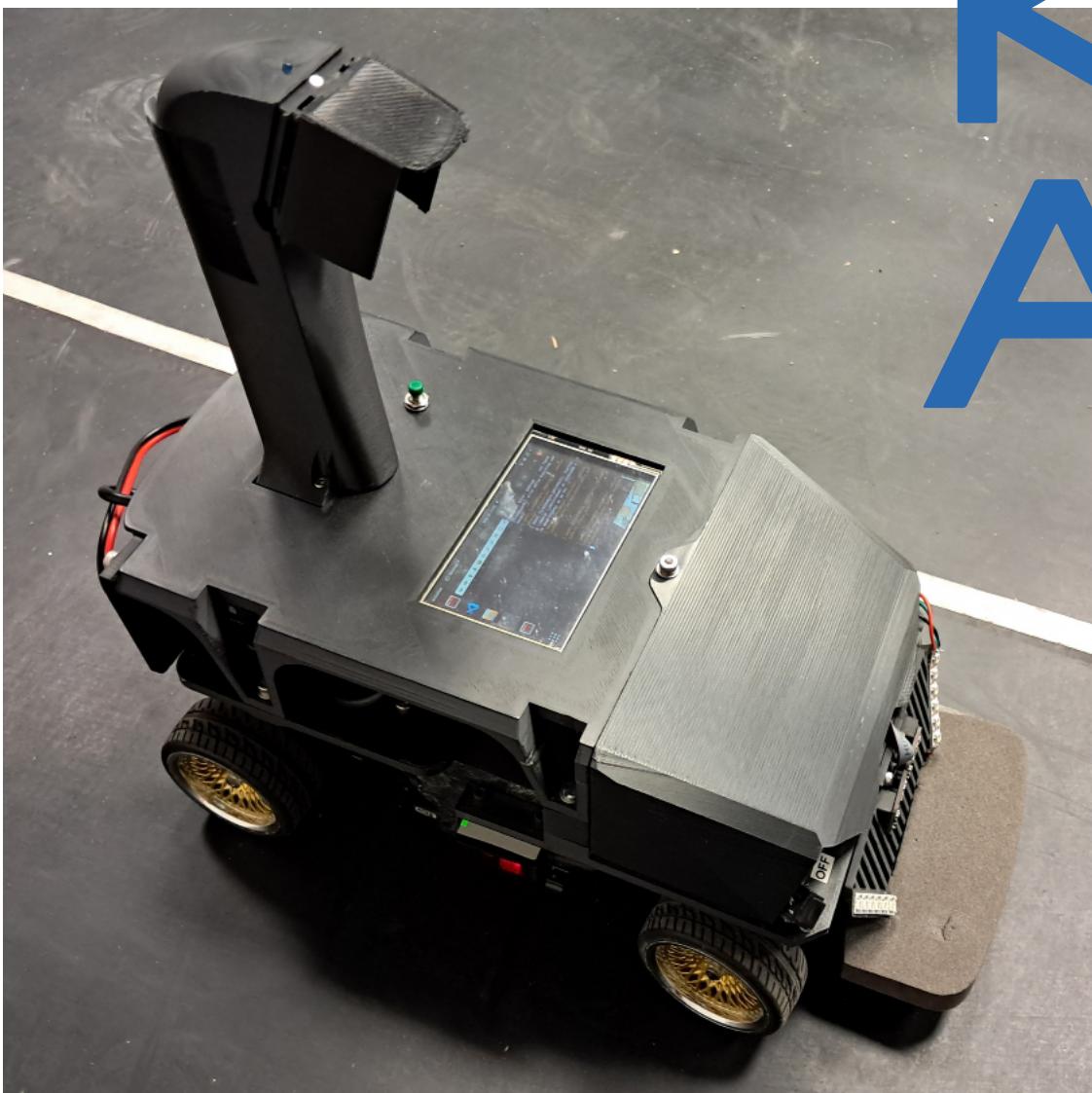


Projektbericht Autonomes Modellfahrzeug

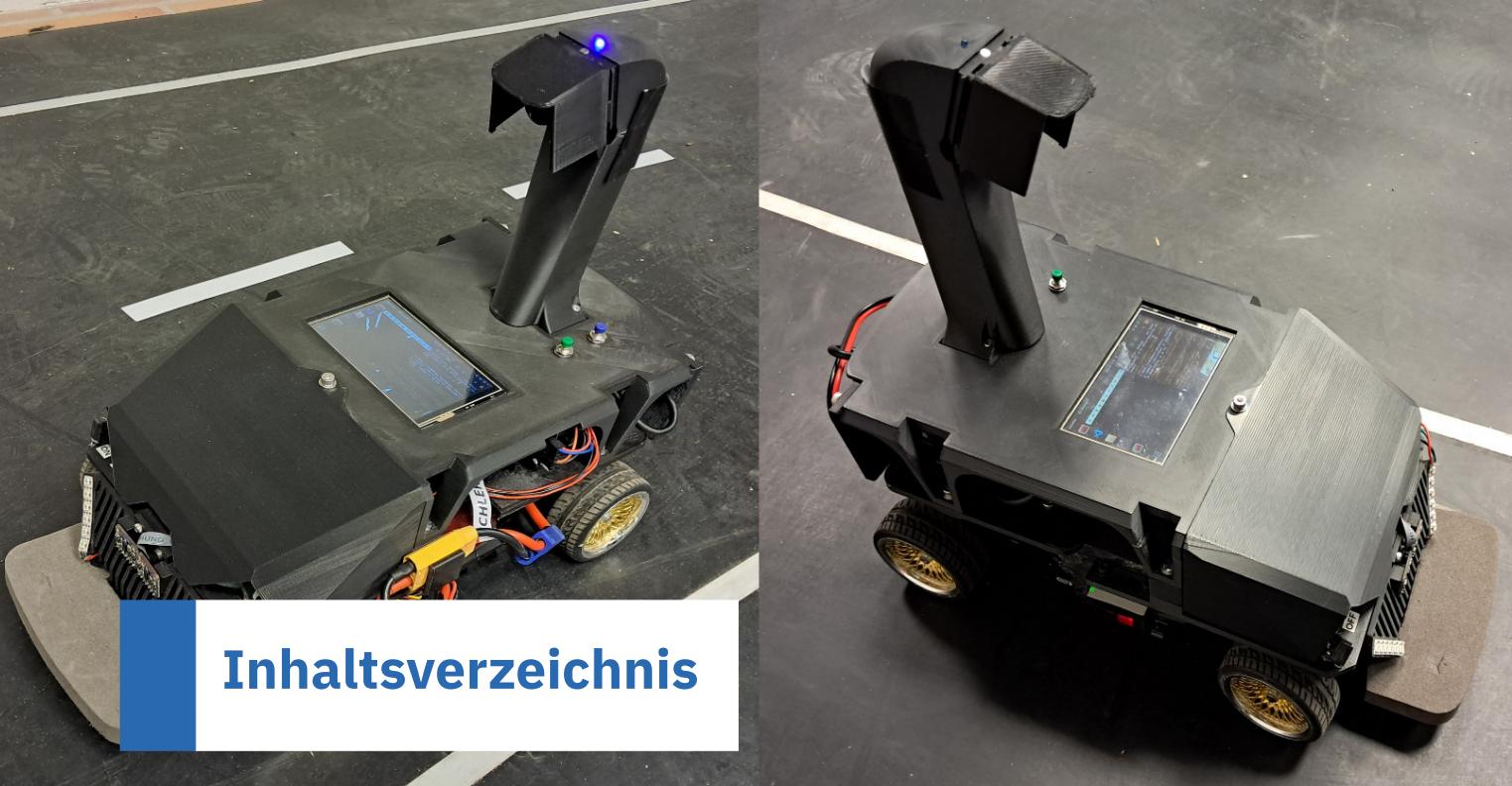
24ws OL AutonomesFahrzeug

Stand 28. März 2025

K
A



Copyright © 2024 Sven Tolkmitt 67570, Lars Pföhler 77920
Published by Sven Tolkmitt, Lars Pföhler
https://ilias.h-ka.de/ilias.php?baseClass=ilWikiHandlerGUI&ref_id=208764&cmd=view
Dieses Skript wird als Lehrmaterial für die Vorlesung Entwicklungsprojekt Wintersemester 2024/2025 an der Hochschule Karlsruhe verwendet. Das Skript wird unter der folgenden Lizenz veröffentlicht: CC BY-NC-SA 3.0 (<http://creativecommons.org/licenses/by-nc-sa/3.0/>).
Erste Ausgabe, Stand: Oktober 2024



Inhaltsverzeichnis

Erklärung 6

Abkürzungsverzeichnis 7

Abbildungsverzeichnis 8

I

Einleitung

1	Aufgabenstellung	10
2	Zuständigkeit der Projektgruppen	13
2.1	Hardware-Projektgruppe	13
2.2	Software-Projektgruppen	13
3	Motivation und Ziel der Projektes	14
4	CAuDri-Challenge	15
4.1	Regularien	15
4.2	CAuDri-Events	16
4.2.1	Freie Fahrt	16
4.2.2	Hinternisumfahrung	16
4.2.3	Navigations-Kurs	16

II	Ausgangssituation	
5	Hardware	18
5.1	NUC	18
5.2	Spannungsversorgung	18
5.3	NUC	18
5.4	Motortreiber	19
5.5	Verkleidung des Fahrzeuges	19
5.6	Anzeige der Desktopumgebung	20
6	Software	21
6.1	Dateistruktur	21
6.1.1	Startskript reales Fahrzeug	23
6.1.2	MCT_arduino_mega	23
6.1.3	MCT_arduino_mega_test	23
6.1.4	MCT_arduino_nano	23
6.1.5	MCT_arduino_nano_test	24
6.2	NUC	24
6.3	Catkin-Make	24
III	Umsetzung	
7	Hardware	26
7.1	Bestellte Teile	26
7.2	Spannungsversorgung	26
7.2.1	Akku	26
7.2.2	DC-DC-Wandler	27
7.2.3	Verbau DC-DC-Wandler	28
7.2.4	Spannungswerte	32
7.3	Anzeige der Desktopumgebung	32
7.4	Motortreiber	32
7.5	Prüf- und Ladekebel mit XT90	33
8	Software	35
8.1	Startskript reales Fahrzeug	35
8.2	NUC	36
IV	Inbetriebnahme	
9	Handhabung	38
9.1	Akkuladegerät	38
9.2	Auto starten	40
9.3	Software starten	41

10 Zusammenfassung **44**

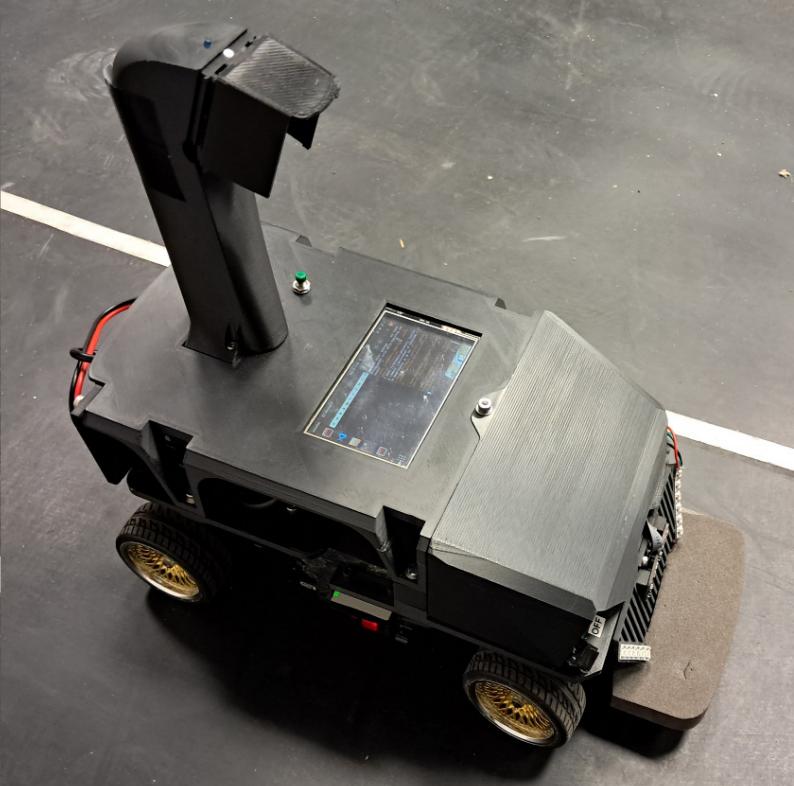
11 Erforderliche zukünftige Arbeiten **45**

Bibliography

Bibliography **47**



Erklärung



Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst habe, dass ich sie zuvor an keiner anderen Hochschule und in keinem anderen Studiengang als Prüfungsleistung eingereicht habe und dass ich keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle Stellen der Arbeit, die wörtlich oder sinngemäß aus Veröffentlichungen oder aus anderweitigen fremden Äußerungen entnommen wurden, sind als solche kenntlich gemacht.

