Prototypische Entwicklung eines ferngesteuerten Kettenfahrzeugs auf Basis eines Einplatinencomputers

Bachelorarbeit

vorgelegt von

Dominik Dachs
Matrikelnummer: 1018570

Frankfurt University of Applied Sciences
Fachbereich 2 - Studiengang Informatik

Erstprüfer: Prof. Dr. Christian Baun
Zweitprüfer: Prof. Dr. Doina Logofatu

1 Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbständig verfasst und dabei keine anderen als die angegebenen Hilfsmittel benutzt habe. Sämtliche Stellen der Arbeit, die im Wortlaut oder dem Sinn nach Publikationen oder Vorträgen anderer Autoren entnommen sind, habe ich als solche kenntlich gemacht. Die Arbeit wurde bisher weder gesamt noch in Teilen einer anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Datum: _____ Unterschrift: _____

2 Danksagung

Hiermit möchte ich mich bei allen bedanken, die mich bei meinem Studium unterstützt haben. Besonderer Dank geht dabei an Yalçın Eren.
Auch möchte ich mich bei allen Korrekturlesern bedanken. Außerdem danke ich Prof. Dr. Baun für die Hilfe bei der Themenfindung und der Betreuung der Bachelorarbeit, sowie Prof. Dr. Logofatu für die Verfügbarkeit als Zweitprüfer.

3 Zusammenfassung

Diese Bachelorarbeit beschreibt den Aufbau eines Kettenfahrzeuges und dessen Fernsteuerung über eine Webseite mithilfe eine Einplatinencomputers. Dies geschieht unter Verwendung eine Raspberry Pi 3 Model B+, sowie zweier bürstenloser Gleichstrommotoren, welche das Kettenfahrzeug antreiben. Die Steuerung selbst erfolgt über eine, auf dem Raspberry Pi gehosteten Webseite, sowie einem Java Backend. Des Weiteren werden detailliert die einzelnen Schritte für den Aufbau des Kompletten Kettenfahrzeugs beschrieben. Darunter fallen die Implementierung der einzelnen Programme so wie der Aufbau der physischen Komponenten.

4 Abstract

This bachelor thesis describes the construction of a tracked vehicle and its remote control via a website using a single-board computer. This is done using a Raspberry Pi 3 Model B + and two brushless DC motors driving the tracked vehicle. The control itself is done via a website hosted on the Raspberry Pi, as well as a Java backend. Furthermore, the individual steps for the construction of the complete tracked vehicle are described in detail. This includes the implementation of the individual programs as well as the structure of the physical components.

Inhaltsverzeichnis

Abkürzungsverzeichnis

ABS Acrylnitril-Butadien-Styrol-Copolymere

BCM Broadcom

CLI Command Line Interface

CSS Cascading Stylesheet

DC Direct Current

DIY Do It Yourself

ESC Electronic Speed Controller

FDM Fused Deposition Modeling

GPIO General Purpose Input/Output

HTML Hypertext Markup Language

IDE Integrated Development Environment

JSON JavaScript Object Notation

mAh Milliamperestunden

npm Node.js Package Manager

PLA Polylactide

PWM Pulsweitenmodulation

RC Remote Controlled

SASS Syntactically Awesome Stylesheets

SCSS Sassy CSS

TCP Transmission Control Protocol

URL Uniform Resource Locator

5 Einleitung

Den Einsatzmöglichkeiten von modernen Einplatinencomputern, von der Steuerungsplatine einer RC Drohne, über das Steuermodul eines Smart-Home-Systems, bis hin zum Einsatz als Steuermodul für dedizierte Bitcoin-Mining-Hardware[1], sind kaum Grenzen gesetzt. Dabei zeichnen sich die heutigen Einplatinencomputer durch ihren geringen Formfaktor, einer sparsamen Energieaufnahme, sowie ihrer Leistungsfähigkeit und Modularität aus. Unter den heutigen Einplatinencomputer verschiedener Hersteller, wie zum Beispiel dem Arduino oder dem PandaBoard, sticht besonders der, von der Raspberry Pi Foundation entwickelte, Raspberry Pi hervor. Dieser hat sich in den ersten fünf Jahren seines Verkaufsstarts mit über zwölfeinhalb Millionen verkauften Exemplare zum dritt meist verkauften Allzweckcomputer aller Zeiten avanciert[2].

Der erste Raspberry Pi erschien im Februar 2012 und wurde ursprünglich für den Informatikunterricht in Schulen und dritte Welt Ländern entwickelt[3]. Mittlerweile ist der Raspberry Pi in mehr als zehn verschiedenen Modellen mit unterschiedlichen Spezifikationen verfügbar. Die verschiedenen Modelle sind für rund 10 bis 30 Euro[4] erhältlich und lassen sich durch eine Fülle von zusätzlich verfügbaren Erweiterungsmodulen, wie zum Beispiel einem LCD Touch Bildschirm oder einem 4G Kommunikationsmodul, aufrüsten. Auch in meiner Bachelorarbeit kommt ein Raspberry Pi, der dritten Generation, zum Einsatz, da dieser sich durch seine Rechenleistung, integriertes WLAN-Modul und verfügbaren Anschlüsse perfekt für den Einsatz als Internetfähiges Steuermodul in einem Kettenfahrzeug eignet.

[1] Piksa 2013.
[2] Miller 2017.
[3] Cellan-Jones 2011.
[4] Vergleich: `https://www.idealo.de/07.03.2019`

5.1 Zielstellung

Das Ziel dieser Bachelorarbeit ist die Entwicklung eines, über eine Webseite steuerbares, Kettenfahrzeuges. Das Fahrzeug selbst soll dabei einen über WLAN erreichbaren Server hosten, welcher wiederum für die Ansteuerung der verwendeten Motoren zuständig ist. Weiterhin wurde bei der Entwicklung des Prototypen versucht alle Komponenten möglichst ausbaufähig zu gestalten, mit der Absicht einer möglichst einfachen Weiterentwicklung nach der Fertigstellung. Dies spiegelt sich Beispielsweise in der Implementierung eines bidirektionalen Kommunikationsmethode zwischen der Steuerungsanwendung und dem Server wieder, welches bei späterer Weiterentwicklung ermöglicht, dass das Kettenfahrzeug Feedback in Echtzeit an die Steuerungsanwendung senden kann.

5.2 Motivation

Die Motivation hinter der Umsetzung dieser prototypischen Entwicklung eines Kettenfahrzeuges stammt von der Vielzahl an Einsatzmöglichkeiten eines Einplatinencomputers. Dies gepaart mit meiner Faszination für 3-D-Drucker und den Möglichkeiten, welche in Kombination dieser beiden Technologien entstehen und den erlernten Fähigkeiten aus meinem Studium und meiner Praxisphase, führten mich dazu diese Arbeit zu realisieren. Teil meiner Motivation für diese Umsetzung war es auch, die Möglichkeiten der Programmiersprache Java und verschiedener verschiedener Frameworks, von denen einige speziell für die Anwendung auf einem Raspberry Pi geschaffen worden sind, in Verbindung mit einem Raspberry Pi auszutesten.

6 Stand der Technik

Im folgenden Kapitel werde ich den auf den aktuellen Stand der Technik in Bezug auf die verwendete Hardware eingehen. Dabei werde ich zuerst auf den Technologiestand des Raspberry Pi eingehen, darauf auf die verwendeten Motoren und ihre Steuermodule und abschließend auf verfügbare Modellkettenfahrzeuge welche für den Einsatz mit einem Raspberry Pi nutzbar wären.

6.1 Der Raspberry Pi

Der Raspberry Pi ist ein, von der Raspberry Pi Foundation entwickelter Einplatinencomputer. Er besteht aus einem Ein-Chip-System mit einem ARM-Mikroprozessor und hat etwa die Grundfläche einer Kreditkarte. Die aktuelle Version ist der auf Abbildung 6.1.1 zu sehende Raspberry Pi 3 B+.

