



UNIVERSITÄT AUGSBURG

Fakultät für Angewandte Informatik

Praktikum für Produktionstechnik

Entwicklungsprozess eines lenkbaren Fahrzeuges TEGGLA

vorgelegt von:	Jonas Wilfert: 1541778 Niklas Paprotta: 1543350 Johannes Evertz: 1463672 Marcel Khodabakhsh: 1333430
eingereicht am:	30.07.2019
Studiengang:	B.Sc. Ingenieurinformatik
Anfertigung am Lehrstuhl:	Produktionsinformatik
	Fakultät für Angewandte Informatik
Verantwortlicher Professor:	Prof. Dr. Ing. Johannes Schilp
Wissenschaftliche Betreuer:	M.Sc. Michael Aumüller M.Sc. Fabian Herzer M.Sc. Shuang Lu M.Sc. Paul Haase

Kurzfassung

Dieser Bericht soll als Dokumentation des Entwicklungsprozesses und der technischen Komponenten des TEGGLA dienen. Hierbei handelt es sich um ein im Rahmen des Praktikum für Produktionstechnik entwickelten Fahrzeugs, welches durch die besonderen Mecanum-Räder mehr Freiheitsgrade hat als ein konventionelles Automobil.

Inhaltsverzeichnis

1 Einleitung & Motivation	1
2 Anforderungen	2
2.1 Anforderungsliste	2
2.2 Lastenheft	3
3 Planung	6
3.1 Entwürfe	6
3.1.1 Seggway	6
3.1.2 Weggchair	7
3.1.3 TEGGLA	8
3.2 Morphologischer Kasten	9
3.3 Online Bestellungen	11
3.4 Verwendete Technologien	11
3.4.1 Git	12
3.4.2 Creo	12
3.4.3 PlatformIO	13
4 Entwicklung der Schlüsselemente	15
4.1 Übersicht des gesamten Modells	15
4.2 Schaltplan	16
4.3 BMS und Laden	17
4.4 Mecanum	17
4.5 Planetengetriebe	19
4.6 ESP-32 vs. ESP-8266	22
4.7 User-Interface	24
4.7.1 Java (obsolete)	24
4.7.2 HTML5 und Controller-Anbindung	25
4.7.3 Protokoll	28
4.8 Steuerung per (XBox) Controller	28

4.9	PLA vs. TPU	29
4.10	Eierhalter	29
4.11	Leichtbau	30
5	Zukünftige Entwicklungsmöglichkeiten	31
5.1	Regler	31
5.2	Verbesserungen	32
5.2.1	Hardware	32
5.2.2	Software	33
6	Zusammenfassung	34
6.1	Fazit	34
6.2	Errungene Erfahrung	35
A	Quellcode	37
A.1	ESP32	37
A.2	Webpage	50
A.3	Java	61
B	Technische Zeichnungen	87
Literaturverzeichnis		102
Listings		103
Abbildungsverzeichnis		104
Tabellenverzeichnis		106

1 Einleitung & Motivation

2 Anforderungen

Für den ersten Meilenstein, dessen Abgabe am 12.05.2019 war eine Anforderungsliste und ein Lastenheft abzugeben. Eine Liste mit 18 Anforderungen wurde noch am gleichen Tag erstellt und im Laufe der nächsten Woche ausformuliert.

Die Entscheidung, die Abgabefristen und Termine in das Lastenheft aufzunehmen, wurde getroffen da sie elementar für das Bestehen des Projektes sind, auch wenn sie schlussendlich nicht relevant für das Fahrzeug sind.

2.1 Anforderungsliste

w4 > w3 > w2 > w1		Anforderungsliste	Erstellt am 7.5.	
Lfd.	F/W	Anforderung	Änderung	Verantwortlich
1	F	Fahren		Marcel
2	F	Lenken können		Jonas
3	F	Mindestdistanz 4 Meter		Johannes
4	F	Ei transportieren können		Niklas
5	F	Vorpräsentation: 28.5.		Marcel
6	F	Finales Konzept und CAD Modell: 12.6.		Jonas
7	F	Abgabetermin: 23.7.		Johannes
8	F	Abgabe Bericht: 30.7.		Niklas
9	W4	Ei unbeschädigt transportieren		Marcel
10	W3	Leichtbauweise		Jonas
11	W3	Ressourceneffizienz		Johannes
12	W3	Kosteneffizienz		Niklas
13	W3	innovatives Design		Marcel
14	W2	Wartbarkeit		Jonas
15	W2	externe Steuerung oder autonomes Fahren		Johannes
16	W1	schnelles Fahren		Niklas
17	W1	fahren durch unwegsames Gelände		Marcel
19	W1	Modularisierung/Erweiterbarkeit		Johannes

Tabelle 2.1: Anforderungsliste

2.2 Lastenheft

Fahren (F)

Das finale Fahrzeug muss in der Lage sein, von Startpunkt 1 zu einem definierten Endpunkt 2 fahren zu können. Die Art und Weise der Fortbewegung steht dabei nicht im Mittelpunkt, lediglich die fehlerfreie Funktionalität muss gegeben sein.

Lenken können (F)

Da Fahren allein nicht ausreicht, um die S-förmige Strecke zu absolvieren, muss das Fahrzeug ebenfalls lenkbar sein. Hierfür ist es nötig mit ausreichender Genauigkeit lenken zu können, um nicht den Bereich der Strecke zu verlassen.

Mindestdistanz 4 Meter (F)

Das Fahrzeug muss in der Lage sein, beladen mit einem Ei, eine Distanz von mindestens 4 Metern zu überwinden, damit das Ei es auch bis zum Zielpunkt schafft und nicht auf halber Strecke stehen bleibt.

Ei transportieren können (F)

Es wird gefordert, dass das finale Fahrzeug in der Lage ist, ein Ei über eine gewisse Distanz transportieren zu können.

Ei unbeschädigt transportieren (W4)

Bei dieser Anforderung handelt es sich um eine optionale Anforderung, welche jedoch eine hohe Priorität erhalten hat (W4). Das Ei sollte also am Ende des Projekts unbeschädigt von Startpunkt 1 zu Endpunkt 2 transportiert werden können, um diese Anforderung zu erfüllen.

Kosteneffizienz (W3)

Da das Budget dieses Projektes begrenzt ist, ist darauf zu achten, dass das Fahrzeug sehr kosteneffizient konstruiert und produziert wird.

Ressourceneffizienz (W3)

Aufgrund der Vermeidung zusätzlicher Kosten, der limitierten Baumaterialien und der Vermeidung eines größeren Zeitaufwandes soll das Projekt möglichst ressourceneffizient geplant und umgesetzt werden. Bei dieser Anforderung handelt es sich um eine optionale Anforderung, welche jedoch eine hohe Priorität erhalten hat (W3).

Leichtbauweise (W3)

Diese Anforderung ist ebenfalls optional mit recht hoher Wichtigkeit (W3). Bei der Entwicklung und Konzipierung des Fahrzeugs sollte auf ein möglichst geringes Gewicht geachtet werden.

Innovatives Design (W3)

Bei dieser Anforderung handelt es sich um eine optionale Anforderung, welche eine relativ hohe Priorität erhalten hat (W3). Zusätzlich zur grundlegenden Funktionalität des Fahrzeugs wird auch noch Wert auf das optische Design des Fahrzeugs gelegt sowie die Art und Weise der Fortbewegung bewertet. Hierbei werden insbesondere kreative und einzigartige Denkansätze wertgeschätzt.

Externe Steuerung oder autonomes Fahren (W2)

Um das Fahrzeug sicher durch den Parcours zu navigieren, benötigt es entweder eine externe Steuerung (z.B. Fernbedienung + IR Empfänger, Pfeiltasten der Tastatur + WLAN-Verbindung etc.) oder es muss in der Lage sein, völlig autonom (durch Sensorik und „künstliche Intelligenz“) den Parcours zu meistern.

Wartbarkeit (W2)

Bei dieser Anforderung handelt es sich um eine optionale Anforderung, welche jedoch eine relativ niedrige Priorität erhalten hat (W2). Durch Achtung auf Wartbarkeit und Reparaturfreundlichkeit des Fahrzeugs kann besser auf unvorhergesehene Fehlfunktionen reagiert werden.

Modularisierung/Erweiterbarkeit (W1)

Für die Aspekte „Kosteneffizienz“, „Ressourceneffizienz“ und „Wartbarkeit“ spielt die modulare Entwicklung und Konstruktion eine große Rolle. Durch die Modularisierung

kann das Fahrzeug leichter überarbeitet, erweitert und gewartet werden. Des Weiteren ist eine zukünftige Erweiterung des Projekts dadurch leichter umsetzbar. Bei dieser Anforderung handelt es sich um eine optionale Anforderung, welche eine niedrige Priorität erhalten hat (W1).

Schnelles Fahren (W1)

Da in der Logistikbranche Zeitdruck ein wesentlicher Faktor ist, besteht der Wunsch, dass das Fahrzeug die zurückzulegende Strecke in möglichst kurzer Zeit bewältigt und sich deshalb mit hohem Tempo fortbewegt.

Fahren durch unwegsames Gelände (W1)

Das finale Fahrzeug kann sich im Optimalfall nach dem Abschluss des Projekts auch auf unwegsamen Geländen effizient und zielsicher fortbewegen. Diese Anforderung ist nicht explizit durch den Auftraggeber vorgegeben, würde jedoch einen Mehrwert des Produkts erzielen. Daher handelt es sich um eine optionale Anforderung, welche eine niedrige Priorität erhalten hat (W1).

3 Planung

Nachdem das Lastenheft abgegeben wurde war der nächste Meilenstein der Abschluss der Konzeptphase, zu dem eine kurze Präsentation über Gesamt- und Teilfunktionen, Lösungsprinzipien, Morphologischer Kasten gehalten werden sollte. Außerdem sollte die Präsentation pro Gruppenmitglied ein vorläufiges Konzept mit Freihandskizze enthalten.

3.1 Entwürfe

Da die Entscheidung für das finale Konzept sehr früh gefallen ist, wurde sich darauf geeinigt, dass zwei Gruppenmitglieder dieses Konzept bearbeiten, und dabei viel tiefer ins Detail gehen. Die übrigen zwei Gruppenmitglieder haben jeweils eine eigene Idee behandelt, blieben dabei aber deutlich oberflächlicher.

Insgesamt sind dabei die drei im Folgenden vorgestellte Konzepte entwickelt worden.

3.1.1 Seggway

Das Konzept des (S)Eggways (Abb. 3.1) ist, wie der Name schon verrät, inspiriert durch die immer populärer werdende Fortbewegungsart des Segways. Dieses Konzept würde viele Vorteile mit sich bringen: Das Gesamtgewicht des Seggways ist relativ gering, da es keine massiven Bauteile gibt und technische Bauteile wie das Breadboard gleichzeitig mehrere Zwecke erfüllen (Stabilisierung und Verbindung).

Zusätzlich könnte der Seggway einen sicheren Ei-Transport durch eine passende Regelung sicherstellen, da diese ermöglicht, dass auch bei starken Steigungen und unebenen Strecken immer das Gleichgewicht beibehalten beziehungsweise wiederhergestellt werden kann. Das gelingt durch das Prinzip des inversen Pendels.

Die Steuerung des Fahrzeugs erfolgt über das eingezeichnete Bluetooth-Modul, welches die erhaltenen Daten an den Arduino weitergibt. Der Arduino verarbeitet die Daten und steuert anschließend (falls nötig) die zwei Stepper-Motoren an.

Die Stepper-Motoren sind für diese Anwendung ideal geeignet und können auch direkt ohne Getriebe mit den zwei Reifen verbunden werden. Das Gyroskop dient dem Messen von Geschwindigkeit, Beschleunigung und aktuellem Winkel.

Der Eierhalter ist genau so konzipiert wie in dem realisierten TEGGLA Modell, nämlich werden mehrere einzelne Segmente verbunden um möglichst wenig Gewicht mit möglichst viel Stabilität zu verbinden.

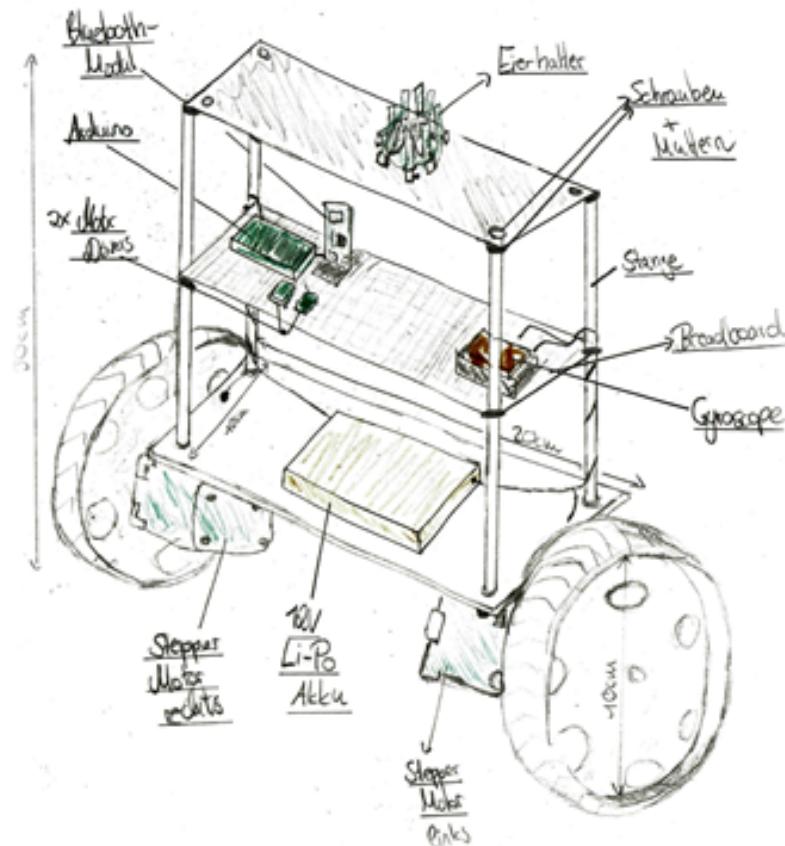


Abbildung 3.1: Skizze Seggway

Siehe Tabelle 3.1 für eine genaue Aufzählung der benötigten Komponenten. Alle restlichen Bauteile wie die Stangen, Ebenen und Reifen könnten problemlos via 3D-Druck mit PLA hergestellt werden.

3.1.2 Weggchair

Beim Weggchair ist die Namensgebung leider ein wenig misslungen. Der Rest vom Konzept wäre mit den vom Lehrstuhl bereitgestellten Materialien recht leicht zu

Bezeichnung	Anzahl
Bluetooth-Modul	1
Arduino	1
Motor Driver	2
Stepper Motor	2
12V Li-Po Akku	1
Gyroscope	1
Breadboard	1
Schraube + Mutter	8

Tabelle 3.1: Stückliste Seggway

realisieren gewesen. Deswegen wäre es der Plan B gewesen, falls das präferierte Konzept nicht realisiert werden kann.

Das Konzept orientiert sich, wie der Name andeuten soll, recht nahe an einem Rollstuhl. Die gesamte Elektronik befindet sich in einer Zwischenebene unter der "Sitzfläche". Gelenkt wird der Weggchair indem sich beide Motoren verschieden schnell drehen. Wie in der Skizze (Abb. 3.2) zu sehen, ist das "Stützrad" eher eine "Stützkugel", die in alle Richtungen über den Boden gleiten kann. Statt einer Kugel wären auch eine oder zwei drehbar gelagerte Rollen denkbar gewesen, ähnlich wie bei einem Einkaufswagen oder natürlich dem originalen Rollstuhl.

Da die verwendeten DC-Motoren unter Vollast über 7000 Umdrehungen schaffen, dafür aber weniger Drehmoment haben, wäre – wie in der finalen Idee – ein Planetengetriebe innerhalb der großen Räder zum Einsatz gekommen.

Das Ei wäre beim Weggchair an der Stelle befestigt, wo normalerweise der Rollstuhlfahrer sitzt. Dafür wäre eine Federung und Halterung an dieser Stelle gewesen. Vermutlich wäre ein mit Watte oder einem ähnlichen elastischen, polsternden Material ausgestatteter Eierbecher zum Einsatz gekommen.

3.1.3 TEGGLA

Da sich die nächsten drei Kapitel nur mit dem finalen Konzept auseinandersetzen, wird an dieser Stelle nur ganz kurz auf das Konzept des TEGGLA (Abb. 3.3) eingegangen, damit die nachfolgenden Kapitel nachvollziehbar sind.

Das besondere Feature beim TEGGLA sind die omnidirektionalen Räder, auch als Me-canum bezeichnet, die neben dem normalen Vorwärtsfahren auch Seitwärtsbewegungen

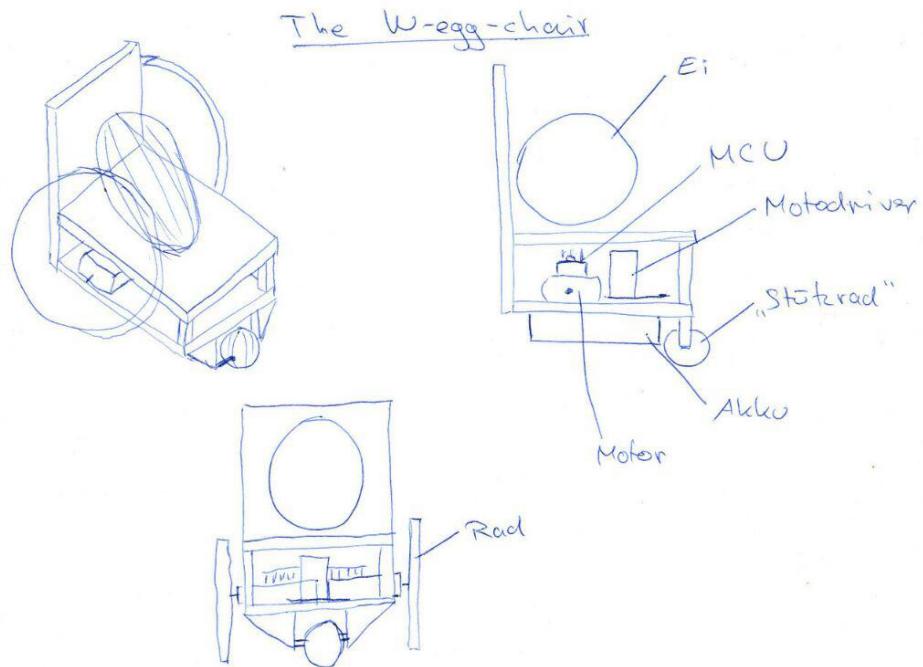


Abbildung 3.2: Skizze Weggchair

und Rotation zulassen. Diese drei Freiheitsgrade lassen sich auch beliebig kombinieren. Dadurch ist in der Ebene jede denkbare Richtung befahrbar. Dazu sind allerdings vier Motoren notwendig.

Daher wächst die Elektronikteileliste wie folgt: Für die zwei Extramotoren wird eine zusätzliche H-Brücke benötigt. Leider hat das bereitgestellte ESP-8266 nicht genug Pins für die zusätzliche Elektronik, weswegen zum ESP-32 upgegraded werden musste. Es folgen noch viel mehr Details, aber um die folgenden Kapitel zu verstehen, muss noch gesagt werden, dass der TEGGLA in der Entwicklungsphase noch Omni-Move hieß. Beide Namen sind im Folgenden also synonym.

3.2 Morphologischer Kasten

Mithilfe des Morphologischen Kasten (Abb. 3.4) lassen sich die benötigten Komponenten auf eine einfach ersichtliche Weise vergleichen.

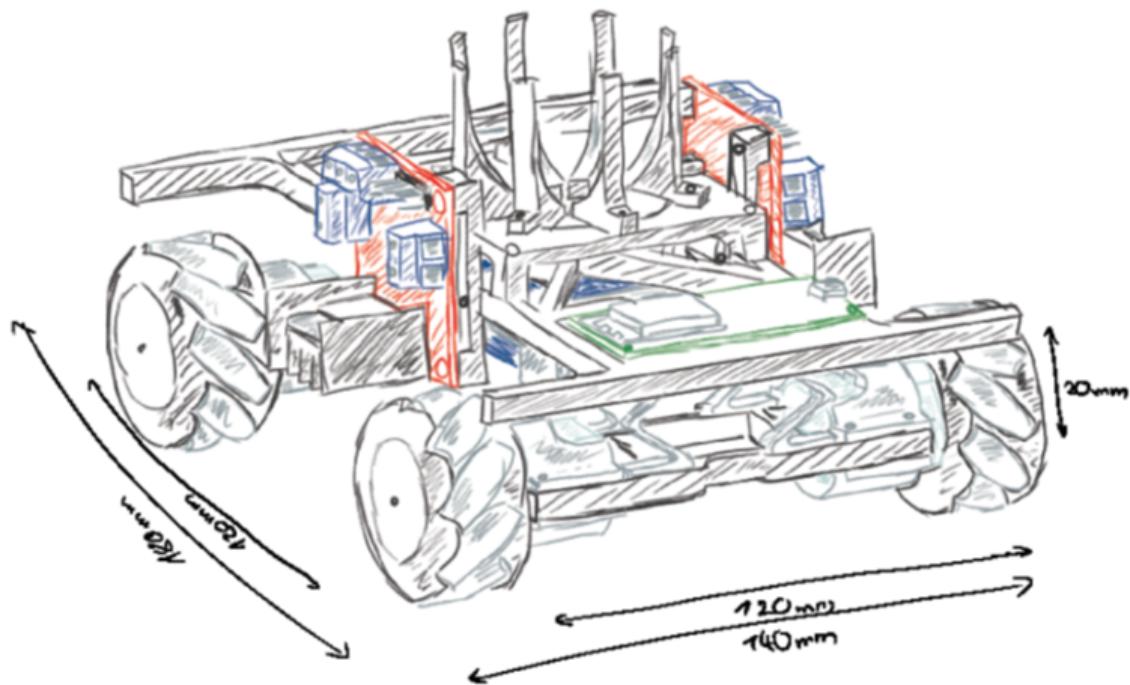


Abbildung 3.3: Skizze TEGGLA

	OmniMove	(S)eg(g)way	W-egg-chair	
	Variante A	Variante B	Variante C	Variante D
Antrieb	Servo	DC-Motor	Step-Motor	
Lenkung	Lenkachse	Einzelne gesteuerte Motoren		
Räder	Mecanum	Rad	Kette	Spinnenbeine
Getriebe	Keins	Planetengetriebe	Gearbox	
Controller	ESP32	ESP8266	Arduino	
Akku	LiPo 2S	Battery	LiPo 3S	
Ladetechnik	Keiner	BMS + 9V Netzteil	BMS + 5V USB	
Steuerung	Laptop	Spiele-Controller	Handy	Autonom
Federung	Keine	3D-Druck	Feder-Dämpfer	
Lage-Sensorik	Keine	Gyroskop	Pendel	Encoder
Material	PLA	PLA + TPU	Holz	ABS
Verbindung	Schrauben	Pins		

Abbildung 3.4: Morphologischer Kasten

Der Seggway bräuchte, um sich aufrecht zu halten auf jeden Fall Step-Motoren statt der DC-Motoren. Dadurch könnte das Getriebe gespart werden. Allerdings braucht der

Seggway auf jeden Fall ein Gyroskop (und eine gute Regelung) um aufrecht stehen zu bleiben und zu fahren. Bei der Steuerung setzen alle Konzepte auf einen Spiele-Controller, da es keine leichtere und intuitivere Steuerung gibt.

Der Weggchair würde, wie oben angemerkt, mit den vom Lehrstuhl bereitgestellten Materialien auskommen. Für die Übersetzung wäre ein Planetengetriebe im Rad zuständig.

Der TEGGLA braucht ein ESP-32, da das ESP-8266 zu wenig Anschlüsse hat. Das Highlight, die Räder, sind bei diesem Konzept die Mecanum-Wheels. Als Energiespeicher kommt ein zweizelliger LiPo in Kombination mit einem BMS (Battery Management System) zum Einsatz. Hier ist der Einsatz des Spiele Controllers besonders wichtig, da damit alle drei Achsen (X-Achse, Y-Achse und Rotation) analog gesteuert werden können. Für die Sicherheit des Eies sorgt eine Federung aus TPU. Zum TEGGLA werden alle Teile und Entscheidungen in diesem Bericht noch näher beleuchtet.

3.3 Online Bestellungen

Um das Fahrzeug nach dem Praktikum behalten zu können, war eine Voraussetzung, dass nur Teile verbaut werden, die nicht Eigentum des Lehrstuhls sind. Aus diesem Grund wurden die nötigen Bauteile bei unterschiedlichen Onlineshops herausgesucht und bestellt. Hierbei fiel die Entscheidung auf Pollin, einem deutschen Elektronik-Händler und AliExpress, einem chinesischen Großhändler. In der ursprünglichen Planung wurden die Kosten pro Fahrzeug auf circa 20€ – 25€ überschlagen. In der finalen Bestellung beliefen sich die Kosten auf insgesamt etwa 32€.

Da die Bauteile in Sammelbestellungen für insgesamt 4 Fahrzeuge getätigt wurden, verteilten sich die Versandkosten auf die Fahrzeuge, wodurch sich der Preis nach unten hin anpasste.

Siehe Tabelle 3.2 für eine genaue Aufteilung der Kosten.

3.4 Verwendete Technologien

Damit alle Teammitglieder an dem Projekt arbeiten können, musste sich vor dem Beginn auf die Software geeinigt werden, die verwendet wird. Diese Entscheidungsfindung wird in diesem Kapitel genauer betrachtet.

Artikel	Stk	€/Stk	€	Laden
Netzteil 9V 1A	1	0,95	0,95	Pollin
DC Motor	4	0,95	3,80	Pollin
2S LiPo	1	9,95	9,95	Pollin
XT60 5er Satz	0,5	1,8	0,90	AliExpress
ESP32	1	3,77	3,77	AliExpress
2s BMS	1	0,89	0,89	AliExpress
Kabelset 20cm	0,5	3,30	1,65	AliExpress
Kabelset F – F 10cm	0,5	0,68	0,34	AliExpress
Gyroskop	1	0,93	0,93	AliExpress
H-Brücken	2	1,17	2,34	Bestand
Filament [kg]	0,05	20,00	1,00	Bestand
Schrauben + Muttern [Set]	1	1,00	1,00	Bestand
Motorkabel	1	0,50	0,50	Bestand
Versandkosten AliExpress	0,25	9,00	2,25	
Versandkosten Pollin	0,25	5,00	1,25	
Total				31,52 €

Tabelle 3.2: Kostenübersicht

3.4.1 Git

Zu Beginn des Projekts war bereits klar, dass eine Datenversionskontrolle für alle Projektdateien benötigt wird. Durch verschiedene Vorerfahrungen fiel die Entscheidung sehr leicht. Git löst genau diese Aufgaben perfekt. Das Projekt wurde in einem privaten Repository auf Github gehostet. Die Wahl der graphischen Oberflächen für Git war jedem Gruppenmitglied selbst überlassen. Vereinzelt wurde hier auf “Git Extensions” gesetzt, wobei meist eher “GitKraken” (Abb. 3.6) benutzt wurde, aufgrund der übersichtlichen graphischen Oberfläche.

3.4.2 Creo

Bei der CAD-Software standen mehrere unterschiedliche Programme unterschiedlicher Hersteller zur Auswahl. Diese waren “Catia” von Dassault Systemes, “Sketchup” von Trimble Inc., “Creo Parametrics” von PTC Inc., “Blender” von Blender Foundation,

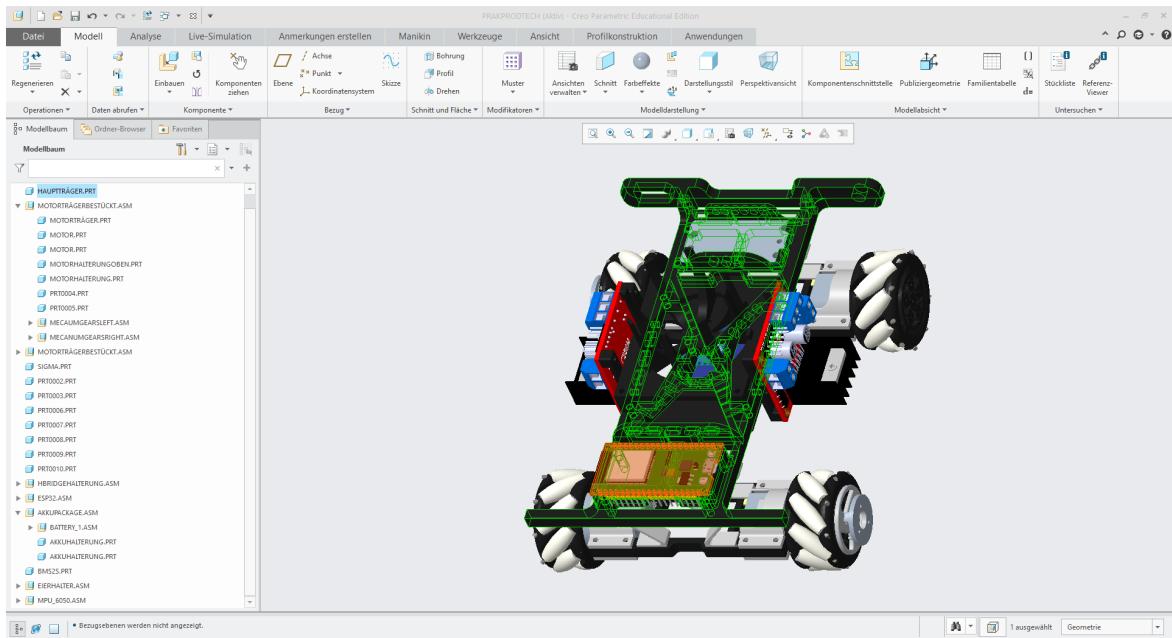


Abbildung 3.5: CAD Programm “Creo Parametrics”

“OpenSCAD” (Opensource Programm) und “FreeCAD” (ebenfalls Opensource). Nach längerem Abwägen fiel die Entscheidung auf das Programm Creo Parametrics (Abb. 3.6), da dieses vergleichbar einfach zu bedienen ist, aber trotzdem einen sehr großen Funktionsumfang bieten kann. Des Weiteren bietet Creo eine gute 3D-Maus Unterstützung, was das Modellieren deutlich vereinfacht. Trotz des großen Funktionsumfangs reichten die Möglichkeiten, die Creo bietet, nicht immer aus, weshalb in Einzelfällen auf andere CAD-Programme zurückgegriffen wurde. Diese werden allerdings an den betroffenen Stellen extra erwähnt.