.....
Ort, Datum

.....
(Sven Tolkmitt)

.....
(Lars Pföhler)



Abkürzungsverzeichnis

ARP	Address Resolution Protocol
CAuDri	Cognitive Autonomous Driving
HDMI	High Definition Multimedia Interface
LiPo	Lithium-Ionen-Polymer
NUC	Next Unit of Computing
RC	Remote Controlled
ROS	Robot Operating System
SPI	Serial Peripheral Interface
ToF	Time of Flight
USB	Universal Serial Bus



Abbildungsverzeichnis

5.1 Rückwärtige Verkleidung des Fahrzeugs	19
6.1 Vereinfachte Datei- und Ordnerstruktur	22
7.1 DC-DC-Wandler	27
7.2 Versuchsaufbau DC-DC-Wandler	28
7.3 Grundplatte der Rückwärtigen Verkleidung	29
7.4 Arretierung des DC-DC-Wandlers	30
7.5 Rückwärtige Abdeckplatte mit verbautem DC-DC-Wandler	31
7.6 Montage des DC-Dc-Wandlers am Fahrzeug	31
7.7 Zerlegter Motortreiber mit Anschlusskabeln	33
7.8 Prüfkabel für Labornetzteil	33
7.9 Ladekabel XT90	34
9.1 Verbindung Ladegerät mit Akku	38
9.2 Vorgehensweise beim Laden der Akkus	39
9.3 An- und Aus-Schalter des Fahrzeug	40

T

Einleitung

1	Aufgabenstellung	10
2	Zuständigkeit der Projektgruppen	13
2.1	Hardware-Projektgruppe	
2.2	Software-Projektgruppen	
3	Motivation und Ziel der Projektes	14
4	CAuDri-Challenge	15
4.1	Regularien	
4.2	CAuDri-Events	



1. Aufgabenstellung

Die Aufgabenstellung umfasst die Instandsetzung und Weiterentwicklung des bisherigen Modellfahrzeugs. Die anliegenden Projektthemen und damit verbundene Aufgabenstellungen sind sehr vielfältig und umfassen u.a. Bildverarbeitung, Regelung/Steuerung, Mapping, Hardware (Sensoren, Aktoren) und Simulation.

Für diese Projektarbeit liegt der Fokus auf Hardwareüberarbeitung und der Inbetriebnahme des Modellfahrzeugs. Die Entwicklung erfolgt in enger Zusammenarbeit mit den anderen Projektgruppen. Das Ziel dieses Projektes setzt ein lauffähiges und nach Möglichkeit autonom fahrendes Modellfahrzeug dar.

Aufgabenstellung Projektarbeit

Projekttitle: Autonomes Modellfahrzeug

Stichworte: autonomes Fahren, NUC-PC, Arduinos, ROS, Sensoren, Modellbau, ...

Hintergrund zum Projektthema:

In den letzten Jahren wurde von Studierenden der Fakultät MMT ein autonomes Modellauto entwickelt, um am CaroloCup (siehe z. B. YouTube) der TU Braunschweig teilnehmen zu können. Das Modelfahrzeug soll in der Lage sein, autonom auf einer Straße mit Kreuzungen, Kurven, Hindernissen, Fehlern in der Fahrbahnmarkierung und Verkehrsschildern fahren zu können. Nach einem neuen Wettbewerb wird gesucht.

Aufgabenstellung des Projektes:

Das bisherige Modellfahrzeug soll instandgesetzt und weiterentwickelt werden. Die anliegenden Projektthemen/Aufgabenstellungen sind sehr vielfältig und umfassen u. a. Bildverarbeitung, Regelung/Steuerung, Mapping, Hardware (Sensoren, Aktoren), Simulation, ...

Die Projektarbeit wird/kann folgende Ziele bzw. Ergebnisse beinhalten:

- Einarbeiten in LINUX, ROS und Gazebo, sowie in das Fahrzeug (Aufbau, Kommunikation zwischen NUC und Arduino, ...) und Inbetriebnahme des Fahrzeugs
 - Weiterentwicklung des Fahrzeugs, z. B.
 1. Verbessern der Quer- und Längsregelung
 2. Erstellen einer Straßenkarte mit Hilfe des IMUs und Nutzen der Informationen für die Regelung
 3. Testen und Weiterentwickeln der Kommunikation zwischen Arduino und NUC (Übertragung von Sensordaten)
 4. Hardwareüberarbeitung (Stromversorgung, Sensoren, ...)
 5. Umsetzen/Verbessern der Einparkkonzepte
 6. Weiterentwicklung der Simulationsumgebung für das autonome Modelfahrzeug in Gazebo
 7. Weiterentwicklung der Bildverarbeitung (innerhalb der realen Umgebung) und der Regelung
 - Die möglichen Weiterentwicklungsthemen sind vielfältig und können an die Interessen der Studierenden angepasst werden.
 - Die Projektergebnisse müssen in einem Bericht dokumentiert werden. Anstelle des geforderten Projektposters wäre ein Projektvideo wünschenswert.

Bearbeitung im: WS 2024/25, jedes Semester	Zielgruppe: hauptsächlich: MECB,FZTB,MECM,ASEM	Projekt-Code: 24ws_OL_AutonomesFahrzeug		
Betreuer (Professor): Prof. Dr. Ferdinand Olawsky; Tel.: 1710; E-Mail: Ferdinand.Olawsky@h-ka.de				
weitere Betreuer: Jürgen Ewert				
Unterschrift mit Datum (Betreuer):		22.10.24 F.Olawsky		
Projektbearbeiter				
Vorname	Name	Studiengang	Matr.-Nr.	Unterschrift
Sven	Talkmitt	Fahrzeugtechnologie	67570	
Lars	Pföhler	ASEM	77920	

Name: Sven Tolkmitt, Lars Pföhler		Titel der Projektarbeit Autonomes Fahrzeug															Statusdatum 20.02.2025		
Betreuer: Prof. Dr.-Ing. Ferdinand Olawsky																			
Koordinator: Prof. Dr.-Ing. Ferdinand Olawsky																			
Projektcode: 24ws_OL_AutonomesFahrzeug		Bearbeitungs-																	
		Status																	
		0-9																	
		Kalenderwoche																	
		41 42 43 44 45 46 47 48 49 50 51 52																	
		1 2 3 4 5 6 7 8 9 10 11 12 13 14																	
Nr.	Aktivitäten:	Offizieller Starttermin:																	
	1 Einarbeitung in Grundlagen	22.10.2024																	
	2 Fahrzeugsbestandsaufnahme	9																	
	3 Spannungsversorgung NUC	9																	
	4 Versorgung Fahrzeug	9																	
	5 Fahrzeugtest	8																	
	6 Software Update/ Migration	9																	
	7 Inbetriebnahme	9																	
	8 Schriftliche Ausarbeitung	9																	
	9 Korrektur, Drucken, Binden	9																	
10																			
11																			
	Abgabedatum:	31.03.2025																	
Meilensteine		1 2 3 4 5 6 7 8																	
Meilensteine (Liste siehe unten)																			
Betreuer in der Hochschule nicht erreichbar																			
Koordinator in der Hochschule nicht erreichbar																			
Prüfungszeit und Vorbereitungen																			
Eigener Urlaub																			



2. Zuständigkeit der Projektgruppen

2.1 Hardware-Projektgruppe

Die in diese Projektarbeit involvierte Gruppe nimmt das Fahrzeug in Betrieb. Das umfasst die Bestandsaufnahme der verbauten Komponenten und deren Funktionsprüfung. Essenziell ist auch die Software, sofern diese von den anderen betreffenden Gruppen lauffähig vorliegt. Diese wird mit den letzten vorliegenden Softwarestand auf dem Modelfahrzeug in Betrieb genommen und auf Ihre Funktion geprüft.

Die Software der Arduinos und deren Funktionsabläufe fallen in den Zuständigkeitsbereich der Hardware-Gruppe. Die Arduinos steuern die Beleuchtung und den Antriebsstrang.

2.2 Software-Projektgruppen

Die Software und die damit verbundenen Projektgruppen lassen sich in zwei Themenbereiche eingliedern. Ein Team überarbeitet die Bildverarbeitung. Damit verbunden sind Linien-, Verkehrsschild- und Hindernis-Erkennung. Diese Daten sind für eine autonome Fahrt unerlässlich.

Eins weiteres Team optimiert die Verarbeitung der Sensordaten und die damit verbundene Regelung. Die Regelung verarbeitet die eingehenden Größen der Sensoren und hat als Ergebnis eine Bahnkurve, welcher das Fahrzeug folgen soll.

Die Ergebnisse und Einzelheiten der Software-Projektgruppen lassen sich den entsprechenden Berichten und Softwareständen entnehmen.



3. Motivation und Ziel der Projektes

Das Ziel der Projektarbeit kann entsprechend der Aufgabenstellung um die folgenden Punkte konkretisiert werden.

- Einarbeiten in LINUX, ROS und Gazebo, sowie in das Fahrzeug (Aufbau, Kommunikation zwischen NUC und Arduino, ...) und Inbetriebnahme des Fahrzeugs
- Testen und Weiterentwickeln der Kommunikation zwischen Arduino und NUC (Übertragung von Sensordaten)
- Hardwareüberarbeitung (Stromversorgung, Sensoren, ...)

Die Motivation für die Entwicklung eines autonom fahrenden Modelfahrzeugs basiert neben der aktuellen Entwicklung der Automobilindustrie auf der Teilnahme am Wettbewerb Carolo-Cup [5]. Der Carolo-Cup bietet einen Rahmen für kompetitive Herausforderungen im Bereich des autonomen Fahrens. Da der Wettbewerb derzeit nicht stattfindet, wird trotzdem entsprechend dessen Regularien entwickelt, um einen repräsentativen Standard zu verwenden.

Ein aktuell noch ausgetragener und vergleichbarer Wettbewerb ist die CAuDri-Challenge. Die Rahmenbedingungen werden im Folgenden kurz erläutert. Der Wettbewerb wurde erstmals 2023 ausgetragen und ist die Nachfolgeveranstaltung des Carolo-Cups der Technischen Universität Braunschweig.



4. CAuDri-Challenge

Die CAuDri ist ein jährlich stattfindender Wettbewerb, der Studierenden eine Möglichkeit bietet, autonome Modelfahrzeuge im Maßstab 1:10 zu entwickeln und ihre Fähigkeiten in realitätsähnlichen Szenarien zu demonstrieren [1].