Abbildung 6.1.1: Raspberryy Pi 3 B+
Quelle: `https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/`

Seit seinem Verkaufsstart Anfang 2012, wurden die verschiedenen Modelle des Raspberry Pi bis Ende 2018 insgesamt über dreiundzwanzig Millionen mal verkauft[5], was den ursprünglichen Plan von der Herstellung von rund 10 bis 20.000 Stück bei weitem übersteigt[6]. Der Raspberry ist inzwischen in 9 verschiedenen Modellen, unterteilt in 3 Generationen, plus dem Raspberry Pi Zero, welcher in 2 verschiedenen Varianten erhältlich ist.

Auflistung aller momentan erhältlicher Raspberry Pi Modelle inklusive Datum der Erstveröffentlichung:

1. Generation
   - Model A (2013)
   - Model A+ (2014)
   - Model B (2012)
   - Model B+ (2014)
2. Generation
   - Model B (2015)
   - Model B v1.2 (2016)
3. Generation
   - Model A+ (2018)
   - Model B (2016)
   - Model B+ (2018)

Raspberry Pi Zero

- Zero (2015)
- Zero W/WH (2017/18)

[5] Torrone 2018.
[6] *SALES SOAR AND RASPBERRY PI BEATS COMMODORE 64* 2018.

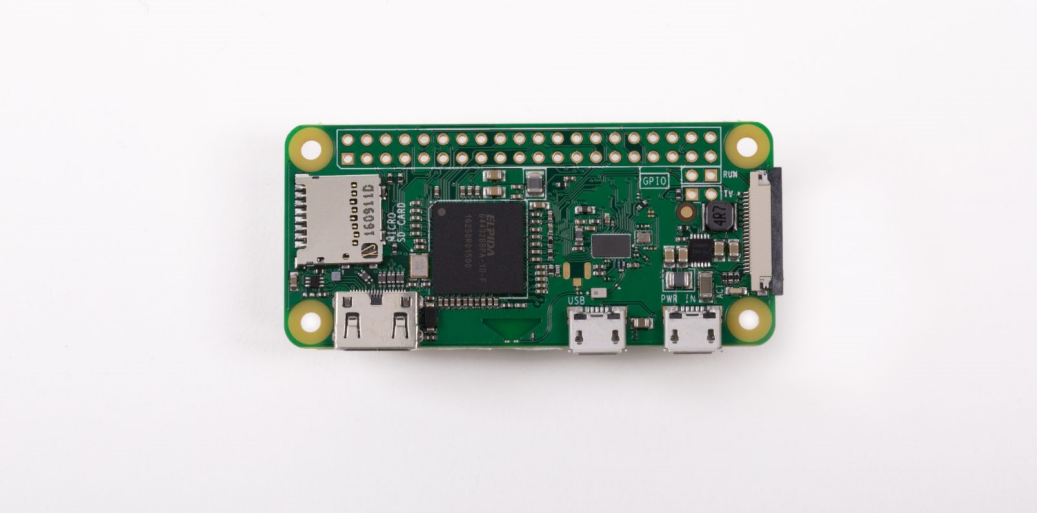

Abbildung 6.1.2: Raspberryy Pi Zero W
Quelle: `https://www.raspberrypi.org/products/raspberry-pi-zero-w/`

Der 2018 erschienene Raspberry Pi 3 B+ ist das neuste und zugleich leistungsstärkste Modell. Wie auch sein Vorgängermodell, dem Raspberry Pi 3 B, besitzt dieser eine CPU mit 64-bit-ARMv8 Architektur, 1 Gigabyte Arbeitsspeicher, integriertes WLAN und Bluetooth Low Energy[7]. Im Vergleich zu seinem Vorgänger hat dieser eine um 200 MHz gesteigerte Taktung seines Prozessors und einen verbesserten Funkmodul, welches 5 GHz WLAN nach den IEEE-802.11ac-Standard und auch Bluetooth 4.2 beherrscht.[8]

[7]Upton 2016.
[8]Storm 2018.

Die Technischen Merkmale des aktuellsten Raspberry Pi Modells, des Raspberry Pi 3 B+ lauten wie folgt:[9]

- Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
- 1GB LPDDR2 SDRAM
- 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, Bluetooth Low Energy
- Gigabit Ethernet over USB 2.0 (maximaler Durchsatz von 300 MB/s)
- Erweiterter 40-pin GPIO header
- Full-size HDMI
- 4 USB 2.0 Anschlüsse
- CSI Camera Port für das Anschließen einer Raspberry Pi Kamera
- DSI Display Port für den Anschluss eines Raspberry Pi Touchscreens
- 4-poliger Stereoausgang und Composite-Video-Anschluss
- Micro SD Pport für das laden des Betriebssystems und Speicherung der Daten
- 5V/2.5A DC Power input
- Power-over-Ethernet (PoE) unterstützung (benötigt separat erhältlichen PoE HAT)

[9]Quelle: `https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/`

6.2 Motoren und Motorsteuerung

Bei den verfügbaren Motoren für RC-Modelle (englisch für **R**emote **C**ontrolled) ist sind diese in zwei Kategorien zu unterteilen, den Verbrennungsmotoren und den Elektromotoren. Dabei arbeiten verschiedene Verbrennungsmotoren mit unterschiedlichen Kraftstoffen, wie Methanol, Benzin, Diesel oder Kerosin. Ein Beispiel für einen Kraftstoffmotor ist der Glühzündermotor, welcher ein Zweitakt- oder Viertaktmotor ist. Er kommt ohne eine gesteuerte Zündung aus, stattdessen wir der Treibstoff über eine dauerhaft glühende Glühkerzen gezündet. Der Glühzündermotor wird im Modellbau mit Methanol betrieben.

Abbildung 6.2.1: Beispiel eines Glühzündermotors
Quelle: http://sebastianfahrni.blogspot.com/2013/06/juni-2013-gluhzundermotor.html

Kraftstoffmotoren für RC-Modelle haben den Vorteil, das diese weniger temperaturanfällig sind und im Vergleich zu Elektromotoren auch bei sehr hohen oder sehr niedrigen Temperaturen eingesetzt werden könne, bei denen der Akku eines Elektromotors schaden nehmen würde. Weiterhin bringen Kraftstoffmotoren realistische Geräusche und einen Auspuff mit sich, welche wiederum dazu führen, dass Kraftstoffmotoren betriebene RC-Modelle nur im freien genutzt werden können[10].

Elektromotoren sind hauptsächlich in zwei verschiedenen Kategorien erhältlich: Dem Bürstenmotor und dem bürstenlosen Motoren.

Der Bürstenmotor ist ein Gleichstrommotor, welcher mittels Kohlebürsten mit dem sich rotierenden Stromwandler in Kontakt steht. Aufgrund dieser Reibung weisen Bürstenmotoren allerdings eine durch, Verschleiß verursachte, Lebensdauer auf, welchem durch regelmäßige Wartung entgegengewirkt werden kann. Auch gilt zu beachten dass es bei dieser Art von Motor, bei sehr hohen Drehzahlen zu, durch die Bürsten verursachten, Funkenflug kommt, dem sogenannten "Bürstenfeuer"[11]. Ein Vorteil des Bürstenmotors ist, dass diese, durch ihre simple Bauweise günstiger in der Anschaffung, im Vergleich zu Verbrennungsmotoren oder auch bürstenlosen Motoren, sind.

Abbildung 6.2.2: Beispiel eines bürstenlosen Motors
Quelle: `http://www.rcsunnysky.com/content/157.html`

[10] *Alles, was man über RC-Motoren wissen sollte* 2014.
[11] *Gleichstrommaschine* 2019.

