oder Ver-

schlüsselung der Kommunikation hätte einen Remote-Angriff wie von Miller und Valasek beschrieben oder das Einbringen eines, mit Schadsoftware ausgestatteten, Steuergerätes wie bei der Arbeit von Palanca et al. erheblich erschwert. Im April 2018 veröffentlichte die niederländische Firma Computest [2018] ihre Recherchen zur erfolgreichen Anwendung der von Miller und Valasek beschriebenen Sicherheitslücken. Mit ähnlichen Mitteln erfolgte im Mai 2018 der Hack eines BMW durch die chinesischen Sicherheitsforscher des Tencent Keen Security Lab [2018]. Es besteht deshalb der dringende Bedarf weiterer Forschung auf diesem Gebiet.

Anhang A: Schaltplan

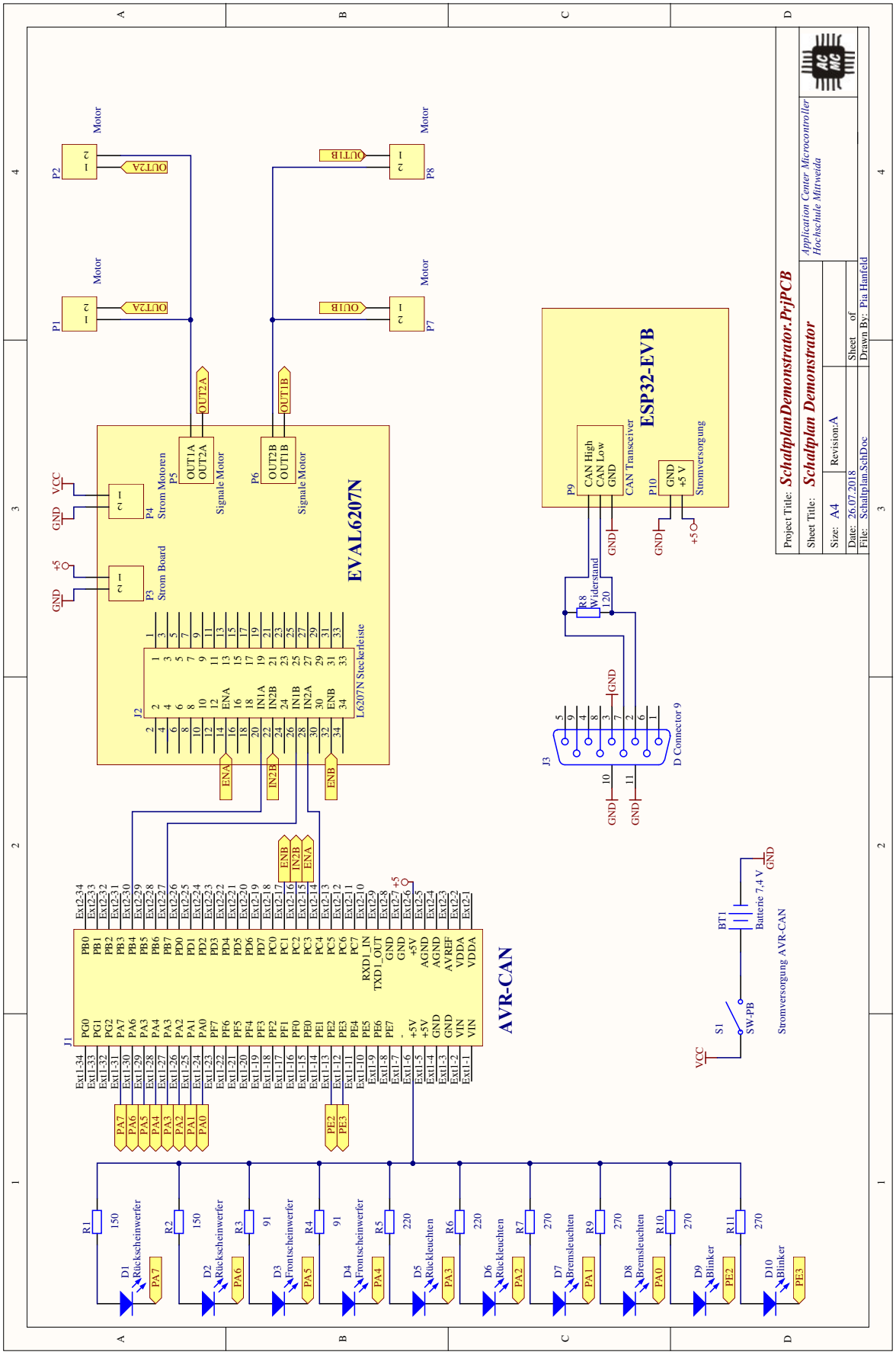

Abbildung A.1: Schaltplan des Demonstrators. Quelle: eigene Arbeit

Anhang B: Programmablauf CAN-Knoten

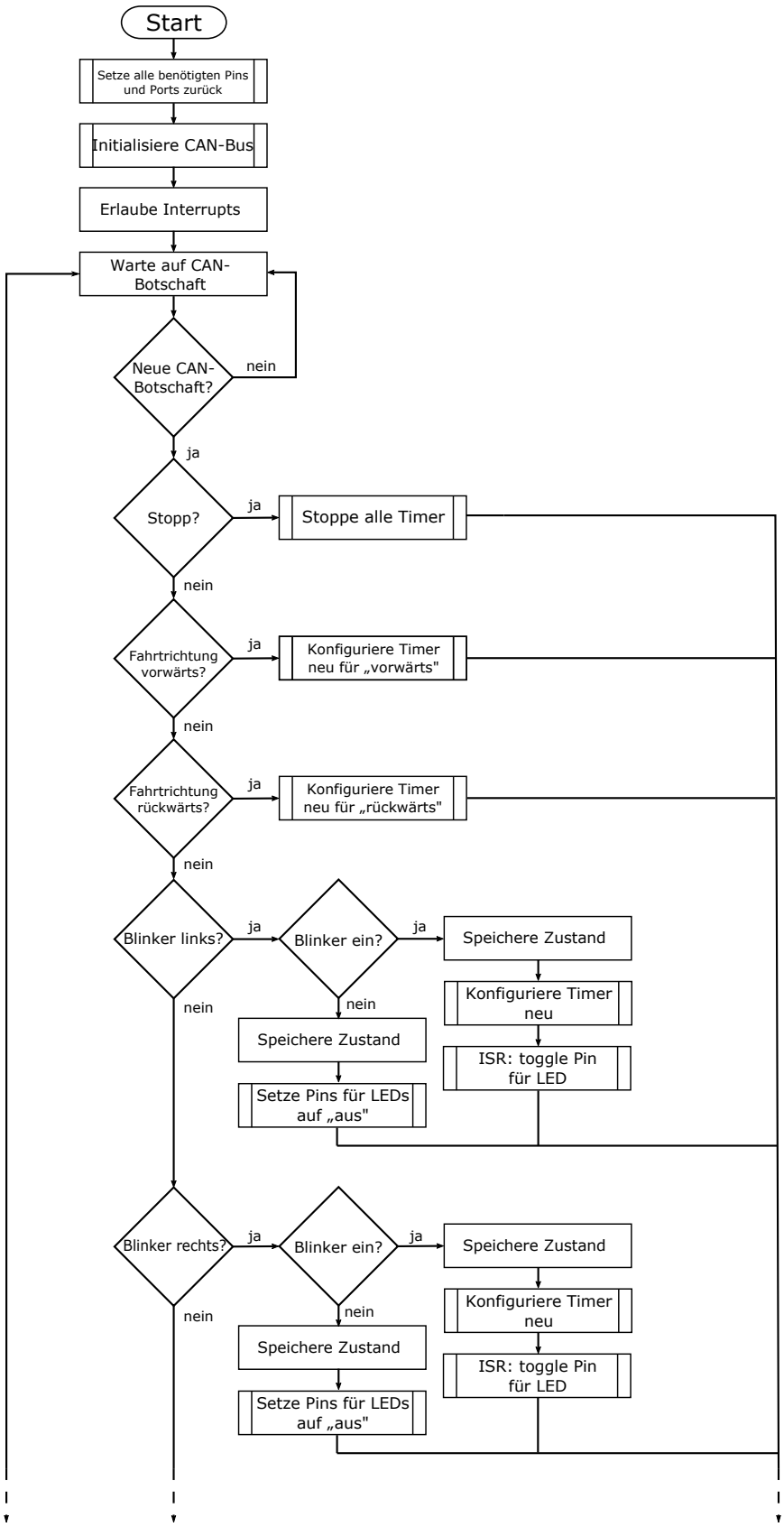

Abbildung B.1: Skizze über den Programmablauf des CAN-Knotens (Teil 1). Quelle: eigene Arbeit