4.1 Regularien

Die Regularien für die CAuDri-Challenge [7] werden regelmäßig aktualisiert und umfassen u.a. folgende Kernelemente:

- Schnellstmögliche Absolvieren verschiedener Szenarien
- Elektromotor muss für Bewegung verwendet werden
- Energie muss aus Batterien stammen, welche zwischen Events ausgetauscht werden dürfen
- Modell im Maßstab 1:10
- Radstand muss mindestens 200mm betragen
- Spurweite muss mindestens 160mm betragen
- Das Fahrzeug darf inklusive Aufbau nicht breiter als 300mm sein
- Die Höhe darf 300mm (von Fahrbahn) nicht überschreiten
- Mindestens eine Achse muss lenkbar sein
- Verkleidungen müssen zur Inspektion schnell abgenommen werden können
- Steuern über RC-Mode muss in Notsituationen möglich sein
- RC-Mode muss über blaues Licht angezeigt werden
- Drei Bremslichter und zwei Blinklichter pro Seite müssen verbaut und funktionsfähig sein

4.2 CAuDri-Events

Die CAuDri-Challenge umfasst bis dato drei verschiedene dynamische Events, die im Folgenden kurz erläutert werden.

4.2.1 Freie Fahrt

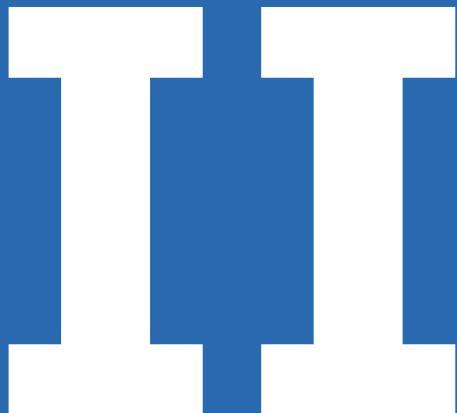
- Größtmögliche Distanz auf zweispuriger Strecke zurücklegen
- Strecke ähnelt ländlicher Straße mit Geraden, Kurven und Kreuzungen
- Fahrbaumanmarkierungen variieren: gestrichelte/durchgezogene Linien
- Verlassen der Spur, falsche Nutzung des Blinkers und Aktivierung des RC-Modus werden bestraft

4.2.2 Hinternisumfahrung

- Erweiterung der Freien Fahrt durch statische und dynamische Hindernisse
- Parkaufgabe
- Hindernisse müssen umfahren werden
- Städtischer Abschnitt mit Geschwindigkeitsbegrenzung und Parkbereich
- Falsches Parken, zu schnelles Fahren und Spurwechsel ohne Blinken werden bestraft

4.2.3 Navigations-Kurs

- Testet Fähigkeit eine unbekannte Strecke zu kartieren
- Orientierungspunkte müssen in der richtigen Reihenfolge angefahren werden
- Unterteilt in Mapping- und Navigations-Phasellus
- Ziel ist es kürzeste Strecke zwischen Orientierungspunkten zurückzulegen



Ausgangssituation

5	Hardware	18
5.1	NUC		
5.2	Spannungsversorgung		
5.3	NUC		
5.4	Motortreiber		
5.5	Verkleidung des Fahrzeuges		
5.6	Anzeige der Desktopumgebung		
6	Software	21
6.1	Dateistruktur		
6.2	NUC		
6.3	Catkin-Make		



5. Hardware

In diesem Kapitel wird die Hardware des Fahrzeugs bei Übernahme des Projektes beschrieben.

5.1 NUC

Der Hauptbestandteil des Modelfahrzeugs, damit dieses Autonom fahren kann ist der Intel NUC. Dieser verarbeitet die Kameradaten und regelt den Lenkwinkel sowie die Motorspannung. Auf dem Intel NUC ist das Betriebssystem Ubuntu installiert.

5.2 Spannungsversorgung

Für die Spannungsversorgung des Fahrzeugs wird ein Akkumulator verwendet. Dieser wurde auch dazu verwendet, den NUC mit Spannung zu versorgen. Da der NUC eine Nenn-Eingangsspannung von 19 Volt benötigt und durch den Akkumulator im ungünstigsten Fall mit elf Volt versorgt wird, besteht hier der Bedarf einer Änderung.

5.3 NUC

Da kurz vor Übernahme des Projektes ein Defekt des im Fahrzeug verbauten NUC aufgetreten ist, wurde als Ersatz ein funktionstüchtiger aus dem Laboraum verbaut. Der letzte für den Test verwendete Softwarestand befindet sich aber noch auf der Festplatte des defekten NUC. Auch mögliche installierte oder konfigurierte Pakete sind nur auf dem defekten NUC nachvollzehbar.

5.4 Motortreiber

Das Modellfahrzeug verfügt über einen Motortreiber der den 3-Phasigen Brushlessmotor antreibt. Dieser wandelt das Signal vom Arduino kommend in eine Motorspannung um. Dieser Motortreiber muss nach Anstecken des Akkus eingeschalten werden. Hierzu gibt es keine Fehlermeldung im Skript, wenn der Motortreiber ausgeschaltet ist. Sollte der Motortreiber nach dem Anstecken des Akkus und das Inbetriebsetzen des Fahrzeugs schon leuchten, ist dieser aktiv. Da allerdings ein Signal innerhalb kürzester Zeit erwartet wird, muss im RC Mode kurz das "Gaspedal" betätigt werden und am Fahrzeug sollten sich die Räder drehen. Danach gibt es keine Probleme mit dem Motortreiber.

5.5 Verkleidung des Fahrzeuges

Das Fahrzeug ist auf der Rückseite mit einer Verkleidung, welche einem Diffusor ähnelt verkleidet. Diese Verkleidung sitzt direkt hinter dem NUC und lässt wenig Platz für die Verwendung von Universal Serial Bus (USB) - und High Definition Multimedia Interface (HDMI) -Steckern. Einzig die Verwendung von speziellen USB - und HDMI -Steckern mit flexiblem Kabel und kurzem Stecker sind möglich. Auch ein Umstecken der Kabel mit montierter Verkleidung ist nicht möglich. Diese Verkleidung wird hier aufgeführt, da sie im Rahmen dieses Projektes überarbeitet wurde. Die alte Verkleidung ist in Abbildung 5.1 dargestellt.

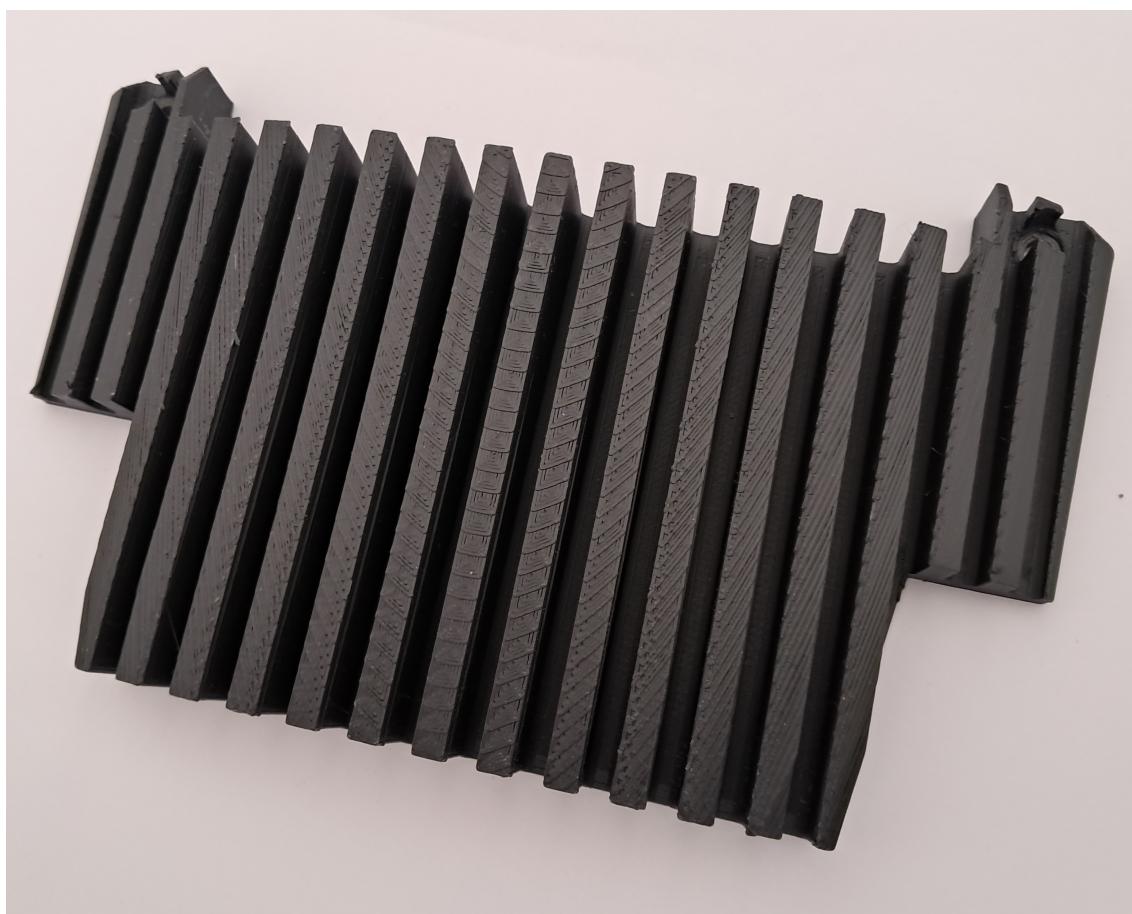
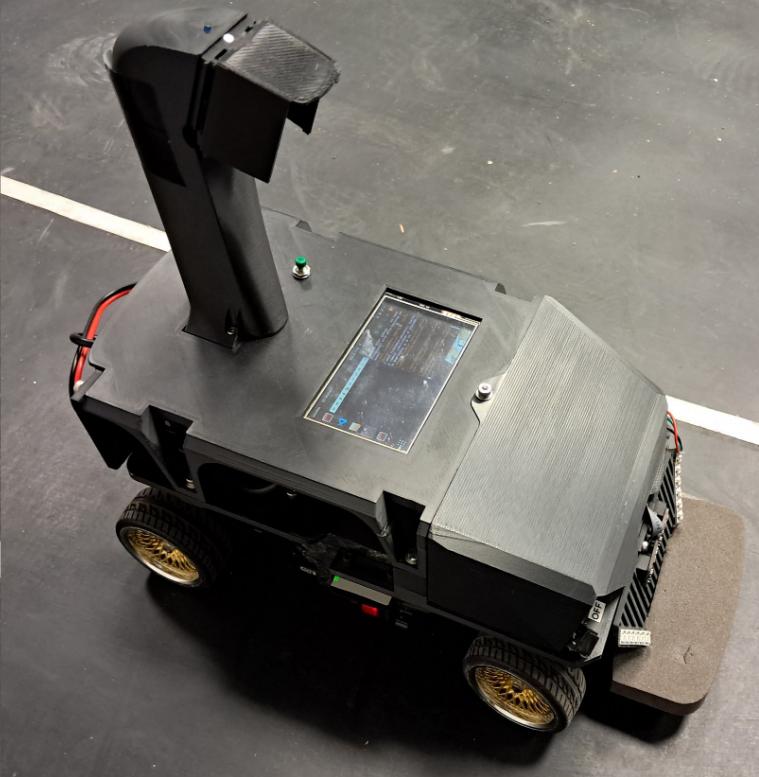


Abbildung 5.1: Rückwärtige Verkleidung des Fahrzeuges (eigene Aufnahme)

5.6 Anzeige der Desktopumgebung

Um beim Testen sowie an Wettkämpfen die Skripte und das Kamerabild zu überwachen, ist ein drahtloser HDMI Sender und Empfänger verbaut. Mit diesem wird das HDMI Signal des Intel NUC an einen externen Monitor übertragen. Hiermit kann das Skript gestartet und überwacht werden. Sollten während des Betriebs Fehler im Skript auftreten, kann dies am Monitor erkannt werden. Dies hilft dabei, das Fahrzeug aus der Ferne schon bevor Unfälle oder Fehlleitungen auftreten in den manuellen Steuermodus zu versetzen.