Bürstenlose Motoren oder auch Brushless Motoren besitzen, wie der Name schon sagt, keine Kohlebürsten, sondern nutzen einen Regler, welcher für den Polwechsel sorgt. Die genaue Funktionsweise eines bürstenlosen Motores ist dabei folgende:

> "Bei einem 3-poligen Motor generiert der Regler ein sogenanntes Drehfeld durch Schaltung einer Gleichspannung auf zwei von drei Motorspulen, wobei die dritte Motorspule als Sensorleitung dient und die jeweilige Position des sich drehenden Permanentmagneten feststellt. Durch das Magnetfeld des Permanentmagneten und das Magnetfeld der Motorspulen, beginnt sich der Permanentmagnet durch Anziehung und Abstoßung der Magnetfelder zu drehen. Diese Rotation setzt sich solange fort bis sich zwei gleich gepolte Magnetpaare fast gegenüberstehen. Jetzt wechselt der Regler die Spannung auf den Motorspulen und der Vorgang beginnt von Neuem."[12]

Vorteile des bürstenlosen Motors sind, das dieser zum einen eine geringere Wärmeentwicklung im Vergleich zu einem Bürstenmotor besitzt, da durch die fehlenden Kohlebürsten keine Reibungswärme entsteht. Auch der Wartungsaufwand ist minimal und du es können höhere Drehzahlen erreicht werden.

6.3 Motorsteuerungen

Für die Nutzung eines Elektromotors wird eine Motorsteuerung benötigt, welche die Spannung und den Stromfluss steuert, um den anzutreiben[13]. Diese werden auch ESC genannt (englisch für Electronic speed controller) und sind sowohl für bürstenlose Motoren wie auch Bürstenmotoren erhältlich. ESCs für Bürstenmotoren steuern die Geschwindigkeit des Motors über die Regulierung der Spannung. ESCs für bürstenlose Motoren Operieren nach einem anderen Prinzip, die Geschwindigkeit des Motors wird durch die Anpassung der Stromimpulse welche an den Motor geschickt werden kontrolliert. Dies geschieht durch sogenannte Pulsweitenmodulations (kurz: PWM).

[12] *Brushless vs. Brushed – Motoren* 2015.
[13] SafeDrone Team 2017.

6.4 Kettenfahrzeuge für den Modellbetrieb

Bei der Wahl des Kettenfahrzeuges gibt es mehrere Faktoren zu beachten. Neben Größe, Gewicht und Platz für zusätzliche Komponenten, Beispielsweise dem Raspberry Pi, ist auch zu beachten ob dieses bereits eingebaute Motoren mit sich bringt und ob diese für die Nutzung in Verbindung mit dem Raspberry Pi in frage kommen. Eine Möglichkeit wäre die Nutzung eines fertigen Modellkettenfahrzeugs, welches dann nachträglich für den Einsatz mit einem Raspberry Pi modifiziert werden muss. Eine weitere Möglichkeit wäre die Verwendung sogenannter DIY-Kits (kurz für do it yourself), wie auf Abbildung 6.4.1 zu sehen ist.

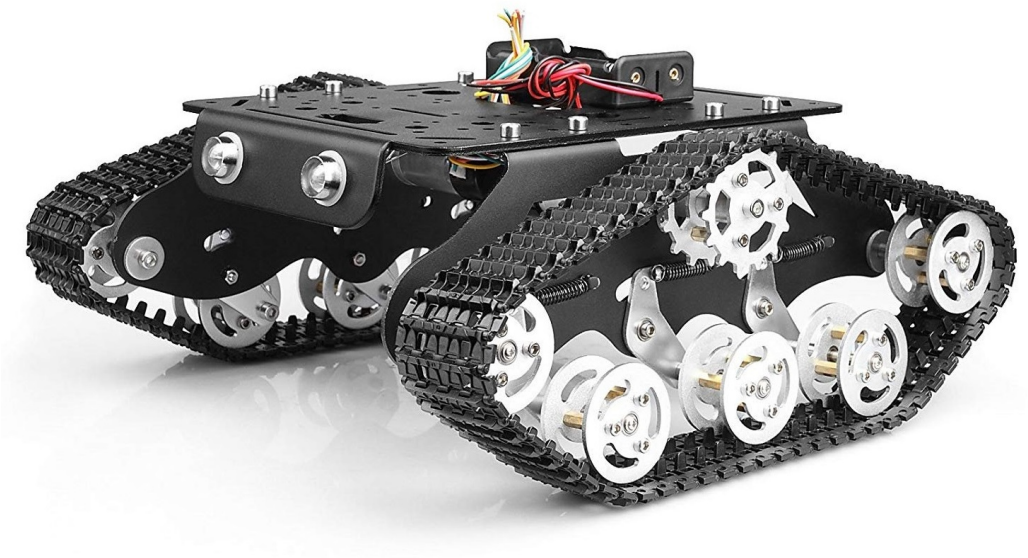

Abbildung 6.4.1: Beispiel für ein DIY-Kit eines Kettenfahrzeugs
Quelle:
`https://www.amazon.de/TSSS-Intelligente-Aluminium-Legierung-D\%C3\%A4mpfung-Effekt-leistungsf\%C3\%A4higem/dp/B07FD74FWX/`

Diese bieten das Chassis eines Kettenfahrzeugs, meist mit einer Plattform auf welche alle nötigen Komponenten befestigt werden können, welches nur noch zusammengebaut werden muss. Eine Dritte Möglichkeit wäre die

Verwendung eines 3D-Druckers um die meisten Teile des Kettenfahrzeugs zu drucken. Diverse Webseiten, wie zum Beispiel `https://thingiverse.com/`, bieten dabei eine Plattform für 3D-Druck Enthusiasten, auf denen sie selbst kreierte 3D-Modelle frei als Open-Source-lizenzierte Dateien teilen können[14].

[14]Baichtal 2018.

7 Design

Im folgenden Kapitel werden wird auf auf die getroffenen Designentscheidungen für die Entwicklung des Kettenfahrzeugs eingegangen. Dabei werden die verwendeten Technologien und Komponenten genauer beschrieben sowie aufgezeigt, welche Alternativlösungen verfügbar sind und wie diese das Design beeinflussen würden.

7.1 Der Einplatinencomputer

Als Einplatinencomputer wurde für diese Arbeit ein Raspberry Pi 3 Model B+ verwendet. Dieser Raspberry Pi der dritten Generation ist, zum Zeitpunkt dieser Arbeit, das neuste und zugleich leistungsstärkste Modell der Raspberry Pi Reihe. Der Raspberry Pi B+ ist im März 2018 erschienen und bringt , im Vergleich zu seinem Vorgängermodell, dem Raspberry Pi 3 Modell B, einige Neuerungen mit sich. Zum einen verfügt das Modell B+ eine im Vergleich, zu seinem Vorgänger, um 200 MHz gesteigerte Taktrate seiner CPU, welche nun eine Taktrate von 1400 MHz besitzt. Weiterhin besitzt dieser ein verbessertes Funkmodul, welches über 5 GHz WLAN und auch Bluetooth 4.2 verfügt. Die Maße von 93 mm x 63,5 mm x 20 mm Länge x Breite x Höhe) und sein Gewicht von 42 g eignen sich sehr gut, um den Raspberry Pi auf einem Modellkettenfahrzeug unterzubringen.