3.4.3 PlatformIO

Die Wahl der Entwicklungsumgebung fiel auf Visual Studio Code mit PlatformIO, anstelle der in der Vorlesung vorgestellten Arduino IDE.

Mit der umfangreichen Erweiterbarkeit von VSCode ist hier die Programmierung einfacher und fehlerfreier durchzuführen, da in Arduino IDE nur bedingtes Syntax-checking vorhanden ist.

Durch PlatformIO stellt sich das Einbringen von externen Bibliotheken ebenfalls als Leichtigkeit heraus. Des Weiteren ist hier die Unterstützung von einer Vielzahl von MicroControllern bereits integriert.

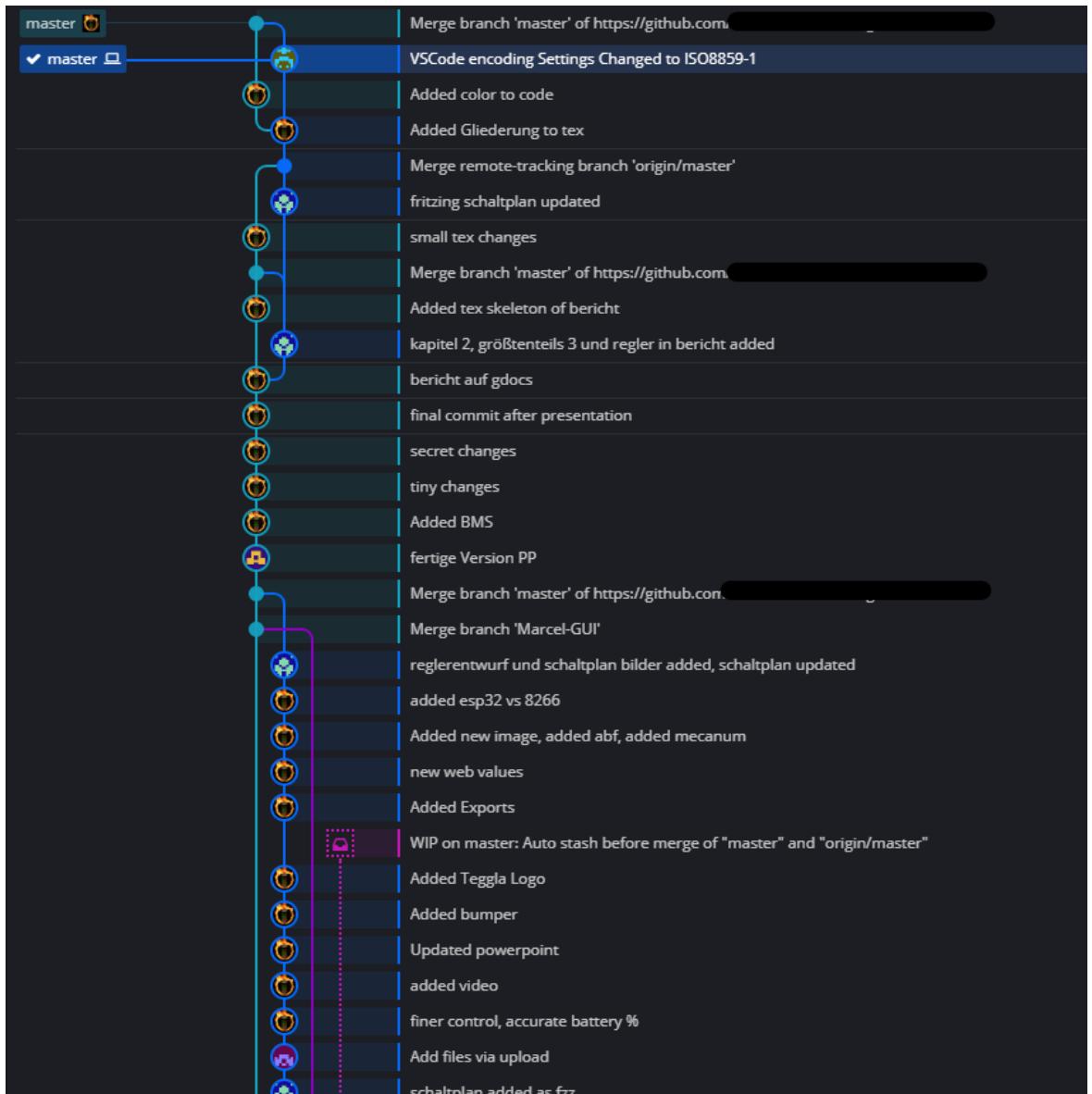


Abbildung 3.6: Graphische Oberfläche für Git (GitKraken)

4 Entwicklung der Schlüsselemente

4.1 Übersicht des gesamten Modells

Wie in Abbildung 4.1 erkenntm, besteht das Fahrzeug aus vielen Einzelteilen. Auf die wichtigsten Komponenten, die in der Stückliste (Tabelle 4.1) genannt werden, wird in den folgenden Kapiteln genauer eingegangen.

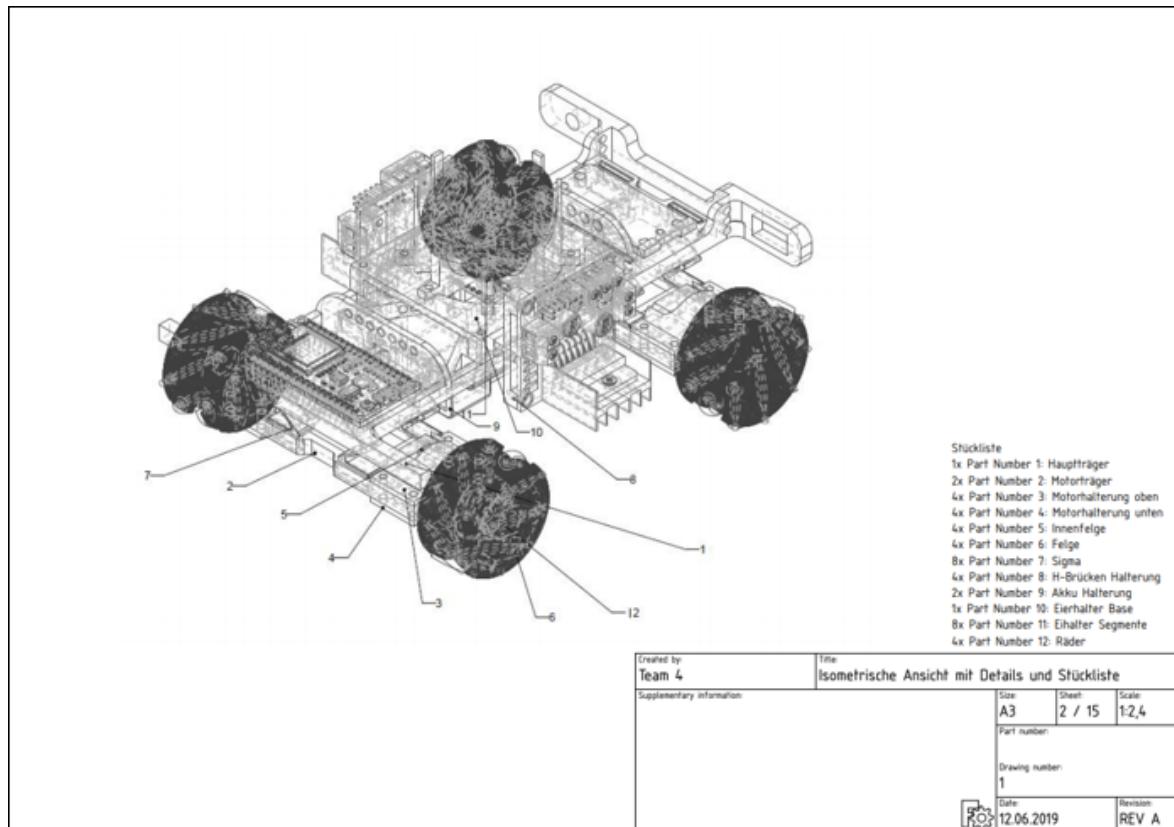


Abbildung 4.1: Überischt der TEGGLA-Komponenten

Bezeichnung	Anzahl
Hauptträger	1
Motorträger	2
Motorhalterung oben	4
Motorhalterung unten	4
Innenfelge und Felge	4
Sigma	8
H-Brücken Halterung	4
Akku Halterung	2
Eihalterung Base	1
Eihalter Segmente	8
Mecanum-Räder	4

Tabelle 4.1: Stückliste der TEGGLA-Komponenten

4.2 Schaltplan

Der Schaltplan (Abb 4.2) zeigt eine grobe Skizze der Verkabelung des TEGGLA Fahrzeuges. Der 2S Lipo Akku wird hierbei durch die beiden Einzellenakkus dargestellt. Der Hauptschalter, ein Dreipositionenschalter, dient zum Ein- und Ausschalten, sowie um das Fahrzeug in den Lademode zu versetzen. Die linke der beiden H-Brücken dient neben der Motoransteuerung auch zur Spannungsregulierung für das ESP. Das ESP32 ist die zentrale Steuereinheit und steuert mit je sechs digitalen Pins die H-Brücken an und kann so die Motordrehrichtung sowie die Geschwindigkeit der Motoren steuern. Das ESP32 kommuniziert auch via dem I2C Bus mit dem Gyroskop und versorgt dieses mit der 3.3V Spannungsversorgung. Die Widerstände am Ausgang des BMS, auf welches im folgenden Abschnitt eingegangen wird, bilden einen 3:1 Spannungsteiler. Dieser wird benötigt, da das ESP32 einen maximalen Spannungseingang von 3.3V verträgt, am BMS jedoch bis zu 9V anliegen können. Durch diesen Spannungsteiler ist es über den Analog-Digital-Wandler in dem ESP32 möglich, den Akkustand in 4095 Schritten auszulesen. Hierdurch wird dem User ein Feedback gegeben, wie lange die Akkuladung noch ausreicht.

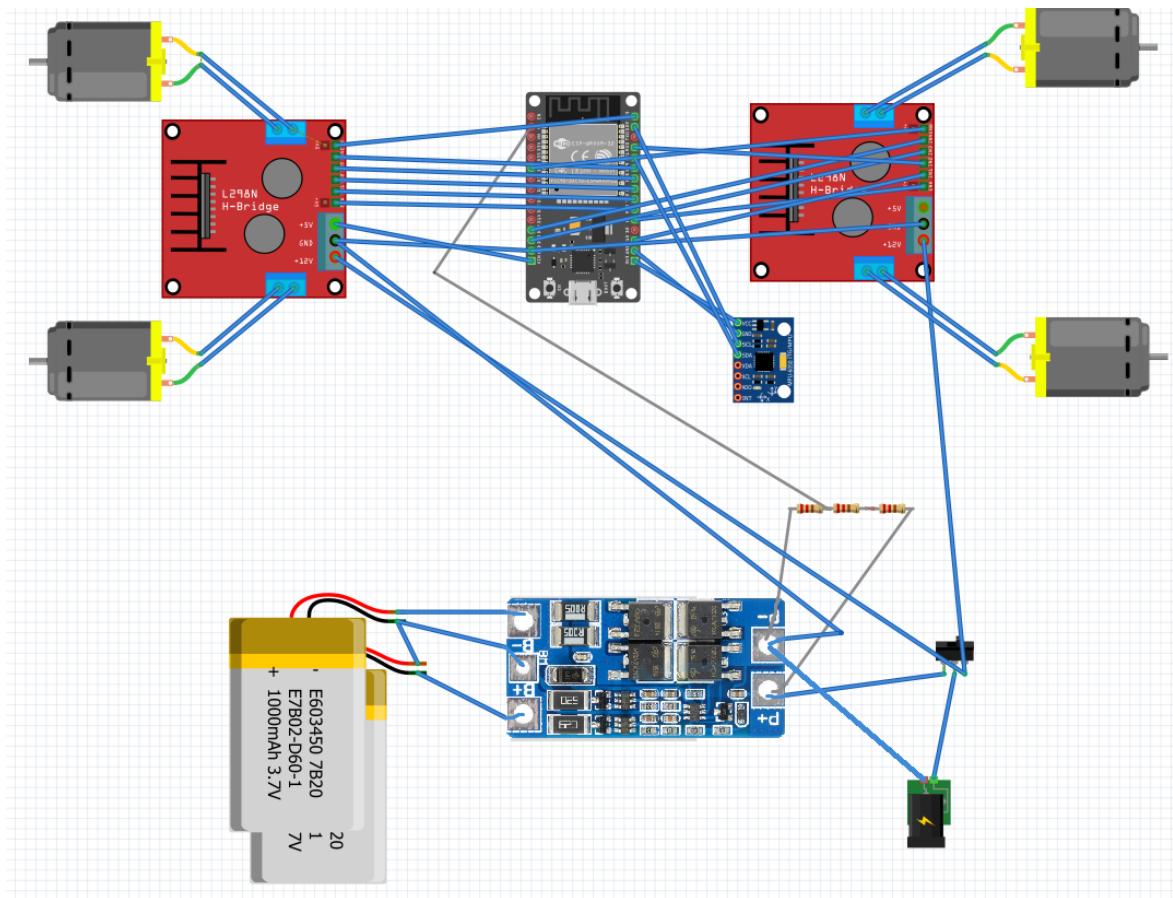


Abbildung 4.2: Schaltplan (gezeichnet mit Fritzing)

4.3 BMS und Laden

Um einem Akkuschaden vorzubeugen, sollte ein User doch einmal übersehen, dass der Akku des Fahrzeugs leer ist, wurde ein BMS (Battery Management System) verbaut. Dieses verhindert durch rechtzeitiges Abschalten ein Über- und Tiefenentladen des Akkus. Um das Fahrzeug wieder aufladen zu können, wurde ein externes Netzteil gekauft, welches über einen XT60 Stecker an das Fahrzeug angeschlossen werden kann. Sobald der Hauptschalter in die Ladeposition gebracht wird, beginnt das Fahrzeug nun den internen Akku zu laden, bis das BMS bei geladenem Akku den Ladevorgang vollautomatisch beendet.

4.4 Mecanum

Als besonderes Merkmal des TEGGLA fallen sofort die besonderen Räder auf. Hierbei handelt es sich um sogenannte Mecanum-Räder.

Hierbei handelt es sich um von Bengt Ilon 1972 erfundene Räder, die es dem Fahrzeug erlauben sich in drei Freiheitsgraden zu bewegen.

Dies wird ermöglicht durch die um 45° gedrehten Rollen auf den Rädern, sodass diese wie ein X aussehen. Durch unterschiedliche Drehrichtung und Drehgeschwindigkeit der Motoren kann das Fahrzeug in jegliche Richtungen innerhalb der Ebene bewegt werden. (vgl. Abb 4.3)

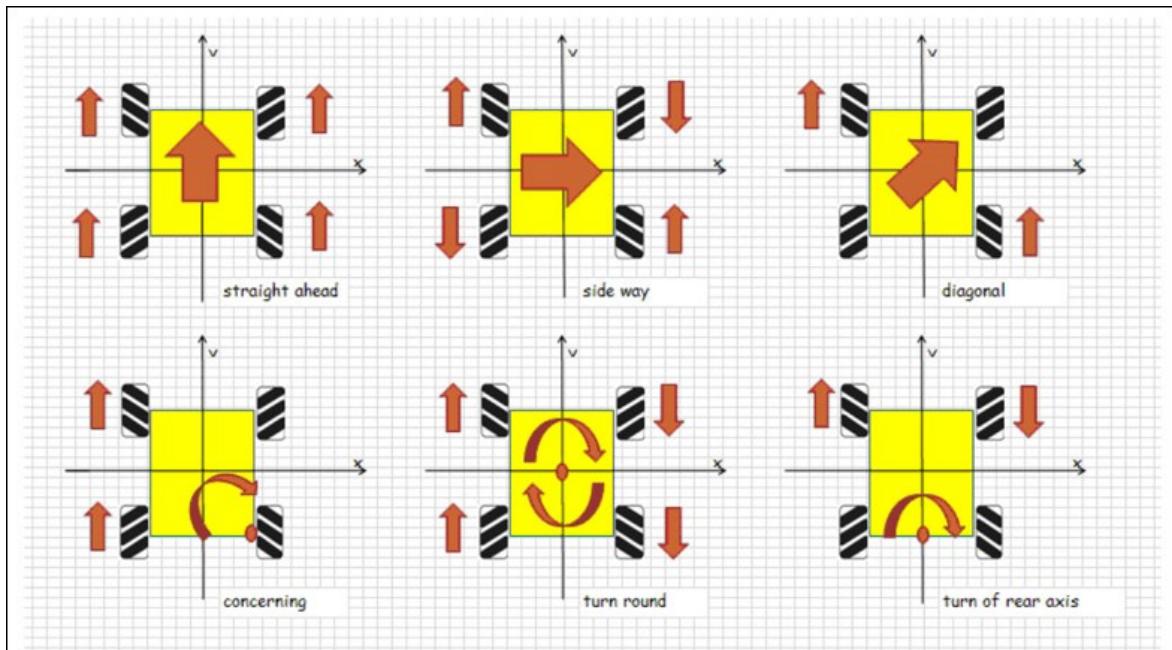


Abbildung 4.3: Freiheitsgrade von Mecanum [5]

Die jeweiligen Motorgeschwindigkeiten lassen sich hierbei durch Formeln 4.1–4.4 berechnen.

V_x = Motorgeschwindigkeit des x-ten Rads im Uhrzeigersinn (beginnend vorderes linkes Rad)

V_d = gewünschte Roboter Geschwindigkeit $[-1, 1]$

θ_d = gewünschter Winkel $[0, 2\pi]$

V_θ = gewünschte Rotationsgeschwindigkeit $[-1, 1]$

$$V_1 = V_d \sin \left(\theta_d + \frac{\pi}{4} \right) + V_\theta \quad (4.1)$$

$$V_2 = V_d \cos \left(\theta_d + \frac{\pi}{4} \right) - V_\theta \quad (4.2)$$

$$V_3 = V_d \cos \left(\theta_d + \frac{\pi}{4} \right) + V_\theta \quad (4.3)$$

$$V_4 = V_d \sin \left(\theta_d + \frac{\pi}{4} \right) - V_\theta \quad (4.4)$$

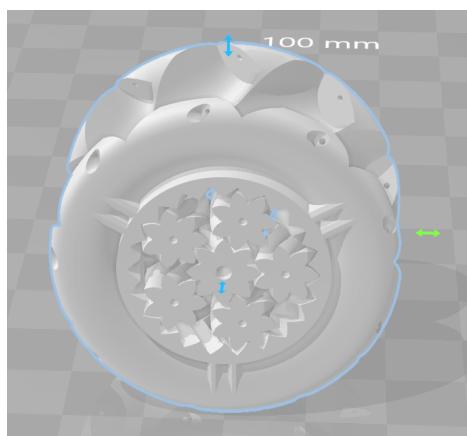
Abbildung 4.4: Formeln für die individuellen Motoren [5]

Bei den in diesem Praktikum verwendeten Rädern handelt es sich um eine Modifizierung der STL Dateien von Jonah Innoart von dem Internetportal Thingiverse [4]. In das Zentrum des Rades wurde mithilfe einer Boolean Operation von Blender3D eine Führung für die Getriebe geschnitten, sodass diese in das Rad eingeschoben werden können.

4.5 Planetengetriebe

Bereits in der 2. Woche des Projekts sind die ersten Entwürfe für die Planetengetriebe entstanden und bestanden aus zwei 3D-Modellen, dem Getriebe (Abb 4.5b) und dem Rad die beide von der Onlineplattform “Thingiverse” bezogen wurden. Diese wurden in Blender zu einem Bauteil zusammenfügt (Abb 4.5a).

Nach einem erfolgreichen Testdruck stellte sich allerdings raus, dass das Getriebe zu viel Reibung hatte, sodass es nicht anlaufen konnte.



(a) Version 1 des Planetengetriebes



(b) Original des verwendeten Planetengetriebes [1]

Um der Reibung entgegen wirken zu können, wurden im nächsten Schritt mit dem CAD Programm “OpenSCAD” (Abb 4.6) eigene Planetengetriebe erstellt. Hierbei lassen sich

vielerlei Parameter einstellen, beispielsweise Größe, Toleranzen, Zahanzahl, Planetenzahl, etc. Dies ermöglicht frei mit der Übersetzung sowie den bruckbaren Toleranzen zu testen um das bestmögliche Getriebe zu bekommen.

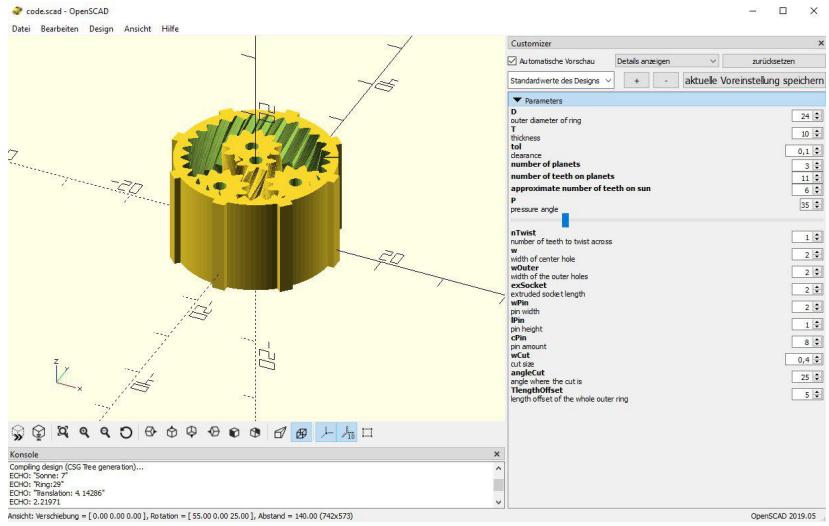


Abbildung 4.6: OpenSCAD zum Erstellen von Planetengetrieben

Durch die Funktionsweise von Additiver Fertigung lassen sich V-förmig verzahnte Getriebe als eine Einheit drucken. Dies wäre mit konventionellen Fertigungsprozessen nicht möglich. Da jedoch aufgrund von mangelnder Genauigkeit des 3D Druckers die Toleranzen so hoch gewählt werden müssen, sodass sich die Zahne nicht verschmelzen, führt dies zu zu viel Spiel in dem Getriebe.

Um jedoch das Getriebe weiterhin platzsparend in das Rad zu senken, wurde hier ebenfalls mit geradverzahnten Getrieben (Abb. 4.7) getestet. Hierbei entsteht jedoch das Problem der Fixierung der Planeten. Es entstand somit der Plan eine Gegenfuge zu benutzen um alle Teile zu halten.

Weiterhin gab es noch das Problem der zu niedrigen Übersetzungen. Hier hat es einige Iterationen gedauert bis eine Übersetzung von Über 1:4 erreicht werden konnte. Probleme die hier aufgetreten sind beispielsweise zu kleine Zahne (Abb. 4.8a), sodass diese nicht mehr ineinander greifen konnten.

Des Weiteren hängt die mögliche Übersetzung auch von der Anzahl der Planeten ab. Sind zu viele Planeten im Ring (Abb. 4.8c) überschneiden sie sich, und man muss somit auf weniger Planeten (Abb. 4.8b) zurückgreifen, was die Stabilität beeinträchtigt. Hierbei lässt sich auch der Kraftwinkel der Zahne einstellen, Ist der Winkel zu niedrig (Abb. 4.8d) rutschen die Zahne einfacher durch, verhaken sich jedoch nicht so stark. Da Versuche mit höheren Übersetzungen als 1:4 nie Erfolg hatten, wurden ebenfalls mehrstufige Getriebe (Abb. 4.8e) getestet, sogenannten "Composite Gear Sets". Hierbei sind

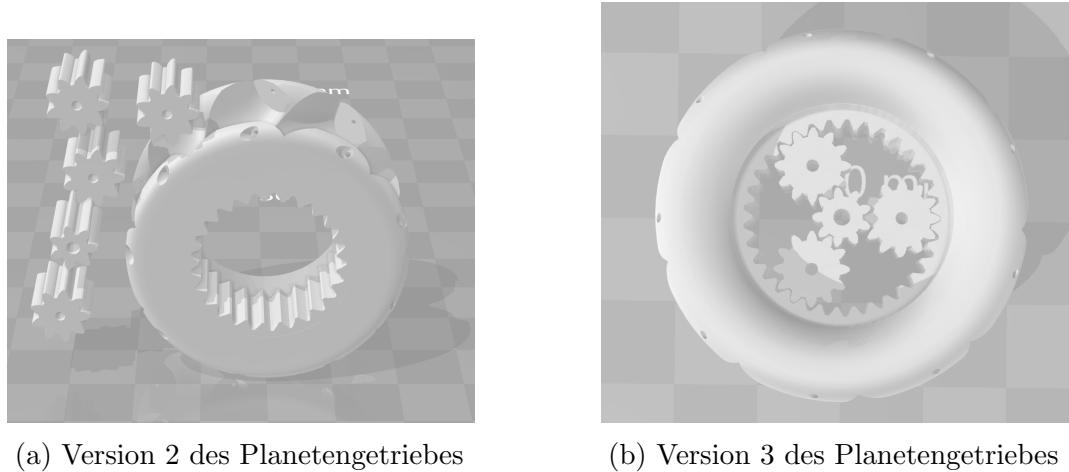


Abbildung 4.7: Im Rad integrierte Getriebe

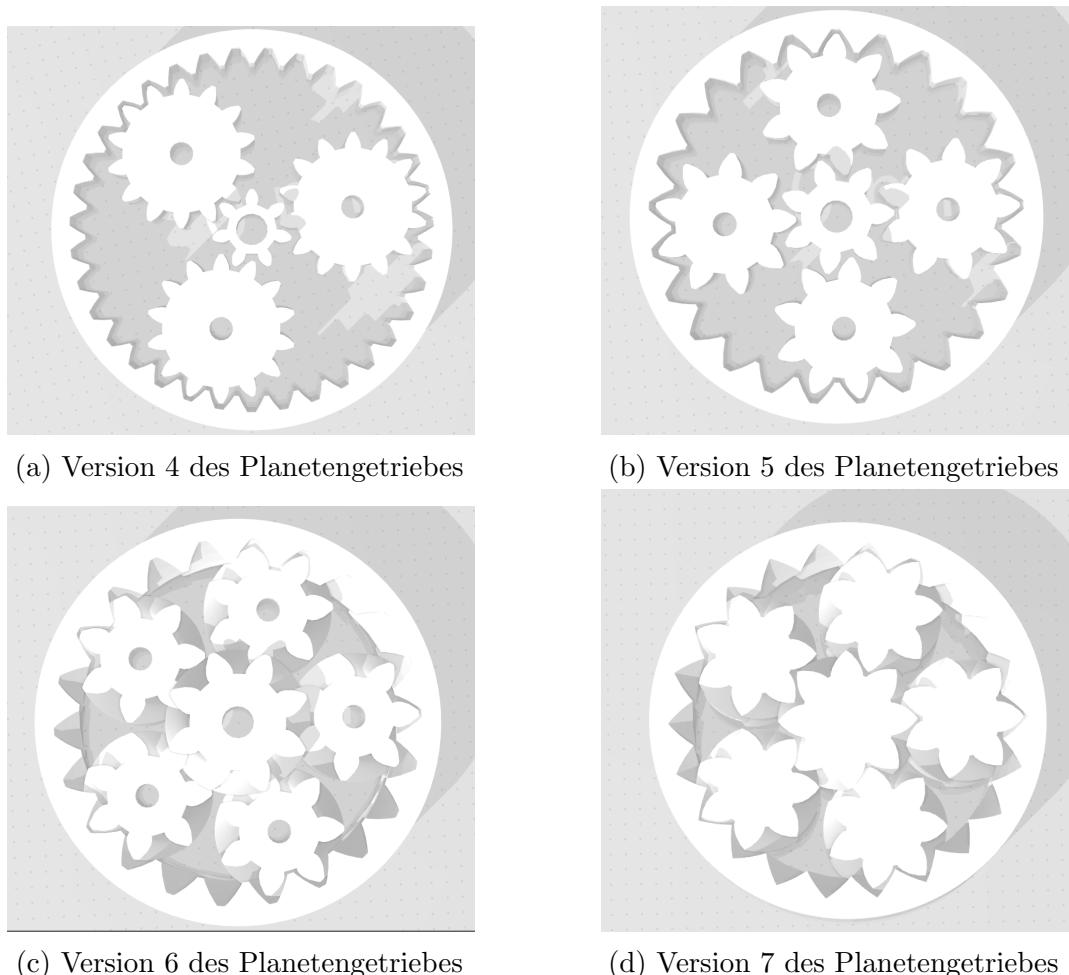


Abbildung 4.8: Iterationen der Getriebe

Übersetzungen von über 1:66 möglich, was für diese Anwendung perfekt wäre. Selbst bei mehrfachen Testdrucken der Zweistufigen Getriebe gelangt es nicht, ein lauffähiges Getriebe mit ausreichendem Wirkungsgrad zu erreichen. Es gelang bei einigen Getrieben sie per Hand mit einiger Gewalt zu drehen, jedoch hatten die Motoren nicht genug Moment um überhaupt aus dem Stillstand los zu drehen.