Neben der Verwendung eines externen Monitors besteht auch die Möglichkeit einen im Gehäuse des Fahzeuges verbauten Monitor anzustecken. Da dieser (interne) Monitor über eine Bildschirmdiagonale von einhundert Millimetern verfügt, eignet er sich in erster Linie zur Oberflächlichen Funktionsprüfung.



6. Software

Die Software lässt sich in zwei Anwendungsbereiche gliedern. Ein Bereich ist für die Simulation des Fahrzeugs in Gazebo konzipiert [6]. Der andere Bereich für die Ausführung der Software auf dem realen Fahrzeug. Die beiden Bereiche teilen sich einige Dateien und Skripte und auch die Startskripte, welche weitere erforderliche Programme ausführen sind ähnlich. Im Folgenden wird nur der für diese Projektgruppe relevante Teil, für die Ausführung auf dem realen Objekt näher erläutert.

6.1 Dateistuktur

Um einen Überblick über die für das reale Fahrzeug relevanten Dateien und Verzeichnisse zu schaffen, werden diese im Folgenden schematisch dargestellt. Die Datei- und Ordnerstruktur ist in Abbildung 6.1 dargestellt. Als Basis des Projektes dient ein Ordner, welcher alle für die Kompilierung und Ausführung benötigten Dateien enthält.

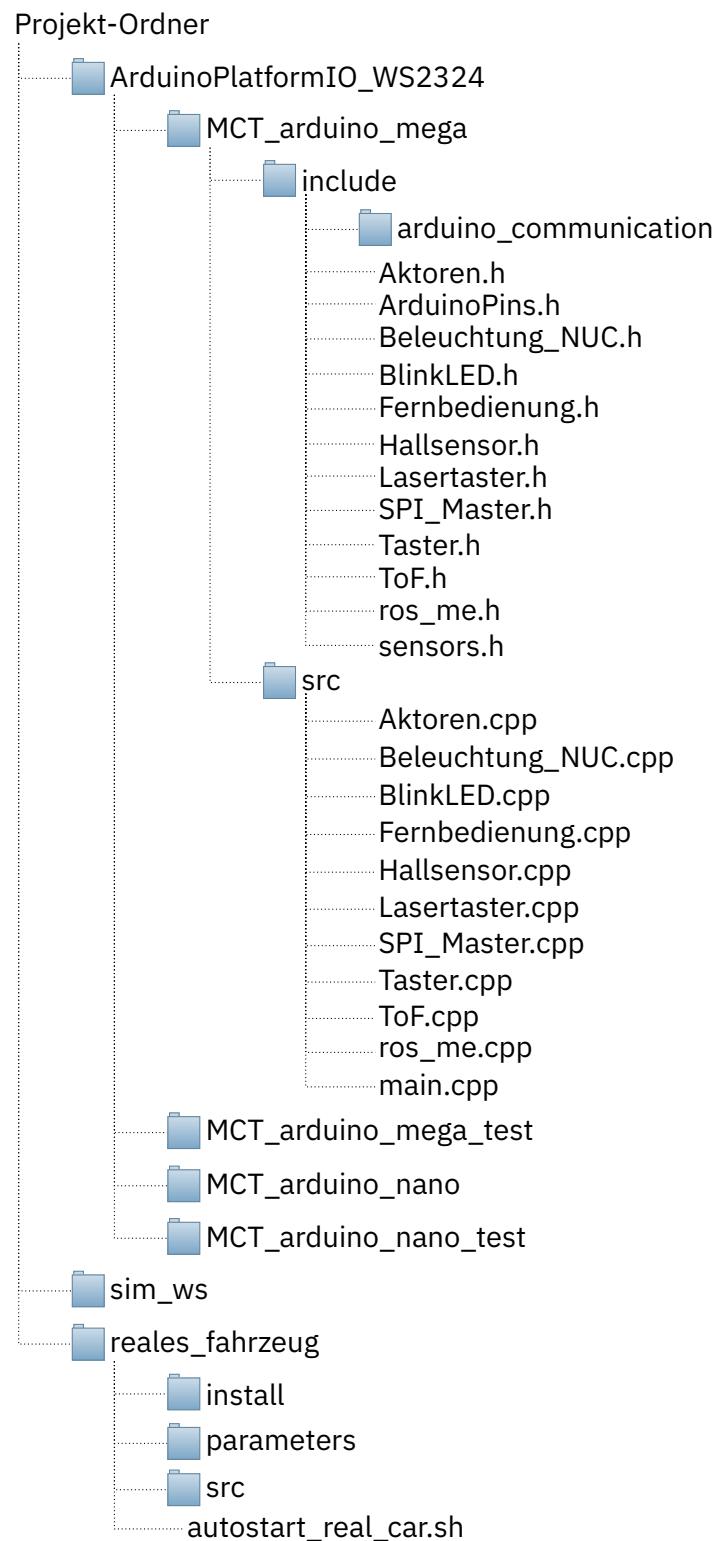


Abbildung 6.1: Vereinfachte Datei- und Ordnerstruktur (eigene Darstellung)

6.1.1 Startskript reales Fahrzeug

Das Skript "autostart_real_car.sh", siehe Abbildung 6.1, startet alle erforderlichen Programme für die Verwendung des realen Fahrzeugs. Das beinhaltet die Ausführung zahlreicher Skripte, welche für die Bilderkennung, Robot Operating System (ROS)-Interaktion, Testausgaben und die Kommunikation mit den Aduinos zuständig sind. Die Fenster, welche jeweils die Ausführung von Skripten darstellen, öffnen sich zeitverzögert und in einer festgelebten Reihenfolge. Einige der Fenster weisen auf Fehler im betreffenden Skript, andere schließen sich aufgrund von Fehlfunktionen direkt nach der Ausführung.

Die Benennung der Fenster lässt nicht auf die ausgeführten Funktionen schließen. Auch das sofortige Schließen der Fenster bei ungeplantem Verhalten macht es unmöglich den aufgetretenen Fehler nachzuvollziehen. Dieses Verhalten birgt Verbesserungspotential. Das Startskript sorgt nicht für eine nachhaltige Ausführung aller Fenster bzw. Skripte, die für einen reibungslosen Ablauf vorgesehen sind. Auch ein aktuelles Kamerabild wird nicht angezeigt.

Nach Einhalten der Wartezeit ist keine autonome Fahrt möglich. Einzig eine Fahrt unter Verwendung des Remote Controlled (RC)-Modus ermöglicht eine Bewegung des Modellfahrzeugs.

6.1.2 MCT_arduino_mega

Der Arduino Mega hat die Aufgabe die Steuerung des Fahrzeugs umzusetzen. Dabei werden Sensordaten ausgelesen und weitergeleitet. Zu den wichtigsten Aufgaben zählen folgende Punkte:

- Auslesen der Sensordaten von Lasertaster, Infrarotsensor und Time of Flight (ToF)-Sensor
- Kommuniziert über Serial Peripheral Interface (SPI) mit dem Arduino Nano (als Master)
- Sendet Sensordaten über ROSSerial an den NUC
- Steuert die Beleuchtungseinstellungen in bestimmten Fahrmodi
- Verarbeitet Sensordaten für das autonome Fahren und erkennt über die Sensorik Hindernisse
- Empfang der Fernbedieungs-Signale

6.1.3 MCT_arduino_mega_test

Um die Funktion des Arduino Mega und der angeschlossenen Peripherie zu überprüfen sind hier Testskripte implementiert. Diese Skripte umfassen die folgenden Prüfungen:

- SPI-Kommunikationstest zwischen Arduino Mega und Arduino Nano
- Funktionskontrolle der ToF-Sensoren und Verifizierung der Daten
- Sicherstellen der Objekterkennung des Lasertasters
- Bodenmarkierungserkennung durch Infrarotsensor

6.1.4 MCT_arduino_nano

Der Arduino Nano ist für die Steuerung der Beleuchtung und die Verarbeitung ausgelagerter Sensoren zuständig. Zu den wichtigsten Aufgaben zählen:

- LED-Streifen für Blinklicht, Bremslicht, etc. ansteuern
- Über SPI empfangene Befehle des Arduino Mega umzusetzen
- Daten der Hallsensoren auslesen

6.1.5 MCT_arduino_nano_test

Diese Testskripte des Arduino Nano umfassen die folgendende Prüfung:

- Funktion der LED-Streifen auf der Vorder- und Rückseite

6.2 NUC

Für die Steuerung und Entwicklung des Projektes kommen kompakte Computer bzw. NUCs zum Einsatz. Diese werden für die Simulation mit Gazebo, aber auch für die Rechnung der Regelungsfunktionen, Linienerkennung und Motoransteuerung kommt ein NUC im Modellfahrzeug zum Einsatz. Das verwendete Betriebssystem ist Ubuntu [4]. Über die in Ubuntu verfügbaren Pakete und Interaktionen mit dem Terminal ist eine gezielte Steuerung realisierbar. Auch Zugriffe auf Schnittstellen, wie Tastatur oder die Kamera, sind für Diagnosezwecke über eine Eingabe im Terminal erfolgreich.

Beim Projektstart waren die Softwarepakete z.T. veraltet und eine Ausführung war dadurch eingeschränkt.

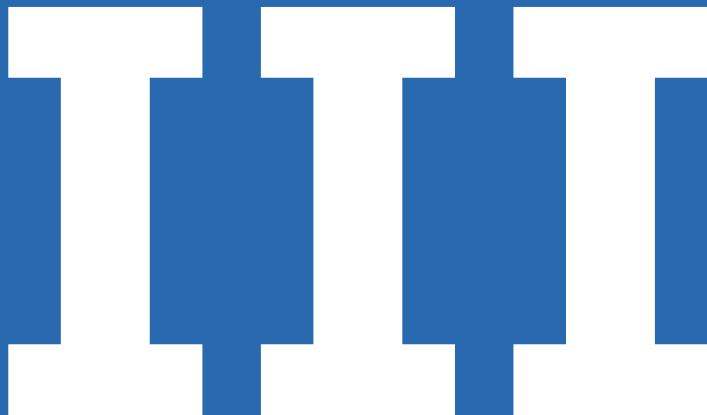
6.3 Catkin-Make

Catkin-Make ist ein Werkzeug um Code in einem Catkin-Workspace zu bauen. Der Workspace muss dabei den Vorgaben des REP128 Standards folgen [8]. Das Kommando zum bauen wird im root-Verzeichnis aufgerufen und ermöglicht bei Übernahme des Projektes eine Kompilierung des Workspaces. Näheres zur Verwendung folgt im Kapitel IV.

Die Voraussetzungen für die Verwendung von Catkin-Make sind:

- Installiertes ROS
- Eingerichteter Catkin-Workspace

Umsetzung



7	Hardware	26
7.1	Bestellte Teile	
7.2	Spannungsversorgung	
7.3	Anzeige der Desktopumgebung	
7.4	Motortreiber	
7.5	Prüf- und Ladekebel mit XT90	
8	Software	35
8.1	Startskript reales Fahrzeug	
8.2	NUC	



7. Hardware

7.1 Bestellte Teile

Folgende Teile mussten für einen reibungslosen Betrieb beschafft werden:

- DC-DC-Wandler
- Steckeradapter von EC5 auf XT90
- Akku Stecker und Buchsen XT90

7.2 Spannungsversorgung

In den folgenden Abschnitten wird die Spannungsversorgung des Fahrzeuges sowie des Intel NUC näher beschrieben.