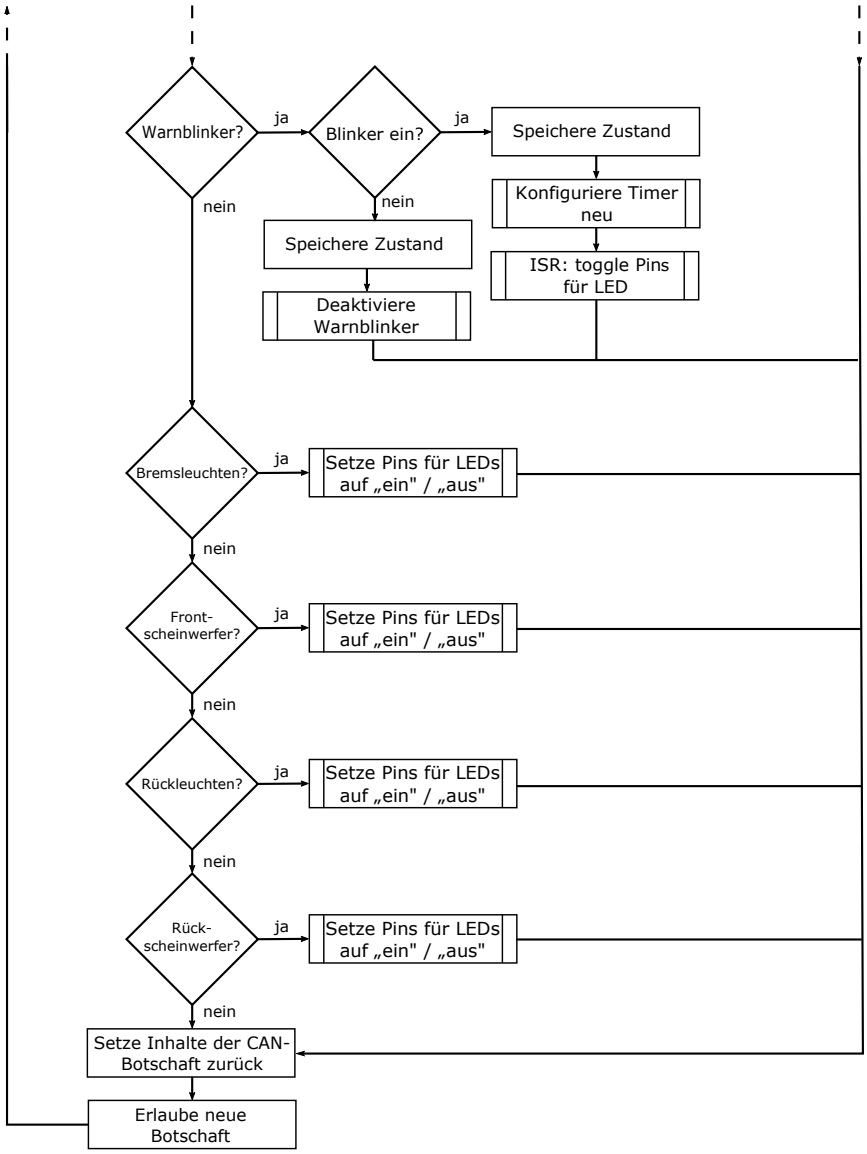

Abbildung B.2: Skizze über den Programmablauf des CAN-Knotens (Teil 2). Quelle: eigene Arbeit