Alternativ hätte auch ein Raspberry Pi Zero W gewählt werden können, dieser ist 2017 erschienen und hat als Neuerung, seinem Vorgänger gegenüber, einen Funkchip verbaut welcher 2,4 GHz WLAN unterstützt. Damit würde dieser sich grundsätzlich auch eignen um sich über eine Webanwendung steuern zu lassen. Vorteilhaft gegenüber dem Raspberry Pi 3 B+ wäre der geringere Formfaktor des Zero und der geringer ausfallende Stromverbrauch, welcher mit einer kleineren und dadurch auch leichteren Stromversorgen verbunden wäre. Nachteilhaft ist allerdings der geringere Arbeitsspeicher (512 MB im Vergleich zu den 1024 MB des Raspberry Pi 3 B+) und ein CPU mit geringerer Leistung (1 Kern mit 1000 MHz Taktung im Vergleich zu den 4 Kernen mit 1400 MHz Taktung des Raspberry Pi 3 B+). Da vorgesehen

ist, dass der Raspberry Pi den Server hosten soll, auf dem später das Steuerungsfrontend, sowie ein Java Backend laufen soll, wurde sich zugunsten der größeren Rechenleistung für den Raspberry Pi 3 B+ entschieden.

Betrieben werden soll der Raspberry Pi mittels einer Powerbank. Eine Powerbank ist ein Akkumulator mit einem oder mehreren Lithium-Ionen-Akkuzellen, welche für das laden mobiler Endgeräte entworfen wurde. Als Ladekapazität wurden 1000 mAh gewählt, da dies für ungefähr 3 Stunden Dauerbetrieb genügen sollte[15]

Abbildung 7.1.1: Beispielabbildung einer Powebank
Quelle: https://www.amazon.de/
Powerbank-10000mAh-Ports-Ausgang-LED-Statusanzeige/dp/
B075KJ7FCF/

[15]Berechnet mit PI POWER ESTIMATOR APP Quelle: https://www.raspberrypi-spy.co.uk/tools/pi-power-estimator-app/

7.2 Auswahl der Motoren und Motorsteuerung

Als Motoren für den Antrieb des Kettenfahrzeugs wurden Bürstenlose Gleichstrom Motoren gewählt (auch brushless DC Motoren genannt, dabei steht DC für das englische Direct Current, also Gleichstrom). Diese Motoren haben den Vorteil gegenüber Bürstenmotoren, dass sie weniger Wartung benötigen und höhere Drehzahlen erreichen können. Die Geschwindigkeit solcher Motoren wird über ein PWM (Pulsweitenmodulation) Signal gesteuert. Dies ist eine Modulationstechnik bei dem das Nutzsignal in unterschiedlichen Pulsweiten an den Empfänger gesendet wird.

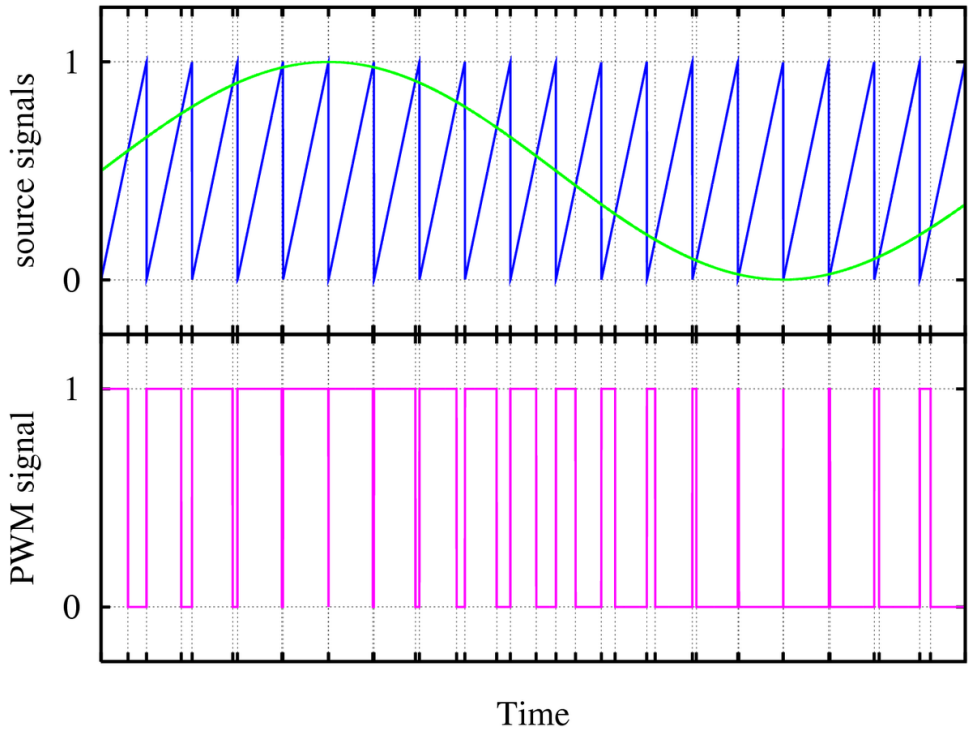

Abbildung 7.2.1: Pulsweitenmodulation
Quelle: `https://de.wikipedia.org/wiki/Pulsdauermodulation`

Dabei bestimmt der Prozentsatz, also wie lange das Signal während eines Zyklus (auch duty cycle gennant) aktiv ist, die Geschwindigkeit mit der sich der Motor dreht. Ein Signal von 100% des duty cycles wäre also maximale Geschwindigkeit für den Motor.

Für die Motorsteuerung wurden sogenannte Electronic Speed Controller

ausgesucht, diese stellen ein Regelungsgerät da, welches die Geschwindigkeit der Motoren an das gesendete PWM-Signal anzupassen. Auch enthalten Sie diverse Sicherheitsvorkehrungen, welche den Motor Beispielsweise bei einem plötzlichen Spannungsabriss vorsichtig abbremst.

7.3 Wahl des Kettenfahrzeugs

Bei der Wahl des Kettenfahrzeuges müssen mehrere Eigenschaften beachtet werden. Zum einen muss dieses entweder kompatibel mit den anderen Komponenten sein, oder die schon vorinstallierten Teile müssen die gewünschten Eigenschaften erfüllen. Auch sollte das Chassis genügend Platz für alle zusätzlich verbauten Elemente des Kettenfahrzeugs haben. Zu diesen Elementen zählt der Raspberry Pi und die Powerbank, aber auch eventuell die Motorsteuerungen und der Akku für die Motoren, falls das gewählte Chassis diese Teile nicht mitbringt oder vorsieht.

Die Wahl viel schließlich auf ein Kettenfahrzeug mittels 3D-Drucker herzustellen. Diese Methode ist, dank eines für mich bereits zur Verfügung stehenden 3D-Druckers, kostengünstiger, als ein bereits fertiges RC-Modellkettenfahrzeug zu kaufen und dieses für den Einsatz mit einem Einplatinencomputer umzubauen. Weiterhin erlaubt dieses Vorgehen, alle Komponenten einzeln auszusuchen und zu verbauen. Weitere Vorteile, eines selbst gedruckten Kettenfahrzeugs sind, dass strukturelle Ersatzteile nicht nachgekauft werden müssen, sondern diese einfach erneut gedruckt werden können und dass Teile beliebig vor dem Druck bearbeitet und angepasst werden können um neuen Anforderungen, wie Beispielsweise einem erhöhtem Platzbedarf, zu genügen.

Gewählt wurde ein 3D-druckbares Kettenfahrzeug, mit dem Namen "RC Tank", zu sehen auf Abbildung 7.3.1. Dieses wurde von André Klaus, auf der Webseite `www.thingiverse.com` unter der creative commons Lizenz "CC BY-NC-SA 3.0", veröffentlicht und steht zum freien Download zur Verfügung. Dieses erfüllt alle nötigen Anforderungen und bietet gute Möglichkeiten zur späteren Modifikation.