7.2.1 Akku

Für den Betrieb des Autonomen Fahrzeuges wird ein Akku benötigt. Dieser Akku ist ein Lithium-Ionen-Polymer (LiPo)4 Modellbauakku. Dieser hat eine Nominalspannung von 11,1 Volt und ist von Conrad Electronics. Der Anschlussstecker am Akku hat eine XT90 Buchse. Die kurzzeitige Belastung des Akkus beträgt 400 Ampere und der maximale Ladestrom beträgt 5 Ampere.

- Mindestspannung 10V
- Ladeschlussspannung 12,3V
- nominal spannung 11,1V
- Adapterkabel EC5 auf XT90

7.2.2 DC-DC-Wandler

Der verbaute DC-DC-Wandler transformiert von der Akkuspannung die Nominell bei 11,1 Volt liegt auf die erforderlichen 19 Volt, die der Intel NUC für einen sicheren Betrieb benötigt. Hierbei musste bei der Beschaffung darauf geachtet werden, dass der DC-DC-Wandler folgende Anforderungen erfüllt:

- Erhöhung der Spannung von 11,1 auf 19 Volt
- Konstante Spannung am Ausgang
- Die erforderliche Stromstärke für den Betrieb des Intel NUC

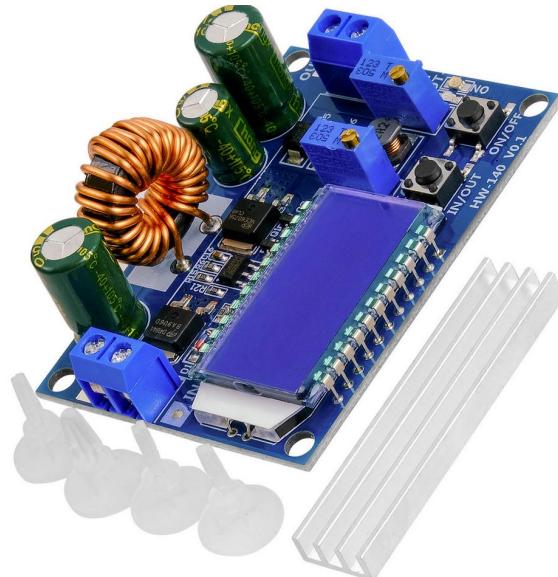


Abbildung 7.1: DC-DC-Wandler [3]

Mit diesen Anforderungen wurde von AZ-Delivery folgender DC-DC-Wandler beschafft(7.1). Dieser DC-DC-Wandler liefert einen konstanten Ausgangsstrom vom 4 Ampere durch den aufgeklebten Kühlkörper. Somit wird eine Leistung von 76 Watt erreicht werden. Das Netzteil des Intel NUC hat eine Leistung von 90 Watt. Nach Erhalt des DC-DC-Wandlers wurde ein Stresstest von 60 Sekunden mit einer CPU-Auslastung von 100% durchgeführt. Bei diesem Stresstest wurde ein maximaler Ausgangsstrom von ca. 3,2 Ampere erreicht. Damit sind noch Reserven vorhanden. Dazu wurde folgender Versuchaufbau wie in Abbildung (7.2) umgesetzt. Zum Schutz des Akkus wurde der Test mit den Labornetzteilen durchgeführt.

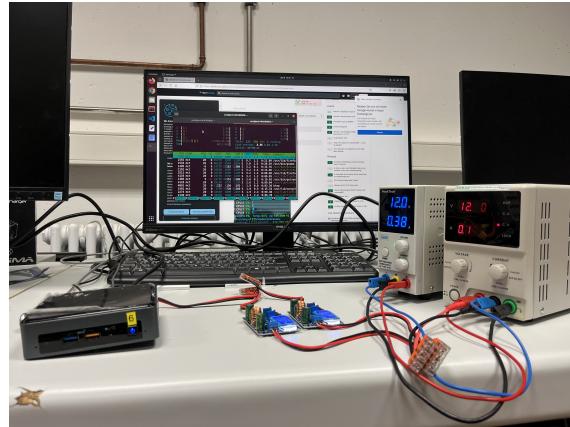


Abbildung 7.2: Versuchsaufbau DC-DC-Wandler (eigene Aufnahme)

7.2.3 Verbau DC-DC-Wandler

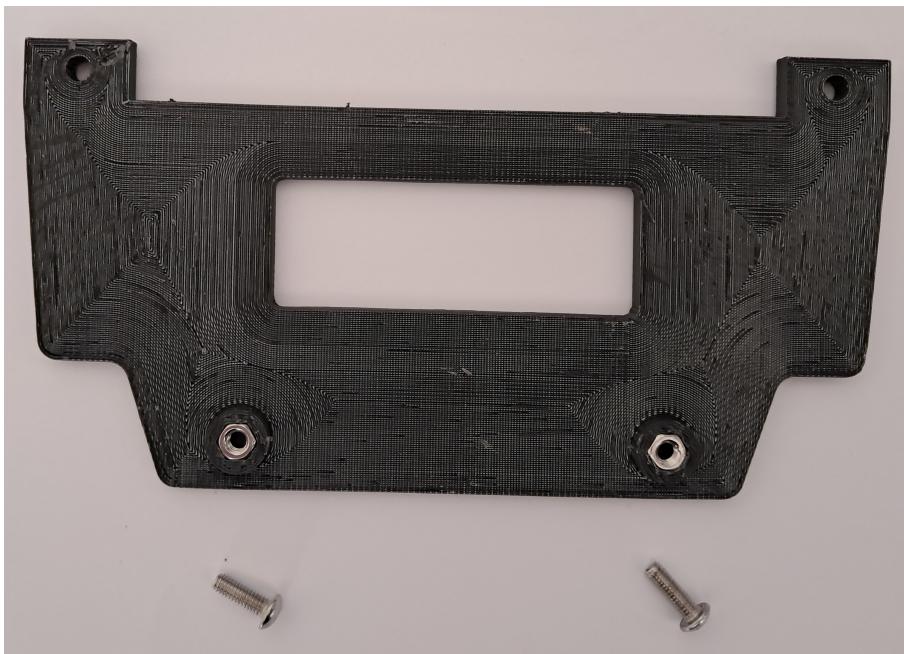
Der DC-DC-Wandler wurde in das Heckbauteil integriert. Somit wird erreicht, dass beim Testen sowie beim Wettbewerb die Akkuspannung abgelesen werden kann. Auf dem LCD-Display kann man sich die Eingangsspannung oder die Ausgangsspannung mit dem jeweiligen Strom anzeigen lassen. Die Funktionen können mittels einer Tasterbetätigung (Taster auf Platine verbaut) umgeschaltet werden. Für die Integration des DC-DC-Wandlers wurde die Rückwärtige Verkleidung, siehe 5.5, neu entworfen bzw. konstruiert. Die neue Konstruktion hat im Vergleich zur vorherigen folgende Verbesserungen:

- Aussparung für HDMI- und USB-Stecker des NUC
- Befestigung des DC-DC-Wandlers (Schraubverbindung)
- Separate an Grundplatte verschraubtes Gehäuse für DC-DC-Wandler
- Gewichtsersparnis und kompaktere Bauweise

Die neue Verkleidung für die Rückseite des Fahrzeugs besteht aus zwei Teilen. Einen Teil bildet die Grundplatte der Verkleidung, siehe 7.3. Diese ist nur 4.5 Millimeter hoch und somit 20 Millimeter dünner als zuvor. Die Grundplatte verfügt über eine Aussparung auf Höhe des NUC, welche die Zugänglichkeit verbessert. Zur Befestigung am Fahrzeug werden die bereits vorhanden Gewinde verwendet.



(a) Oberseite



(b) Unterseite

Abbildung 7.3: Grundplatte der Rückwärtigen Verkleidung (eigene Aufnahme)

Um die Befestigung so modular wie möglich zu halten, wird für den DC-DC-Wandler eine separate Arretierung konstruiert, welche den Wandler aufnimmt und diesen mit Schrauben arretiert. Diese Arretierung ist in Abbildung 7.4 dargestellt. Diese Arretierung schützt den Wandler und macht das Ablesen des Displays, die Verwendung der Potentiometer, sowie die Bedienung der Taster möglich.

Um die Grundplatte mit der Arretierung und den DC-DC-Wandler verschrauben zu können, sind in den Teilen Aussparungen vorgesehen. In diese Aussparungen können Sechskantmutten eingepresst werden. Durch das einpressen wird nur minimaler Bauraum benötigt und ein Herausfallen der Muttern wird verhindert.

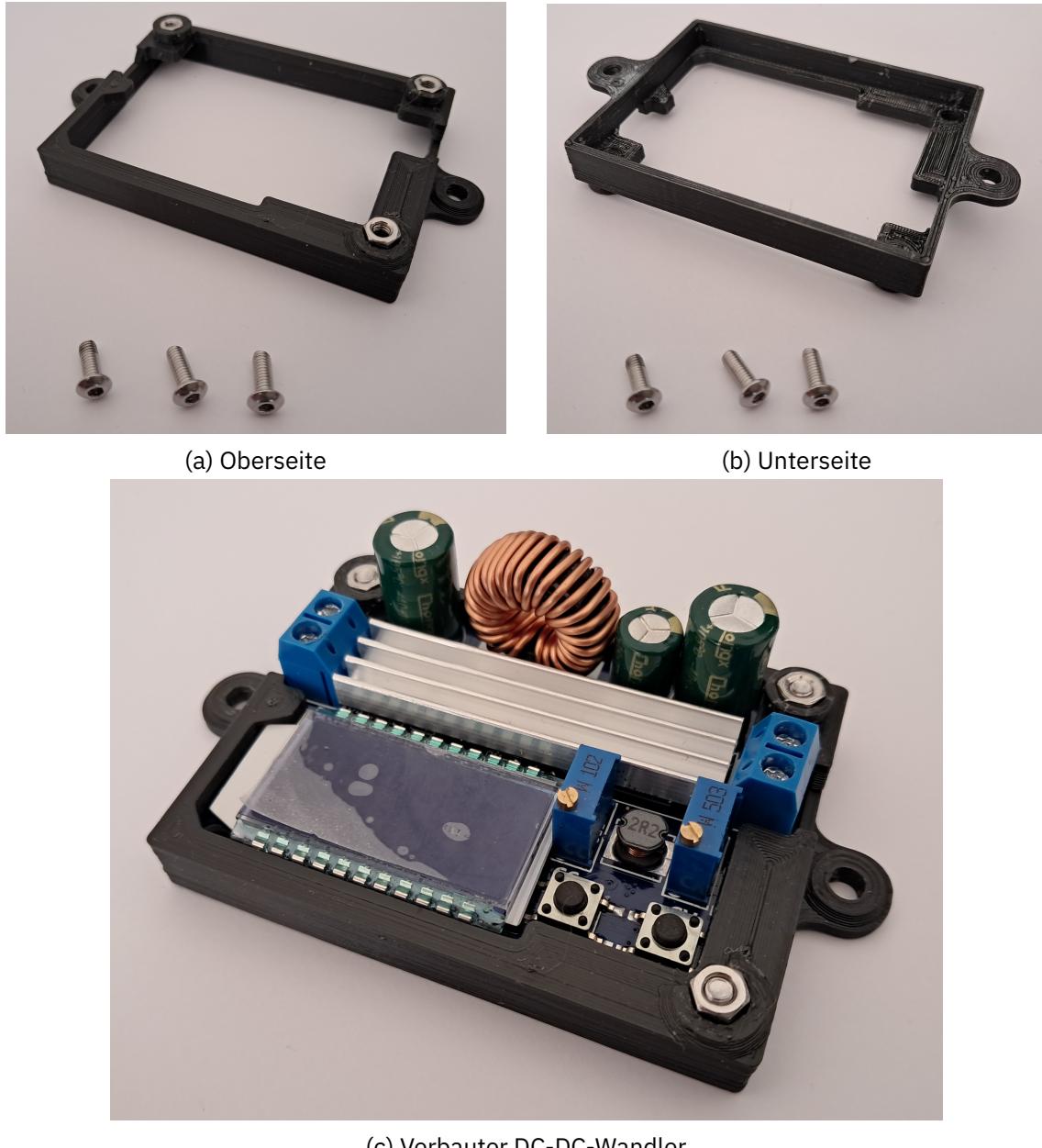


Abbildung 7.4: Arretierung des DC-DC-Wandlers (eigene Aufnahme)

Die Arretierung des DC-DC-Wandlers kann wiederum mit Schrauben an der Grundplatte befestigt werden. Durch diesen Aufbau kann jeder Halter separat demontiert werden und Reparaturen oder Prüfarbeiten sind unkompliziert möglich. Der komplette Aufbau der Verkleidung mit Wandler, Arretierung und Verschraubungen ist in Abbildung 7.5 dargestellt. Der Verbau am Fahrzeug ist Abbildung 7.6 zu entnehmen.