Anhang C: Oszillogramm einer CAN-Botschaft

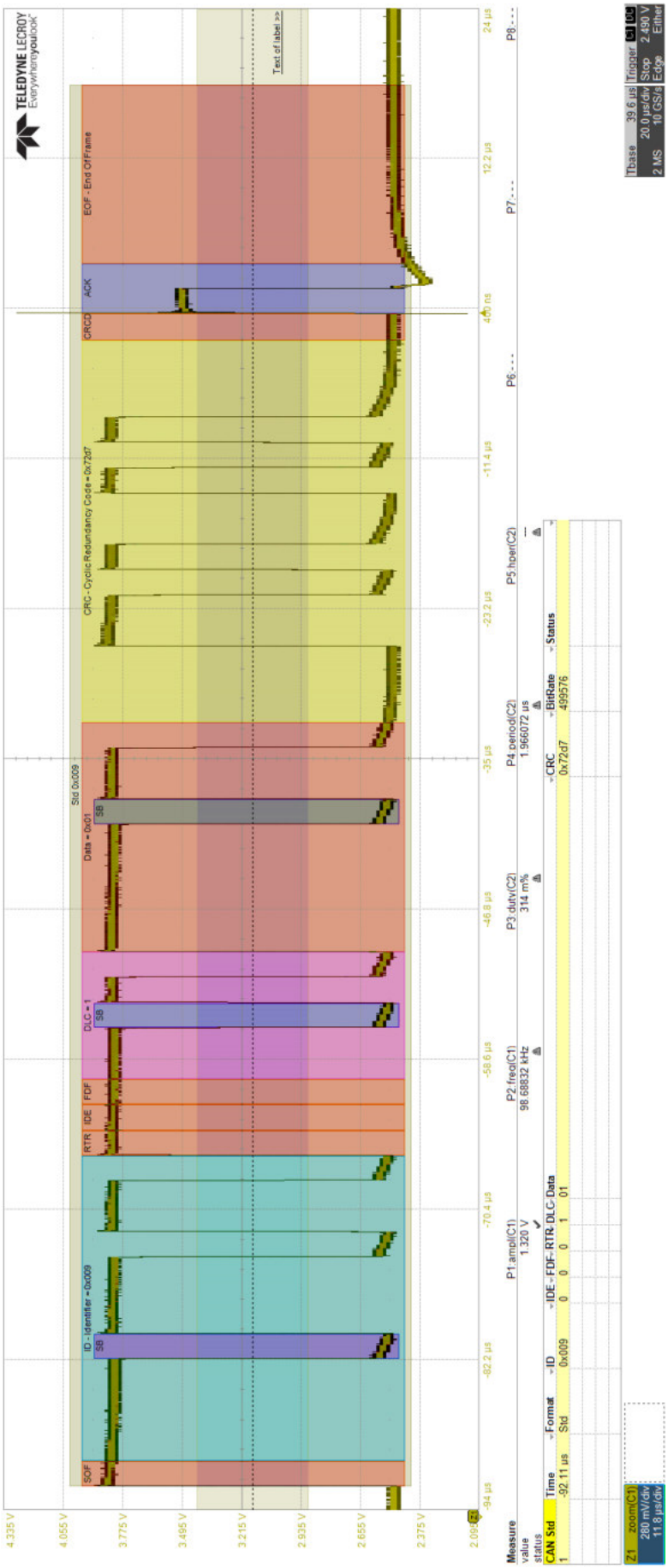

Abbildung C.1: Oszillogramm einer CAN-Botschaft. Quelle: eigene Aufnahme

Literaturverzeichnis

Atmel Corporation. AT90CAN32/64/128, 2008. URL http://ww1.microchip.com/downloads/en/DeviceDoc/doc7679.pdf. Zuletzt geprüft am: 14.08.2018.

T. Beierlein, Hrsg. *Taschenbuch Mikroprozessortechnik*. Fachbuchverl. Leipzig im Carl-Hanser-Verl., München, 4., neu bearb. Aufl., 2011. ISBN 9783446423312.

K. Beiter, C. Rätz, und O. Garnatz. Gesetzliche On-Board-Diagnose und ODX. In B. Bäker und A. Unger, Hrsgg., *Diagnose in mechatronischen Fahrzeugsystemen III*, S. 44–56. expert-Verl., Renningen, 2010. ISBN 9783816929949.

M. Bresch und N. Salman. *Design and implementation of an intrusion detection system (IDS) for in-vehicle networks*. Department of Computer Science and Engineering (Chalmers), Chalmers University of Technology, Göteborg, 2017.