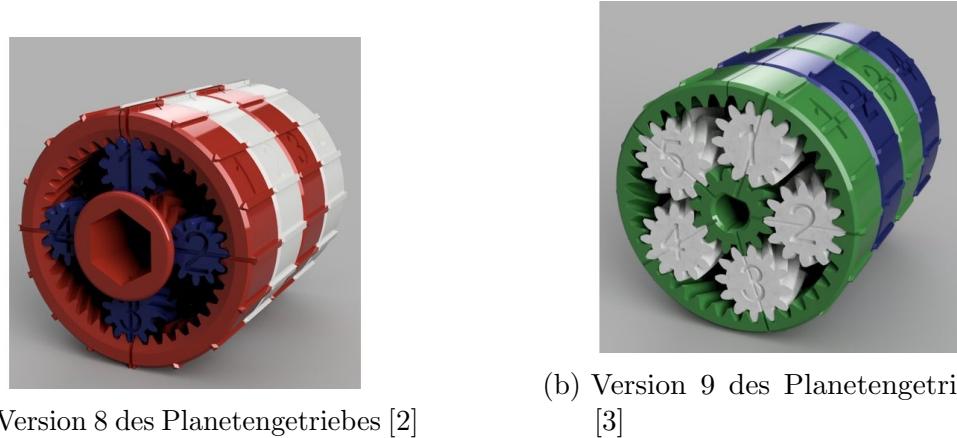


Abbildung 4.9: Mehrstufige Getriebe

Aus diesen Gründen wurde sich letztendlich auf einstufige Getriebe mit einer Übersetzung von 1 : 4.14 geeinigt. Obwohl dies zwar nicht ausreichend ist, um die Motoren von einer Lastdrehzahl von bis zu 6000 rpm auf eine lenkbare Geschwindigkeit zu übersetzen, wurde hier ebenfalls mit bedacht, dass weitere Steuerung durch Software mithilfe der H-Brücken möglich ist.

Dies führt zu dem finalen Design (Abb. 4.10a). Aufgrund der geringeren Reibung wurde sich hierbei für eine Geradverzahnung entschieden. Der 14mm lange Ring ist an einer Seite offen, sodass er aufgebogen werden kann um ihn um den Käfig (Abb. 4.10b) zu schließen. Bei dem Käfig handelt es sich um eine Verbindung der Innenfelge, die auf den Motor aufgesteckt wird mit der Außenfelge. Der Käfig erfüllt ebenfalls den Zweck die Planeten an ihrem Platz zu halten, sowie zusätzliche Stabilität der Achsen zu gewährleisten. Für die Achsen der Planeten dienen hier 1.5mm, die auf der Innenseite eingeklebt sind.

4.6 ESP-32 vs. ESP-8266

Bei dem vom Lehrstuhl gestellten ESP-8266 handelt es sich um einen WiFi-fähigen Microcontroller der chinesischen Firma “espressif”. Dieser besitzt 17 GPIO Pins, wovon jedoch lediglich 11 Pins nutzbar sind, da 6 an den externen SPI Flash angeschlossen

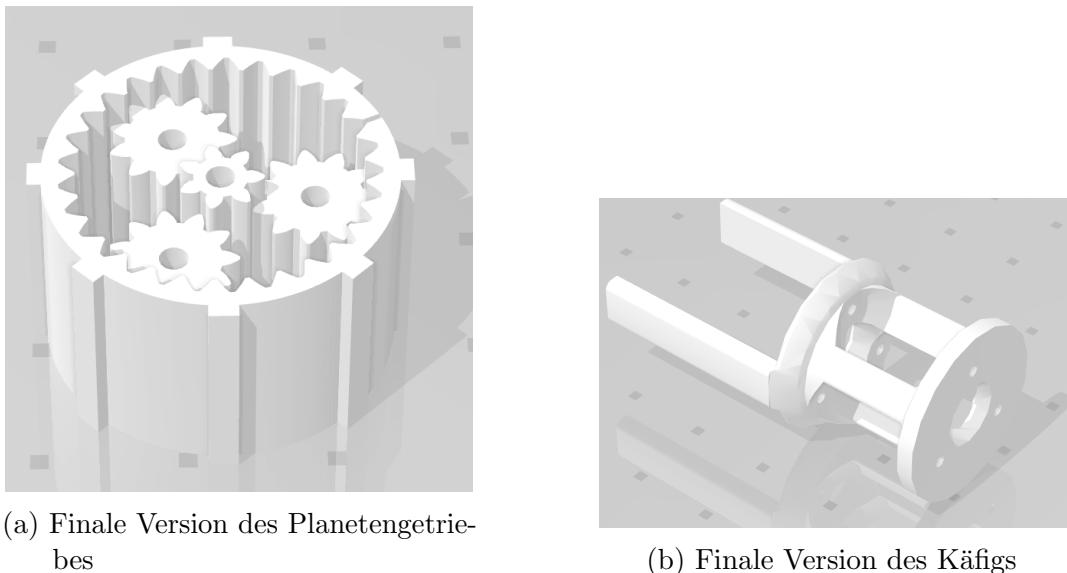


Abbildung 4.10: Finale Getriebebaugruppe

sind. Da dies wie in Tabelle 4.2 aufgezeigt nicht für unsere Zwecke reicht, musste auf den leistungsstärkeren ESP-32 ausgewichen werden.

Benötigte Pins	
4x PWM	Motor Enable
8x Output	Motor Richtung
2x I ² C	Gyroskop
1x ADC	Batteriespannung
15 Pins	

Tabelle 4.2: Aufzählung benötigter Pins

Obwohl die Anzahl der Pins das ausschlaggebende Argument für einen Austausch des MicroControllers war, bringt dieser natürlich weitere Vorteile mit sich.

Beispielsweise profitiert die später genauer erklärte Website stark davon, auf einen zweiten Core ausgelagert werden zu können.

Siehe Tabelle 4.3 für einen Vergleich der beiden MicroController anhand der für dieses Projekt relevanten Faktoren.

	ESP-8266	ESP-32
Cores	single core	dual core
Max Frequenz	160 MHz	240 MHz
GPI	17 (11 nutzbar)	36 (30 nutzbar)
SRAM	160 KB	520 KB
ADC Auflösung	10 bit	12 bit
Stromverbrauch	80 mA	260 mA
Preis (aus China)	2€	4€

Tabelle 4.3: Vorteile des ESP-32

4.7 User-Interface

Eine passende, nutzerfreundliche und optisch ansprechende UI hatte für das Projekt TEGGLA von Anfang an eine hohe Priorität. Es wurden zwei verschiedene UIs für das Projekt entwickelt, welche nachfolgend detailliert beschrieben werden.

4.7.1 Java (obsolete)

Das erste Konzept für eine passende UI wurde mit Java entwickelt. Java (und das Framework Java Swing) bieten grundsätzlich einige Möglichkeiten Benutzeroberflächen zu bauen und da Java eine der meistgenutzten Programmiersprachen ist, haben wir uns vorerst dafür entschieden.

Die Java UI besteht aus zwei Fenstern, welche vorerst die Verbindung zum TEGGLA sicherstellen und anschließend die Steuerung und Kommunikation ermöglichen.

Im ersten Fenster (Abb. 4.11) müssen in die Textfelder „IP Adresse“ und „PORT“ die passenden Daten eingegeben werden, um anschließend mit dem „Connect“ Button eine Verbindung zum TEGGLA aufzubauen. Falls das nicht gelungen ist, bekommt der Nutzer eine Fehlermeldung als Pop-Up und muss die Daten erneut eingeben. Im Erfolgsfall erscheint eine kurze Erfolgsmeldung und es öffnet sich das zweite Fenster.

Im zweiten Fenster (Abb. 4.12) werden alle wichtigen Daten für die Steuerung und Kommunikation des TEGGLA angezeigt, nämlich Geschwindigkeit und Akkustand.

Die Entwicklung des zweiten Fensters war noch nicht ganz abgeschlossen, als Java durch die Verwendung der neuen UI mit JavaScript / HTML5 abgelöst wurde. Daher wäre dieses Fenster noch durch die Visualisierung der Motoransteuerung und – auslastung

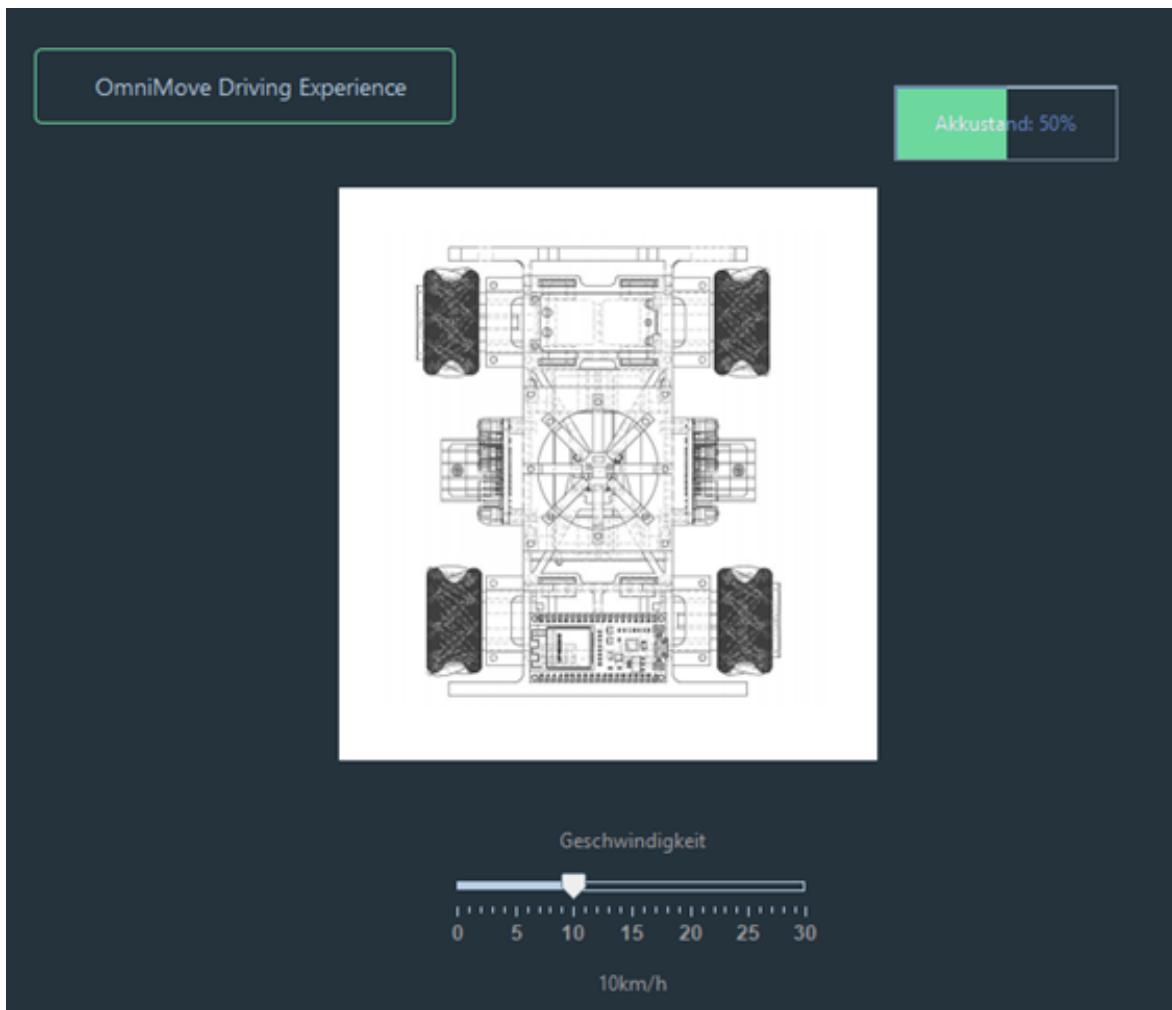


Abbildung 4.11: Java Fenster 1: Verbindungsauflaufbau

ergänzt worden.

Abschließend lässt sich festhalten, dass die Entwicklung einer UI mit Java (und Java Swing) durchaus Vorteile hat, aber das Erstellen einer optisch ansprechenden Benutzeroberfläche sehr zeitaufwändig ist. Für diese zwei Fenster (mit Logikschicht) wurden in etwa 1000 Zeilen Quellcode benötigt, was für größere Projekte mit mehr Komponenten nicht empfehlenswert ist. Der zugehörige Quellcode ist im Anhang des Berichts und kann bei Interesse daher noch genauer betrachtet werden.

4.7.2 HTML5 und Controller-Anbindung

Das Java Programm wurde aus mehreren Gründen zugunsten einer auf HTML5 sowie JavaScript basierten Weboberfläche ersetzt:

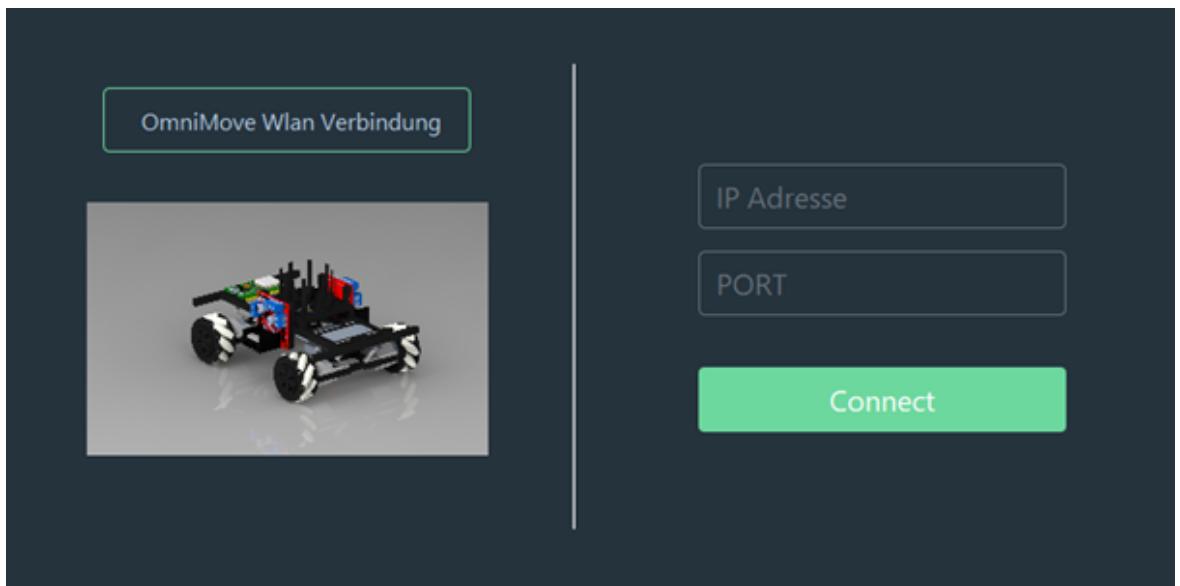


Abbildung 4.12: Java Fenster 2: Steuerung

- Native und einheitliche Unterstützung für Controller unterschiedlichster Marken in HTML5
- Unabhängig von Java Laufzeitumgebung, sowie Verfügbarkeit des Programms. (Hierbei muss nur ein Browser auf dem PC installiert sein.)
- Einfache Übertragung der Daten per WebSockets anstatt von “raw” Sockets, ohne ein eigenes “Frame” um die Daten bauen zu müssen

Die Erstellung der Website lässt sich sehr leicht durch die ESPAsyncWebServer Library für den ESP32 lösen.

Diese hostet direkt auf dem ESP32 einen WebServer der sowohl HTML5, JS, als auch CSS bereitstellen kann. Als Speicherort für diese Dateien wird das sogenannte Dateisystem SPIFFS verwendet, welches den Flashspeicher des ESP32 als Dateisystem benutzt, wie es beispielsweise aus Windows bekannt ist.

Eine Einschränkung ist die Limitierung auf eine gleichzeitige Verbindung zu dem Server. Dies wurde empirisch herausgefunden und somit konnte nicht sicher gesagt werden, ob es sich hier um eine Einschränkung aufgrund mangelnder Rechenleistung handelt oder ob die Library nicht mehr gleichzeitige Verbindungen unterstützt. Als Lösung dieses Problems, wurde nun die Anzahl der Verbindungen, die der WiFi Accesspoint akzeptiert, auf eins gesetzt.

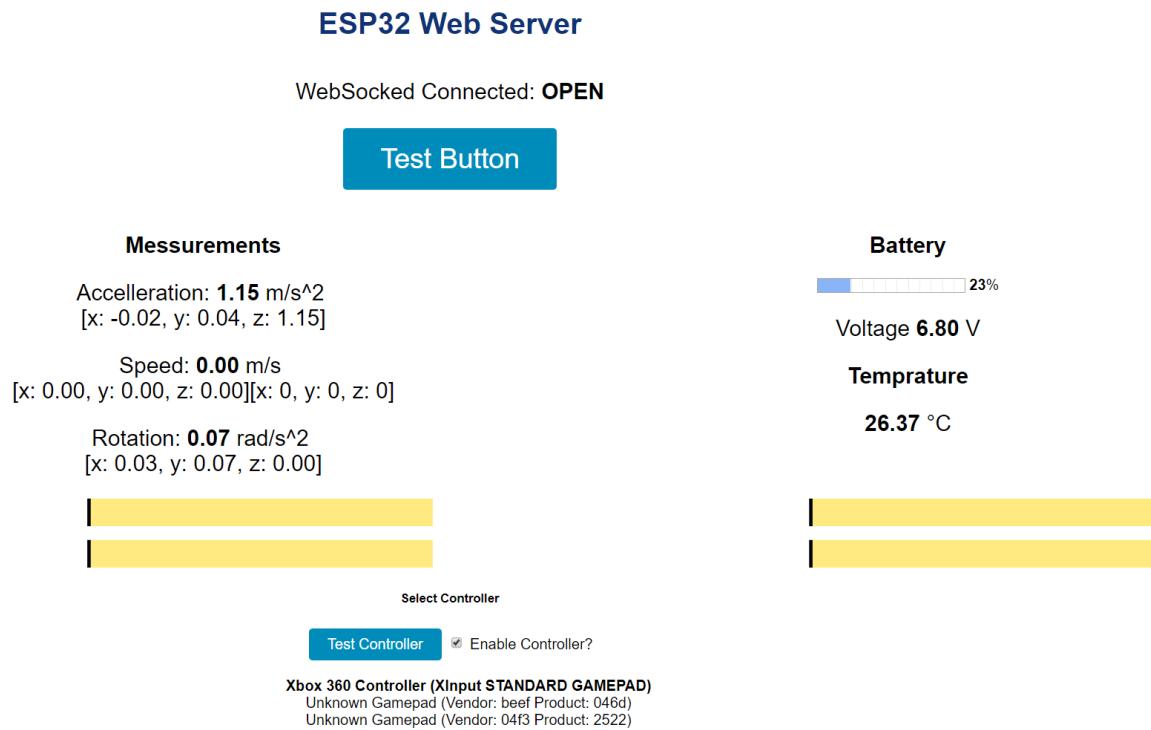


Abbildung 4.13: Bildschirmfoto Weboberfläche

In der Weboberfläche (Abb. 4.13) sind ebenfalls noch jegliche Messwerte hinterlegt, die das Fahrzeug sammelt.

Diese sind:

- Beschleunigung
- Geschwindigkeit
- Rotationsgeschwindigkeit
- Batteriespannung
- Temperatur
- Drehgeschwindigkeit der jeweiligen Motoren

Für die Wahl des Controllers ist eine auswählbare Liste aller angeschlossenen Controller am Ende der Seite vorhanden. Falls der Nutzer keinen Controller besitzt oder angegeschlossen hat, kann durch das Abwählen der Checkbox auf die Steuerung per WASD, sowie die Pfeiltasten gewechselt werden.

4.7.3 Protokoll

Zum Übertragen wurde ein binäres Protokoll (Tabelle 4.4) entwickelt, um die Datenrate gering zu halten, sowie die Verarbeitung auf dem Microcontroller zu vereinfachen.

Da durch Websockets bereits ein Integrierter Frame geschickt wird, muss nicht bei jeder Nachricht die Länge sowie Anfangs- und Endbyte mitgeschickt werden, wie es sonst bei raw Sockets nötig gewesen wäre.

Hierbei wird jeder Wert als Int16 geschickt, um ein ausreichend großes Spektrum bei geringer Datenrate zu gewährleisten.

Am Anfang jeder Nachricht wird die ID ebenfalls als int16 geschickt, somit wären 65536 unterschiedliche Nachrichten erlaubt.

Um die Anzahl an Nachrichten zu verringern, wurden die Batteriespannung und Werte des Gyroskop zu einer kombinierte Nachricht zusammengefasst, um den Overhead gering zu halten.

Name	ID	Werte											
		X-Achse L	Y-Achse L	X-Achse R	Y-Achse R								
DRIVE	0	X-Achse L	Y-Achse L	X-Achse R	Y-Achse R								
GYRO	1	Speed-X	Speed-Y	Speed-Z	Accel-X	Accel-Y	Accel-Z	Rot-X	Rot-Y	Rot-Z			
BATTERY	2	Voltage											
MOTOR	3	Speed-M1	Speed-M2	Speed-M3	Speed-M4								
COMB	4	Speed-X	Speed-Y	Speed-Z	Accel-X	Accel-Y	Accel-Z	Rot-X	Rot-Y	Rot-Z	Battery	Temp	

Tabelle 4.4: Binäres Protokoll

4.8 Steuerung per (XBox) Controller

Wie bereits in der Dokumentation der Weboberfläche erwähnt, wird die Steuerung primär per Spiele Controller gelöst, beispielsweise einem XBox-Controller von Microsoft. Diese Entscheidung ist in der erhöhten Mobilität motiviert.

Mit einer Steuerung per WASD bzw. per Pfeiltasten sind nur binäre Zustände messbar, gedrückt oder nicht gedrückt, volle Geschwindigkeit oder Stillstand. Im Vergleich dazu erlauben uns die JoySticks des Controllers durch unterschiedlich starke Auslenkung die Geschwindigkeit sehr variabel zu bestimmen.

Da hierbei ebenfalls der JoyStick in jegliche Richtungen bewegt werden kann, können ebenfalls die Mecanum-Räder so angesteuert werden, dass sie in genau diese Richtung fahren.

Jedoch sind hiermit nur zwei der drei Freiheitsgrade unseres Fahrzeugs abgedeckt. Um Rotationen um die eigene Achse mit variabler Geschwindigkeit zu steuern, wird die Eingabe des linken und des rechten JoySticks mit folgenden Formeln überlagert.

Sei hier *controlSide* Auslenkung des linken Sticks in X Richtung (nach rechts), *controlFront* Auslenkung des linken Sticks in Y Richtung (nach vorne) und *controlTurn* Auslenkung des rechten Sticks in X Richtung (nach rechts),

$$\theta = \text{atan2}(\text{controlSide}, \text{controlFront}) \quad (4.5)$$

$$V_d = \min(\sqrt{\text{controlFront}^2 + \text{controlSide}^2}, 1023) - \frac{\text{controlTurn}}{2} \quad (4.6)$$

$$V_\theta = \frac{\text{controlTurn}}{2} \quad (4.7)$$

4.9 PLA vs. TPU

Der Großteil der Teile für den TEGGLA wurde mit PLA Filament gedruckt. PLA steht für Polyactide, TPU steht für thermoplastisches Polyurethan. In diesem Kapitel wird auf die Unterschiede zwischen TPU und PLA eingegangen.

PLA hat im Vergleich ein höheres E-Modul, was zu einer höheren Zugfestigkeit und Steifigkeit führt. Auch in Sachen Verarbeitung ist PLA deutlich einfacher als TPU. Im Gegensatz zu TPU ist der Druckprozess mit PLA problemlos. Darüber hinaus ist TPU ein bisschen teurer als PLA.

Warum also TPU? TPU ist im Vergleich zu PLA viel flexibler und elastischer. Deswegen fiel die Wahl beim Rohstoff für die Sigmas und die Motorhalterungen auf TPU. Acht Sigmas verbinden die zwei Motorträger mit dem Hauptträger. Sie dienen hier quasi als Federung für die beiden "Achsen". Die Motorhalterungen sind das Bindeglied zwischen Motor und Motorträger. Es ist naheliegend, eine Federung und eine Befestigung für die Motoren aus einem elastischen Material zu drucken, auch wenn man mit einigen Nachteilen leben muss.

Um ein ansprechendes Design zu erhalten und um die Unterscheidung der Materialien zu erleichtern, wurde für den Robter, sowie in den Skizzen, für PLA die Farbe schwarz verwendet, für TPU die Farbe weiß.

4.10 Eierhalter

Der Leichtbau-Anspruch lässt sich bei allen Komponenten des TEGGLA wiederfinden, sogar beim Eierhalter. Dieser wurde aus mehreren einzelnen Bauteilen erstellt, um mit möglichst wenig Gewicht die größte Stabilität für das Ei zu erreichen. Es wurde vorerst das Fundament des Eihalters gedruckt und anschließend mit 8 Eihalter-Segmenten

verbunden. Somit kann das zu transportierende Ei schnell und sicher auf dem TEGGLA verwahrt werden.

Zusätzlich befindet sich der Eierhalter genau im Schwerpunkt des Fahrzeugs, um sicherzustellen, dass die aufgenommene Last gleichmäßig verteilt werden kann und die omnidirektionale Fahrweise des TEGGLA nicht beeinträchtigt werden kann.

4.11 Leichtbau

Bei der Konstruktion des TEGGLA wurde auf die Verwendung von massiven und schweren Bauteilen komplett verzichtet, wodurch das geringe Gesamtgewicht des Fahrzeugs (ca. 500 Gramm) erreicht worden ist. Dabei sind die schwersten Bauteile die Elektronikbauteile, auf die kein Einfluss genommen werden kann, wie die Motoren (je 36g), die H-Brücken (je 25g) sowie der Akku (62g).

Um ein omnidirektionales Fahren zu ermöglichen, benötigt der TEGGLA vier Motoren und zwei H-Brücken, welches das Gewicht des Fahrzeugs im Vergleich zu konventionellen Fahrzeugen erhöht. Die Leichtbau Anforderung ist daher in gewisser Weise in Konflikt mit einer innovativen Idee, schnellem Fahren und vor allem dem omnidirektionalen Fahren.

Durch die Verwendung einer elastischen Federung konnte auf ein Gehäuse für den Schutz der Elektronik verzichtet werden und zusätzlich konnten die dadurch entstandenen Hohlräume für technische Bauteile und Verbindungskabel genutzt werden.

5 Zukünftige Entwicklungsmöglichkeiten

Obwohl das Praktikum erfolgreich verlief, werden in diesem Kapitel noch Verbesserungsvorschläge und Ausbaumöglichkeiten beleuchtet.

5.1 Regler

Die Messwerte des Gyroskops können für einen Regelkreis (Abb. 5.1) benutzt werden.

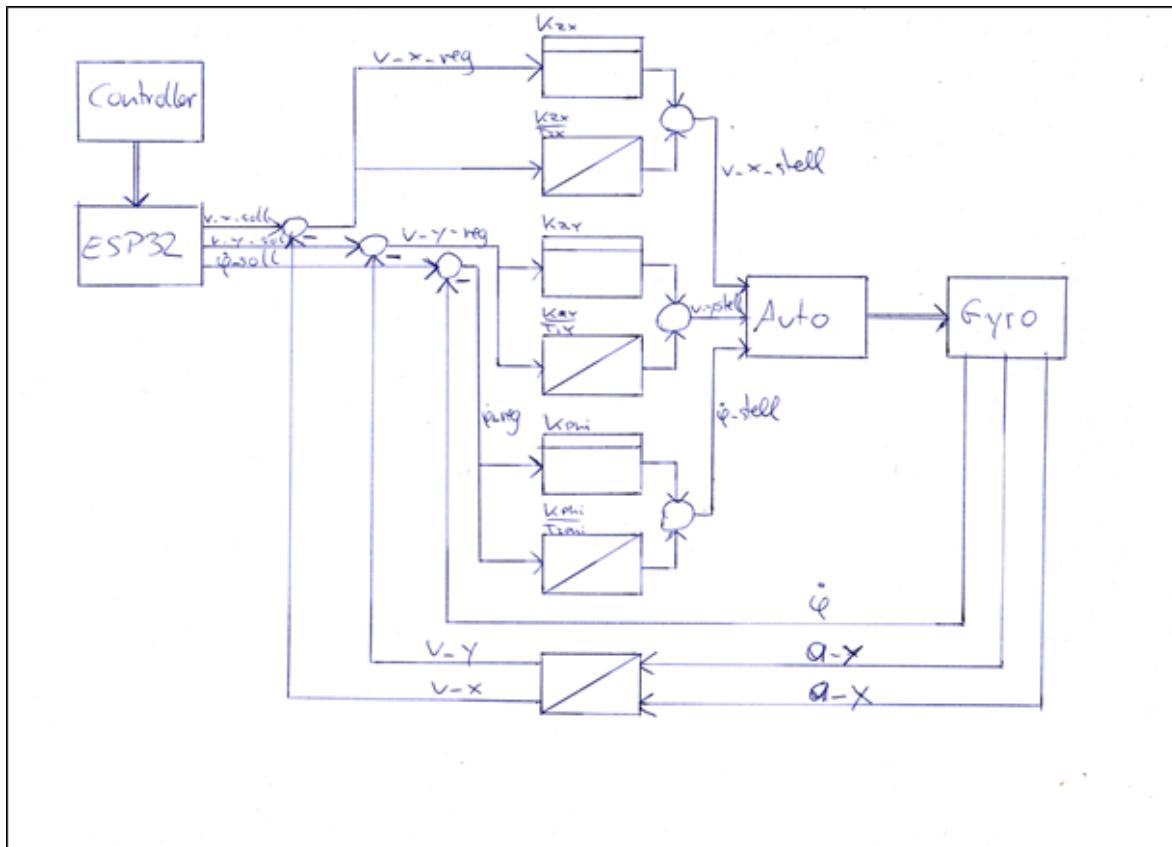


Abbildung 5.1: Blockschaltbild Regler

Durch die Eingaben des Spiele-Controllers und deren Interpretation durch das ESP-32 werden Sollwerte für Geschwindigkeiten in allen Achsen vorgegeben. Das Gyroskop misst Beschleunigung in X- und Y-Richtung und die Winkelgeschwindigkeit der Rotation. Deswegen müssen die Beschleunigungswerte einmal integriert werden, um auf die momentane Ist-Geschwindigkeit in beiden Richtungen zu kommen. Das übernimmt der Integrator ganz unten.