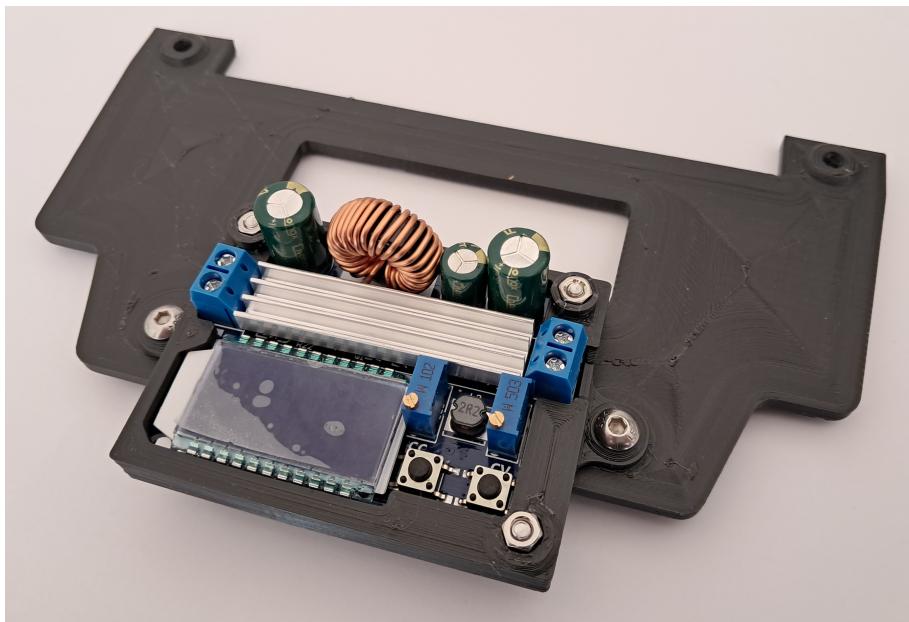


Abbildung 7.5: Rückwärtige Abdeckplatte mit verbautem DC-DC-Wandler (eigene Aufnahme)

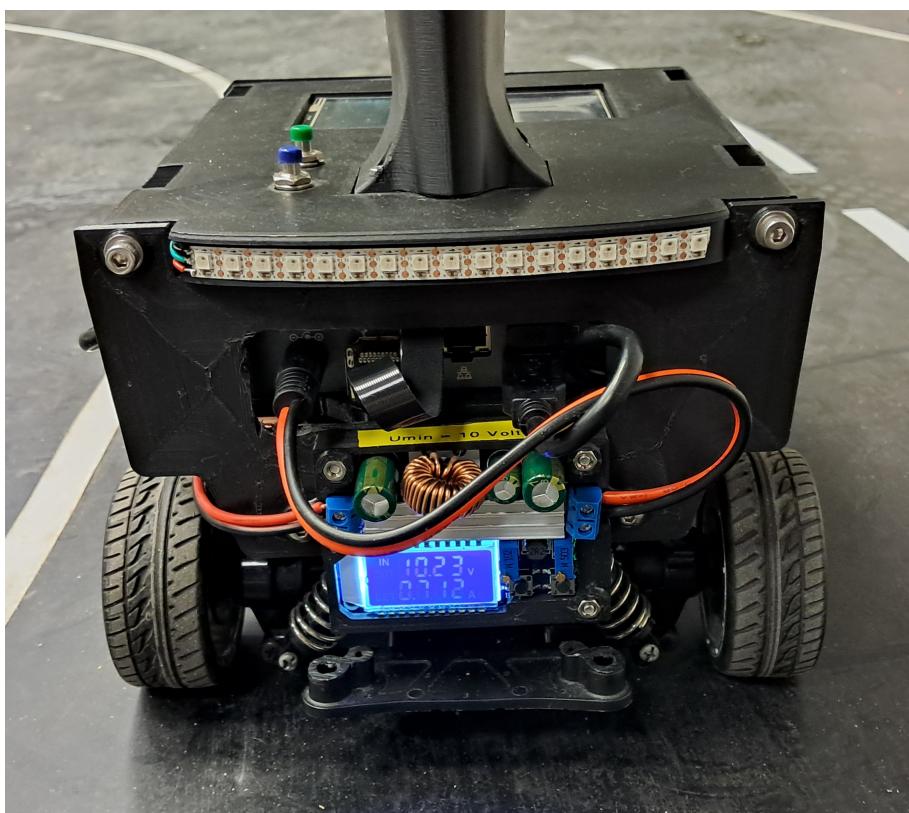


Abbildung 7.6: Montage des DC-Dc-Wandlers am Fahrzeug (eigene Aufnahme)

7.2.4 Spannungswerte

Folgende Spannungswerten sollen für einen reibunglosen Betrieb des Modellfahrzeuges eingehalten werden:

- Minimale Akkuspannung von ca. 10 Volt
- bei vollgeladenem Akku beträgt die Spannung ca. 12,3 Volt
- Versorgung des Intel NUC mit Konstantspannung von 19 Volt (mittels des DC-DC-Wandlers)
- Die Arduinos werden über den Intel NUC mit Spannungen von 5 Volt betrieben über das USB-Kabel

7.3 Anzeige der Desktopumgebung

Durch einen Defekt des Transmitters ist die Verwendung des drahtlosen HDMI-Tranceivers nicht möglich. Der integrierte Monitor konnte in Betrieb genommen werden. Da die Bildschirmgröße des internen Monitors keine zuverlässige Verwendung für den Testbetrieb zulässt, Beispielsweise kann der Text bei einer Eingabe im Terminal nur mit unhandlich großen Symbolen auf dem internen Monitor erkannt werden, wird deshalb auf eine kabelgebundene HDMI-Verbindung zu einem Monitor im Laborraum zurückgegriffen. Hierbei wird ein zehn Meter langes Kabel verwendet, welches bei Fahrt vom Fahrzeug gezogen wird.

7.4 Motortreiber

Der Motortreiber muss nach Verbinden der Spannung separat gestartet werden. Der Start erfolgt über das Drücken des integrierten Tasters. Da der Taster schwer zugänglich ist, wurde dieser mit einem Imbusschlüssel betätigt. Durch die ungünstige Konstruktion und Verbau des Tasters kam es dabei zum Kontakt mit dem Imbusschlüssel und offen liegenden Pins des Treibers. Nach diesem Kontakt war die Funktion des Controllers eingeschränkt.

Um die Funktion wiederherzustellen, wird der Motortreiber zerlegt und die Platine eines Baugleichen Ersatz-Treibers eingelötet. Der zerlegte Treiber ist in Abbildung 7.7 dargestellt.

Nach Tausch des Treibers ist die Funktion wieder gewährleistet. Teilweise kann es jedoch vorkommen, dass der Motortreiber nach Verbinden der Spannungsversorgung ausgeht und nicht wieder eingeschaltet werden kann. Abhilfe schafft hierbei nur ein Trennen und Wiederverbinden der Spannungsversorgung.

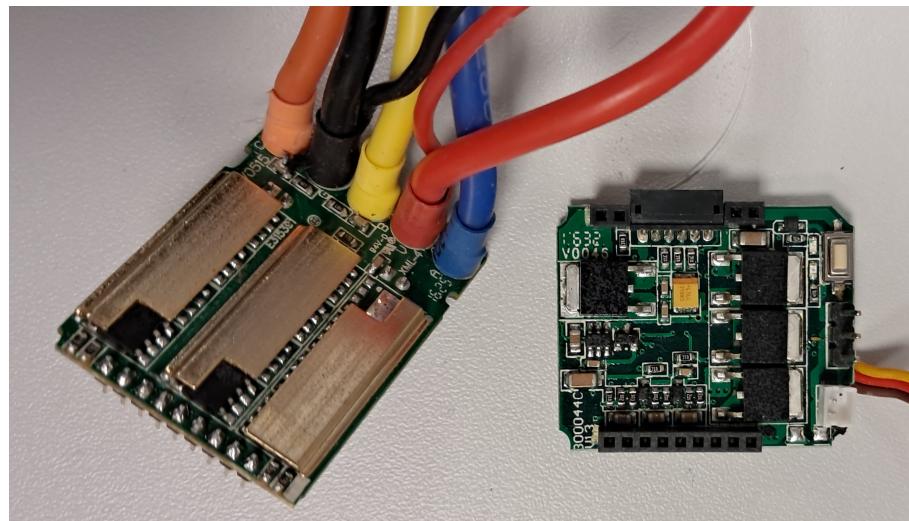


Abbildung 7.7: Zerlegter Motortreiber mit Anschlusskabeln, Endstufe (links), Ansteuerung (rechts), (eigene Aufnahme)

7.5 Prüf- und Ladekabel mit XT90

Um die Akkus für die langen Tests des neuen Softwarepaketes unnötig zu belasten, wurde ein Prüfkabel wie in Abbildung 7.8 hergestellt, dass an ein Labornetzteil angeschlossen werden kann, um die grundelegenden Fahrzeugfunktionen im Stand zu testen. Hierbei ist besonders zu beachten, dass der Intel-NUC über sein eigenes Netzteil betrieben werden muss, da das Labornetzteil hierfür zu wenig Leistung bereitstellt.