Computest. The Connected Car: Ways to get unauthorized access and potential implications, 2018. URL https://www.computest.nl/wp-content/uploads/2018/04/connected-car-rapport.pdf. Zuletzt geprüft am: 05.07.2018.

A. V. Durga Ganesh Reddy, G. Dhadyalla, und N. Kumari. Experimental Validation of CAN to Bluetooth Gateway for In-Vehicle Wireless Networks. In *International Conference on Emerging Trends in Communication, Control, Signal Processing & Computing Applications (C2SPCA), 2013*, S. 1–5, Piscataway, NJ, 2013. IEEE. ISBN 978-1-4799-1085-4. doi: 10.1109/C2SPCA.2013.6749460.

N. Englisch, O. Khan, R. Mittag, F. Hänchen, W. Hardt, und A. Heller. YellowCar Automotive Multi-ECU Demonstrator. In *INFORMATIK 2017, 15. GI Workshop Automotive Software Engineering*, S. 1517–1522. Chemnitz, 2017.

Espressif Systems. ESP32 Series Datasheet, 2018. URL https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf. Zuletzt geprüft am: 14.08.2018.

ETAS GmbH. BUSMASTER, 2017. URL https://rbei-etas.github.io/busmaster/. Zuletzt geprüft am: 18.07.2018.

K. Etschberger, Hrsg. *Controller-Area-Network ; Grundlagen, Protokolle, Bausteine, Anwendungen*. Hanser, München, 2., völlig überarb. Aufl., 2000. ISBN 3446194312.

K. Gerhards. *Konzeption und Umsetzung eines Versuchsfahrzeugs für automotive Software*. Diplomarbeit, Rheinisch-Westfälische Technische Hochschule, Aachen, 2005.

M. Glietsch. Funktionsbibliothek für CAN, 2008. URL https://www.mikrocontroller.net/topic/98697. Zuletzt geprüft am: 03.07.2018.

B. Groza, S. Murvay, A. van Herrewege, und I. Verbauwhede. LiBrA-CAN: A Lightweight Broadcast Authentication Protocol for Controller Area Networks. In J. Pieprzyk, A.-R. Sadeghi, und M. Manulis, Hrsgg., *Cryptology and network security*, Ausgabe 7712 von *Lecture Notes in Computer Science*, S. 185–200. Springer, Berlin, 2012. ISBN 978-3-642-35403-8. doi: 10.1007/978-3-642-35404-5_15.

E. Hering, K. Bressler, und J. Gutekunst, Hrsgg. *Elektronik für Ingenieure und Naturwissenschaftler*. Springer Vieweg, Berlin, 7., aktualisierte und verbesserte Auflage, 2017. ISBN 978-3-662-54213-2. doi: 10.1007/978-3-662-54214-9.

Joy-IT. Robot Car Kit 01 für Arduino, 2017. URL http://anleitung.joy-it.net/wp-content/uploads/2017/11/AnleitungRobot03.pdf. Zuletzt geprüft am: 17.07.2018.

M. Knapp und C. S. Lam. CAN to Wi-Fi Gateway Design Guide, 2015. URL http://www.ti.com/lit/pdf/tidual1. Zuletzt geprüft am: 07.08.2018.

S. Kowalewski. Aufbau des Modellautos, 2011. URL https://embedded.rwth-aachen.de/doku.php?id=forschung:modellauto_aufbau. Zuletzt geprüft am: 06.07.2018.

P. Mandl, A. Bakomenko, und J. Weiß. *Grundkurs Datenkommunikation: TCP/IP-basierte Kommunikation: Grundlagen, Konzepte und Standards*. Technische und Ingenieurinformatik. Vieweg+Teubner Verlag / GWV Fachverlage GmbH Wiesbaden, Wiesbaden, 2008. ISBN 978-3-8348-9270-6. doi: 10.1007/978-3-8348-9270-6.

C. Miller und C. Valasek. Adventures in Automotive Networks and Control Units, 2013. URL http://illmatics.com/car_hacking.pdf. Zuletzt geprüft am: 11.07.2018.