Anschließend wird von den Soll- und Ist-Werten die Differenz gebildet. Über diese Regelungsdifferenz wird jeweils durch einen PI-Regler (Proportional und Integrationsglied) mit empirisch bestimmten Faktoren die Stellgröße ermittelt. Das Integrationsglied der PI-Regler ist dabei wichtig für die stationäre Genauigkeit.

Die Stellgrößen werden nach Umrechnung an die Motoren weitergeleitet, was wiederum zu veränderten Messgrößen am Gyroskop führt. Damit ist der Kreis geschlossen.

5.2 Verbesserungen

Stillstand bedeutet Rückschritt. Daher werden bei zukünftigen TEGGLA Versionen sowohl Hardware – als auch Software – Komponenten angepasst, um jedes Fahrzeug besser als dessen Vorgänger zu machen.

5.2.1 Hardware

Für das omnidirektionale Fahren mit den Mecanum-Rädern sind Stepper-Motoren besser geeignet als die aktuell verwendeten DC-Motoren, da diese eine vorgegebene Drehzahl besser halten können. Daher werden zukünftige TEGGLAs mit vier Stepper-Motoren ausgestattet sein.

Eine weitere hardwarenahe Verbesserung ist die Neugestaltung des Hauptschalter am Rahmen des Fahrzeugs. Dieser ist aktuell so konstruiert, dass der Hauptschalter bei einer Kollision des Fahrzeugs mit Objekten unbeabsichtigt betätigt werden kann. Daher wird in der nächsten Version des TEGGLA der Hauptschalter nach innen zeigen, um vor Kollisionen und unerwünschtem Betätigen geschützt zu sein.

Die Verbesserung des Planetengetriebes ist zwar ein zeitaufwändiges Vorhaben, jedoch würde sich das ebenfalls positiv auf die Übersetzungsrate und somit auf das Fahren auswirken. Durch das Anpassen von Stellgrößen im CAD Modell und durch einige Druckversuche kann immer weniger Spiel im Getriebe erreicht werden. Eine Alternative

wäre die Verwendung eines besseren, genaueren 3D-Druckers, da beim 3D-Druck eines Planetengetriebes bereits sehr geringe Druckgenauigkeiten problematisch sind.

Die vorerst letzten hardwarenahen Verbesserungsmöglichkeiten beziehen sich auf den Ladestecker und die Rollen der Mecanum-Räder. Der aktuell verwendete Ladestecker (XT60) ist für ein häufiges Aufladen des Fahrzeugs nicht gut geeignet, da sich das Kabel schwer ein- und ausstecken lässt.

Die Rollen der Mecanum-Räder sollen bei zukünftigen Versionen des TEGGLA mehr Grip haben, was eventuell durch die Verwendung eines anderen Filaments erreicht werden kann.

5.2.2 Software

Die Weboberfläche für die Kommunikation und Steuerung des TEGGLA soll fortlaufend erweitert und verbessert werden. Beispielsweise kann durch das Integrieren der Beschleunigung, welche vom Gyroscope gemessen wird, zusätzlich auch noch die aktuelle Geschwindigkeit des TEGGLA angezeigt werden.

Eine weitere softwarenahe Verbesserung des TEGGLA wäre das Beheben von Verbindungsproblemen. Bei dem Testen des fertigen Fahrzeugs kam es teilweise zu Verbindungsabbrüchen zum WLAN des TEGGLA. Dadurch konnte das Fahrzeug natürlich nicht mehr gesteuert werden und zusätzlich wurde der letzte erhaltene Befehl dauerhaft ausgeführt.

Dieses Verbindungsproblem kann durch zwei Schritte verbessert werden: Vorerst sollte der Code so angepasst werden, dass bei einem Verbindungsabbruch kein Befehl mehr ausgeführt wird und der TEGGLA somit keine unkontrollierten Bewegungen durchführt. Anschließend können automatische Reconnects implementiert werden, welche bei einem Verbindungsabbruch in kleinen Zeitabständen versuchen, die Verbindung automatisch wiederherzustellen. Dadurch kann die Zeit ohne Verbindung mit dem TEGGLA minimiert werden.

6 Zusammenfassung

In diesem abschließenden Kapitel werden die gesammelten Erfahrungen reflektiert, sowie die wichtigsten erlernten, beziehungsweise verbesserten Fähigkeiten beim Entwickeln des TEGGLA zusammengefasst.

6.1 Fazit

Das Entwickeln des TEGGLA hat dem gesamten Team sehr viel Spaß gemacht. Um das erwünschte Ergebnis zu erreichen, wurde der Umgang mit vielen neuen Technologien erlernt beziehungsweise das bereits vorhandene Wissen vertieft. Vor allem das Konstruieren in CAD, die Anfertigung von technischen Zeichnungen und das tatsächliche Drucken und Zusammenbauen der Baugruppen hatten einen großen Mehrwert für unsere zukünftigen Projekte. Die wichtigsten verwendeten Technologien waren:

Creo

Konstruieren aller Bauteile des TEGGLA.

C++

Programmierung des MicroController auf einer Hardwarenahen Ebene.

JavaScript / HTML5

Weboberfläche mit Sockets für die Kommunikation und Steuerung des TEGGLA.

Github

Repository für die Sicherung, Versionierung, Verteilung und Aktualisierung aller Daten, technischen Zeichnungen, Skizzen, Berichte, Präsentationen und Quellcode.

6.2 Errungene Erfahrung

Der wohl zeitaufwändigste und schwierigste Aspekt des Projekts war der 3D-Druck eines Planetengetriebes, welches die erforderliche Übersetzung für das omnidirektionale Fahren bereitstellen kann. Bereits geringe Druckerungengenauigkeiten führten zu nicht nutzbaren Getriebeverisionen und daher wurden viele Anläufe benötigt, um ein passendes Planetengetriebe zu entwickeln. Für zukünftige Projekte mit 3D-Druckern wird das berücksichtigt und der Fokus bei der Entwicklung darauf gelegt.

Als sehr positiv war die zeitliche Planung und Zusammenarbeit des Teams zu bewerten. Trotz unerwarteten Mehraufwänden für Teilaufgaben, kam es zu keinerlei zeitlichen Engpässen und dadurch zu einem qualitativ hochwertigen Endprodukt. Eine möglichst präzise Aufgabendefinition und -verteilung am Projektanfang ist enorm wichtig, um dies sicherstellen zu können. Zusätzlich waren die Meilensteintermine des Lehrstuhls ebenfalls hilfreich, um sich an den Terminplan zu halten.

Die Nutzung eines GIT Repositories war ebenfalls äußerst gewinnbringend, da die Versierung, Verteilung, Aktualisierung und Sicherung jederzeit und für alle Teammitglieder gewährleistet werden konnte.

Insgesamt wurden enorm viele Erfahrungen über den Entwicklungsprozess eines lenkbaren Fahrzeugs gesammelt, welche sich natürlich auch auf andere Produkte und Projekte abstrahieren lassen. Die einzelnen Entwicklungsschritte, nämlich die Auswahl einer Idee, das Anfertigen eines Lastenhefts, die Rollen- und Aufgabenverteilung innerhalb des Teams sowie die zeitliche Einhaltung von Terminen sind alles wichtige Kenntnisse für unsere zukünftigen Projekte.

Anhang

A Quellcode

A.1 ESP32

```
1 #include "communication.h"
2
3 AsyncWebSocketClient *ws_client = nullptr;
4
5 void Communication::onWSData(AsyncWebSocket *server, AsyncWebSocketClient *client,
6     AwsEventType type, uint8_t *data, size_t len) {
7     ws_client = client;
8
9     int16_t *d16 = (int16_t *)data;
10    int16_t id = d16[0];
11
12    switch (id) {
13        case OmniMessageType::DRIVE:
14            onDrive(d16[1], d16[2], d16[3], d16[4]);
15            break;
16
17        default:
18            Serial.printf("No such command %i\n", id);
19            break;
20    }
21
22 void Communication::onDrive(int16_t x1, int16_t y1, int16_t x2, int16_t y2) {
23     // Serial.printf("Driving with speed: %i, %i and %i, %i now\n", x1, y1, x2,
24     // y2);
25     Movement::drive(y1, x1, x2);
26 }
27
28 void Communication::sendCurrGyro(XYZ speed, XYZ accel, XYZ rot) {
29     int16_t buff[10] = {OmniMessageType::CURR_GYRO, speed.x, speed.y, speed.z,
30     accel.x, accel.y, accel.z, rot.x, rot.y, rot.z};
31     ws->binaryAll((uint8_t *)buff, 20);
```

```

30 }
31
32 void Communication::sendCurrGyBatComb(int16_t sX, int16_t sY, int16_t sZ, int16_t
33     aX, int16_t aY, int16_t aZ, int16_t rX, int16_t rY, int16_t rZ, int16_t bat,
34     int16_t temp) {
35     int16_t buff[12] = {OmniMessageType::CURR_GY_BAT_COMB, sX, sY, sZ, aX, aY, aZ,
36     rX, rY, rZ, bat, temp};
37     ws->binaryAll((uint8_t *)buff, 24);
38 }
39
40 void Communication::sendCurrMotor(int16_t vl, int16_t vr, int16_t hl, int16_t hr) {
41     int16_t buff[5] = {OmniMessageType::CURR_MOTOR, vl, vr, hl, hr};
42     // Serial.println("should send now");
43     ws->binaryAll((uint8_t *)buff, 10);
44 }
45
46 void Communication::sendCurrBattery(int16_t cell1, int16_t cell2) {
47     int16_t buff[5] = {OmniMessageType::CURR_BATTERY, cell1, cell2};
48     // Serial.println("should send now");
49     ws->binaryAll((uint8_t *)buff, 10);
50 }
51
52 AsyncWebSocket *Communication::ws;

```

Listing A.1: communication.cpp

```

1 #pragma once
2
3 #include "movement.h"
4 #include <ESPAsyncWebServer.h>
5
6 /**
7 * Limitations due to js:
8 * Buffer has to be all the same type
9 *
10 * Type is int16 as well at the start of the array
11 *
12 */
13 enum OmniMessageType {
14     DRIVE = 0,           //L4 {x1} int16 [-1023, 1023] :: {y1} int16 [-1023,
15     1023] :: {x2} int16 [-1023, 1023] :: {y2} int16 [-1023, 1023]

```

```

15     CURR_GYRO = 1,           //L18 {speed m/s} (x) int16 (y) int16 (z) int16 :: 
16     {accel m/s^2} (x) int16 (y) int16 (z) int16 :: {rot} (x) int16 (y) int16 (z)
17     int16
18     CURR_BATTERY = 2,       //L1 {cell1} int16 [0, 2^12]
19     CURR_MOTOR = 3,         //L4 {vl} int16 [-1023, 1023] :: {vr} int16 [-1023,
20     1023] :: {hl} int16 [-1023, 1023] :: {hr} int16 [-1023, 1023]
21     CURR_GY_BAT_COMB = 4,   //L20 {speed m/s} (x) int16 (y) int16 (z) int16 :: 
22     {accel m/s^2} (x) int16 (y) int16 (z) int16 :: {rot} (x) int16 (y) int16 (z)
23     int16 :: {bat} int16 [0, 2^12] :: {temp} int16
24 };
25
26
27 struct XYZ {
28     int16_t x;
29     int16_t y;
30     int16_t z;
31 };
32
33
34 class Communication {
35 public:
36     static void onWSData(AsyncWebSocket *server, AsyncWebSocketClient *client,
37     AwsEventType type, uint8_t *data, size_t len);
38     static void onDrive(int16_t x1, int16_t y1, int16_t x2, int16_t y2);
39     static void sendCurrGyro(XYZ speed, XYZ accel, XYZ rot);
40     static void sendCurrBattery(int16_t cell1, int16_t cell2);
41     static void sendCurrGyBatComb(int16_t sX, int16_t sY, int16_t sZ, int16_t aX,
42     int16_t aY, int16_t aZ, int16_t rX, int16_t rY, int16_t rZ, int16_t bat,
43     int16_t temp);
44     static void sendCurrMotor(int16_t vl, int16_t vr, int16_t hl, int16_t hr);
45
46     static AsyncWebSocket *ws;
47 };

```

Listing A.2: communication.h

```

1 #include "movement.h"
2
3 // ----- Motor Class
4 // ##### Functions #####
5 void Motor::setSpeed(int16_t speed) {
6     // disable before before changing direction
7     // (probably not needed) safety to prevent shoutthrough

```

```
8     ledcWrite(channel, 0);
9     if (speed > 0) {
10         digitalWrite(pin_dir1, LOW);
11         digitalWrite(pin_dir2, HIGH);
12     } else if (speed < 0) {
13         digitalWrite(pin_dir1, HIGH);
14         digitalWrite(pin_dir2, LOW);
15     } else {
16         digitalWrite(pin_dir1, LOW);
17         digitalWrite(pin_dir2, LOW);
18         return;
19     }
20
21     ledcWrite(channel, abs(speed));
22 }
23
24 void Motor::stop() {
25     ledcWrite(channel, 0);
26     digitalWrite(pin_dir1, LOW);
27     digitalWrite(pin_dir2, LOW);
28 }
29
30 // ----- Movement Class
31 // ##### Attributes #####
32 Motor Movement::MOTOR_VL = Motor(CH_MOTOR_VL, PIN_MOTOR_VL_DIR1,
33     PIN_MOTOR_VL_DIR2);
33 Motor Movement::MOTOR_VR = Motor(CH_MOTOR_VR, PIN_MOTOR_VR_DIR1,
34     PIN_MOTOR_VR_DIR2);
34 Motor Movement::MOTOR_HL = Motor(CH_MOTOR_HL, PIN_MOTOR_HL_DIR1,
35     PIN_MOTOR_HL_DIR2);
35 Motor Movement::MOTOR_HR = Motor(CH_MOTOR_HR, PIN_MOTOR_HR_DIR1,
36     PIN_MOTOR_HR_DIR2);
36
37 // ##### Functions #####
38 void Movement::initPWM() {
39     Serial.println("Initing PWMs");
40
41     // Setting all pins to output
42     pinMode(PIN_LED, OUTPUT);
43     pinMode(PIN_MOTOR_VL_EN, OUTPUT);
44     pinMode(PIN_MOTOR_VL_DIR1, OUTPUT);
45     pinMode(PIN_MOTOR_VL_DIR2, OUTPUT);
```

```
46
47     pinMode(PIN_MOTOR_VR_EN, OUTPUT);
48     pinMode(PIN_MOTOR_VR_DIR1, OUTPUT);
49     pinMode(PIN_MOTOR_VR_DIR2, OUTPUT);
50
51     pinMode(PIN_MOTOR_HL_EN, OUTPUT);
52     pinMode(PIN_MOTOR_HL_DIR1, OUTPUT);
53     pinMode(PIN_MOTOR_HL_DIR2, OUTPUT);
54
55     pinMode(PIN_MOTOR_HR_EN, OUTPUT);
56     pinMode(PIN_MOTOR_HR_DIR1, OUTPUT);
57     pinMode(PIN_MOTOR_HR_DIR2, OUTPUT);
58
59 // Disable all pins by default to not possibly cause shootthrough
60 digitalWrite(PIN_MOTOR_VL_EN, LOW);
61 digitalWrite(PIN_MOTOR_VL_DIR1, LOW);
62 digitalWrite(PIN_MOTOR_VL_DIR2, LOW);
63
64 digitalWrite(PIN_MOTOR_VR_EN, LOW);
65 digitalWrite(PIN_MOTOR_VR_DIR1, LOW);
66 digitalWrite(PIN_MOTOR_VR_DIR2, LOW);
67
68 digitalWrite(PIN_MOTOR_HL_EN, LOW);
69 digitalWrite(PIN_MOTOR_HL_DIR1, LOW);
70 digitalWrite(PIN_MOTOR_HL_DIR2, LOW);
71
72 digitalWrite(PIN_MOTOR_HR_EN, LOW);
73 digitalWrite(PIN_MOTOR_HR_DIR1, LOW);
74 digitalWrite(PIN_MOTOR_HR_DIR2, LOW);
75
76 // configure LED PWM functionalitites
77 ledcSetup(CH_LED, PWM_FREQ, PWM_RESOLUTION);
78 ledcSetup(CH_MOTOR_VL, PWM_FREQ, PWM_RESOLUTION);
79 ledcSetup(CH_MOTOR_VR, PWM_FREQ, PWM_RESOLUTION);
80 ledcSetup(CH_MOTOR_HL, PWM_FREQ, PWM_RESOLUTION);
81 ledcSetup(CH_MOTOR_HR, PWM_FREQ, PWM_RESOLUTION);
82
83 // attach the channel to the GPIO2 to be controlled
84 ledcAttachPin(PIN_LED, CH_LED);
85 ledcAttachPin(PIN_MOTOR_VL_EN, CH_MOTOR_VL);
86 ledcAttachPin(PIN_MOTOR_VR_EN, CH_MOTOR_VR);
87 ledcAttachPin(PIN_MOTOR_HL_EN, CH_MOTOR_HL);
88 ledcAttachPin(PIN_MOTOR_HR_EN, CH_MOTOR_HR);
```

```
89     Serial.println("Initiated PWMs");
90 }
91
92
93 void Movement::drive(int controlFront, int controlSide, int controlTurn) {
94     if (abs(controlFront) < CONTROLLER_LOWER_LIMIT && abs(controlSide) <
95         CONTROLLER_LOWER_LIMIT && abs(controlTurn) < CONTROLLER_LOWER_LIMIT) {
96         // Serial.println("Stopping Motors");
97         MOTOR_VL.stop();
98         MOTOR_VR.stop();
99         MOTOR_HL.stop();
100        MOTOR_HR.stop();
101
102        Communication::sendCurrMotor(0, 0, 0, 0);
103        return;
104    }
105
106    int speedVL = 0;
107    int speedVR = 0;
108    int speedHL = 0;
109    int speedHR = 0;
110
111    if (abs(controlFront) > 0 && controlSide == 0 && controlTurn == 0) {
112        speedVL = controlFront;
113        speedVR = controlFront;
114        speedHL = controlFront;
115        speedHR = controlFront;
116    } else if (abs(controlSide) > 0 && controlFront == 0 && controlTurn == 0) {
117        speedVL = controlSide;
118        speedVR = -controlSide;
119        speedHL = -controlSide;
120        speedHR = controlSide;
121    } else if (controlSide == 0 && controlFront == 0 && abs(controlTurn) > 0) {
122        speedVL = controlTurn;
123        speedVR = -controlTurn;
124        speedHL = controlTurn;
125        speedHR = -controlTurn;
126    } else {
127        driveAlgorithm(controlFront, controlSide, controlTurn, &speedVL, &speedVR,
128                      &speedHL, &speedHR);
129    }
130}
```

```

129     speedVL = speedVL / 1023.0 * USEABLE_UPPER_LIMIT + (1023 -
130     USEABLE_UPPER_LIMIT) * sgn(speedVL);
131     speedVR = speedVR / 1023.0 * USEABLE_UPPER_LIMIT + (1023 -
132     USEABLE_UPPER_LIMIT) * sgn(speedVR);
133     speedHL = speedHL / 1023.0 * USEABLE_UPPER_LIMIT + (1023 -
134     USEABLE_UPPER_LIMIT) * sgn(speedHL);
135     speedHR = speedHR / 1023.0 * USEABLE_UPPER_LIMIT + (1023 -
136     USEABLE_UPPER_LIMIT) * sgn(speedHR);
137
138
139     MOTOR_VL.setSpeed(speedVL);
140     MOTOR_VR.setSpeed(speedVR);
141     MOTOR_HL.setSpeed(speedHL);
142     MOTOR_HR.setSpeed(speedHR);
143
144
145     Communication::sendCurrMotor(speedVL, speedVR, speedHL, speedHR);
146 }
147
148 void Movement::driveAlgorithm(int controlFront, int controlSide, int controlTurn,
149     int *speedVL, int *speedVR, int *speedHL, int *speedHR) {
150     double phi = atan2(controlSide, controlFront);
151
152     int vd = min((int)sqrt(controlFront * controlFront + controlSide *
153     controlSide), 1023);
154     vd -= controlTurn / 2;
155     int vphi = controlTurn / 2;
156
157     double s = vd * sin(phi + PI / 4);
158     double c = vd * cos(phi + PI / 4);
159
160     *speedVL = s + vphi;
161     *speedVR = c - vphi;
162     *speedHL = c + vphi;
163     *speedHR = s - vphi;
164 }
```

Listing A.3: movement.cpp

```

1 #pragma once
2
3 #include "communication.h"
4 #include "util.h"
5 #include <Arduino.h>
```

```
6 #include <math.h>
7
8 class Motor {
9 private:
10     char channel;
11     char pin_dir1;
12     char pin_dir2;
13
14 public:
15     Motor(char channel, char pin_dir1, char pin_dir2) : channel(channel),
16         pin_dir1(pin_dir1), pin_dir2(pin_dir2){};
17     ~Motor() {}
18
19     /**
20      * speed in [-1023; 1023]
21      */
22     void setSpeed(int16_t speed);
23
24     void stop();
25 };
26
27 class Movement {
28 private:
29     /* data */
30
31     static Motor MOTOR_VL;
32     static Motor MOTOR_VR;
33     static Motor MOTOR_HL;
34     static Motor MOTOR_HR;
35
36
37     /** Amount of values down from 1023 which can be used */
38     static const int USEABLE_UPPER_LIMIT = 700;
39     /** Wont move below this limit */
40     static const int CONTROLLER_LOWER_LIMIT = 50;
41
42     static const char PIN_MOTOR_VL_EN = 23;
43     static const char PIN_MOTOR_VL_DIR1 = 18;
44     static const char PIN_MOTOR_VL_DIR2 = 19;
45
46     static const char PIN_MOTOR_VR_EN = 32;
47     static const char PIN_MOTOR_VR_DIR1 = 12;
```

```

48     static const char PIN_MOTOR_VR_DIR2 = 13;
49
50     static const char PIN_MOTOR_HL_EN = 16;
51     static const char PIN_MOTOR_HL_DIR1 = 17;
52     static const char PIN_MOTOR_HL_DIR2 = 5;
53
54     static const char PIN_MOTOR_HR_EN = 4;
55     static const char PIN_MOTOR_HR_DIR1 = 3;
56     static const char PIN_MOTOR_HR_DIR2 = 15;
57
58     static const int PWM_FREQ = 500;
59     static const char CH_LED = 0;
60     static const char CH_MOTOR_VL = 1;
61     static const char CH_MOTOR_VR = 2;
62     static const char CH_MOTOR_HL = 3;
63     static const char CH_MOTOR_HR = 4;
64     static const char PWM_RESOLUTION = 10; //Resolution 8, 10, 12, 15
65
66     static void initPWM();
67
68     static void driveMotor(char channel, char dir1, char dir2, int speed);
69
70 /**
71  * Converts the given controls into wheel speeds
72  * Algorithm source:
73  * http://eprints.utm.edu.my/16543/1/Omni%20Directional%20Control%20Algorithm%20For%20Mecanu
74  */
75     static void drive(int controlFront, int controlSide, int controlTurn);
76
77 private:
78     static void driveAlgorithm(int controlFront, int controlSide, int controlTurn,
79     int *speedVL, int *speedVR, int *speedHL, int *speedHR);
78 };

```

Listing A.4: movement.h

```

1 #include "util.h"
2
3 ****
4 * high precision sine/cosine
5 *
6 * Source: https://gist.github.com/geraldyeo/988116

```

```
7  *
8  *
9  ****
10 void cossin(float x, float *outCos, float *outSin) {
11     float sin, cos;
12
13     //always wrap input angle to -PI..PI
14     if (x < -3.14159265)
15         x += 6.28318531;
16     else if (x > 3.14159265)
17         x -= 6.28318531;
18
19     //compute sine
20     if (x < 0) {
21         sin = 1.27323954 * x + .405284735 * x * x;
22
23         if (sin < 0)
24             sin = .225 * (sin * -sin - sin) + sin;
25         else
26             sin = .225 * (sin * sin - sin) + sin;
27     } else {
28         sin = 1.27323954 * x - 0.405284735 * x * x;
29
30         if (sin < 0)
31             sin = .225 * (sin * -sin - sin) + sin;
32         else
33             sin = .225 * (sin * sin - sin) + sin;
34     }
35
36     //compute cosine: sin(x + PI/2) = cos(x)
37     x += 1.57079632;
38     if (x > 3.14159265)
39         x -= 6.28318531;
40
41     if (x < 0) {
42         cos = 1.27323954 * x + 0.405284735 * x * x;
43
44         if (cos < 0)
45             cos = .225 * (cos * -cos - cos) + cos;
46         else
47             cos = .225 * (cos * cos - cos) + cos;
48     } else {
49         cos = 1.27323954 * x - 0.405284735 * x * x;
```

```

50
51     if (cos < 0)
52         cos = .225 * (cos * -cos - cos) + cos;
53     else
54         cos = .225 * (cos * cos - cos) + cos;
55 }
56
57 *outSin = sin;
58 *outCos = cos;
59 }
60
61 /**
62 * Branchless signum function
63 */
64
65 int sgn(int val) {
66     return (0 < val) - (val < 0);
67 }

```

Listing A.5: util.cpp

```

1 #pragma once
2
3 void coassin(float x, float *outCos, float *outSin);
4
5 int sgn(int val);

```

Listing A.6: util.h

```

1 #include "MPU6050.h"
2 #include "communication.h"
3 #include "movement.h"
4 #include <Arduino.h>
5 #include <ESPAsyncWebServer.h>
6 #include <SPIFFS.h>
7 #include <WiFi.h>
8 #include <Wire.h>
9
10 // Replace with your network credentials
11 const char *ssid = "ESP32-OmniMove";

```

```
12 const char *password = "123456789";
13
14 // const byte DNS_PORT = 53;
15 const IPAddress apIP = IPAddress(192, 168, 4, 1);
16
17 AsyncWebServer server(80);
18 AsyncWebSocket ws("/ws");
19 MPU6050 mpu;
20
21 void onEvent(AsyncWebSocket *server, AsyncWebSocketClient *client, AwsEventType
22 type, void *arg, uint8_t *data, size_t len) {
23
24     switch (type) {
25         case WS_EVT_CONNECT:
26             Serial.printf("Client connected from %s\n",
27 client->remoteIP().toString().c_str());
28             break;
29
30         case WS_EVT_DATA:
31             Communication::onWSData(server, client, type, data, len);
32             break;
33
34     }
35 }
36
37 void setup() {
38     // enableCore1WDT();
39     Serial.begin(115200);
40
41     Wire.begin();
42
43     mpu.initialize();
44
45     // mpu.CalibrateAccel_MP6500(6);
46     // mpu.CalibrateGyro(6);
47
48     // mpu.PrintActiveOffsets_MP6500();
49     mpu.setXGyroOffset(96);
50     mpu.setYGyroOffset(92);
51     mpu.setZGyroOffset(-20);
52 }
```

```
53 // Serial.printf("\n");
54 Movement::initPWM();
55
56 // enable AP with dns
57 WiFi.mode(WIFI_AP);
58
59 // Setup websockets
60 ws.onEvent(onEvent);
61 server.addHandler(&ws);
62 Communication::ws = &ws;
63 ws.enable(true);
64
65 // Initialize SPIFFS
66 if (!SPIFFS.begin(true)) {
67     while (true) {
68         Serial.println("An Error has occurred while mounting SPIFFS");
69         delay(1000);
70     }
71     return;
72 } else {
73     Serial.println("Mounted SPIFFS successfully");
74 }
75
76 // Route for root / web page
77 server.on("/", HTTP_GET, [] (AsyncWebRequest *request) {
78     // Serial.println("request on index");
79     request->send(SPIFFS, "/index.html", String(), false, nullptr);
80 });
81
82 // Route to load style.css file
83 server.on("/style.css", HTTP_GET, [] (AsyncWebRequest *request) {
84     // Serial.println("request on style");
85     request->send(SPIFFS, "/style.css", "text/css");
86 });
87
88 // Route to load code.js file
89 server.on("/code.js", HTTP_GET, [] (AsyncWebRequest *request) {
90     // Serial.println("request on code");
91     request->send(SPIFFS, "/code.js", "text/javascript");
92 });
93
94 // Route to load code.js file
95 server.on("/progressbar.min.js", HTTP_GET, [] (AsyncWebRequest *request) {
```

```

96     // Serial.println("request on progressbar");
97     request->send(SPIFFS, "/progressbar.min.js", "text/javascript");
98 }
99
100 // WiFi.softAPConfig(apIP, apIP, IPAddress(255, 255, 255, 0));
101 WiFi.setSleep(false);
102 WiFi.softAP(ssid, password, 6, 1);
103 delay(1000);
104
105 server.begin();
106 }
107
108 void loop() {
109
110 // put your main code here, to run repeatedly:
111 // dnsServer.processNextRequest();
112 uint16_t v = analogRead(39);
113 int16_t x, y, z, gx, gy, gz, temp;
114
115 temp = mpu.getTemperature();
116 mpu.getMotion6(&x, &y, &z, &gx, &gy, &gz);
117
118 // Serial.printf("x: %6.2fg, y: %6.2fg, z: %6.2fg, gx: %6.2f°/s, gy: %6.2f°/s,
119 // gz: %6.2f°/s, temp: %6.2f°C\r", x / 16384.0, y / 16384.0, z / 16384.0, gx /
250.0, gy / 250.0, gz / 250.0, temp / 340.0 + 36.53);
120 // power = (v / 4095 * 3.1 + .1) * 3;
121 Communication::sendCurrGyBatComb(0, 0, 0, x, y, z, gx, gy, gz, v, temp);
122 delay(1000);
123 }
```