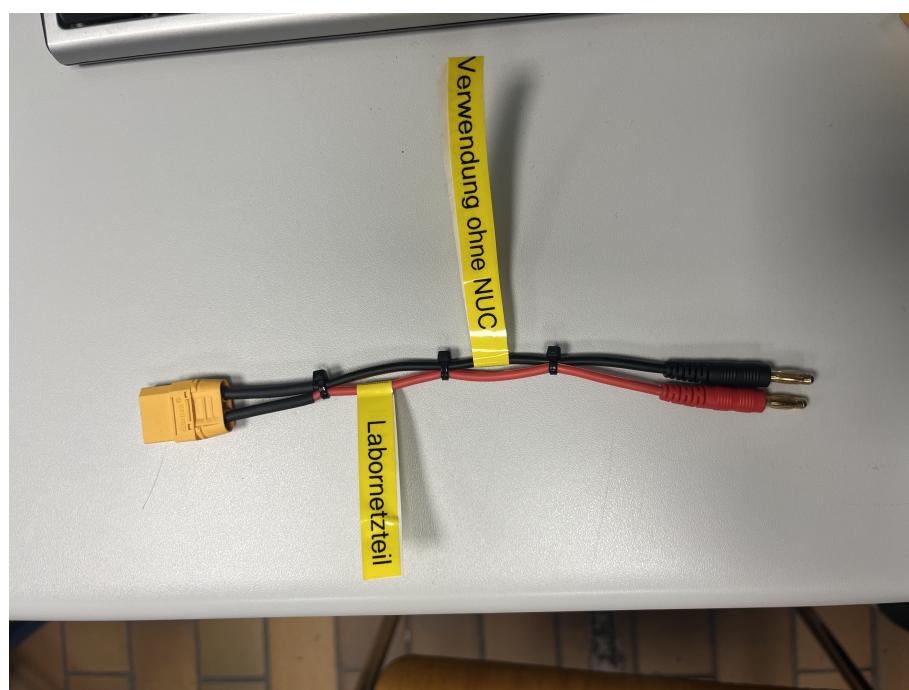


Abbildung 7.8: Prüfkabel für Labornetzteil (eigene Aufnahme)

Da die neuen Akkus einen anderen Stecker haben wurde hierzu auch das passende Ladekabel siehe Abbildung 7.9 hergestellt. Dieser hat die Anschlüsse XT90-Stecker und Bananenstecker. Dieses Kabel muss zum Laden des Akkus verwendet werden. Wie man den Akku lädt, wird in Kapitel 9.1 näher beschrieben.

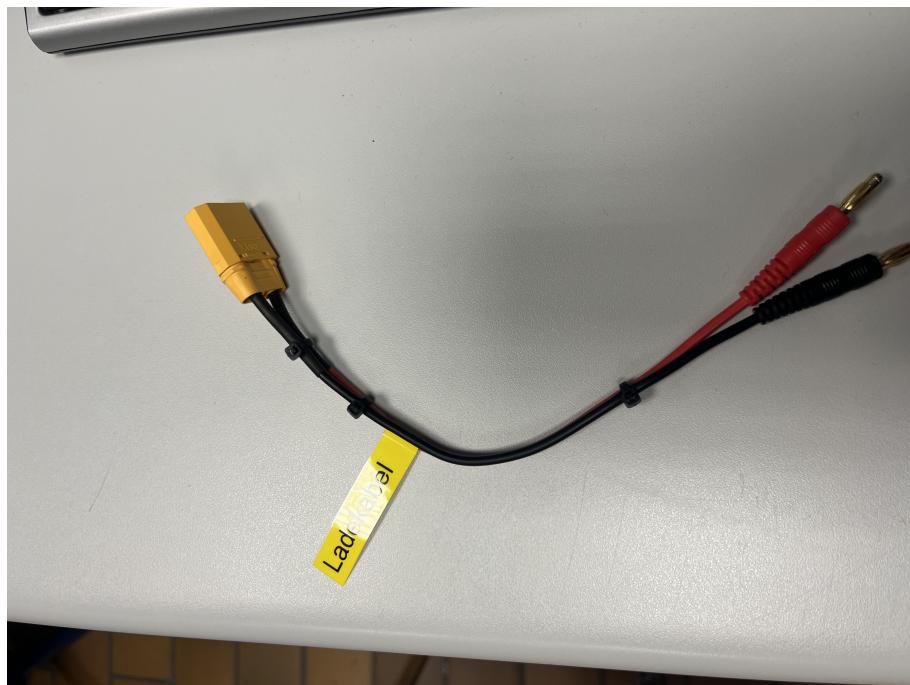


Abbildung 7.9: Ladekabel XT90 (eigene Aufnahme)



8. Software

Damit das Modelfahrzeug autonom fahren kann, wird Software in Form verschiedener Skripte benötigt. Diese werden in den weiteren Abschnitten näher beschrieben.

8.1 Startskript reales Fahrzeug

Um die Funktion des Startskriptes besser nachvollziehen zu können und auftretende Fehler identifizieren erkennen zu können, werden den Fenstern, in welchen die einzelnen Skripte ausgeführt werden sinnvoll benannt, um sie zuordnen zu können. Die Fenster erhalten dabei den Namen des Skriptes, welches sie ausführen. Der Name wird in der Übersicht der Fenster unter Ubuntu und auch im Kopf des Fensters selbst angezeigt. Zusätzlich zur Namensgebung wird das Verhalten geöffneter Fenster nach erfolgter Ausführung bzw. Auftreten eines Fehlers angepasst. Um zu verhindern, dass die Fenster direkt nach Ausführung geschlossen werden, wird ein spezielles Terminal-Profil für die Ausführung festgelegt. Dieses Profil umfasst die folgenden Einstellungen:

- Fenster nach Ausführung geöffnet lassen
- Anzeigelimit/Anzeigezeilen nicht begrenzen

Durch diese Einstellungen bleibt ein Fenster bei einem Fehler geöffnet und die Gesamtheit der Ausgaben ist ersichtlich. Ein exemplarischer Aufruf kann dem Beispiel unten entnommen werden. Bei diesem Aufruf ist der Fenstername "start_rosserial.sh" und das zu verwendende Profil "hold".

```
#!/bin/bash
gnome-terminal --title="start_rosserial.sh"
--window-with-profile hold -- ./start_rosserial.sh
```

8.2 NUC

Am NUC bzw. der Ubuntu-Installation wurden die folgenden Änderungen vorgenommen:

- Updates installierter Pakete (Ubuntu)
- Installation von Testtools und Nutzungsvereinfachungen
- Erstellung eines Profils für das Terminalfenster, welche für ausgeführte Skripte verwendet wird. Weitere Details sind im Abschnitt 8.1 erläutert

IV

Inbetriebnahme

9	Handhabung	38
9.1	Akkuladegerät	
9.2	Auto starten	
9.3	Software starten	
9.4	Testskripte	



9. Handhabung

In diesem Kapitel wird der Umgang mit dem realen Fahrzeug näher beschrieben. Die folgenden Abschnitte sollen eine Anleitung zur Inbetriebnahme des Fahrzeugs und der zugehörigen Peripherie geben.

9.1 Akkuladegerät

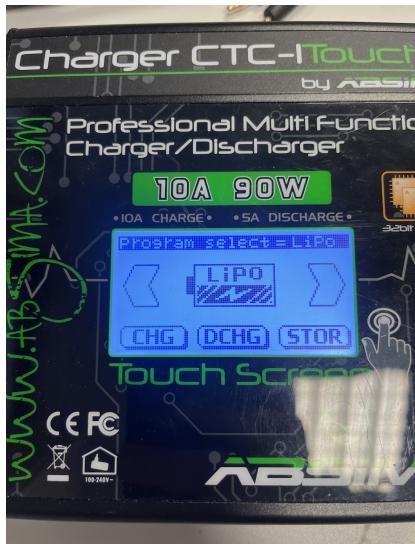
Das Lagegerät mit angeschlossenem Akku ist in Abbildung 9.1 dargestellt. Die Akkus für das Fahrzeug müssen regelmäßig geladen werden, da diese maximal nur 20 Minuten verwendet werden können, bevor der Akku eine Restspannung von ca. 10 Volt erreicht. Zur Ladung des Akkus wird das Ladegerät "Charger CTC-I Touch" verwendet. In der Sicherheitsunterweisung wurde folgende Anschlußreihenfolge erklärt.



Abbildung 9.1: Verbindung Ladegerät mit Akku (eigene Aufnahme)

1. Bananenstecker des Ladekabels an Ladegerät in richtiger Polung anschließen
2. Den Stecker des Ladegerätes in die Steckdosen Steckdosen
3. Den Akku mit dem Ladekabel verbinden und den Balancingstecker am Anschluss 3S anschließen siehe 9.1
4. Im Ladegerät die Option **CHG** auswählen siehe 9.2a
5. Dann müssen die Ladesettings eingestellt werden siehe 9.2b und mit langen Touchdruck auf **Enter** bestätigt werden
6. Wenn die eingestellten Werte wie in der Abbildung 9.2b übereinstimmen, kann der Ladevorgang mit Touchdruck auf **START** begonnen werden siehe 9.2c

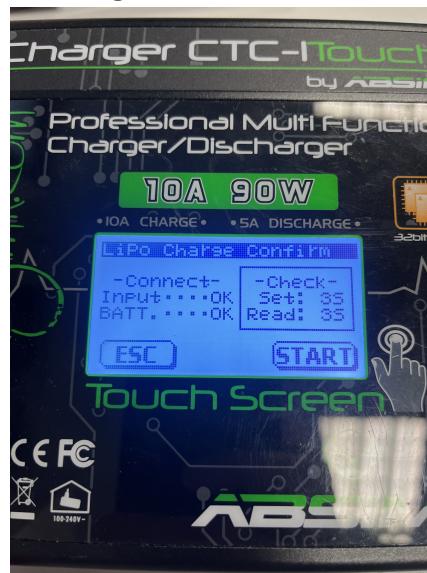
Folgende Einstellungen starten den Ladevorgang siehe Abbildung 9.2.



(a) Startbildschirm Ladegerät



(b) Screen Ladeauswahl



(c) Screen Ladestart

Abbildung 9.2: Vorgehensweise beim Laden der Akkus (eigene Aufnahme)

9.2 Auto starten

Nachdem der Akku am Fahrzeug angeschlossen ist, wird das Fahrzeug mit dem Taster **ON** siehe Abbildung 9.3 gestartet. Hierbei werden die Arduinos, der Intel NUC sowie alle Sensoren und Aktoren mit Spannung versorgt. Sollte der Motortreiber nicht grün blinken, muss das Fahrzeug von dem Akku entfernt und wieder angesteckt werden.

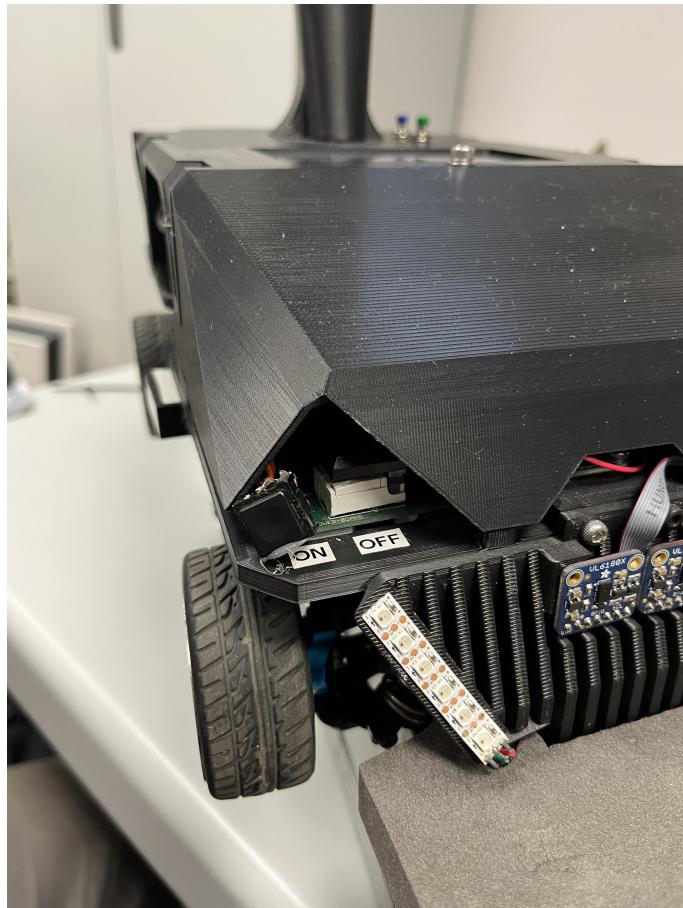


Abbildung 9.3: An- und Aus-Schalter des Fahrzeug (eigene Darstellung)

Wird der Akku des Fahrzeuges entfernt, muss vorher der Intel NUC heruntergefahren werden und mit Taster **OFF** siehe Abbildung 9.3 betätigt werden. Danach kann der Akku vom Fahrzeug getrennt werden.