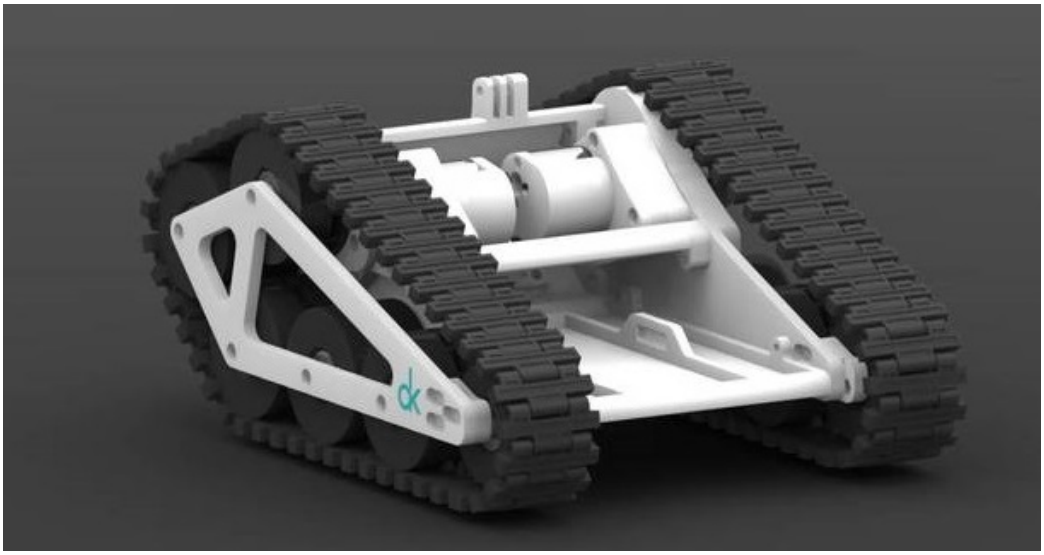

Abbildung 7.3.1: Gerendertes Bild des gewählten Kettenfahrzeugs
Quelle: `https://www.thingiverse.com/thing:2414983`

7.4 Technologien für Steuerung über eine Webanwendung

Um das Kettenfahrzeug über eine Webanwendung steuern zu können, werden zwei Komponenten benötigt, welche sich in Front-End, also die Webanwendung selbst, welche der Nutzer über seinen Webbrowser lädt und nutzt um die Steuerungsbefehle zu senden, und das Back-End, welches die Steuerungsbefehle des Nutzers empfängt, verarbeitet und an die Motorsteuerung sendet.

Für das Front-End wurde sich für eine Angular Anwendung entschieden. Angular ist einen plattformunabhängiges, auf TypeScript-basierendes Webapplikationsframework, welches von Google entwickelt wird. Einer der drößten Vorteile von Angular entsteht dadurch, dass dieses in TypeScript entwickelt wird. Während des Buildprozesses wird der TypeScript Code in JavaScript transpiliert. Dadurch ist man während der Entwicklung nicht gezwungen Browserabhängig zu arbeiten, von manchen Webbrowsern nicht unterstützte Funktionalitäten können beim transpilieren einfach durch sogenannte Polyfills bereitgestellt werden. Auch bringt TypeScript die Eigenschaft der Typisierung mit und unterteilt eine Webseite in kleiner Kompo-

nenten, sogenannte components, was der Struktur von Java ähnelt und den Umstieg von Java auf TypeScript erleichtert[16].

Als Back-End Technologie wurde nicht, das auf Einplatinencomputern häufig genutzte, Python gewählt. Stattdessen wurde sich für die Verwendung von Java entschieden. Diese Entscheidung beruht teilweise auf der Zielsetzung dieser Arbeit, zu erforschen, welche Möglichkeiten, in der Verwendung eines Einplatinencomputers in Verbindung mit der Programmiersprache Java, bestehen. Auch wurde Java gewählt, da diese mir selbst gut vertraut ist und ich deshalb zuversichtlich bin, gesetzte Ziele so erreichen zu können.

Besonders zu erwähnen ist dabei die Verwendung der Spring Frameworks und des Build-Management-Tools Maven, welche die Entwicklung besonders beeinflusst haben. Das Spring Framework ist ein quelloffenes Java Framework, dessen Ziel es ist, die Entwicklung mit Java, sowie JavaEE, zu vereinfachen und eine Entkopplung der Entwicklungskomponenten zu ermöglichen. Eine der auffälligsten Eigenschaften dieses Frameworks ist die sogenannte Dependency Injection, welche dafür sorgt, dass Objekte ihre benötigten Ressourcen und Objekte automatisch zugewiesen bekommen, wodurch keine Abhängigkeiten in der Programmierung entstehen. Diese Eigenschaft hilft dabei den für diese Arbeit geschriebenen Code auch noch in Zukunft gut nutzen und ausbauen zu können. Ein weiterer, für diese Arbeit wichtiger, Teil des Spring Frameworks ist die Spring Boot Erweiterung, welche unter anderem mit einem eingebetteten Tomcat kommt, welches das mühsame aufsetzen eines Webserver und dessen Konfiguration obsolet machen.

Das eingesetzte Build-Managemenet-Tool Maven der Apache Software Foundation, wird in dieser Arbeit genutzt, um den kompletten Build-Prozess des Back-Ends, sowie des Front-Ends zu übernehmen und diese, in einem sogenannten Web Application Archive, zusammen zufassen. Dieses daraus entstehende Archiv kann dann anschließend einfach auf dem Raspberry Pi, über die Kommandozeile ausgeführt werden. Damit erleichtert Maven den

[16]Dorn 2018.

Build-Prozess erheblich, weshalb sich für den Einsatz dieses Tools entschieden worden ist.

Eine weitere verwendete Technologie ist das Pi4J Projekt, welches Java Bibliotheken für die Nutzung der Ein- und Ausgabefähigkeiten, des Raspberry Pi, bereitstellt. Diese Bibliotheken ermöglichen es, die GPIO-Pins (abgekürzt für general purpose input/output) mittels Java nutzen zu können und so der Motorsteuerungen die benötigten PWM-Signale senden zu können.

Als Technologie, für die Übertragung der Steuerungsbefehle des Nutzerfrontends an das Back-End, wurde das auf TCP (abgekürzt für Transmission Control Protocol) basierende WebSocket-Protokoll genutzt. Dieses erlaubt einerseits den Aufbau eines bidirektionalen Kommunikationskanals zwischen dem Front- und dem Back-End und reduziert andererseits die Menge an Belastungsdaten, da bei der Kommunikation über WebSockets, im Vergleich zu einer reinen HTTP-Verbindung, mittels REST, der HTTP-Header entfällt.

8 Implementierung

Dieses Kapitel beschreibt die Vorgehensweise bei der Entwicklung, der Einrichtung und der Inbetriebnahme aller wichtigen Komponenten, um das Kettenfahrzeug mittels Webanwendung fernsteuern zu können.

Zuerst wird der Prozess der Herstellung, des Kettenfahrzeugmodells, mittels 3D-Drucker genauer erläutert. Anschließend wird die Programmierung des Front- und Back-Ends, mittels Angular, beziehungsweise Java, beschrieben und dabei die wichtigsten Schritte aufgezeigt. Danach wird die Einrichtung des Raspberry Pis beschrieben und wie dieser vorzubereiten ist, um alle Voraussetzungen für die Inbetriebnahme als Steuerungsmodul in dem Kettenfahrzeug zu funktionieren. Abschließend folgt das Zusammenbringen aller physischen Bauteile und ihrer Verkabelung, sowie aller digitalen Komponenten und ihrer letztendlichen Einrichtung.

8.1 Druck des Kettenfahrzeugs

Wie bereits in Kapitel 7.3 beschrieben, wurde entschieden, das Kettenfahrzeug mittels 3D-Drucker herzustellen. Dies wurde mit der Nutzung eines 3D-Druckers, ähnlich Abbildung 8.1.1, welcher nach dem Fused Depositiong Modeling Verfahren (kurz FDM; zu deutsch: Schmelzschichtung) arbeitet.