C. Miller und C. Valasek. Remote exploitation of an unaltered passenger vehicle. *Black Hat USA*, 2015.

D. K. Nilsson und U. E. Larson. Conducting Forensic Investigations of Cyber Attacks on Automobile In-vehicle Networks. In M. Sorell, Hrsg., *Proceedings of the 1st International Conference on Forensic Applications and Techniques in Telecommunications, Information, and Multimedia and Workshop*, e-Forensics 2008, S. 8:1–8:6, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). ISBN 978-963-9799-19-6.

Olimex Ltd. AVR-CAN development board, 2010. URL https://www.olimex.com/Products/AVR/Development/AVR-CAN/resources/AVR-CAN.pdf. Zuletzt geprüft am: 03.07.2018.

A. Palanca, E. Evenchick, F. Maggi, und S. Zanero. A Stealth, Selective, Link-Layer Denial-of-Service Attack Against Automotive Networks. In M. Polychronakis und M. Meier, Hrsgg., *Detection of intrusions and malware, and vulnerability assessment*, Ausgabe 10327 von *Lecture Notes in Computer Science*, S. 185–206. Springer, Cham, 2017. ISBN 978-3-319-60875-4. doi: 10.1007/978-3-319-60876-1_9.

Robert Bosch GmbH. CAN Specification: Version 2.0, 1991. URL http://esd.cs.ucr.edu/webres/can20.pdf. Zuletzt geprüft am: 05.07.2018.

M. Sauter, Hrsg. *Grundkurs mobile Kommunikationssysteme: LTE-Advanced, UMTS, HSPA, GSM, GPRS, Wireless LAN und Bluetooth*. Springer Vieweg, Wiesbaden, 6., überarb. und erw. Aufl., 2015. ISBN 978-3-658-08341-0. doi: 10.1007/978-3-658-08342-7.

M. Stebner. *Entwurf eines frei konfigurierbaren CAN-Manipulators*. Bachelorarbeit, Hochschule Mittweida, Mittweida, 2017.

STMicroelectronics. L6205, L6206, L6207 DUAL FULL BRIDGE DRIVERS, 2003. URL https://www.mouser.de/ds/2/389/cd00004482-953611.pdf. Zuletzt geprüft am: 03.07.2018.

STMicroelectronics. ST Industrial Communication Board - EVALCOMMBOARD, 2006. URL https://www.mouser.de/ds/2/389/cd00004482-953611.pdf. Zuletzt geprüft am: 03.07.2018.

STMicroelectronics. DMOS dual full bridge driver with PWM current controller, 2014. URL www.st.com/resource/en/datasheet/l6207.pdf. Zuletzt geprüft am: 03.07.2018.

Team BHP. Abbildung eines Dashboard im Fahrzeug, 2014. URL http://www.team-bhp.com/forum/attachments/technical-stuff/1221050d1395126616-about-non-linear-speedometers-b.jpg. Zuletzt geprüft am: 22.08.2018.

Tencent Keen Security Lab. Experimental Security Assessment of BMW Cars: A Summary Report, 2018. URL https://keenlab.tencent.com/en/Experimental_Security_Assessment_of_BMW_Cars_by_KeenLab.pdf. Zuletzt geprüft am: 05.07.2018.

Vector CANtech. *Programming With CAPL*. Novi, Michigan 48375 USA, 2004.

Q. Wang und S. Sawhney. VeCure: A practical security framework to protect the CAN bus of vehicles. In *2014 International Conference on the Internet of Things (IOT)*, S. 13–18, Piscataway, NJ, 2014. IEEE. ISBN 978-1-4799-5154-3. doi: 10.1109/IOT.2014.7030108.

F. Zhou, S. Li, und X. Hou. Development Method of Simulation and Test System for Vehicle Body CAN Bus based on CANoe. In *7th World Congress on Intelligent Control and Automation, 2008*, S. 7515–7519, Piscataway, NJ, 2008. IEEE. ISBN 978-1-4244-2113-8. doi: 10.1109/WCICA.2008.4594092.

W. Zimmermann und R. Schmidgall. *Bussysteme in der Fahrzeugtechnik*. Springer Fachmedien Wiesbaden, Wiesbaden, 2014. ISBN 978-3-658-02418-5. doi: 10.1007/978-3-658-02419-2.

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die Arbeit noch nicht anderweitig für Prüfungszwecke vorgelegt habe.

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

Mittweida, 22. August 2018

Pia Hanfeld