Listing A.7: main.cpp

A.2 Webpage

```

1 <!DOCTYPE html>
2 <html>
3
4 <head>
5   <title>ESP32 Web Server</title>
6   <meta charset="utf-8" />
```

```
7  <meta name="viewport" content="width=device-width, initial-scale=1">
8  <link rel="icon" href="data:, ">
9  <link rel="stylesheet" type="text/css" href="style.css">
10 </head>
11
12 <body>
13  <h1>ESP32 Web Server</h1>
14  <p>WebSocked Connected: <strong id="wsState">CONNECTING</strong></p>
15  <p><button class="button" onclick="sendSpeed()">Test Button</button></p>
16
17
18 <div class="wrapper">
19  <div class="left">
20   <h2>Messurements</h2>
21   <p>Accelleration: <strong id="valAccelTotal">0</strong> m/s2
22   <br /> <span id="valAccel">[x: 0, y: 0, z: 0]</span>
23  </p>
24
25  <p>Speed: <strong id="valSpeedTotal">0</strong> m/s
26  <br /> <span id="valSpeed">[x: 0, y: 0, z: 0]</span></p>
27
28  <p>Rotation: <strong id="valRotTotal">0</strong> rad/s2
29  <br /> <span id="valRot">[x: 0, y: 0, z: 0]</span></p>
30
31 </div>
32
33 <div class="right">
34  <h2>Battery</h2>
35  <progress id="progBattery" value="60" max="100"></progress>
36
37  <strong id="valBatteryTotal">-1</strong>%
38  <p>Voltage <strong id="valVolt">-1</strong> V</p>
39
40  <h2>Temprature</h2>
41  <p><strong id="valTemp">-1</strong> °C</p>
42 </div>
43 </div>
44
45
46 <div>
47  <svg viewBox="0 0 205 10" preserveAspectRatio="none" style="width: 100%; height: 100%;">
```

```

48   <path id="pathVL" d="M50,2 L100,2 0,2 50,2" stroke="#FFEA82"
49   stroke-width="4" fill-opacity="0"
50   style="stroke-dasharray: 100, 100; stroke-dashoffset: 0;"/></path>
51 <path id="pathHL" d="M50,8 L100,8 0,8 50,8" stroke="#FFEA82"
52   stroke-width="4" fill-opacity="0"
53   style="stroke-dasharray: 100, 100; stroke-dashoffset: 0;"/></path>
54
55 <path id="pathVR" d="M155,2 L205,2 105,2 155,2" stroke="#FFEA82"
56   stroke-width="4" fill-opacity="0"
57   style="stroke-dasharray: 100, 100; stroke-dashoffset: 0;"/></path>
58 <path id="pathHR" d="M155,8 L205,8 105,8 155,8" stroke="#FFEA82"
59   stroke-width="4" fill-opacity="0"
60   style="stroke-dasharray: 100, 100; stroke-dashoffset: 0;"/></path>
61
62 <path d="M 50 0 L 50 4" stroke="black" stroke-width="0.5" fill="none" />
63 <path d="M 50 6 L 50 10" stroke="black" stroke-width="0.5" fill="none" />
64 <path d="M 155 0 L 155 4" stroke="black" stroke-width="0.5" fill="none" />
65 <path d="M 155 6 L 155 10" stroke="black" stroke-width="0.5" fill="none" />
66 </svg>
67 </div>
68
69 <div>
70   <h5>Select Controller</h5>
71   <button class="buttonsmall" onclick="updateControllerList()">Test
72     Controller</button>
73   <input type="checkbox" id="controllerEnabled" checked>
74   <label for="controllerEnabled">Enable Controller?</label>
75   <ul id="controllerList">
76     <li onclick="onControllerSelected(0)">Test</li>
77
78   </ul>
79 </div>
80
81 </body>
82
83 <script type="text/javascript" src="progressbar.min.js"></script>
84 <script type="text/javascript" src="code.js"></script>
85
86 </html>

```

Listing A.8: index.html

```
1 let ws = new WebSocket("ws://192.168.4.1/ws");
2 // let ws = new WebSocket("ws://demos.kaazing.com/echo");
3 ws.binaryType = 'arraybuffer';
4
5 let barVL = new ProgressBar.Path('#pathVL', { easing: 'easeInOut', duration: 140,
6 });
7 let barVR = new ProgressBar.Path('#pathVR', { easing: 'easeInOut', duration: 140,
8 });
9 let barHL = new ProgressBar.Path('#pathHL', { easing: 'easeInOut', duration: 140,
10 });
11 let barHR = new ProgressBar.Path('#pathHR', { easing: 'easeInOut', duration: 140,
12 });
13 barVL.set(0.1)
14 barVR.set(0.1)
15 barHL.set(0.1)
16 barHR.set(0.1)
17
18 let selectedControllerIndex = -1;
19 window.onload = function () {
20     setInterval(controllerFunc, 100);
21     updateControllerList();
22 };
23
24 //##region >>> WebSocket
25 ws.onopen = (event) => {
26     document.getElementById("wsState").innerHTML = "OPEN";
27 };
28
29 ws.onerror = (event) => {
30     document.getElementById("wsState").innerHTML = "ERROR";
31 };
32
33 ws.onclose = (event) => {
34     document.getElementById("wsState").innerHTML = "CLOSED";
35 };
36
37 ws.onmessage = function (event) {
38     // console.log("WebSocket message received:", event);
39     let dv = new DataView(event.data);
40     let id = dv.getInt16(0, true);
41
42     switch (id) {
43         case 1: //gyro
```

```

40     let a = [dv.getInt16(2, true), dv.getInt16(4, true), dv.getInt16(6, true)];
41     let s = [dv.getInt16(8, true), dv.getInt16(10, true), dv.getInt16(12, true)];
42     let r = [dv.getInt16(2, true), dv.getInt16(4, true), dv.getInt16(6, true)];
43
44     document.getElementById("valAccelTotal").innerHTML = Math.sqrt(a[0] * a[0] +
45       a[1] * a[1] + a[2] * a[2]).toFixed(2);
46     document.getElementById("valSpeedTotal").innerHTML = Math.sqrt(a[0] * s[0] +
47       s[1] * s[1] + s[2] * s[2]).toFixed(2);
48     document.getElementById("valRotTotal").innerHTML = Math.sqrt(r[0] * r[0] +
49       r[1] * r[1] + r[2] * r[2]).toFixed(2);
50
51     document.getElementById("valAccel").innerHTML = "[x: " + a[0] + ", y: " +
52       a[1] + ", z: " + a[2] + "]";
53     document.getElementById("valSpeed").innerHTML = "[x: " + s[0] + ", y: " +
54       s[1] + ", z: " + s[2] + "]";
55     document.getElementById("valRot").innerHTML = "[x: " + r[0] + ", y: " + r[1] +
56       ", z: " + r[2] + "]";
57     break;
58
59 case 2: //battery
60   let c1 = (dv.getInt16(2, true) / 4095 * 3.1 + .1) * 3;
61
62   document.getElementById("valVolt").innerHTML = c1.toFixed(2);
63   break;
64
65 case 3: //motors
66   const UPPER_LIMIT = 300;
67
68   let vl = dv.getInt16(2, true) / (1023 * 4);
69   let vr = dv.getInt16(4, true) / (1023 * 4);
70   let hl = dv.getInt16(6, true) / (1023 * 4);
71   let hr = dv.getInt16(8, true) / (1023 * 4);
72
73   barVL.animate(vl);
74   barVR.animate(vr);
75   barHL.animate(hl);
76   barHR.animate(hr);
77   break;
78
79 case 4: //gyroBatComb
80   let s1 = [dv.getInt16(2, true), dv.getInt16(4, true), dv.getInt16(6, true)];
81   let a1 = [dv.getInt16(8, true) / 16384.0, dv.getInt16(10, true) / 16384.0,
82     dv.getInt16(12, true) / 16384.0];

```

```

75     let r1 = [dv.getInt16(14, true) / 250.0, dv.getInt16(16, true) / 250.0,
76     dv.getInt16(18, true) / 250.0];
77
78     document.getElementById("valAccelTotal").innerHTML = Math.sqrt(a1[0] * a1[0]
79     + a1[1] * a1[1] + a1[2] * a1[2]).toFixed(2);
80     document.getElementById("valSpeedTotal").innerHTML = Math.sqrt(s1[0] * s1[0]
81     + s1[1] * s1[1] + s1[2] * s1[2]).toFixed(2);
82     document.getElementById("valRotTotal").innerHTML = Math.sqrt(r1[0] * r1[0] +
83     r1[1] * r1[1] + r1[2] * r1[2]).toFixed(2);
84
85     document.getElementById("valAccel").innerHTML = "[x: " + a1[0].toFixed(2) +
86     ", y: " + a1[1].toFixed(2) + ", z: " + a1[2].toFixed(2) + "J";
87     document.getElementById("valSpeed").innerHTML = "[x: " + s1[0].toFixed(2) +
88     ", y: " + s1[1].toFixed(2) + ", z: " + s1[2].toFixed(2) + "J";
89     document.getElementById("valRot").innerHTML = "[x: " + r1[0].toFixed(2) + ",
90     y: " + r1[1].toFixed(2) + ", z: " + r1[2].toFixed(2) + "J";
91
92     let bat = (dv.getInt16(20, true) / 4095 * 3.1 + .1) * 3;
93     let batPercent = (bat - 6.5) / (7.8 - 6.5) * 100;
94     let temp = dv.getInt16(22, true) / 340.0 + 36.53;
95
96     document.getElementById("valVolt").innerHTML = bat.toFixed(2);
97     document.getElementById("progBattery").value = batPercent.toFixed(2);
98     document.getElementById("valBatteryTotal").innerHTML = batPercent.toFixed(0);
99
100    document.getElementById("valTemp").innerHTML = temp.toFixed(2);
101    break;
102
103    default:
104      break;
105  }
106
107  //endregion
108
109 //##region >>> Controller
110 window.addEventListener("gamepadconnected", (event) => {
111   console.log("A gamepad connected:");
112   console.log(event.gamepad);
113
114   updateControllerList();
115 });

```

```
111
112 window.addEventListener("gamepaddisconnected", (event) => {
113   console.log("A gamepad disconnected:");
114   console.log(event.gamepad);
115
116   updateControllerList();
117 });
118
119
120
121
122 // Lets the user select which connected controller to use
123 function updateControllerList() {
124   let inner = ""
125
126   let pads = navigator.getGamepads();
127   console.log(pads);
128
129   for (let i = 0; i < pads.length; i++) {
130     const element = pads[i];
131     if (element === null)
132       continue;
133     if (selectedControllerIndex == i) {
134       inner += "<li onclick=\"onControllerSelected(" + i + ")\"><strong>" +
135       element.id + "</strong></li>";
136     } else {
137       inner += "<li onclick=\"onControllerSelected(" + i + ")\">" + element.id +
138       "</li>";
139     }
140   }
141
142   document.getElementById("controllerList").innerHTML = inner;
143
144 // gets called when a Controller gets selected in the List
145 function onControllerSelected(index) {
146   selectedControllerIndex = index;
147   updateControllerList();
148
149
150 let lastX1 = 0, lastX2 = 0, lastY1 = 0, lastY2 = 0;
151 function controllerFunc() {
```

```
152 if (!document.getElementById("controllerEnabled").checked)
153     return;
154
155 let pads = navigator.getGamepads();
156 if (pads === null || pads === undefined || pads[selectedControllerIndex] === null
157     || pads[selectedControllerIndex] === undefined)
158     return;
159 x1 = Math.floor(pads[selectedControllerIndex].axes[0] * 1023);
160 y1 = -Math.floor(pads[selectedControllerIndex].axes[1] * 1023);
161 x2 = Math.floor(pads[selectedControllerIndex].axes[2] * 1023);
162 y2 = -Math.floor(pads[selectedControllerIndex].axes[3] * 1023);
163
164 if (Math.abs(x1) < 200)
165     x1 -= x1;
166 if (Math.abs(y1) < 200)
167     y1 -= y1;
168 if (Math.abs(x2) < 200)
169     x2 -= x2;
170 if (Math.abs(y2) < 200)
171     y2 -= y2;
172
173 if (lastX1 != x1 || lastY1 != y1 || lastX2 != x2 || lastY2 != y2) {
174     console.log(x1 + ", " + y1 + " : " + x2 + ", " + y2);
175     if (ws.readyState == WebSocket.OPEN) {
176         let buf = new Int16Array([0, x1, y1, x2, y2])
177         ws.send(buf);
178     }
179     lastX1 = x1;
180     lastX2 = x2;
181     lastY1 = y1;
182     lastY2 = y2;
183 }
184
185 }
186 //endregion
187
188 //#region >>> Keyboard Control
189 let dirX = 0;
190 let dirY = 0;
191 let rotX = 0;
192 //a: 65, s: 83, d:68, w:87
193 document.addEventListener('keydown', function (event) {
```

```
194 let c = event.keyCode;
195 switch (event.keyCode) {
196     case 65:
197         dirX = -1;
198         break;
199     case 68:
200         dirX = 1;
201         break;
202     case 87:
203         dirY = 1;
204         break;
205     case 83:
206         dirY = -1;
207         break;
208     case 39:
209         rotX = 1;
210         break;
211     case 37:
212         rotX = -1;
213         break;
214     case 27:
215         dirY = 0;
216         dirX = 0;
217         break;
218     default:
219         break;
220 }
221
222 if (lastX1 != dirX * 1023 || lastY1 != dirY * 1023 || lastX2 != rotX * 1023 ||
223     lastY2 != 0) {
224     console.log(dirX * 1023 + ", " + dirY * 1023 + " : " + rotX * 1023 + ", " + 0);
225     if (ws.readyState == WebSocket.OPEN) {
226         let buf = new Int16Array([0, dirX * 1023, dirY * 1023, rotX * 1023, 0])
227         ws.send(buf);
228     }
229     lastX1 = dirX * 1023;
230     lastY1 = dirY * 1023;
231     lastX2 = rotX * 1023;
232     lastY2 = 0;
233
234 // console.log("x: " + dirX + " y: " + dirY);
235});
```

```
236
237 document.addEventListener('keyup', function (event) {
238     let c = event.keyCode;
239     switch (event.keyCode) {
240         case 65:
241         case 68:
242             dirX = 0;
243             break;
244         case 87:
245         case 83:
246             dirY = 0;
247             break;
248         case 39:
249         case 37:
250             rotX = 0;
251             break;
252         default:
253             break;
254     }
255     console.log("x: " + dirX + " y: " + dirY);
256
257     if (lastX1 != dirX * 1023 || lastY1 != dirY * 1023 || lastX2 != rotX * 1023 ||
258         lastY2 != 0) {
259         console.log(dirX * 1023 + ", " + dirY * 1023 + " : " + rotX * 1023 + ", " + 0);
260         if (ws.readyState == WebSocket.OPEN) {
261             let buf = new Int16Array([0, dirX * 1023, dirY * 1023, rotX * 1023, 0])
262             ws.send(buf);
263         }
264         lastX1 = dirX * 1023;
265         lastY1 = dirY * 1023;
266         lastX2 = rotX * 1023;
267         lastY2 = 0;
268     }
269 });
270
271
272
273 function sendSpeed() {
274     console.log("speed");
275     let arr = new Int16Array([2, Math.random() * 4048, Math.random() * 4048]);
276     let arr2 = new Int16Array([1,
277         Math.random() * 1023, Math.random() * 1023, Math.random() * 1023,
```

```
278     Math.random() * 1023, Math.random() * 1023, Math.random() * 1023,
279     Math.random() * 1023, Math.random() * 1023, Math.random() * 1023]);
280     ws.send(arr);
281     ws.send(arr2);
282 }
```

Listing A.9: code.js

```
1 html {
2     font-family: Helvetica;
3     display: inline-block;
4     margin: 0px auto;
5     text-align: center;
6 }
7 h1{
8     color: #0F3376;
9     padding: 2vh;
10 }
11 p{
12     font-size: 1.5rem;
13 }
14 .button {
15     display: inline-block;
16     background-color: #008CBA;
17     border: none;
18     border-radius: 4px;
19     color: white;
20     padding: 16px 40px;
21     text-decoration: none;
22     font-size: 30px;
23     margin: 2px;
24     cursor: pointer;
25 }
26
27 .buttonsmall {
28     display: inline-block;
29     background-color: #008CBA;
30     border: none;
31     border-radius: 4px;
32     color: white;
33     padding: 8px 20px;
34     text-decoration: none;
```

```
35     font-size: 16px;  
36     margin: 2px;  
37     cursor: pointer;  
38 }  
39  
40 .button2 {  
41     background-color: #f44336;  
42 }  
43  
44 .wrapper {  
45     display: flex;  
46 }  
47  
48 .left {  
49     flex: 0 0 65%;  
50 }  
51  
52 .right {  
53     flex: 1;  
54 }
```

Listing A.10: style.css

A.3 Java

```
1 package main;  
2 import Toaster.Toaster;  
3 import Utils.*;  
4  
5 import java.awt.*;  
6 import java.awt.event.*;  
7 import javax.swing.*;  
8  
9 public class ConnectUI extends JFrame {  
10  
11     /**  
12      *  
13      */  
14     private static final long serialVersionUID = 1L;  
15 }
```

```
16     private final Toaster toaster;
17
18     public static void main(String[] args) {
19         new ConnectUI();
20     }
21
22     private ConnectUI() {
23         JPanel mainJPanel = getMainJPanel();
24
25         addLogo(mainJPanel);
26
27         addSeparator(mainJPanel);
28
29         addIPTextField(mainJPanel);
30
31         addPortTextField(mainJPanel);
32
33         addConnectButton(mainJPanel);
34
35         getContentPane().add(mainJPanel);
36
37         pack();
38         setVisible(true);
39         toFront();
40
41         Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
42         setLocation(screenSize.width / 2 - getWidth() / 2, screenSize.height / 2 -
43         getHeight() / 2);
44
45         toaster = new Toaster(mainJPanel);
46     }
47
48     private JPanel getMainJPanel() {
49         setUndecorated(true);
50
51         Dimension size = new Dimension(800, 400);
52
53         JPanel panel1 = new JPanel();
54         panel1.setSize(size);
55         panel1.setPreferredSize(size);
56         panel1.setBackground(UIUtils.COLOR_BACKGROUND);
57         panel1.setLayout(null);
```

```
58     TextFieldPort titel = new TextFieldPort();
59     titel.setEnabled(false);
60     titel.setEditable(false);
61     titel.setHorizontalAlignment(SwingConstants.CENTER);
62     titel.setBorderColor(UIUtils.COLOR_INTERACTIVE);
63     titel.setFont(new Font("Segoe UI", Font.PLAIN, 16));
64     titel.setText("OmniMove Wlan Verbindung");
65     titel.setBounds(68, 57, 250, 44);
66     panel1.add(titel);
67
68     panel1.setFocusable(true);
69     panel1.requestFocus();
70
71     MouseAdapter ma = new MouseAdapter() {
72         int lastX, lastY;
73
74         @Override
75         public void mousePressed(MouseEvent e) {
76             lastX = e.getXOnScreen();
77             lastY = e.getYOnScreen();
78         }
79
80         @Override
81         public void mouseDragged(MouseEvent e) {
82             int x = e.getXOnScreen();
83             int y = e.getYOnScreen();
84             setLocation(getLocationOnScreen().x + x - lastX,
85             getLocationOnScreen().y + y - lastY);
86             lastX = x;
87             lastY = y;
88         }
89     };
90
91     panel1.addMouseListener(ma);
92     panel1.addMouseMotionListener(ma);
93
94     addWindowListener(new WindowAdapter() {
95         @Override
96         public void windowClosing(WindowEvent e) {
97             System.exit(0);
98         }
99     });

```

```
100     return panel1;
101 }
102
103 private void addSeparator(JPanel panel1) {
104     JSeparator separator1 = new JSeparator();
105     separator1.setOrientation(SwingConstants.VERTICAL);
106     separator1.setForeground(UIUtils.COLOR_OUTLINE);
107     panel1.add(separator1);
108     separator1.setBounds(387, 41, 2, 316);
109 }
110
111 private void addLogo(JPanel panel1) {
112     JLabel label1 = new JLabel();
113     //label1.setSize(300,300);
114     label1.setFocusable(false);
115     label1.setBounds(57, 123, 273, 197);
116     //ImageIcon robot = new ImageIcon(new
117     ImageIcon("/robot.png").getImage().getScaledInstance(20, 20,
118     Image.SCALE_DEFAULT));
119     //label1.setIcon(robot);
120     Image robot = new
121     ImageIcon(this.getClass().getResource("/robot.png")).getImage();
122     label1.setIcon(new ImageIcon(robot));
123     panel1.add(label1);
124 }
125
126
127 private void addIPTextField(JPanel panel1) {
128     TextFieldPort usernameField = new TextFieldPort();
129     //usernameField.setBorderColor(new Color(102, 205, 170));
130
131     usernameField.setText(UIUtils.PLACEHOLDER_TEXT_IP);
132     usernameField.setForeground(UIUtils.COLOR_OUTLINE);
133     usernameField.setBorderColor(UIUtils.COLOR_OUTLINE);
134
135     usernameField.setBounds(473, 109, 250, 44);
136     usernameField.addFocusListener(new FocusListener() {
137         @Override
138         public void focusGained(FocusEvent e) {
139             if (usernameField.getText().equals(UIUtils.PLACEHOLDER_TEXT_IP)) {
140                 usernameField.setText("");
141             }
142             usernameField.setForeground(Color.white);
143             usernameField.setBorderColor(UIUtils.COLOR_INTERACTIVE);
144         }
145     });
146 }
```

```
140     }
141
142     @Override
143     public void focusLost(FocusEvent e) {
144         if (usernameField.getText().isEmpty()) {
145             usernameField.setText(UIUtils.PLACEHOLDER_TEXT_IP);
146         }
147         usernameField.setForeground(UIUtils.COLOR_OUTLINE);
148         usernameField.setBorderColor(UIUtils.COLOR_OUTLINE);
149     }
150 );
151
152     panel1.add(usernameField);
153 }
154
155 private void addPortTextField(JPanel panel1) {
156     TextFieldIP passwordField = new TextFieldIP();
157     //passwordField.setBorderColor(new Color(102, 205, 170));
158
159     passwordField.setText(UIUtils.PLACEHOLDER_TEXT_PORT);
160     passwordField.setForeground(UIUtils.COLOR_OUTLINE);
161     passwordField.setBorderColor(UIUtils.COLOR_OUTLINE);
162
163     passwordField.setBounds(473, 168, 250, 44);
164     passwordField.addFocusListener(new FocusListener() {
165         @Override
166         public void focusGained(FocusEvent e) {
167             if (passwordField.getText().equals(UIUtils.PLACEHOLDER_TEXT_PORT))
168             {
169                 passwordField.setText("");
170             }
171             passwordField.setForeground(Color.white);
172             passwordField.setBorderColor(UIUtils.COLOR_INTERACTIVE);
173         }
174
175         @Override
176         public void focusLost(FocusEvent e) {
177             if (passwordField.getText().isEmpty()) {
178                 passwordField.setText(UIUtils.PLACEHOLDER_TEXT_PORT);
179             }
180             passwordField.setForeground(UIUtils.COLOR_OUTLINE);
181             passwordField.setBorderColor(UIUtils.COLOR_OUTLINE);
182         }
183     });
184 }
```

```
182     });
183
184     passwordField.addKeyListener(new KeyAdapter() {
185         @Override
186         public void keyTyped(KeyEvent e) {
187             if (e.getKeyChar() == KeyEvent.VK_ENTER)
188                 loginEventHandler();
189         }
190     });
191
192     panel1.add(passwordField);
193 }
194
195 private void addConnectButton(JPanel panel1) {
196     final Color[] connectColors = {UIUtils.COLOR_INTERACTIVE, Color.white};
197
198     JLabel connectButton = new JLabel() {
199         /**
200          *
201          */
202         private static final long serialVersionUID = 1L;
203
204         @Override
205         protected void paintComponent(Graphics g) {
206             Graphics2D g2 = UIUtils.get2dGraphics(g);
207             super.paintComponent(g2);
208
209             Insets insets = getInsets();
210             int w = getWidth() - insets.left - insets.right;
211             int h = getHeight() - insets.top - insets.bottom;
212             g2.setColor(connectColors[0]);
213             g2.fillRoundRect(insets.left, insets.top, w, h, UIUtils.ROUNDNESS,
214             UIUtils.ROUNDNESS);
215
216             FontMetrics metrics = g2.getFontMetrics(UIUtils.FONT_GENERAL_UI);
217             int x2 = (getWidth() -
218             metrics.stringWidth(UIUtils.BUTTON_TEXT_CONNECT)) / 2;
219             int y2 = ((getHeight() - metrics.getHeight()) / 2) +
220             metrics.getAscent();
221             g2.setFont(UIUtils.FONT_GENERAL_UI);
222             g2.setColor(connectColors[1]);
223             g2.drawString(UIUtils.BUTTON_TEXT_CONNECT, x2, y2);
224         }
225     }
```

```
222     };
223
224     connectButton.addMouseListener(new MouseAdapter() {
225
226         @Override
227         public void mousePressed(MouseEvent e) {
228             loginEventHandler();
229         }
230
231         @Override
232         public void mouseEntered(MouseEvent e) {
233             connectColors[0] = UIUtils.COLOR_INTERACTIVE_DARKER;
234             connectColors[1] = UIUtils.OFFWHITE;
235             connectButton.repaint();
236         }
237
238         @Override
239         public void mouseExited(MouseEvent e) {
240             connectColors[0] = UIUtils.COLOR_INTERACTIVE;
241             connectColors[1] = Color.white;
242             connectButton.repaint();
243         }
244     });
245
246     connectButton.setBackground(UIUtils.COLOR_BACKGROUND);
247     connectButton.setBounds(473, 247, 250, 44);
248     connectButton.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
249     panel1.add(connectButton);
250 }
251
252 private void loginEventHandler() {
253     // hier Connect Code reinschreiben
254     toaster.warn("Login event");
255 }
256 }
```