Bei diesem Verfahren wird das zu druckende Objekt mittels sogenannter Slicer-Software in eine, für 3D-Drucker lesbare Datei, umgewandelt. Dabei wird das zu druckende Objekt in mehrere Schichten, welche wiederum aus Pfaden bestehen, welche der Drucker später mit seinem Druckkopf abfährt und das geschmolzene Druckmaterial aufträgt. Das zu druckende Modell wird, langsam in mehreren Schichten, welche je nach Anwendungsfall 0,025 und 1,25 mm dick sind, durch den Drucker hergestellt.

Für den Druck kann in der Slicing-Software noch weitere Parameter eingestellt werden, wie zum Beispiel, die Anzahl an Außenwänden und der Prozentsatz an Füllung für das zu druckende Modell, denn der Druck mittels FDM-Verfahren erfolgt seltenst als massives Objekt. Dies würde zwar maxi-

Abbildung 8.1.1: Abbildung eines FDM-Druckers
Quelle: `https://tevo3dprinterstore.com/collections/3d-printers`

male Teilstabilität bringen, aber im Gegenzug einen massiv erhöhten Materialverbrauch, eine stark erhöhte Druckzeit und ein erhöhtes Gewicht darstellen. Deshalb kann einm 3D-Objekt in der Slicing-Software ein Prozentsatz gegeben werden, welcher den prozentualen Anteil an Füllmaterial angibt, welches in verschiedenen Mustern, je nach Anwendungsgebiet, gedruckt werden kann. So kann die strukturelle Integrität des zu druckenden Objektes gewahrt werden und zugleich Materialverbrauch und Druckzeit minimiert werden.

Das, für diese Arbeit verwendete, Kettenfahrzeug besteht aus insgesamt 134 einzelnen zu druckenden Teilen. Davon sind 96 dieser Teile identische Kettenglieder, welche verwendet werden um die beiden Ketten des Fahrtzeugs herzustellen. Der Druck dieser Teile benötigt ungefähr 1,7 Kilogramm

Druckmaterial. Das für diese Arbeit verwendete Material ist ein Filament aus Polylactide (kurz PLA). Dieses Filament ist für das häufigst verwendete Material für FDM-Drucker, da dieses relativ einfach einzusetzen und kostengünstig im Vergleich zu beispielsweise Filament aus ABS (kurz für Acrylnitril-Butadien-Styrol-Copolymere) ist.
Die Druckzeit, also die reine Zeit, welche der Drucker braucht um alle 134 benötigten Teile herzustellen, summiert sich auf ungefähr 120 Stunden.

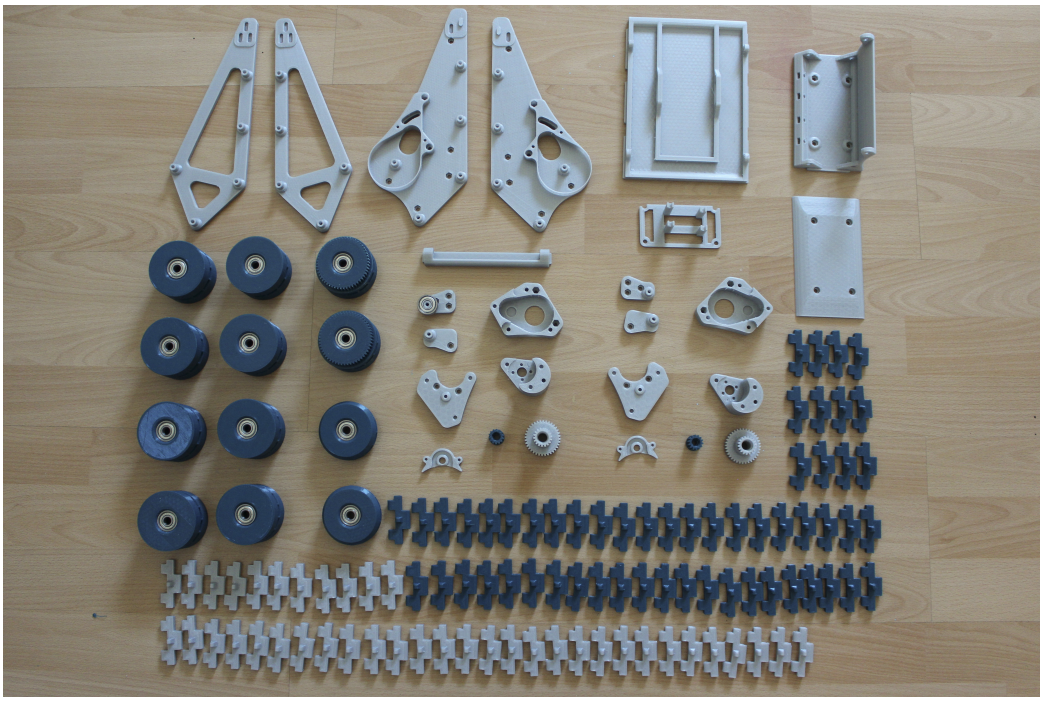

Abbildung 8.1.2: Abbildung aller gedruckten Teile
Quelle: eigene Darstellung

Nachdem alle Teile gedruckt sind können alle gedruckten teile Teile zusammengebaut werden. Dies geschieht bei diesem Modell mittels Schrauben, welches ein einfaches Austauschen ermöglicht, falls ein Teil kaputt geht oder ein Teil nachträglich modifiziert werden, um neue Anforderungen zu erfüllen, sollte.

8.2 Programmierung der Webanwendung

Wie bereits erwähnt, wurde für das Front-End, also die Anwendung, welche der Nutzer, sprich die Person, welche das Kettenfahrzeug steuert, später nutzen wird, Angular gewählt.
Für die Entwicklung von Angular Anwendungen, wird Node.js benötigt. Dies ist eine plattformunabhängige Laufzeitumgebung, welche es ermöglicht JavaScript außerhalb eines Webservers auszuführen. Node.js kommt mit einem vorinstalliertem Packet-Management-Tool, dem Node.js Package Manager (kurz npm). npm wird später bei der Entwicklung der Angular Anwendung benötigt, um die benötigten Angular Bibliotheken als npm-Pakete herunterzuladen.

Um zu überprüfen, ob Node.js bereits auf der Maschine, auf welcher Entwickelt werden soll, bereits installiert ist, kann folgender Befehl in die Konsole eingegeben werden:

```
node -v
```

Falls nicht vorhanden muss Node.js von der offiziellen Webseite `https://nodejs.org` heruntergeladen und installiert werden.
Anschließend kann über npm das Angular CLI (kurz für Command Line Interface) heruntergeladen werden, dazu muss folgender Befehl in die Konsole eingegeben werden:

```
npm install -g @angular/cli
```

Nun kann über die Konsole das Angular CLI genutzt werden um Angular Projekte zu generieren, Test auszuführen oder das Deployment der Anwendung auszuführen. Um dein Angular Projekt zu generieren muss folgender Befehl in die Konsole eingegeben werden:

```
ng new meine-angular-app
```

Dies erstellt eine neues Angular Projekt, inklusive aller anfänglich benötigten Angular npm Pakete sowie folgende Dateien[17]:

- Einem neuen Workspace, mit dem root-Verzeichnis namens “meine-angular-app”

[17] *Getting started* 2018.

- Einer minimalen Beispielanwendung im src Unterverzeichnis mit dem Namen “meine-angular-ap”
- Einem end-to-end Test-Ptojekt, im e2e Unterverzeichnis
- Benötigte Konfigurationsdateien

Damit sind alle Vorbereitungen abgeschlossen um das Front-End in form einer Angular Anwendung programmieren zu können. Jetzt können alle nötigen Komponenten der Angular Anwendung programmiert werden. Besonders erwähnenswert dabei ist die controller Component, welche die Eingaben des Nutzers verarbeitet und diese in ein für das Back-End vereinheitlichtes Format unmwandelt, sowie der WebSocket Service, welcher für die Herstellung der WebSocket-Verbindung zuständig ist.