9.3 Software starten

Um das Fahrzeug autonom fahren lassen zu können, muss das Skript autostart_real_car.sh gestartet werden. Dies wird wie folgt ausgeführt für neue Softwareversion.

- Abspeicherung der neuen Softwarepakete auf dem NUC
- Öffnen des Programmes Terminator (Terminal)
- Zum Abgespeicherten Ordner navigieren mit dem Befehl

cd "Pfad des Ordners"

- Entzippen der Datei mit dem Befehl

unzip "Name des Ordners"

- mittels Terminal in den Ordner navigieren mit dem Befehl

cd "Name des Ordners"

- im Terminal zum Ordner reales_Fahrzeug navigieren

cd "reales_Fahrzeug"

- In diesem Ordner muss erst die Software mit catkin_make gebaut werden

catkin_make

- Nach erfolgreichem Bauen der Software kann diese mit dem Befehl gestartet werden

./ autostart_real_car.sh

Wenn das Autostart Skript ausgeführt wird, öffnen sich mehrere Terminalfenster. Zu beachten ist, dass sich keines der Fenster wieder schließt oder rote Fehlermeldungen angezeigt werden. Allerdings sind angezeigte Warnungen zu ignorieren. Bei einem muss das Passwort noch eingegeben werden. Zusätzlich öffnen sich mehrere Programme. Das eine Programm ist für die Erkennung der Fahrspurlinien und das andere Programm sollte die Karte anzeigen, dies war zum Zeitpunkt unseres Projekt noch nicht umgesetzt. Des Weiteren werden noch weitere Programme geöffnet für verschiedene Ansichten der Kamera.

9.4 Testskripte

Die Testskripte werden zur Prüfung der Aktoren und Sensoren verwendet. Diese müssen für die Arduinos extra geflasht werden. Zusätzlich gibt es ein Testskript für den Intel NUC, mit dem über die Tastatur das Fahrzeug gesteuert werden kann.

V

Fazit

10	Zusammenfassung	44
11	Erforderliche zukünftige Arbeiten	45

Das Ziel dieser Projektarbeit, die Inbetriebnahme des Fahrzeuges, war erfolgreich. Bedingt durch die Komplexität des bereits bei Antritt des Projektes bestehenden Entwicklungsstandes war die Einarbeitung in die Thematik zeitintensiv. Auch die Unterschiede zwischen realem und simuliertem Fahrzeug waren Anspruchsvoll. Funktionen wie die Linienerkennung und das autonome Fahren waren zum Projektstart nur in der simulierten Umgebung getestet und auf dem realen Fahrzeug nicht funktionstüchtig.

Nach erfolgreicher Einarbeitung und in Zusammenarbeit mit den anderen Projektgruppen war es möglich eine lauffähige Software auf dem Fahrzeug auszuführen, welche eine eigenständige Bewegung, wenn auch noch ungerichtet, zu Folge hatte.

Auch die Verwendung von Gitlab [2] hat das Testen vereinfacht und die Verfügbarkeit des aktuellen Softwarestandes für die Projektgruppen sichergestellt.



10. Zusammenfassung

Für die Umsetzung dieser Projektarbeit wurde zunächst eine Bestandsaufnahme des realen Fahrzeuges und der verbauten Komponenten durchgeführt. Das umfasste neben dem Modellfahrzeug auch die benötigte Peripherie zur Inbetriebnahme, wie beispielsweise Akkus und Anschlussmöglichkeiten. Auch die vorhandene Software wurde gesichtet.

Die Software war nicht lauffähig und eine autonome Funktion somit nicht gegeben. Da das defekte Skript in den Zuständigkeitsbereich einer anderen Projektgruppe fällt, wurden zunächst die Funktionen des Arduino geprüft und eine Spannungsversorgung des NUC über den Akku entwickelt und verbaut.

Nach der Bereitstellung eines auf dem realen Fahrzeug ausführbaren Softwarepaketes durch die Regelungs-Projektgruppe konnten die Funktionen erstmals getestet werden. Beim Testen stellte sich heraus, dass die Linienführung nicht mit dem realen Fahrzeug kompatibel ist und nur in der Simulation funktioniert. Ein eigenständiges Fahren, ohne gerichtete Lenkung war trotzdem möglich. Auch das automatische Stoppen vor Hindernissen und das Umschalten in den manuellen Kontrollmodus konnte durch das regelmäßige Verlassen des vorgesehenen Fahrweges ausgiebig getestet werden. Auch der Test von Hindernis- und Schildererkennung war nicht möglich, da der Code nicht in verwertbarer Form vorlag.

Zusammenfassend lässt sich sagen, dass die Hardware des Fahrzeuges erfolgreich in Betrieb genommen wurde, die Software für eine autonome Fahrfunktion aber noch überarbeitet werden muss.

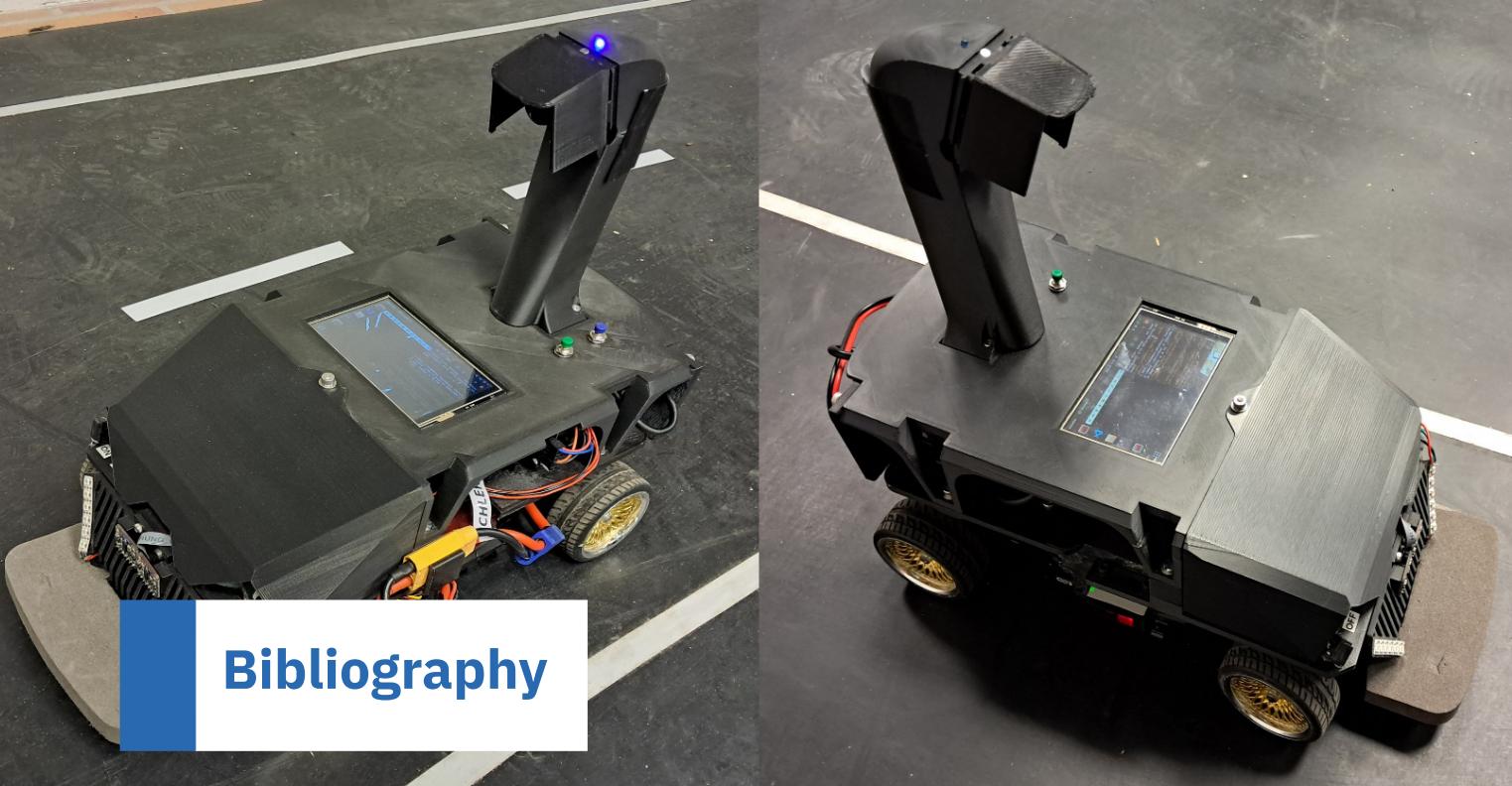
Kurz vor Ende der Projektarbeit wurde im Rahmen von Testarbeiten der Fahrzeugregelung bzw. Linienerkennung der Kameraturm beschädigt. Beim Versuch die Kamera neu auszurichten kam es dabei zu einem horizontalen Riss den Kameraturms über den kompletten Querschnitt. Dieser wurde als Folge von der betreffenden Projektgruppe mit Flüssigklebstoff und faserverstärktem Klebeband instandgesetzt. Diese Lösung ist provisorisch einsatzfähig, sollte aber bei nächster Gelegenheit instandgesetzt werden.



11. Erforderliche zukünftige Arbeiten

Um eine reibunglose Funktion in Zukunft gewährleisten zu können sollten die folgenden Arbeiten noch durchgeführt werden:

- Überarbeitung der Linienerkennung für Erkennung von Linien im realen Betrieb
- Funktionstest der Linienerkennung am realen Fahrzeug
- Funktionstest der Schilder- und Objekterkennung
- Reparatur und Überarbeitung des Kameraturms bzw. der Kameraaufnahme um eine Änderung des Winkels möglich zu machen



Bibliography

- [1] CAuDri-Challenge – caudri-challenge.de. <https://caudri-challenge.de/>. [Accessed 20-02-2025]. 2024 (siehe Seite 15).
- [2] Jonathan M Hethay. *GitLab repository management*. Packt Publishing, 2013 (siehe Seite 43).
- [3] HW-140 DC-DC Buck Boost Converter Step Up/Down, LCD-Anzeige, 3A 5,5V-30V zu 0,5V-30V kompatibel mit Arduino – az-delivery.de. <https://www.az-delivery.de/products/hw-140-buck-boost-converter-mit-anzeige?variant=32922979336288>. [Accessed 26-02-2025] (siehe Seite 27).
- [4] Fabiano B Menegidio u. a. „Dugong: a Docker image, based on Ubuntu Linux, focused on reproducibility and replicability for bioinformatics analyses“. In: *Bioinformatics* 34.3 (2018), Seiten 514–515 (siehe Seite 24).
- [5] Marcus Nolte u. a. „Carolo-Cup—Zehn Jahre automatisiertes Fahren im Maßstab 1: 10“. In: *Sonderprojekte ATZ/MTZ* 22 (2017), Seiten 16–21 (siehe Seite 14).
- [6] Wei Qian u. a. „Manipulation task simulation using ROS and Gazebo“. In: *2014 IEEE International Conference on Robotics and Biomimetics (ROBIO 2014)*. IEEE. 2014, Seiten 2594–2598 (siehe Seite 21).
- [7] Releases · CAuDri/regulations – github.com. <https://github.com/CAuDri/regulations/releases>. [Accessed 20-02-2025]. 2025 (siehe Seite 15).
- [8] REP 128 – Naming Conventions for Catkin Based Workspaces (ROS.org) – ros.org. <https://www.ros.org/reps/rep-0128.html>. [Accessed 26-02-2025] (siehe Seite 24).