Listing A.11: main/ConnectUI.java

```
1 package components;
2
3 import javax.swing.SwingUtilities;
4 import javax.swing.JFrame;
```

```
5 import javax.swing.JPanel;
6 import javax.swing.JComponent;
7
8 import java.awt.BorderLayout;
9 import java.awt.Graphics;
10 import java.awt.Color;
11 import java.awt.Dimension;
12 import javax.swing.border.EmptyBorder;
13
14 public class SpeedMeter extends JComponent {
15     private int meterWidth = 10;
16
17     private float speed = 0f;
18     private boolean breaks = false;
19     private boolean colorfade = false;
20
21     private Color background = Color.LIGHT_GRAY;
22     private Color border = Color.BLACK;
23     private Color fill = Color.GREEN;
24     private Color maxSpeedFill = Color.ORANGE;
25     private Color breakfill = Color.RED;
26
27
28     public void setSpeed(float speed) {
29         this.speed = speed;
30         repaint();
31     }
32
33     public void setBreaks(boolean breaks) {
34         this.breaks = breaks;
35         repaint();
36     }
37
38     public void setColorfade(boolean colorfade) {
39         this.colorfade = colorfade;
40     }
41
42     public void setMaxSpeedFill(Color maxSpeedFill) {
43         this.maxSpeedFill = maxSpeedFill;
44     }
45
46     public boolean isColorfade() {
47         return colorfade;
```

```
48     }
49
50     public Color getMaxSpeedFill() {
51         return maxSpeedFill;
52     }
53
54     public void toggleBreaks() {
55         setBreaks(!breaks);
56     }
57
58     public void setMeterWidth(int meterWidth) {
59         this.meterWidth = meterWidth;
60     }
61
62     public void setBackgroundFill(Color background) {
63         this.background = background;
64         repaint();
65     }
66
67     public void setBorderFill(Color border) {
68         this.border = border;
69         repaint();
70     }
71
72     public void setFill(Color fill) {
73         this.fill = fill;
74         repaint();
75     }
76
77     public void setBreakfill(Color breakfill) {
78         this.breakfill = breakfill;
79         repaint();
80     }
81
82     public Color getBackgroundFill() {
83         return background;
84     }
85
86     public Color getBorderFill() {
87         return border;
88     }
89
90     public Color getFill() {
```

```
91         return fill;
92     }
93
94     public Color getBreakfill() {
95         return breakfill;
96     }
97
98     @Override
99     protected void paintComponent(Graphics g) {
100         int w = Math.min(meterWidth, getWidth());
101         int h = getHeight();
102         int x = getWidth() / 2 - w / 2;
103         int y = 0;
104
105         g.setColor(background);
106         g.fillRect(x, y, w, h);
107
108         g.setColor(border);
109         g.drawRect(x, y, w - 1, h - 1);
110
111         int a = Math.round(speed * (h - 2));
112         if (colorfade)
113             g.setColor(new Color(Math.min(fill.getRed() +
114                 Math.round((maxSpeedFill.getRed() - fill.getRed()) * (Math.abs(speed))), 255),
115                 Math.min(fill.getGreen() + Math.round((maxSpeedFill.getGreen() -
116                 fill.getGreen()) * (Math.abs(speed))), 255),
117                 Math.min(fill.getBlue() + Math.round((maxSpeedFill.getBlue() -
118                 fill.getBlue()) * (Math.abs(speed))), 255),
119                 Math.min(fill.getAlpha() + Math.round((maxSpeedFill.getAlpha() -
120                 fill.getAlpha()) * (Math.abs(speed))), 255)));
121         else
122             g.setColor(fill);
123         if (a > 0)
124             g.fillRect(x + 1, y + h / 2 - a / 2, w - 2, a / 2);
125         else
126             g.fillRect(x + 1, y + h / 2 - 1, w - 2, -(a / 2) + 1);
127
128         int p = Math.round(h / 2);
129         g.setColor(breakfill);
130         g.drawLine(x + 1, y + h - 1 - p, x + w - 1, y + h - 1 - p);
131
132         if (breaks) {
133             g.setColor(breakfill);
134             g.fillRect(x + 1, y + h - 1 - p, w - 2, 1);
135         }
136     }
137
138     public void setBreakfill(Color c) {
139         breakfill = c;
140     }
141
142     public void setBreaks(boolean b) {
143         breaks = b;
144     }
145
146     public void setSpeed(int s) {
147         speed = s;
148     }
149
150     public void setWidth(int w) {
151         meterWidth = w;
152     }
153
154     public void setHeight(int h) {
155         meterHeight = h;
156     }
157
158     public void setAlpha(int a) {
159         alpha = a;
160     }
161
162     public void setRed(int r) {
163         red = r;
164     }
165
166     public void setGreen(int g) {
167         green = g;
168     }
169
170     public void setBlue(int b) {
171         blue = b;
172     }
173
174     public void setBackground(int r, int g, int b) {
175         background = new Color(r, g, b);
176     }
177
178     public void setBorder(int r, int g, int b) {
179         border = new Color(r, g, b);
180     }
181
182     public void setFill(int r, int g, int b) {
183         fill = new Color(r, g, b);
184     }
185
186     public void setBreakfill(int r, int g, int b) {
187         breakfill = new Color(r, g, b);
188     }
189
190     public void setColor(int r, int g, int b) {
191         color = new Color(r, g, b);
192     }
193
194     public void setAlpha(int a) {
195         alpha = a;
196     }
197
198     public void setRed(int r) {
199         red = r;
200     }
201
202     public void setGreen(int g) {
203         green = g;
204     }
205
206     public void setBlue(int b) {
207         blue = b;
208     }
209
210     public void setWidth(int w) {
211         meterWidth = w;
212     }
213
214     public void setHeight(int h) {
215         meterHeight = h;
216     }
217
218     public void setAlpha(int a) {
219         alpha = a;
220     }
221
222     public void setRed(int r) {
223         red = r;
224     }
225
226     public void setGreen(int g) {
227         green = g;
228     }
229
230     public void setBlue(int b) {
231         blue = b;
232     }
233
234     public void setBackground(int r, int g, int b) {
235         background = new Color(r, g, b);
236     }
237
238     public void setBorder(int r, int g, int b) {
239         border = new Color(r, g, b);
240     }
241
242     public void setFill(int r, int g, int b) {
243         fill = new Color(r, g, b);
244     }
245
246     public void setBreakfill(int r, int g, int b) {
247         breakfill = new Color(r, g, b);
248     }
249
250     public void setColor(int r, int g, int b) {
251         color = new Color(r, g, b);
252     }
253
254     public void setAlpha(int a) {
255         alpha = a;
256     }
257
258     public void setRed(int r) {
259         red = r;
260     }
261
262     public void setGreen(int g) {
263         green = g;
264     }
265
266     public void setBlue(int b) {
267         blue = b;
268     }
269
270     public void setWidth(int w) {
271         meterWidth = w;
272     }
273
274     public void setHeight(int h) {
275         meterHeight = h;
276     }
277
278     public void setAlpha(int a) {
279         alpha = a;
280     }
281
282     public void setRed(int r) {
283         red = r;
284     }
285
286     public void setGreen(int g) {
287         green = g;
288     }
289
290     public void setBlue(int b) {
291         blue = b;
292     }
293
294     public void setBackground(int r, int g, int b) {
295         background = new Color(r, g, b);
296     }
297
298     public void setBorder(int r, int g, int b) {
299         border = new Color(r, g, b);
300     }
301
302     public void setFill(int r, int g, int b) {
303         fill = new Color(r, g, b);
304     }
305
306     public void setBreakfill(int r, int g, int b) {
307         breakfill = new Color(r, g, b);
308     }
309
310     public void setColor(int r, int g, int b) {
311         color = new Color(r, g, b);
312     }
313
314     public void setAlpha(int a) {
315         alpha = a;
316     }
317
318     public void setRed(int r) {
319         red = r;
320     }
321
322     public void setGreen(int g) {
323         green = g;
324     }
325
326     public void setBlue(int b) {
327         blue = b;
328     }
329
330     public void setWidth(int w) {
331         meterWidth = w;
332     }
333
334     public void setHeight(int h) {
335         meterHeight = h;
336     }
337
338     public void setAlpha(int a) {
339         alpha = a;
340     }
341
342     public void setRed(int r) {
343         red = r;
344     }
345
346     public void setGreen(int g) {
347         green = g;
348     }
349
350     public void setBlue(int b) {
351         blue = b;
352     }
353
354     public void setBackground(int r, int g, int b) {
355         background = new Color(r, g, b);
356     }
357
358     public void setBorder(int r, int g, int b) {
359         border = new Color(r, g, b);
360     }
361
362     public void setFill(int r, int g, int b) {
363         fill = new Color(r, g, b);
364     }
365
366     public void setBreakfill(int r, int g, int b) {
367         breakfill = new Color(r, g, b);
368     }
369
370     public void setColor(int r, int g, int b) {
371         color = new Color(r, g, b);
372     }
373
374     public void setAlpha(int a) {
375         alpha = a;
376     }
377
378     public void setRed(int r) {
379         red = r;
380     }
381
382     public void setGreen(int g) {
383         green = g;
384     }
385
386     public void setBlue(int b) {
387         blue = b;
388     }
389
390     public void setWidth(int w) {
391         meterWidth = w;
392     }
393
394     public void setHeight(int h) {
395         meterHeight = h;
396     }
397
398     public void setAlpha(int a) {
399         alpha = a;
400     }
401
402     public void setRed(int r) {
403         red = r;
404     }
405
406     public void setGreen(int g) {
407         green = g;
408     }
409
410     public void setBlue(int b) {
411         blue = b;
412     }
413
414     public void setBackground(int r, int g, int b) {
415         background = new Color(r, g, b);
416     }
417
418     public void setBorder(int r, int g, int b) {
419         border = new Color(r, g, b);
420     }
421
422     public void setFill(int r, int g, int b) {
423         fill = new Color(r, g, b);
424     }
425
426     public void setBreakfill(int r, int g, int b) {
427         breakfill = new Color(r, g, b);
428     }
429
430     public void setColor(int r, int g, int b) {
431         color = new Color(r, g, b);
432     }
433
434     public void setAlpha(int a) {
435         alpha = a;
436     }
437
438     public void setRed(int r) {
439         red = r;
440     }
441
442     public void setGreen(int g) {
443         green = g;
444     }
445
446     public void setBlue(int b) {
447         blue = b;
448     }
449
450     public void setWidth(int w) {
451         meterWidth = w;
452     }
453
454     public void setHeight(int h) {
455         meterHeight = h;
456     }
457
458     public void setAlpha(int a) {
459         alpha = a;
460     }
461
462     public void setRed(int r) {
463         red = r;
464     }
465
466     public void setGreen(int g) {
467         green = g;
468     }
469
470     public void setBlue(int b) {
471         blue = b;
472     }
473
474     public void setBackground(int r, int g, int b) {
475         background = new Color(r, g, b);
476     }
477
478     public void setBorder(int r, int g, int b) {
479         border = new Color(r, g, b);
480     }
481
482     public void setFill(int r, int g, int b) {
483         fill = new Color(r, g, b);
484     }
485
486     public void setBreakfill(int r, int g, int b) {
487         breakfill = new Color(r, g, b);
488     }
489
490     public void setColor(int r, int g, int b) {
491         color = new Color(r, g, b);
492     }
493
494     public void setAlpha(int a) {
495         alpha = a;
496     }
497
498     public void setRed(int r) {
499         red = r;
500     }
501
502     public void setGreen(int g) {
503         green = g;
504     }
505
506     public void setBlue(int b) {
507         blue = b;
508     }
509
510     public void setWidth(int w) {
511         meterWidth = w;
512     }
513
514     public void setHeight(int h) {
515         meterHeight = h;
516     }
517
518     public void setAlpha(int a) {
519         alpha = a;
520     }
521
522     public void setRed(int r) {
523         red = r;
524     }
525
526     public void setGreen(int g) {
527         green = g;
528     }
529
530     public void setBlue(int b) {
531         blue = b;
532     }
533
534     public void setBackground(int r, int g, int b) {
535         background = new Color(r, g, b);
536     }
537
538     public void setBorder(int r, int g, int b) {
539         border = new Color(r, g, b);
540     }
541
542     public void setFill(int r, int g, int b) {
543         fill = new Color(r, g, b);
544     }
545
546     public void setBreakfill(int r, int g, int b) {
547         breakfill = new Color(r, g, b);
548     }
549
550     public void setColor(int r, int g, int b) {
551         color = new Color(r, g, b);
552     }
553
554     public void setAlpha(int a) {
555         alpha = a;
556     }
557
558     public void setRed(int r) {
559         red = r;
560     }
561
562     public void setGreen(int g) {
563         green = g;
564     }
565
566     public void setBlue(int b) {
567         blue = b;
568     }
569
570     public void setWidth(int w) {
571         meterWidth = w;
572     }
573
574     public void setHeight(int h) {
575         meterHeight = h;
576     }
577
578     public void setAlpha(int a) {
579         alpha = a;
580     }
581
582     public void setRed(int r) {
583         red = r;
584     }
585
586     public void setGreen(int g) {
587         green = g;
588     }
589
590     public void setBlue(int b) {
591         blue = b;
592     }
593
594     public void setBackground(int r, int g, int b) {
595         background = new Color(r, g, b);
596     }
597
598     public void setBorder(int r, int g, int b) {
599         border = new Color(r, g, b);
600     }
601
602     public void setFill(int r, int g, int b) {
603         fill = new Color(r, g, b);
604     }
605
606     public void setBreakfill(int r, int g, int b) {
607         breakfill = new Color(r, g, b);
608     }
609
610     public void setColor(int r, int g, int b) {
611         color = new Color(r, g, b);
612     }
613
614     public void setAlpha(int a) {
615         alpha = a;
616     }
617
618     public void setRed(int r) {
619         red = r;
620     }
621
622     public void setGreen(int g) {
623         green = g;
624     }
625
626     public void setBlue(int b) {
627         blue = b;
628     }
629
630     public void setWidth(int w) {
631         meterWidth = w;
632     }
633
634     public void setHeight(int h) {
635         meterHeight = h;
636     }
637
638     public void setAlpha(int a) {
639         alpha = a;
640     }
641
642     public void setRed(int r) {
643         red = r;
644     }
645
646     public void setGreen(int g) {
647         green = g;
648     }
649
650     public void setBlue(int b) {
651         blue = b;
652     }
653
654     public void setBackground(int r, int g, int b) {
655         background = new Color(r, g, b);
656     }
657
658     public void setBorder(int r, int g, int b) {
659         border = new Color(r, g, b);
660     }
661
662     public void setFill(int r, int g, int b) {
663         fill = new Color(r, g, b);
664     }
665
666     public void setBreakfill(int r, int g, int b) {
667         breakfill = new Color(r, g, b);
668     }
669
670     public void setColor(int r, int g, int b) {
671         color = new Color(r, g, b);
672     }
673
674     public void setAlpha(int a) {
675         alpha = a;
676     }
677
678     public void setRed(int r) {
679         red = r;
680     }
681
682     public void setGreen(int g) {
683         green = g;
684     }
685
686     public void setBlue(int b) {
687         blue = b;
688     }
689
690     public void setWidth(int w) {
691         meterWidth = w;
692     }
693
694     public void setHeight(int h) {
695         meterHeight = h;
696     }
697
698     public void setAlpha(int a) {
699         alpha = a;
700     }
701
702     public void setRed(int r) {
703         red = r;
704     }
705
706     public void setGreen(int g) {
707         green = g;
708     }
709
710     public void setBlue(int b) {
711         blue = b;
712     }
713
714     public void setBackground(int r, int g, int b) {
715         background = new Color(r, g, b);
716     }
717
718     public void setBorder(int r, int g, int b) {
719         border = new Color(r, g, b);
720     }
721
722     public void setFill(int r, int g, int b) {
723         fill = new Color(r, g, b);
724     }
725
726     public void setBreakfill(int r, int g, int b) {
727         breakfill = new Color(r, g, b);
728     }
729
730     public void setColor(int r, int g, int b) {
731         color = new Color(r, g, b);
732     }
733
734     public void setAlpha(int a) {
735         alpha = a;
736     }
737
738     public void setRed(int r) {
739         red = r;
740     }
741
742     public void setGreen(int g) {
743         green = g;
744     }
745
746     public void setBlue(int b) {
747         blue = b;
748     }
749
750     public void setWidth(int w) {
751         meterWidth = w;
752     }
753
754     public void setHeight(int h) {
755         meterHeight = h;
756     }
757
758     public void setAlpha(int a) {
759         alpha = a;
760     }
761
762     public void setRed(int r) {
763         red = r;
764     }
765
766     public void setGreen(int g) {
767         green = g;
768     }
769
770     public void setBlue(int b) {
771         blue = b;
772     }
773
774     public void setBackground(int r, int g, int b) {
775         background = new Color(r, g, b);
776     }
777
778     public void setBorder(int r, int g, int b) {
779         border = new Color(r, g, b);
780     }
781
782     public void setFill(int r, int g, int b) {
783         fill = new Color(r, g, b);
784     }
785
786     public void setBreakfill(int r, int g, int b) {
787         breakfill = new Color(r, g, b);
788     }
789
790     public void setColor(int r, int g, int b) {
791         color = new Color(r, g, b);
792     }
793
794     public void setAlpha(int a) {
795         alpha = a;
796     }
797
798     public void setRed(int r) {
799         red = r;
800     }
801
802     public void setGreen(int g) {
803         green = g;
804     }
805
806     public void setBlue(int b) {
807         blue = b;
808     }
809
810     public void setWidth(int w) {
811         meterWidth = w;
812     }
813
814     public void setHeight(int h) {
815         meterHeight = h;
816     }
817
818     public void setAlpha(int a) {
819         alpha = a;
820     }
821
822     public void setRed(int r) {
823         red = r;
824     }
825
826     public void setGreen(int g) {
827         green = g;
828     }
829
830     public void setBlue(int b) {
831         blue = b;
832     }
833
834     public void setBackground(int r, int g, int b) {
835         background = new Color(r, g, b);
836     }
837
838     public void setBorder(int r, int g, int b) {
839         border = new Color(r, g, b);
840     }
841
842     public void setFill(int r, int g, int b) {
843         fill = new Color(r, g, b);
844     }
845
846     public void setBreakfill(int r, int g, int b) {
847         breakfill = new Color(r, g, b);
848     }
849
850     public void setColor(int r, int g, int b) {
851         color = new Color(r, g, b);
852     }
853
854     public void setAlpha(int a) {
855         alpha = a;
856     }
857
858     public void setRed(int r) {
859         red = r;
860     }
861
862     public void setGreen(int g) {
863         green = g;
864     }
865
866     public void setBlue(int b) {
867         blue = b;
868     }
869
870     public void setWidth(int w) {
871         meterWidth = w;
872     }
873
874     public void setHeight(int h) {
875         meterHeight = h;
876     }
877
878     public void setAlpha(int a) {
879         alpha = a;
880     }
881
882     public void setRed(int r) {
883         red = r;
884     }
885
886     public void setGreen(int g) {
887         green = g;
888     }
889
890     public void setBlue(int b) {
891         blue = b;
892     }
893
894     public void setBackground(int r, int g, int b) {
895         background = new Color(r, g, b);
896     }
897
898     public void setBorder(int r, int g, int b) {
899         border = new Color(r, g, b);
900     }
901
902     public void setFill(int r, int g, int b) {
903         fill = new Color(r, g, b);
904     }
905
906     public void setBreakfill(int r, int g, int b) {
907         breakfill = new Color(r, g, b);
908     }
909
910     public void setColor(int r, int g, int b) {
911         color = new Color(r, g, b);
912     }
913
914     public void setAlpha(int a) {
915         alpha = a;
916     }
917
918     public void setRed(int r) {
919         red = r;
920     }
921
922     public void setGreen(int g) {
923         green = g;
924     }
925
926     public void setBlue(int b) {
927         blue = b;
928     }
929
930     public void setWidth(int w) {
931         meterWidth = w;
932     }
933
934     public void setHeight(int h) {
935         meterHeight = h;
936     }
937
938     public void setAlpha(int a) {
939         alpha = a;
940     }
941
942     public void setRed(int r) {
943         red = r;
944     }
945
946     public void setGreen(int g) {
947         green = g;
948     }
949
950     public void setBlue(int b) {
951         blue = b;
952     }
953
954     public void setBackground(int r, int g, int b) {
955         background = new Color(r, g, b);
956     }
957
958     public void setBorder(int r, int g, int b) {
959         border = new Color(r, g, b);
960     }
961
962     public void setFill(int r, int g, int b) {
963         fill = new Color(r, g, b);
964     }
965
966     public void setBreakfill(int r, int g, int b) {
967         breakfill = new Color(r, g, b);
968     }
969
970     public void setColor(int r, int g, int b) {
971         color = new Color(r, g, b);
972     }
973
974     public void setAlpha(int a) {
975         alpha = a;
976     }
977
978     public void setRed(int r) {
979         red = r;
980     }
981
982     public void setGreen(int g) {
983         green = g;
984     }
985
986     public void setBlue(int b) {
987         blue = b;
988     }
989
990     public void setWidth(int w) {
991         meterWidth = w;
992     }
993
994     public void setHeight(int h) {
995         meterHeight = h;
996     }
997
998     public void setAlpha(int a) {
999         alpha = a;
1000    }
1001
1002    public void setRed(int r) {
1003        red = r;
1004    }
1005
1006    public void setGreen(int g) {
1007        green = g;
1008    }
1009
1010    public void setBlue(int b) {
1011        blue = b;
1012    }
1013
1014    public void setBackground(int r, int g, int b) {
1015        background = new Color(r, g, b);
1016    }
1017
1018    public void setBorder(int r, int g, int b) {
1019        border = new Color(r, g, b);
1020    }
1021
1022    public void setFill(int r, int g, int b) {
1023        fill = new Color(r, g, b);
1024    }
1025
1026    public void setBreakfill(int r, int g, int b) {
1027        breakfill = new Color(r, g, b);
1028    }
1029
1030    public void setColor(int r, int g, int b) {
1031        color = new Color(r, g, b);
1032    }
1033
1034    public void setAlpha(int a) {
1035        alpha = a;
1036    }
1037
1038    public void setRed(int r) {
1039        red = r;
1040    }
1041
1042    public void setGreen(int g) {
1043        green = g;
1044    }
1045
1046    public void setBlue(int b) {
1047        blue = b;
1048    }
1049
1050    public void setWidth(int w) {
1051        meterWidth = w;
1052    }
1053
1054    public void setHeight(int h) {
1055        meterHeight = h;
1056    }
1057
1058    public void setAlpha(int a) {
1059        alpha = a;
1060    }
1061
1062    public void setRed(int r) {
1063        red = r;
1064    }
1065
1066    public void setGreen(int g) {
1067        green = g;
1068    }
1069
1070    public void setBlue(int b) {
1071        blue = b;
1072    }
1073
1074    public void setBackground(int r, int g, int b) {
1075        background = new Color(r, g, b);
1076    }
1077
1078    public void setBorder(int r, int g, int b) {
1079        border = new Color(r, g, b);
1080    }
1081
1082    public void setFill(int r, int g, int b) {
1083        fill = new Color(r, g, b);
1084    }
1085
1086    public void setBreakfill(int r, int g, int b) {
1087        breakfill = new Color(r, g, b);
1088    }
1089
1090    public void setColor(int r, int g, int b) {
1091        color = new Color(r, g, b);
1092    }
1093
1094    public void setAlpha(int a) {
1095        alpha = a;
1096    }
1097
1098    public void setRed(int r) {
1099        red = r;
1100    }
1101
1102    public void setGreen(int g) {
1103        green = g;
1104    }
1105
1106    public void setBlue(int b) {
1107        blue = b;
1108    }
1109
1110    public void setWidth(int w) {
1111        meterWidth = w;
1112    }
1113
1114    public void setHeight(int h) {
1115        meterHeight = h;
1116    }
1117
1118    public void setAlpha(int a) {
1119        alpha = a;
1120    }
1121
1122    public void setRed(int r) {
1123        red = r;
1124    }
1125
1126    public void setGreen(int g) {
1127        green = g;
1128    }
1129
1130    public void setBlue(int b) {
1131        blue = b;
1132    }
1133
1134    public void setBackground(int r, int g, int b) {
1135        background = new Color(r, g, b);
1136    }
1137
1138    public void setBorder(int r, int g, int b) {
1139        border = new Color(r, g, b);
1140    }
1141
1142    public void setFill(int r, int g, int b) {
1143        fill = new Color(r, g, b);
1144    }
1145
1146    public void setBreakfill(int r, int g, int b) {
1147        breakfill = new Color(r, g, b);
1148    }
1149
1150    public void setColor(int r, int g, int b) {
1151        color = new Color(r, g, b);
1152    }
1153
1154    public void setAlpha(int a) {
1155        alpha = a;
1156    }
1157
1158    public void setRed(int r) {
1159        red = r;
1160    }
1161
1162    public void setGreen(int g) {
1163        green = g;
1164    }
1165
1166    public void setBlue(int b) {
1167        blue = b;
1168    }
1169
1170    public void setWidth(int w) {
1171        meterWidth = w;
1172    }
1173
1174    public void setHeight(int h) {
1175        meterHeight = h;
1176    }
1177
1178    public void setAlpha(int a) {
1179        alpha = a;
1180    }
1181
1182    public void setRed(int r) {
1183        red = r;
1184    }
1185
1186    public void setGreen(int g) {
1187        green = g;
1188    }
1189
1190    public void setBlue(int b) {
1191        blue = b;
1192    }
1193
1194    public void setBackground(int r, int g, int b) {
1195        background = new Color(r, g, b);
1196    }
1197
1198    public void setBorder(int r, int g, int b) {
1199        border = new Color(r, g, b);
1200    }
1201
1202    public void setFill(int r, int g, int b) {
1203        fill = new Color(r, g, b);
1204    }
1205
1206    public void setBreakfill(int r, int g, int b) {
1207        breakfill = new Color(r, g, b);
1208    }
1209
1210    public void setColor(int r, int g, int b) {
1211        color = new Color(r, g, b);
1212    }
1213
1214    public void setAlpha(int a) {
1215        alpha = a;
1216    }
1217
1218    public void setRed(int r) {
1219        red = r;
1220    }
1221
1222    public void setGreen(int g) {
1223        green = g;
1224    }
1225
1226    public void setBlue(int b) {
1227        blue = b;
1228    }
1229
1230    public void setWidth(int w) {
1231        meterWidth = w;
1232    }
1233
1234    public void setHeight(int h) {
1235        meterHeight = h;
1236    }
1237
1238    public void setAlpha(int a) {
1239        alpha = a;
1240    }
1241
1242    public void setRed(int r) {
1243        red = r;
1244    }
1245
1246    public void setGreen(int g) {
1247        green = g;
1248    }
1249
1250    public void setBlue(int b) {
1251        blue = b;
1252    }
1253
1254    public void setBackground(int r, int g, int b) {
1255        background = new Color(r, g, b);
1256    }
1257
1258    public void setBorder(int r, int g, int b) {
1259        border = new Color(r, g, b);
1260    }
1261
1262    public void setFill(int r, int g, int b) {
1263        fill = new Color(r, g, b);
1264    }
1265
1266    public void setBreakfill(int r, int g, int b) {
1267        breakfill = new Color(r, g, b);
1268    }
1269
1270    public void setColor(int r, int g, int b) {
1271        color = new Color(r, g, b);
1272    }
1273
1274    public void setAlpha(int a) {
1275        alpha = a;
1276    }
1277
1278    public void setRed(int r) {
1279        red = r;
1280    }
1281
1282    public void setGreen(int g) {
1283        green = g;
1284    }
1285
1286    public void setBlue(int b) {
1287        blue = b;
1288    }
1289
1290    public void setWidth(int w) {
1291        meterWidth = w;
1292    }
1293
1294    public void setHeight(int h) {
1295        meterHeight = h;
1296    }
1297
1298    public void setAlpha(int a) {
1299        alpha = a;
1300    }
1301
1302    public void setRed(int r) {
1303        red = r;
1304    }
1305
1306    public void setGreen(int g) {
1307        green = g;
1308    }
1309
1310    public void setBlue(int b) {
1311        blue = b;
1312    }
1313
1314    public void setBackground(int r, int g, int b) {
1315        background = new Color(r, g, b);
1316    }
1317
1318    public void setBorder(int r, int g, int b) {
1319        border = new Color(r, g, b);
1320    }
1321
1322    public void setFill(int r, int g, int b) {
1323        fill = new Color(r, g, b);
1324    }
1325
1326    public void setBreakfill(int r, int g, int b) {
1327        breakfill = new Color(r, g, b);
1328    }
1329
1330    public void setColor(int r, int g, int b) {
1331        color = new Color(r, g, b);
1332    }
1333
1334    public void setAlpha(int a) {
1335        alpha = a;
1336    }
1337
1338    public void setRed(int r) {
1339        red = r;
1340    }
1341
1342    public void setGreen(int g) {
1343        green = g;
1344    }
1345
1346    public void setBlue(int b) {
1347        blue = b;
1348    }
1349
1350    public void setWidth(int w) {
1351        meterWidth = w;
1352    }
1353
1354    public void setHeight(int h) {
1355        meterHeight = h;
1356    }
1357
1358    public void setAlpha(int a) {
1359        alpha = a;
1360    }
136
```

```
130         g.fillRect(x + 1, y + 1, w - 2, h / 6);
131         g.fillRect(x + 1, y + h - 1 - h / 6, w - 2, h / 6);
132     }
133 }
134
135 @Override
136 public Dimension getMinimumSize() {
137     Dimension min = super.getMinimumSize();
138     if (min.width < meterWidth)
139         min.width = meterWidth;
140     if (min.height < meterWidth)
141         min.height = meterWidth;
142     return min;
143 }
144
145 @Override
146 public Dimension getPreferredSize() {
147     Dimension pref = super.getPreferredSize();
148     pref.width = meterWidth;
149     return pref;
150 }
151
152 @Override
153 public void setPreferredSize(Dimension pref) {
154     super.setPreferredSize(pref);
155     setMeterWidth(pref.width);
156 }
157
158 public static void main(String[] args) {
159     SwingUtilities.invokeLater(() -> {
160         JFrame frame = new JFrame("Meter");
161         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
162
163         JPanel content = new JPanel(new BorderLayout());
164         content.setBorder(new EmptyBorder(25, 50, 25, 50));
165
166         SpeedMeter meter = new SpeedMeter();
167         meter.setPreferredSize(new Dimension(9, 100));
168         content.add(meter, BorderLayout.CENTER);
169
170         frame.setContentPane(content);
171         frame.pack();
172         frame.setLocationRelativeTo(null);
```

```
173         frame.setVisible(true);
174
175         new Thread(new SpeedTester(meter)).start();
176     });
177 }
178
179 static class SpeedTester implements Runnable {
180     final SpeedMeter meter;
181
182     SpeedTester(final SpeedMeter meter) {
183         this.meter = meter;
184     }
185
186     @Override
187     public void run() {
188         meter.setColorfade(true);
189         float speed = 0;
190         while (true) {
191             if (speed > 1) {
192                 speed = -1;
193                 meter.toggleBreaks();
194                 try {
195                     Thread.sleep(1000);
196                 } catch (InterruptedException e) {
197                     e.printStackTrace();
198                 }
199             } else
200                 speed += 0.01;
201             meter.setSpeed(speed);
202             meter.setBreaks(false);
203             try {
204                 Thread.sleep(20);
205             } catch (InterruptedException e) {
206                 e.printStackTrace();
207             }
208         }
209     }
210 }
211 }
```

Listing A.12: components/SpeedMeter.java

```
1 package Toaster;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.awt.event.MouseAdapter;
6 import java.awt.event.MouseEvent;
7 import java.util.ArrayList;
8 import java.util.concurrent.atomic.AtomicInteger;
9
10 public class Toaster {
11     private static final int STARTING_Y_POS = 15;
12     private static final int SPACER_DISTANCE = 15;
13     private static final ArrayList<ToasterBody> toasterBodies = new ArrayList<>();
14     private final static AtomicInteger CURRENT_Y_OFFSET = new AtomicInteger();
15     private final JPanel panelToToastOn;
16
17     public Toaster(JPanel panelToToastOn) {
18         this.panelToToastOn = panelToToastOn;
19     }
20
21     public void error(String... messages) {
22         for (String s : messages) {
23             toast(s, new Color(181, 59, 86));
24         }
25     }
26
27     public void success(String... messages) {
28         for (String s : messages) {
29             toast(s, new Color(33, 181, 83));
30         }
31     }
32
33     public void info(String... messages) {
34         for (String s : messages) {
35             toast(s, new Color(13, 116, 181));
36         }
37     }
38
39     public void warn(String... messages) {
40         for (String s : messages) {
41             toast(s, new Color(181, 147, 10));
42         }
43     }
44 }
```

```
44
45     private void toast(String message, Color bgColor) {
46         ToasterBody toasterBody;
47
48         if (toasterBodies.isEmpty()) {
49             toasterBody = new ToasterBody(panelToToastOn, message, bgColor,
50 STARTING_Y_POS);
51             CURRENT_Y_OFFSET.set(STARTING_Y_POS + toasterBody.getHeightOfToast());
52         } else {
53             toasterBody = new ToasterBody(panelToToastOn, message, bgColor,
54 CURRENT_Y_OFFSET.get() + SPACER_DISTANCE);
55             CURRENT_Y_OFFSET.addAndGet(SPACER_DISTANCE +
56 toasterBody.getHeightOfToast());
57         }
58
59         toasterBodies.add(toasterBody);
60
61         new Thread(() -> {
62             toasterBody.addMouseListener(new MouseAdapter() {
63                 @Override
64                 public void mousePressed(MouseEvent e) {
65                     removeToast(toasterBody);
66                 }
67             });
68         });
69
70         panelToToastOn.add(toasterBody, 0);
71         panelToToastOn.repaint();
72
73         try {
74             Thread.sleep(6000);
75             removeToast(toasterBody);
76         } catch (InterruptedException e) {
77             e.printStackTrace();
78         }
79     }).start();
80
81
82     private synchronized void removeToast(ToasterBody toasterBody) {
83         if (!toasterBody.getStopDisplaying()) {
84             toasterBody.setStopDisplaying(true);
85
86             toasterBodies.forEach(toasterBody1 -> {
```

```

83         if (toasterBodies.indexOf(toasterBody1) >=
84             toasterBodies.indexOf(toasterBody)) {
85             toasterBody1.setyPos(toasterBody1.getyPos() -
86             toasterBody.getHeightOfToast() - SPACER_DISTANCE);
87         }
88     });
89
89     toasterBodies.remove(toasterBody);
90
90     CURRENT_Y_OFFSET.set(CURRENT_Y_OFFSET.get() - SPACER_DISTANCE -
91     toasterBody.getHeightOfToast());
92
92     panelToToastOn.remove(toasterBody);
93     panelToToastOn.repaint();
94 }
95 }
96 }
```