8.2.1 Die Controller Component

Die Controller Component besteht, wie alle Angular Components, aus drei Dateien. Diese drei Dateien sind eine TypeScript Datei (Dateiendung .ts), welche die Logik der Komponente bereitstellt, einer Hypertext Markup Language Datei (kurz HTML; Dateiendung html), diese beschreibt die semantische Struktur, der später im Webbrowser zu sehenden Komponente auf der Webseite, sowie einer Stylesheet Datei, welche die visuelle Darstellung der Komponente beschreibt.
Als Stylesheet-Art wurde SCSS (Kurz für Sassy CSS; wobei CSS kurz für Cascading Stylesheet steht) benutzt. Dieses ist eine Syntax der Stylesheet-Sprache SASS (kurz für Syntactically Awesome Stylesheets), welches die SASS-Syntax insofern verändert, dass nicht das einrücken des Quelltextes für die Verschachtelung der Selektoren zuständig ist, sonder geschweifte Klammern.

Für die Darstellung des Nutzereingabe wurden zwei Controll-Sticks, wie man sie von modernen Gamecontrollern kennt, gewählt. Diese wurden so programmiert, dass sie jeweils drei Zustände besitzen, einen Nullpunkt, einen

nach vorne geneigten Zustand und einen nach hinten geneigten Zustand. Diese drei Zustände sind durch das drücken, im Code vordefinierter Tasten auf der Tastatur, zu erreichen.
Dabei zeigt folgender Codeausschnitt das joystick Objekt,welches eine Funktion bereitstellt, die durch ihren Aufruf erlaubt, die Logik für die Controll-Sticks zu implementieren:

```
joystick = {
   function(selector, up, down, side, websocket) {
     const elem = document.querySelectorAll(selector);
     let firedUp = false;
     let firedDown = false;
     document.addEventListener('keypress', (e) => {
       switch (e.key) {
         case up: {
           if (!firedUp && !firedDown) {
             elem.forEach(element => { element.classList.add('top'); });
             firedUp = true;
             websocket.send('GO', side);
           }
           break;
         }
         case down: {
           if (!firedDown && !firedUp) {
             elem.forEach(element => { element.classList.add('bottom'); });
             firedDown = true;
             websocket.send('REVERSE', side);
           }
           break;
         }
       }
     });
     document.addEventListener('keyup', (e) => {
       switch (e.key) {
         case up: {
           elem.forEach(element => { element.classList.remove('top'); });
```

```
30              if (firedUp) {
31                websocket.send('STOP', side);
32                firedUp = false;
33              }
34              break;
35            }
36            case down: {
37              elem.forEach(element => { element.classList.remove('
    bottom'); });
38              if (firedDown) {
39                websocket.send('STOP', side);
40                firedDown = false;
41              }
42              break;
43            }
44          }
45        });
46     }
47   };
```

Dabei werden 5 Parameter benötigt um die Funktion des joystick Objekts aufzurufen. Der selector Parameter wird dafür verwendet, dem HTML Elementen die jeweilige SCSS Klasse zuzuweisen, welche im Stylesheet der controller Komponente definiert sind und für einen Übergangsanimation sorgen. Zwei weitere Parameter sind die up und down Parameter, welche die jeweilige Taste für die Eingabe der Kontrollbefehle festlegen. Der side Parameter wird verwendet, um bei bei dem, im websocket Service zusammengestellten Kommando, welches an das Back-End gesendet wird, anzugeben, welcher der beiden Motoren den Befehl verarbeiten soll. Der letzte der 5 Parameter ist der websocket Parameter, welcher angibt an welchen Service für den Aufruf der send() Methode verwendet werden soll.

Anschließend kann, in der ngOnInit() Methode, einem sogennanten Lifecycle Hooks, welcher bei der Initialisierung der Komponente aufgerufen wird, nach der Initialisierung der WebSocket Verbindung, die Funktion des joystick Objectes zwei mal aufgerufen werden:

```
ngOnInit() {
    this.webSocketService.connect();
    this.joystick.function('.right-stick #stick, .right-stick #base', 'o', 'k', 'RIGHT', this.webSocketService);
    this.joystick.function('.left-stick #stick, .left-stick #base', 'w', 's', 'LEFT', this.webSocketService);
}
```

Nicht vergessen werden sollte, im ngOnDestroy() Lifecycle Hook, welcher aufgerufen wird, bevor Angular die Komponente zerstört, die WebSocket Verbindung zu trennen:

```
ngOnDestroy(): void {
    this.webSocketService.disconnect();
}
```

8.2.2 WebSocket Service

Die Logik, welche im Front-End benötigt wird, um die WebSocket Verbindung zu implementieren, befindet sich im websocket Service. Services werden in Angular für Logik, welche keiner bestimmten View angehören, genutzt. Diese können dann im Konstruktor, der jeweiligen Komponente, als Instanz erstellt werden und anschließend auf ihre Funktionen zugegriffen werden.

Der websocket Service besitzt drei Funktionen:

- connect()
- send()
- disconnect()

In der connect() Methode wird dabei, mithilfe der Javascript Bibliotheken SockJS und STOMP.js, eine WebSocket Verbindung mit dem Java Back-End

aufgebaut. Die Variable socket beinhaltet dabei die URL (kurz für Uniform Resource Locator) des Endpunktes, welcher vom Java Back-End bereitgestellt wird. Um diese beiden Bibliotheken in der Angular Anwendung nutzen zu können, müssen diese zuerst mit folgenden Kommandozeilenbefehlen, durch den Node Package Manager, heruntergeladen werden:

```
npm install stompjs
```

```
npm install sockjs-client
```

```
// Etablish backend socket connection
 public connect() {
   this.stompClient = Stomp.over(this.socket);
   this.connected = true;
   console.log('websocket connection established');
 }
```

Die send() Methode des websocket Service, wird verwendet um ein Kommando in einem, innerhalb der send() Methode vordefiniertem, Format zu senden. Dabei werden von der Methode zwei Parameter benötigt, throttle und side. throttle gibt dabei an, ob nach Gas gegeben werden soll, gebremst werden soll oder rückwärts gefahren werden soll. Der side Parameter ist notwendig, um dabei anzuzeigen ob das Kommando für den linken oder für den rechten Motor des Kettenfahrzeugs bestimmt ist.
Beide Parameter werden dann zusammen in eine JSON (kurz für JavaScript Object Notation) Struktur geparst und mittels der send() Methode des, in der connect() Methode erstellten, stompClients an den Endpunkt /app/control des Backends gesendet.

```
 public send(throttle: string, side: string) {
   this.stompClient.send('/app/control', {}, JSON.stringify({ '
  throttle': throttle, 'side': side }));
 }
```

8.3 Das Java Back-End

Für die Erstellung des Back-Ends, welches die Befehle des Front-Ends entgegennimmt und an die Motorsteuerungen sendet, wurde Java genutzt. Zur Unterstützung im Buildprozess, also dem Vorgang, welcher aus dem Code eine ausfühbare Datei erstellt, wurde Apache Maven benutzt.
Zusätzlich empfiehlt es sich beim Entwickeln mit Java eine vollwertige IDE (kurz für Integrated Development Environment) zu nutzten. Dies wurde für die Arbeit in Form der IntelliJ IDEA genutzt, welche, durch ihre eingebauten Werkzeuge, die Arbeit beim Erstellen sowie auch beim Deployment des Projektes stark vereinfacht.