Listing A.13: Toaster/Toaster.java

```

1 package Toaster;
2
3 import Utils.UIUtils;
4
5 import javax.swing.*;
6 import java.awt.*;
7
8 class ToasterBody extends JPanel {
9     /**
10      *
11     */
12     private static final long serialVersionUID = 1L;
13     private static final int TOAST_PADDING = 15;
14     private final int toastWidth;
15     private final String message;
16     private final Color c;
17     private volatile boolean stopDisplaying;
18     private int heightOfToast, stringPosX, stringPosY, yPos;
19     private JPanel panelToToastOn;
20
21     public ToasterBody(JPanel panelToToastOn, String message, Color bgColor, int
yPos) {
```

```
22     this.panelToToastOn = panelToToastOn;
23     this.message = message;
24     this.yPos = yPos;
25     this.c = bgColor;
26
27     FontMetrics metrics = getFontMetrics(UIUtils.FONT_GENERAL_UI);
28     int stringWidth = metrics.stringWidth(this.message);
29
30     toastWidth = stringWidth + (TOAST_PADDING * 2);
31     heightOfToast = metrics.getHeight() + TOAST_PADDING;
32     setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
33     setOpaque(false);
34     setBounds((panelToToastOn.getWidth() - toastWidth) / 2, (int)
35 -(Math.round(heightOfToast / 10.0) * 10), toastWidth, heightOfToast);
36
37     stringPosX = (getWidth() - stringWidth) / 2;
38     stringPosY = ((getHeight() - metrics.getHeight()) / 2) +
39 metrics.getAscent();
40
41     new Thread(() -> {
42         while (getBounds().y < yPos) {
43             int i1 = (yPos - getBounds().y) / 10;
44             i1 = i1 <= 0 ? 1 : i1;
45             setBounds((panelToToastOn.getWidth() - toastWidth) / 2,
46             getBounds().y + i1, toastWidth, heightOfToast);
47             repaint();
48             try {
49                 Thread.sleep(5);
50             } catch (Exception ignored) {
51             }
52         }
53     }).start();
54 }
55
56 @Override
57 protected void paintComponent(Graphics g) {
58     Graphics2D g2 = UIUtils.get2dGraphics(g);
59     super.paintComponent(g2);
60
61     //Background
62     g2.setColor(c);
63     g2.fillRoundRect(0, 0, getWidth(), getHeight(), UIUtils.ROUNDNESS,
64 UIUtils.ROUNDNESS);
```

```
61         // Font
62         g2.setFont(UIUtils.FONT_GENERAL_UI);
63         g2.setColor(Color.white);
64         g2.drawString(message, stringPosX, stringPosY);
65     }
66
67
68     public int getHeightOfToast() {
69         return heightOfToast;
70     }
71
72     public synchronized boolean getStopDisplaying() {
73         return stopDisplaying;
74     }
75
76     public synchronized void setStopDisplaying(boolean hasStoppedDisplaying) {
77         this.stopDisplaying = hasStoppedDisplaying;
78     }
79
80     public void setyPos(int yPos) {
81         this.yPos = yPos;
82         // setBounds((panelToToastOn.getWidth() - toastWidth) / 2, yPos,
83         // toastWidth, heightOfToast);
84
85         new Thread(() -> {
86             while (getBounds().y > yPos) {
87                 int i1 = Math.abs((yPos - getBounds().y) / 10);
88                 i1 = i1 <= 0 ? 1 : i1;
89                 setBounds((panelToToastOn.getWidth() - toastWidth) / 2,
90                         getBounds().y - i1, toastWidth, heightOfToast);
91                 repaint();
92                 try {
93                     Thread.sleep(5);
94                 } catch (Exception ignored) {
95                 }
96             }
97         }).start();
98     }
99
100    public int getyPos() {
101        return yPos;
102    }
103}
```

Listing A.14: Toaster/ToasterBody.java

```
1 package Utils;  
2  
3 import java.awt.*;  
4 import java.util.HashMap;  
5  
6 public class UIUtils {  
7     public static final Font FONT_GENERAL_UI = new Font("Segoe UI", Font.PLAIN,  
8             20);  
9  
10    public static final Color COLOR_OUTLINE = new Color(103, 112, 120);  
11    public static final Color COLOR_BACKGROUND = new Color(37, 51, 61);  
12    public static final Color COLOR_INTERACTIVE = new Color(108, 216, 158);  
13    public static final Color COLOR_INTERACTIVE_DARKER = new Color(87, 171, 127);  
14    public static final Color OFFWHITE = new Color(229, 229, 229);  
15  
16    public static final String BUTTON_TEXT_CONNECT = "Connect";  
17  
18    public static final String Bild = "robot.png";  
19  
20    public static final String PLACEHOLDER_TEXT_IP = "IP Adresse";  
21    public static final String PLACEHOLDER_TEXT_PORT = "PORT";  
22  
23    public static final int ROUNDNESS = 8;  
24  
25    public static Graphics2D get2dGraphics(Graphics g) {  
26        Graphics2D g2 = (Graphics2D) g;  
27        g2.addRenderingHints(new HashMap<RenderingHints.Key, Object>() {/**  
28             *  
29             */  
30             private static final long serialVersionUID = 1L;  
31  
32             {  
33                 put(RenderingHints.KEY_ANTIALIASING,  
34                     RenderingHints.VALUE_ANTIALIAS_ON);  
35                 put(RenderingHints.KEY_TEXT_ANTIALIASING,  
36                     RenderingHints.VALUE_TEXT_ANTIALIAS_LCD_HRGB);  
37             }  
38             return g2;  
39         }  
40     }  
41 }
```

```
36     }
37 }
```

Listing A.15: Utils/UIUtils.java

```
1 package Utils;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.awt.geom.RoundRectangle2D;
6
7
8 public class TextFieldIP extends JTextField {
9     /**
10      *
11     */
12     private static final long serialVersionUID = 1L;
13
14     private Shape shape;
15     private Color borderColor = UIUtils.COLOR_OUTLINE;
16
17     public TextFieldIP() {
18         setOpaque(false);
19         setBackground(UIUtils.COLOR_BACKGROUND);
20         setForeground(Color.white);
21         setCaretColor(Color.white);
22         setCursor(Cursor.getPredefinedCursor(Cursor.TEXT_CURSOR));
23         setMargin(new Insets(2, 10, 2, 2));
24         setHorizontalAlignment(SwingConstants.LEFT);
25         setFont(UIUtils.FONT_GENERAL_UI);
26     }
27
28     protected void paintComponent(Graphics g) {
29         Graphics2D g2 = UIUtils.get2dGraphics(g);
30         g2.setColor(getBackground());
31         g2.fillRoundRect(0, 0, getWidth() - 1, getHeight() - 1, UIUtils.ROUNDNESS,
32                         UIUtils.ROUNDNESS);
33         super.paintComponent(g2);
34     }
35
36     protected void paintBorder(Graphics g) {
37         Graphics2D g2 = UIUtils.get2dGraphics(g);
```

```

37     g2.setColor(borderColor);
38     g2.drawRoundRect(0, 0, getWidth() - 1, getHeight() - 1, UIUtils.ROUNDNESS,
39                       UIUtils.ROUNDNESS);
40   }
41
42   public boolean contains(int x, int y) {
43     if (shape == null || !shape.getBounds().equals(getBounds())) {
44       shape = new RoundRectangle2D.Float(0, 0, getWidth() - 1, getHeight() -
45           1, UIUtils.ROUNDNESS, UIUtils.ROUNDNESS);
46     }
47     return shape.contains(x, y);
48   }
49
50   public void setBorderColor(Color color) {
51     borderColor = color;
52     repaint();
53   }
54 }
```

Listing A.16: Utils/TextFieldIP.java

```

1 package Utils;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.awt.geom.RoundRectangle2D;
6
7
8 public class TextFieldPort extends JTextField {
9   /**
10    *
11   */
12   private static final long serialVersionUID = 1L;
13
14   private Shape shape;
15   private Color borderColor = UIUtils.COLOR_INTERACTIVE;
16
17   public TextFieldPort() {
18     setOpaque(false);
19     setBackground(UIUtils.COLOR_BACKGROUND);
20     setForeground(Color.white);
21     setCaretColor(Color.white);
```

```

22     setCursor(Cursor.getTextCursor());
23     setMargin(new Insets(2, 10, 2, 2));
24     setHorizontalAlignment(SwingConstants.LEFT);
25     setFont(UIUtils.FONT_GENERAL_UI);
26 }
27
28 protected void paintComponent(Graphics g) {
29     Graphics2D g2 = UIUtils.get2dGraphics(g);
30     g2.setColor(getBackground());
31     g2.fillRoundRect(0, 0, getWidth() - 1, getHeight() - 1, UIUtils.ROUNDNESS,
32     UIUtils.ROUNDNESS);
33     super.paintComponent(g2);
34 }
35
36 protected void paintBorder(Graphics g) {
37     Graphics2D g2 = UIUtils.get2dGraphics(g);
38     g2.setColor(borderColor);
39     g2.drawRoundRect(0, 0, getWidth() - 1, getHeight() - 1, UIUtils.ROUNDNESS,
40     UIUtils.ROUNDNESS);
41 }
42
43 public boolean contains(int x, int y) {
44     if (shape == null || !shape.getBounds().equals(getBounds())) {
45         shape = new RoundRectangle2D.Float(0, 0, getWidth() - 1, getHeight() -
46         1, UIUtils.ROUNDNESS, UIUtils.ROUNDNESS);
47     }
48     return shape.contains(x, y);
49 }
50
51 public void setBorderColor(Color color) {
52     borderColor = color;
53     repaint();
54 }
55
56 }

```

Listing A.17: Utils/TextFieldPort.java

```

1 package zweitesFenster;
2
3 public class Main {
4
5     public static void main(String[] args) {

```

```
6     new ZweitesFenster();  
7  
8 }  
9  
10}
```

Listing A.18: main/Main.java

```
1 package zweitesFenster;  
2  
3 import java.awt.BorderLayout;  
4 import java.awt.Dimension;  
5 import java.awt.Font;  
6 import java.awt.Image;  
7 import java.awt.Toolkit;  
8 import java.awt.event.MouseAdapter;  
9 import java.awt.event.MouseEvent;  
10 import java.awt.event.WindowAdapter;  
11 import java.awt.event.WindowEvent;  
12 import javax.swing.ImageIcon;  
13 import javax.swing.JFrame;  
14 import javax.swing.JLabel;  
15 import javax.swing.JPanel;  
16 import javax.swing.JProgressBar;  
17 import javax.swing.JSlider;  
18 import javax.swing.JTextField;  
19 import javax.swing.SwingConstants;  
20 import Utils.TextFieldPort;  
21 import Utils.UIUtils;  
22 import java.awt.Color;  
23  
24 public class ZweitesFenster extends JFrame{  
25  
26     /**  
27     *  
28     */  
29  
30     private static final long serialVersionUID = 1L;  
31  
32  
33     ZweitesFenster(){  
34 }
```

```
35
36     JPanel mainPanel = getMainJPanel();
37
38     addLogo(mainPanel);
39     battery(mainPanel);
40     speed(mainPanel);
41
42     getContentPane().add(mainPanel, BorderLayout.SOUTH);
43
44     pack();
45     setVisible(true);
46     toFront();
47
48     Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
49     setLocation(screenSize.width / 2 - getWidth() / 2, screenSize.height / 2 -
50     getHeight() / 2);
51 }
52
53
54     private JPanel getMainJPanel() {
55         setUndecorated(true);
56
57         Dimension size = new Dimension(700, 600);
58
59         JPanel panel1 = new JPanel();
60         panel1.setSize(size);
61         panel1.setPreferredSize(size);
62         panel1.setBackground(UIUtils.COLOR_BACKGROUND);
63         panel1.setLayout(null);
64
65         TextFieldPort titel = new TextFieldPort();
66         titel.setEnabled(false);
67         titel.setEditable(false);
68         titel.setHorizontalAlignment(SwingConstants.CENTER);
69         //titel.setBorderColor(UIUtils.COLOR_INTERACTIVE);
70         titel.setFont(new Font("Segoe UI", Font.PLAIN, 14));
71         titel.setText("OmniMove Driving Experience");
72         titel.setBounds(20, 25, 250, 45);
73         panel1.add(titel);
74         panel1.setFocusable(true);
75         panel1.requestFocus();
76 }
```

```
77     MouseAdapter ma = new MouseAdapter() {
78         int lastX, lastY;
79
80         @Override
81         public void mousePressed(MouseEvent e) {
82             lastX = e.getXOnScreen();
83             lastY = e.getYOnScreen();
84         }
85
86         @Override
87         public void mouseDragged(MouseEvent e) {
88             int x = e.getXOnScreen();
89             int y = e.getYOnScreen();
90             setLocation(getLocationOnScreen().x + x - lastX,
91             getLocationOnScreen().y + y - lastY);
92             lastX = x;
93             lastY = y;
94         }
95     };
96
97     panel1.addMouseListener(ma);
98     panel1.addMouseMotionListener(ma);
99
100    addWindowListener(new WindowAdapter() {
101        @Override
102        public void windowClosing(WindowEvent e) {
103            System.exit(0);
104        }
105    });
106
107    return panel1;
108
109
110
111    private void addLogo(JPanel panel1) {
112        JLabel label1 = new JLabel();
113        label1.setFocusable(false);
114        label1.setVisible(true);
115        label1.setBounds(201,94,320,369);
116        Image robot = new
117        ImageIcon(this.getClass().getResource("/robot2.png")).getImage();
118        label1.setIcon(new ImageIcon(robot));
119    }
120}
```

```
118     panel1.add(label1);
119 }
120
121
122 private void battery(JPanel panel2) {
123     int akkustand;
124     akkustand = 50;
125
126     JProgressBar battery_number = new JProgressBar();
127     battery_number.setBackground(UIUtils.COLOR_BACKGROUND);
128     battery_number.setForeground(UIUtils.COLOR_INTERACTIVE);
129     battery_number.setEnabled(false);
130     battery_number.setLocation(531, 47);
131     battery_number.setMinimum(0);
132     battery_number.setMaximum(100);
133     battery_number.setFocusable(false);
134     battery_number.setSize(133,45);
135     battery_number.setString("Akkustand: "+akkustand+" %");
136     battery_number.setStringPainted(true);
137     battery_number.setFont(new Font("Segoe UI", Font.PLAIN, 12));
138
139     //default values
140     battery_number.setValue(50);
141
142     panel2.add(battery_number);
143
144 }
145
146
147 private void speed(JPanel panel3) {
148
149     int kmh = 10;
150
151     JPanel combination = new JPanel();
152     combination.setSize(222, 101);
153     combination.setVisible(true);
154     combination.setBackground(UIUtils.COLOR_BACKGROUND);
155
156     JLabel geschwindigkeit = new JLabel();
157     geschwindigkeit.setHorizontalAlignment(SwingConstants.CENTER);
158     geschwindigkeit.setText("Geschwindigkeit");
159     geschwindigkeit.setFont(new Font("Segoe UI", Font.PLAIN, 12));
160     geschwindigkeit.setFocusable(false);
```

```
161     geschwindigkeit.setEnabled(false);  
162  
163     JSlider speed = new JSlider(JSlider.HORIZONTAL, 0, 30, 0);  
164     speed.setSize(123, 35);  
165     speed.setBackground(UIUtils.COLOR_BACKGROUND);  
166     speed.setEnabled(false);  
167     speed.setFocusable(false);  
168     speed.setLocation(292,495);  
169     speed.setMajorTickSpacing(5);  
170     speed.setMinorTickSpacing(1);  
171     speed.setPaintTicks(true);  
172     speed.setPaintLabels(true);  
173  
174     speed.setMaximum(30);  
175     speed.setMinimum(0);  
176     speed.setValue(kmh);  
177  
178     JLabel geschwindigkeit2 = new JLabel();  
179     geschwindigkeit2.setHorizontalAlignment(SwingConstants.CENTER);  
180     geschwindigkeit2.setText(" " +kmh+ " km/h");  
181     geschwindigkeit2.setVisible(true);  
182     geschwindigkeit2.setFont(new Font("Segoe UI", Font.PLAIN, 12));  
183     geschwindigkeit2.setFocusable(false);  
184     geschwindigkeit2.setEnabled(false);  
185  
186     combination.setLayout(new BorderLayout());  
187     combination.add(geschwindigkeit, BorderLayout.NORTH);  
188     combination.add(speed, BorderLayout.CENTER);  
189     combination.add(geschwindigkeit2, BorderLayout.SOUTH);  
190     combination.setLocation(264,488);  
191  
192     panel13.add(combination);  
193  
194 }  
195  
196  
197 }
```

Listing A.19: main/ZweitesFenster.java

B Technische Zeichnungen

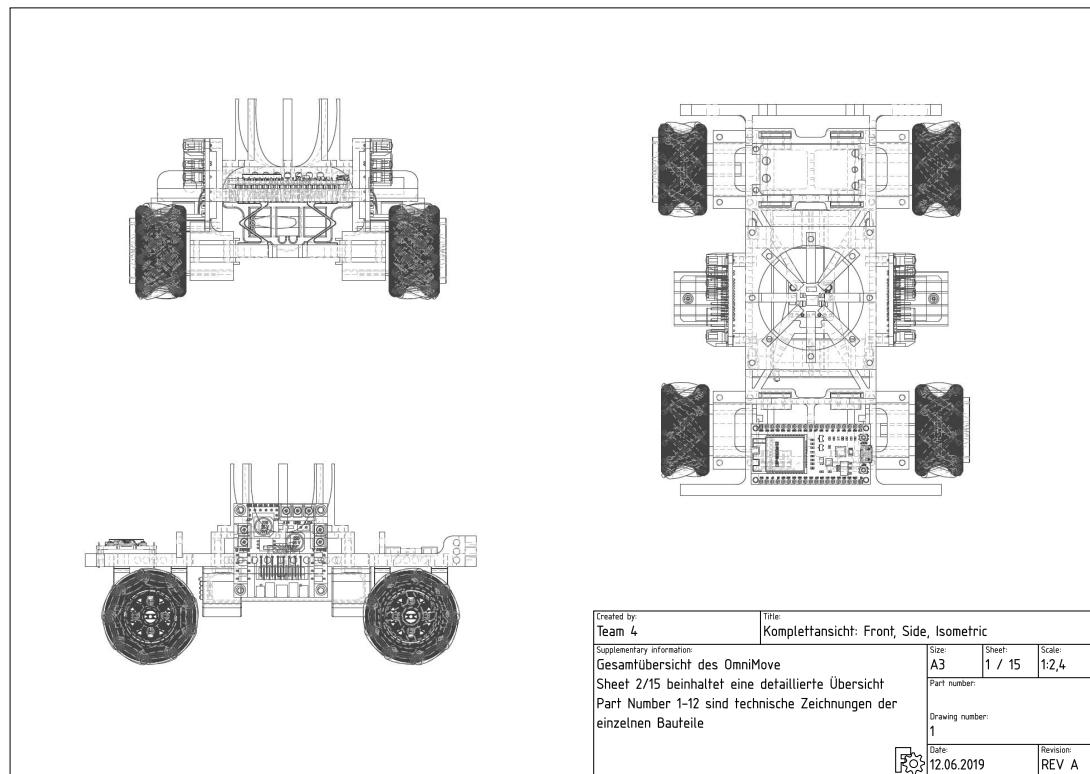


Abbildung B.1: Komplettansicht

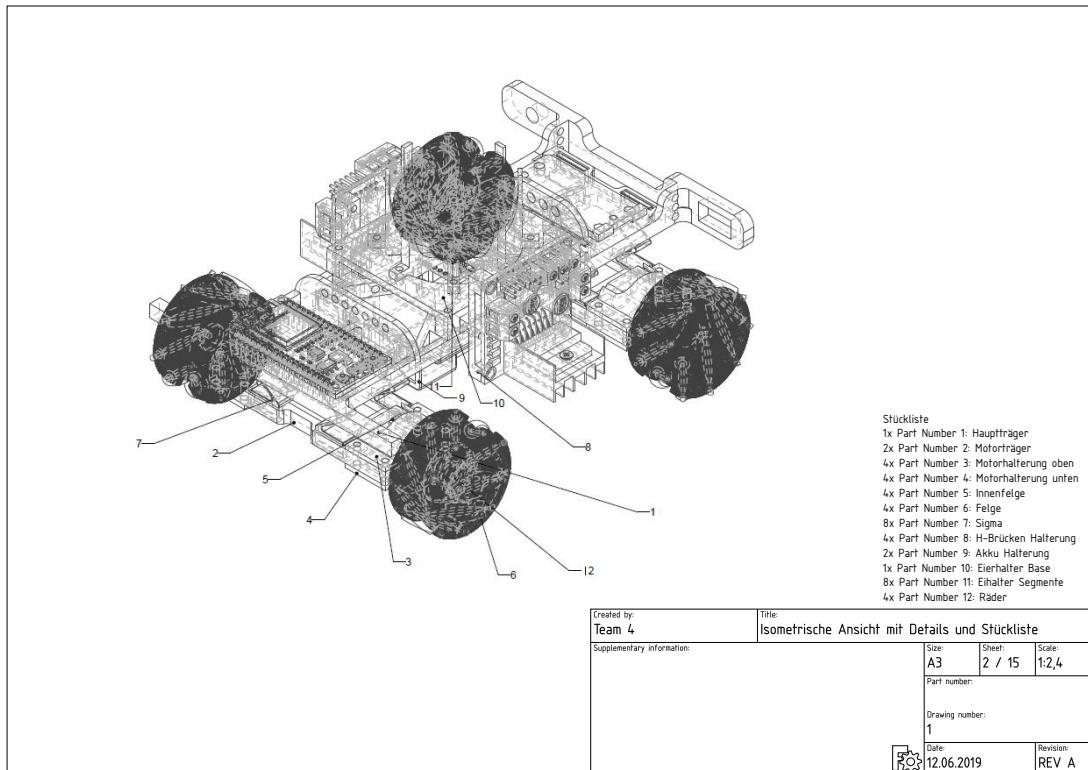


Abbildung B.2: Isometrisch und Stückliste

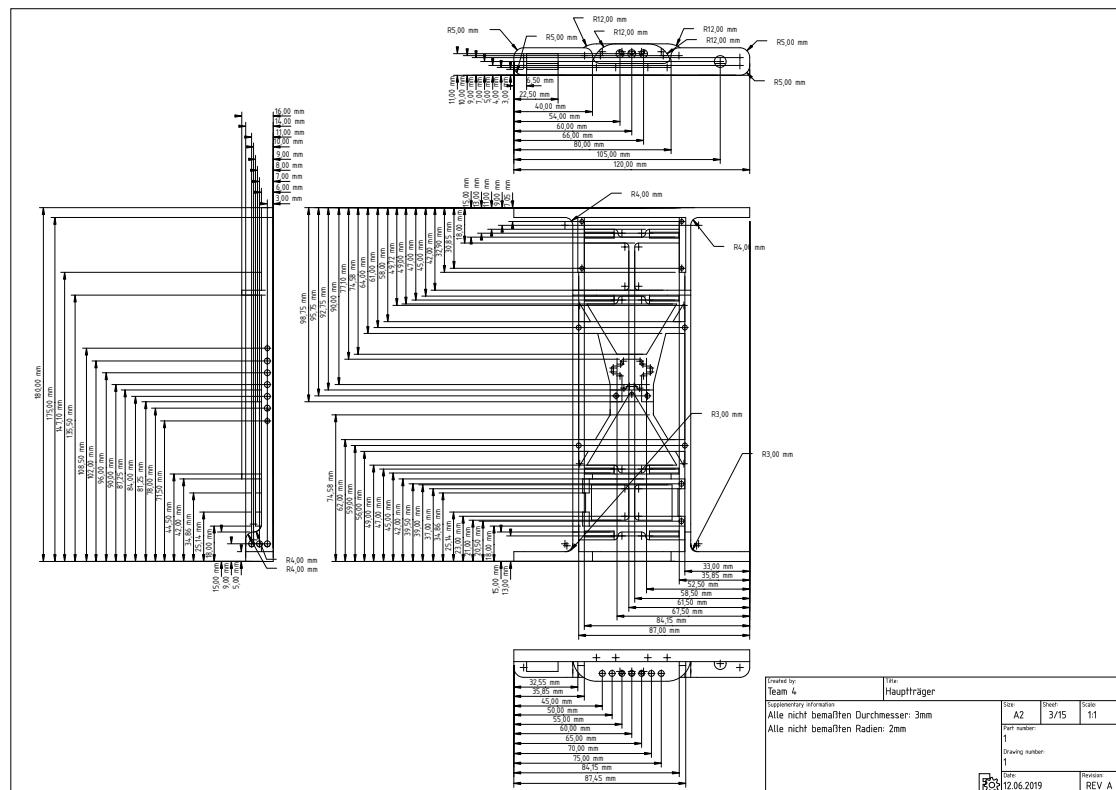


Abbildung B.3: Hauptträger

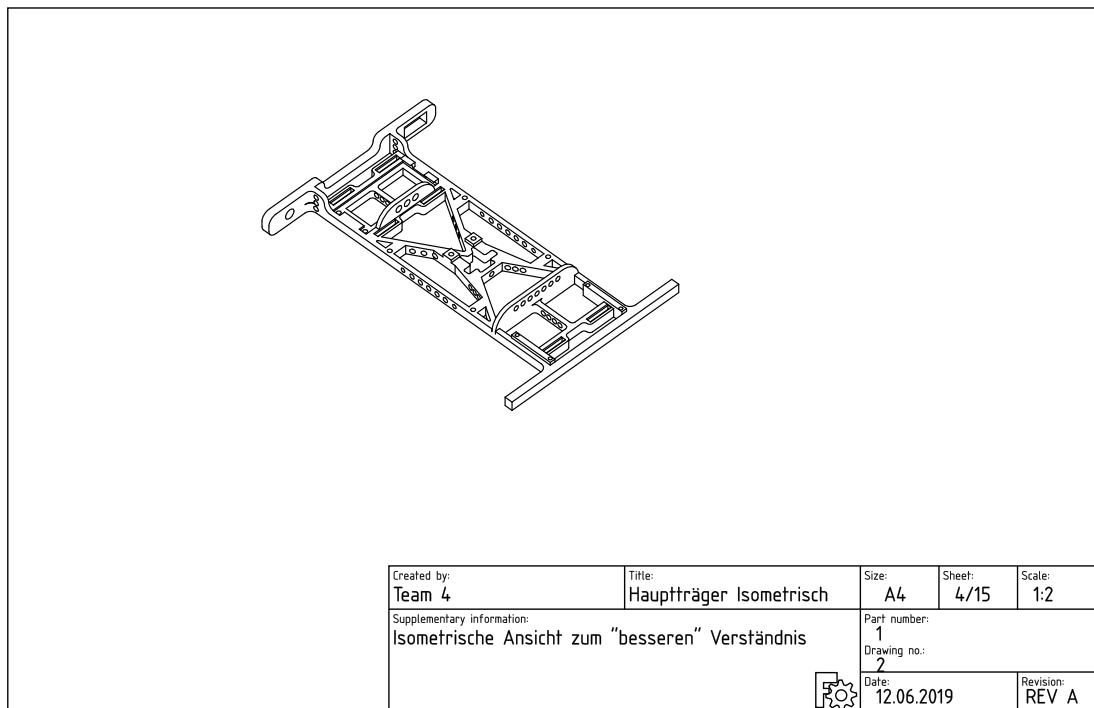


Abbildung B.4: Hauptträger Isometrisch

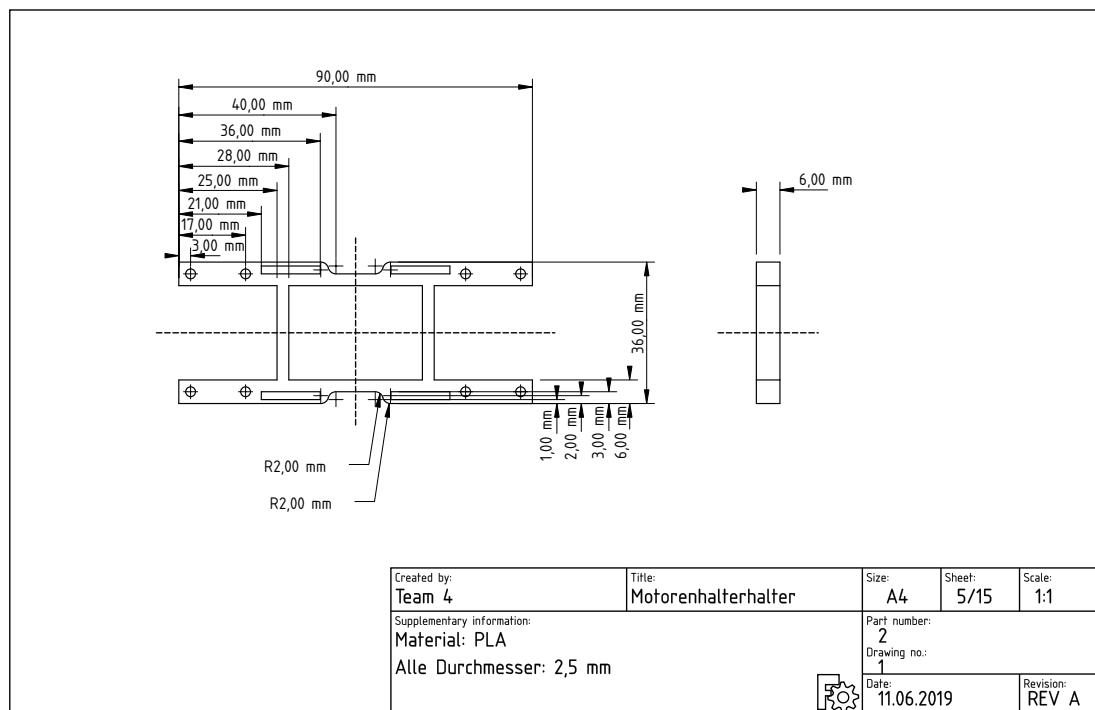


Abbildung B.5: Motorenhalterhalter

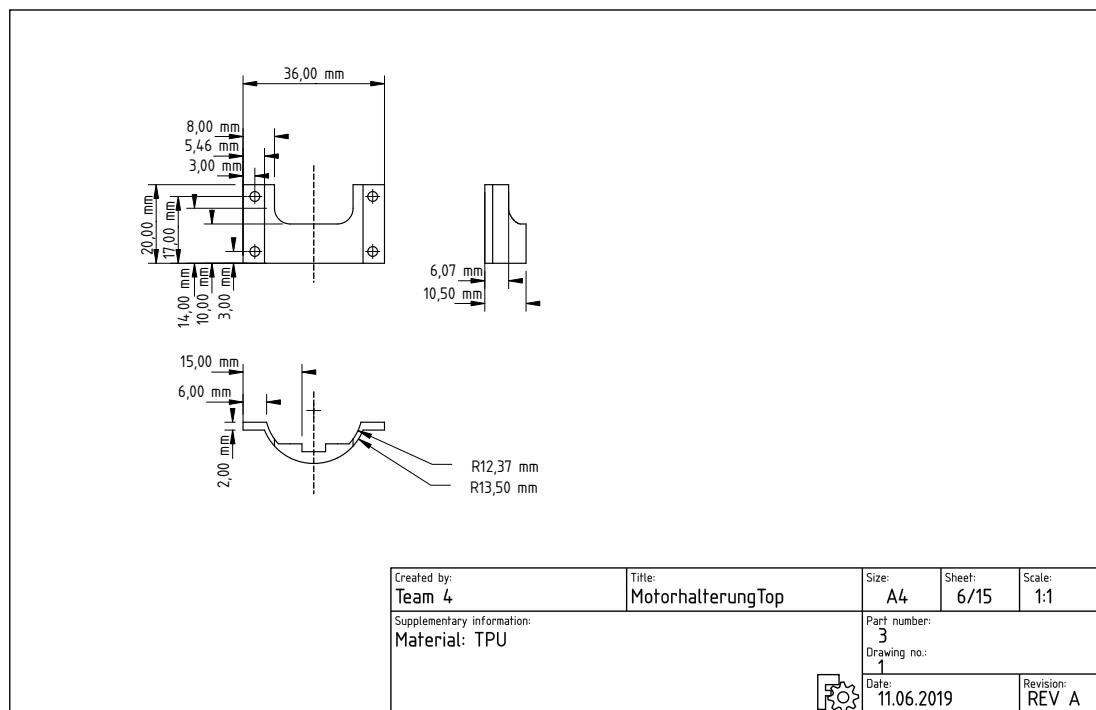


Abbildung B.6: Motorenhalterung Oben

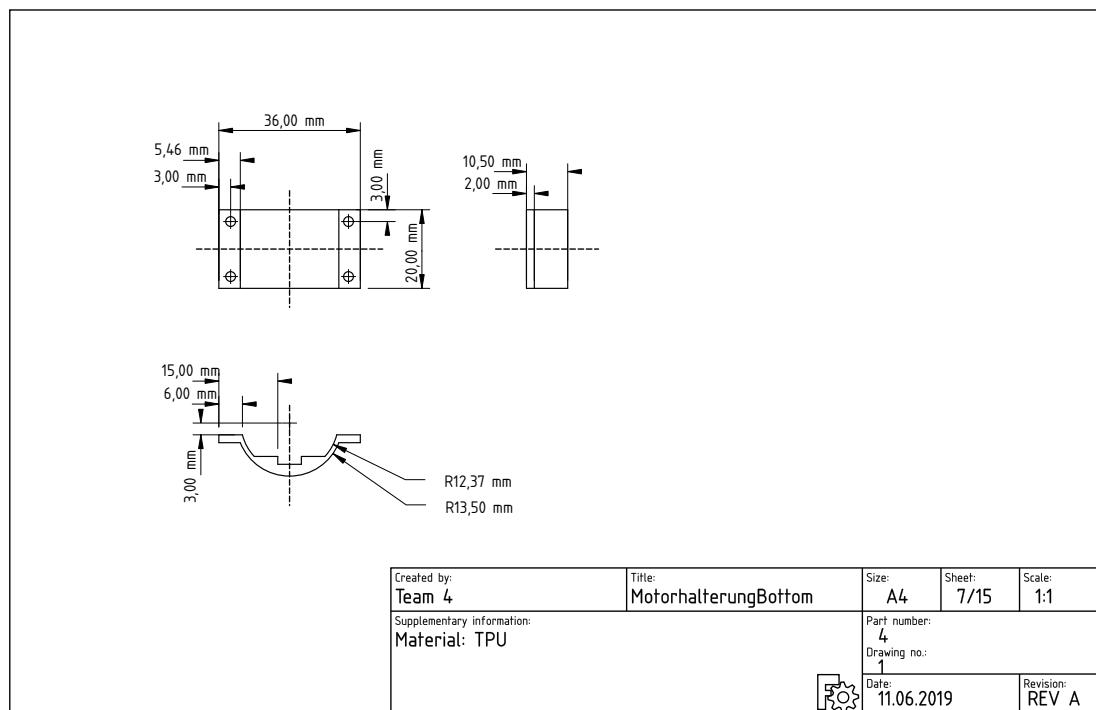


Abbildung B.7: Motorenhalterung Unten

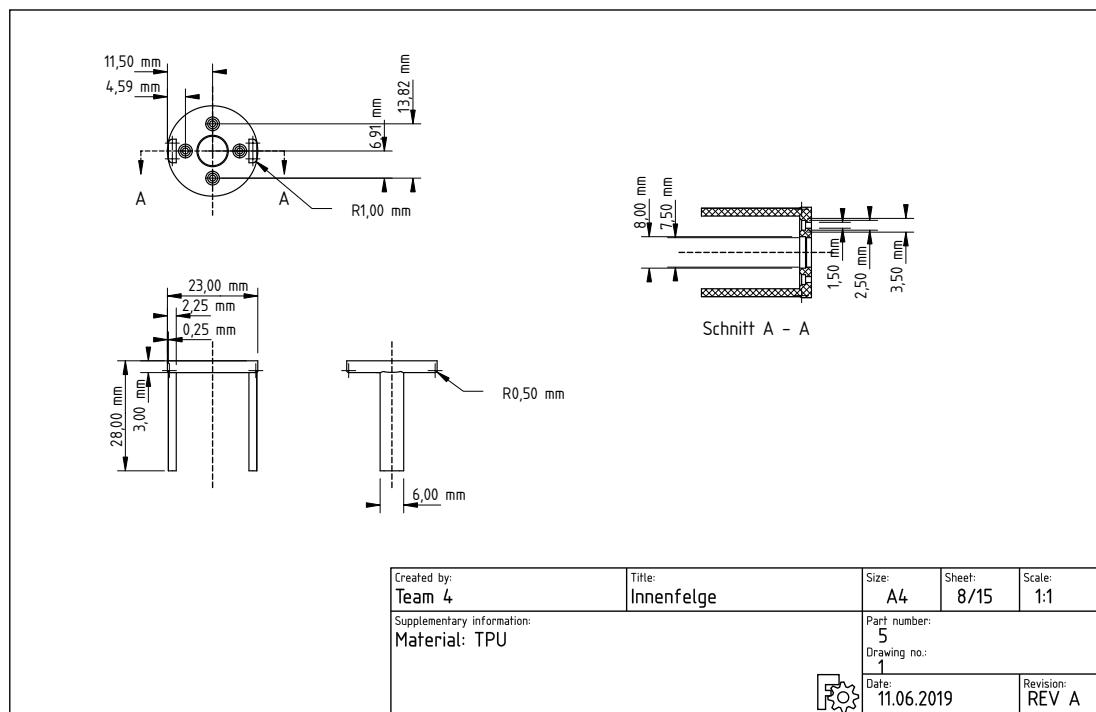


Abbildung B.8: Innenfelge

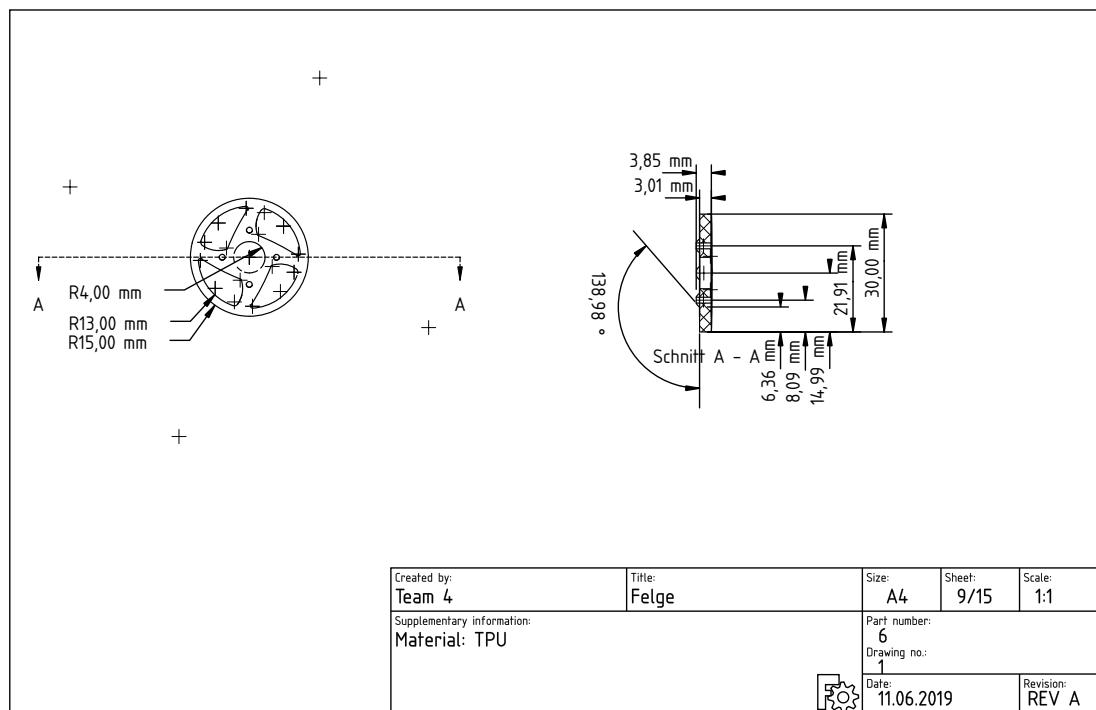


Abbildung B.9: Felge

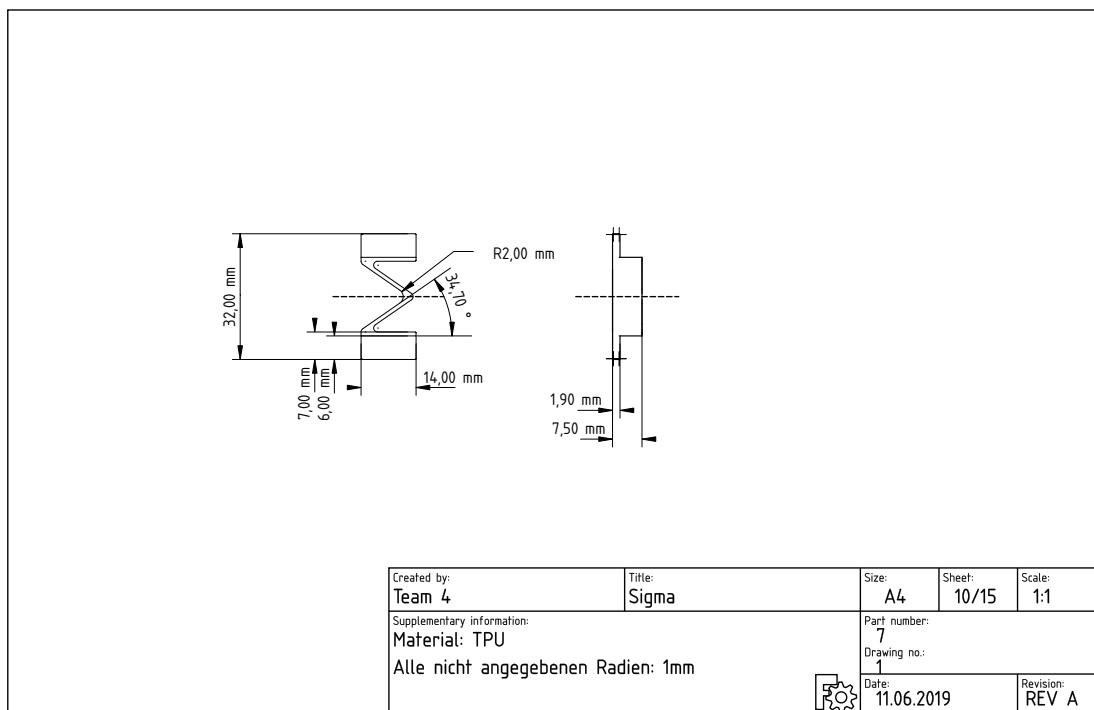


Abbildung B.10: Sigma

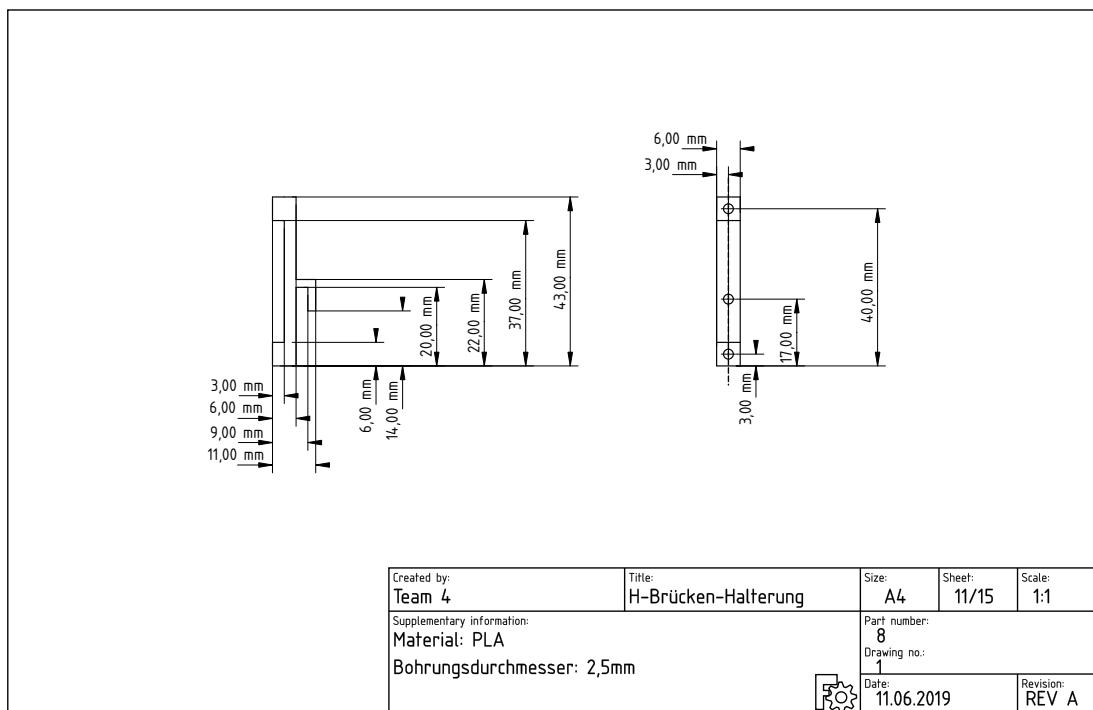


Abbildung B.11: H-Brücke-Halterung

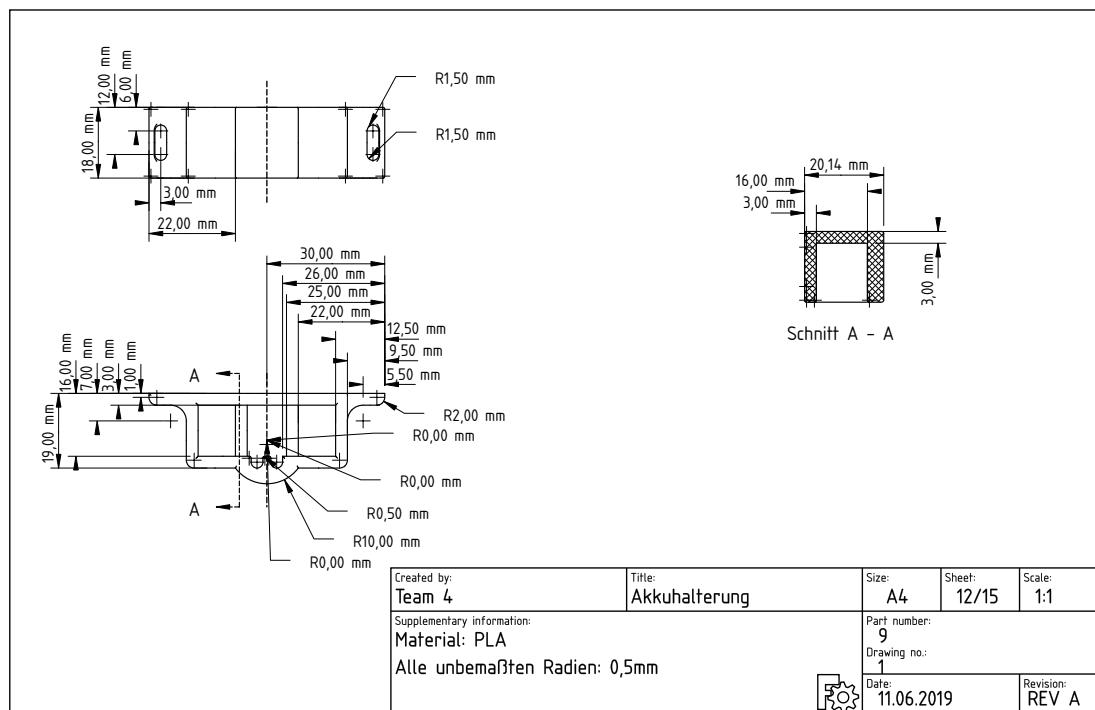


Abbildung B.12: Akkuhalterung

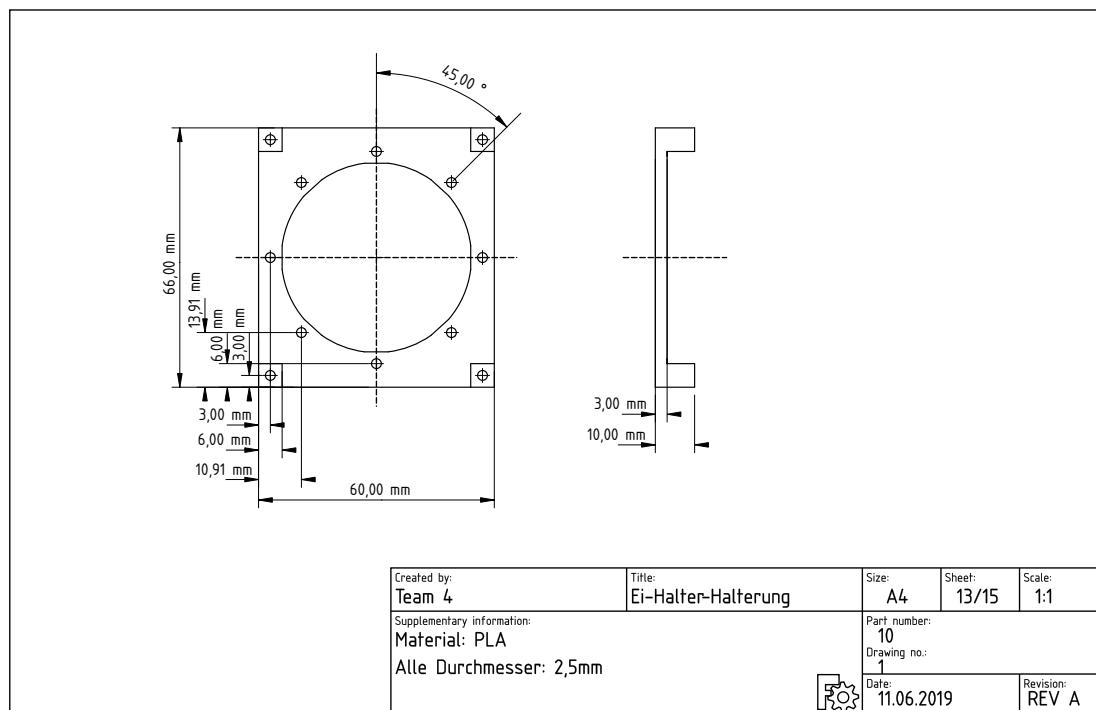


Abbildung B.13: Ei-Halterung-Halterung

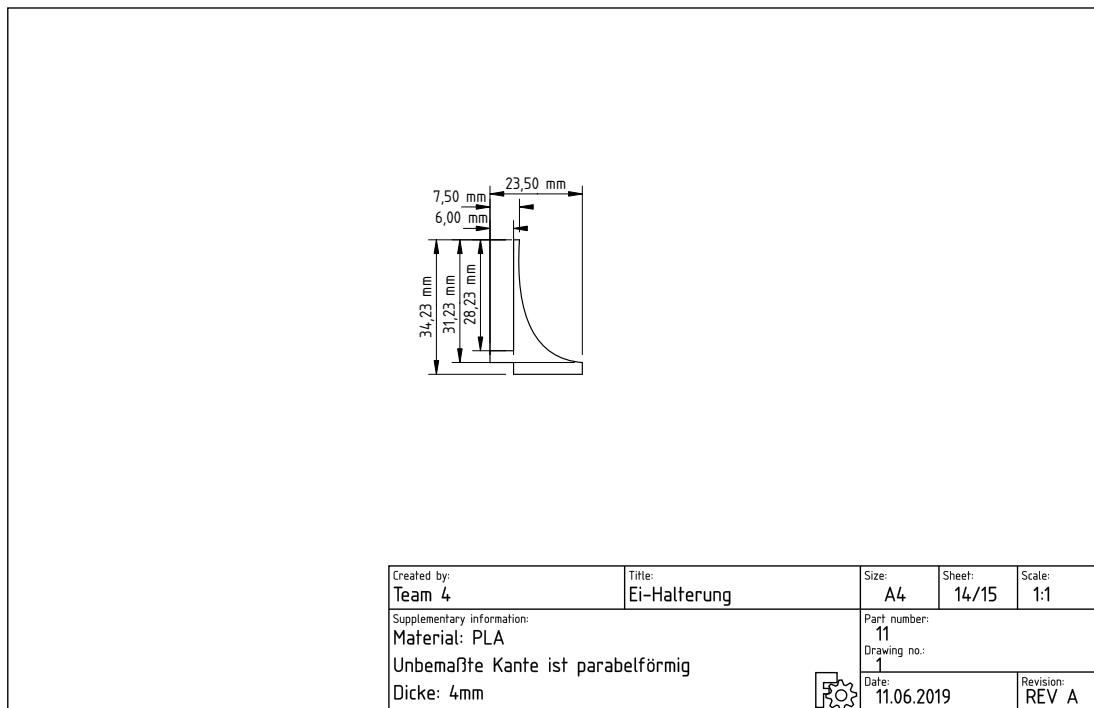


Abbildung B.14: Ei-Halterung

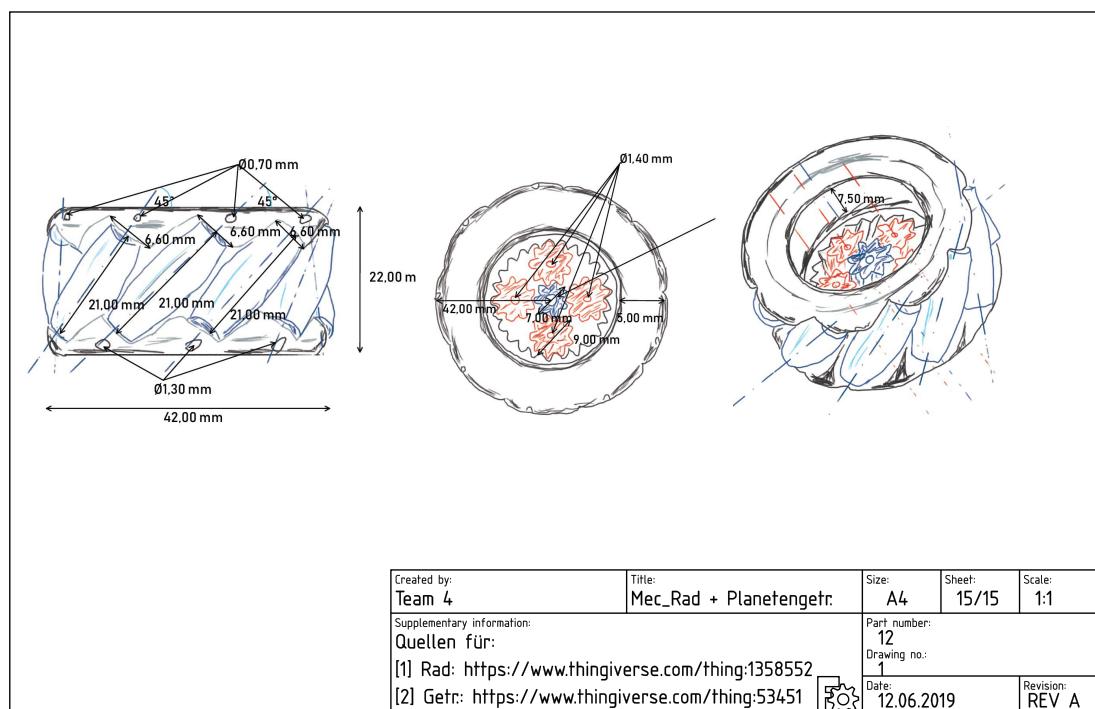


Abbildung B.15: Reifen und Planetengetriebe

Literaturverzeichnis

- [1] EMMETT: *Gear Bearing*. <https://www.thingiverse.com/thing:53451>, Februar 2016
- [2] EMMETT: *Gear Bearing*. <https://www.thingiverse.com/thing:2666785>, Februar 2016
- [3] EMMETT: *Gear Bearing*. <https://www.thingiverse.com/thing:2277105>, Februar 2016
- [4] INNOART, Jonah: *44mm Mecanum Wheel*. <http://www.thingiverse.com/thing:1358552>, Februar 2016
- [5] ROBOTEQ (Hrsg.): *Driving Mecanum Wheels Omnidirectional Robots*. <https://www.roboteq.com/index.php/applications/applications-blog/entry/driving-mecanum-wheels-omnidirectional-robots>: RoboteQ, Oktober 2015

Listings

A.1	communication.cpp	37
A.2	communication.h	38
A.3	movement.cpp	39
A.4	movement.h	43
A.5	util.cpp	45
A.6	util.h	47
A.7	main.cpp	47
A.8	index.html	50
A.9	code.js	53
A.10	style.css	60
A.11	main/ConnectUI.java	61
A.12	components/SpeedMeter.java	67
A.13	Toaster/Toaster.java	73
A.14	Toaster/ToasterBody.java	75
A.15	Utils/UIUtils.java	78
A.16	Utils/TextFieldIP.java	79
A.17	Utils/TextFieldPort.java	80
A.18	main/Main.java	81
A.19	main/ZweitesFenster.java	82

Abbildungsverzeichnis

3.1	Skizze Seggway	7
3.2	Skizze Weggchair	9
3.3	Skizze TEGGLA	10
3.4	Morphologischer Kasten	10
3.5	CAD Programm “Creo Parametrics”	13
3.6	Graphische Oberfläche für Git (GitKraken)	14
4.1	Übersicht der TEGGLA-Komponenten	15
4.2	Schaltplan (gezeichnet mit Fritzing)	17
4.3	Freiheitsgrade von Mecanum [5]	18
4.4	Formeln für die individuellen Motoren [5]	19
4.6	OpenSCAN zum Erstellen von Planetengetrieben	20
4.7	Im Rad integrierte Getriebe	21
4.8	Iterationen der Getriebe	21
4.9	Mehrstufige Getriebe	22
4.10	Finale Getriebekonstruktion	23
4.11	Java Fenster 1: Verbindungsbaufbau	25
4.12	Java Fenster 2: Steuerung	26
4.13	Bildschirmfoto Weboberfläche	27
5.1	Blockschaltbild Regler	31
B.1	Komplettansicht	87
B.2	Isometrisch und Stückliste	88
B.3	Hauptträger	89
B.4	Hauptträger Isometrisch	90
B.5	Motorenhalterhalter	91
B.6	Motorenhalterung Oben	92
B.7	Motorenhalterung Unten	93
B.8	Innenfelge	94

B.9 Felge	95
B.10 Sigma	96
B.11 H-Brücken-Halterung	97
B.12 Akkuhalterung	98
B.13 Ei-Halterung-Halterung	99
B.14 Ei-Halterung	100
B.15 Reifen und Planetengetriebe	101

Tabellenverzeichnis

2.1	Anforderungsliste	2
3.1	Stückliste Seggway	8
3.2	Kostenübersicht	12
4.1	Stückliste der TEGGLA-Komponenten	16
4.2	Aufzählung benötigter Pins	23
4.3	Vorteile des ESP-32	24
4.4	Binäres Protokoll	28