Es werden also vorab diese drei Komponenten benötigt, um mit der Entwicklung der Java Anwendung beginnen zu können:

Java Development Kit 1.8 oder neuer[18]

Apache Maven 3.3 oder neuer[19]

Eine IDE nach Wahl; hier verwendet: IntelliJ IDEA Ultimate[20]

Nach der Installation der benötigten Komponenten, kann in ItelliJ IDEA unter **File** | **New** | **Project** ein neues Maven Projekt, mit dem Java Development Kit als Project SDK, erstellt werden. Dieser Prozess erstellt uns ein fast leeres Java Projekt, welches der Maven Namenskonvention folgt und eine pom.xml besitzt, in welcher unter anderem benötigte Bibliotheken, als Abhängigkeiten eingetragen werden können.

[18] `https://www.oracle.com/technetwork/java/javase/downloads/index.html`
[19] `https://maven.apache.org/download.cgi`
[20] `https://www.jetbrains.com/idea/`

Dieses neu erstellte Project enthält bereits eine Klasse welche als Einstiegskomponente des Java Programms dient.

```
@SpringBootApplication
public class ResourceServerApplication extends
    SpringBootServletInitializer {

    public static void main(String[] args) {
        SpringApplication.run(ResourceServerApplication.class,
    args);
    }

}
```

In den nächsten beiden Unterkapiteln wird die Implementation der WebSockets sowie der Ansteuerung der Motoren unter Zuhilfenahme der Pi4J Bibliothek erläutert. Der vollständige Quellcode des Projektes ist auf dem Datenträger zu finden, welcher dieser Arbeit beiliegt. Auf die Verwendung von Maven wird in Unterkapitel 8.3.3 genauer eingegangen.

8.3.1 WebSockets im Backend

Um die WebSocket Kommunikation mit dem Front-End zu ermöglichen, werden im Java Programm zwei Klassen benötigt. Eine Konfigurationsklasse, welche die WebSocket Verbindung und den STOMP Endpoint zur Verfügung stellt und eine Controller-Klasse, welche die eingehenden Nachrichten verarbeitet.

Zuerst muss die Konfigurationsklasse erstellt werden. Hierzu wird eine neue Java-Klasse, mit dem Namen WebSocketConfig erstellt.

```
1  @Configuration
2  @EnableWebSocketMessageBroker
3  public class WebSocketConfig implements
       WebSocketMessageBrokerConfigurer {
4
5      @Override
6      public void registerStompEndpoints(StompEndpointRegistry
       stompEndpointRegistry) {
7          stompEndpointRegistry.addEndpoint("/test").
       setAllowedOrigins("*").withSockJS();
8      }
9
10     @Override
11     public void configureMessageBroker(MessageBrokerRegistry
       messageBrokerRegistry) {
12         messageBrokerRegistry.enableSimpleBroker("/topic");
13         messageBrokerRegistry.setApplicationDestinationPrefixes(
       "/app");
14     }
15
16 }
```

Diese muss mit der Annotation @Configuration versehen werden, damit die Klasse als Konfiguration erkannt wird. Die @EnableWebSocketMessage-Broker Annotation macht die Kommunikation über WebSockets, unter Verwendung eines Subprotokolls, in diesem Fall SockJS, möglich.
Die registerStompEndpoints() Funktion registriert den STOMP-Endpoint, welcher hier "/test" heißt. Dieser Endpoint wird in der Angular Anwendung verwendet, um die WebSocket Verbindung herzustellen.
Mittels configureMessageBroker() wird der Message Broker konfiguriert, welcher die eingehenden STOMP-Nachrichten verarbeitet. Zusätzlich wird in dieser Methode der Endpoint Prefix für den Message Broker als "/app" konfiguriert. Dies ist der Endpoint, welcher von der Angular Anwendung, für das Senden der Steuerungskommandos genutzt wird.

Jetzt kann die Controller-Klasse erstellt werden, welche dann die Nachrichten der Angular Anwendung empfangen und verarbeiten kann. Hierzu muss einer weitere Java-Klasse erstellt werden, welche den Namen WebsocketController bekommt.

```
1  @Controller
2  public class WebsocketController {
3
4      @Autowired
5      CommandService commandService;
6
7      @MessageMapping("/control")
8      public void CommandReceiver(Command command) throws
       Exception {
9
10         this.commandService.drive(command);
11     }
12 }
```

Diese Klasse wird anschließend mit @Controller annotiert und kann eine Methode hinzugefügt werden, welche mittels der Annotation @MessageMapping(“/control”) STOMP-Nachrichten verarbeitet, welche an den Endpoint “/app/controll” gesendet werden. Diese Nachrichten werden dann anschließend als Command Objekt verarbeitet und mit diesem, als Parameter, die drive() Methode der CommandService Instanz aufgerufen.

8.3.2 Motorsteuerung mittels pigpio

Die CommandService Klasse fungiert als Bindeglied zwischen dem STOMP-Endpunkt und der Simon20ASpeedController Klasse, welche die Empfangenen STOMP-Nachrichten als Command Objekt übergeben bekommt, diese dann Interpretiert und die benötigten Methoden der, zuvor angelegten, Simon20ASpeedController Objekte aufruft.

Die Struktur des Command ist folgende:

```
public class Command {
    private Throttle throttle;
    private Side side;


    public Command(Throttle throttle, Side side) {
        this.throttle = throttle;
        this.side = side;
    }

    ...

}
```

Ein Command Objekt besitzt eine throttle Variable und eine side Variable. Diese variablen wiederum sind jeweils als Enum definiert:

```
public enum Throttle {
        GO, STOP, REVERSE
}
```

```
public enum Side {
        LEFT, RIGHT
}
```

Die CommandService Klasse erstellt zunächst zwei Simon20ASpeedController Objekte mit den Pinnummern 18 und 13:

```
    //Use bcm numbering
    Simon20ASpeedController rightESC = new
    Simon20ASpeedController(18);
    Simon20ASpeedController leftESC = new
    Simon20ASpeedController(13);
```

Für die Pinnummern muss die BCM (kurz für Broadcom) Anschlussnummer verwendet werden.
Graphik 8.3.1 zeigt die Nummern aller auf dem Raspberry Pi 3 B+ verfügbaren Pins, dabei stehen die Nummern in den Kreisen für die physische Nummerierung und die äußeren Nummern, in den Rechtecken, für die BCM Nummerierung.

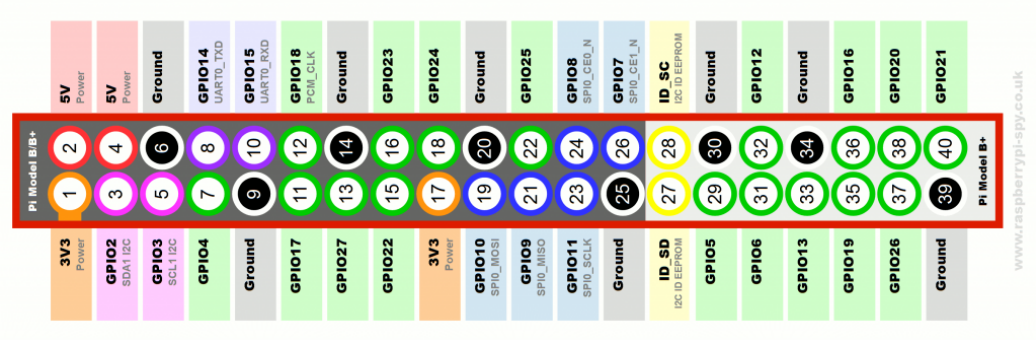

Abbildung 8.3.1: Infografik für Pinnummern des Raspberry Pi
Quelle: `https://www.raspberrypi-spy.co.uk/2012/06/simple-guide-to-the-rpi-gpio-header-and-pins/`

Die Klasse, welche für Ansprechen der Motorsteuerungen mittels PWM ermöglicht, ist die Simon20ASpeedController Klasse. Diese Implementiert ein zuvor erstelltes Interface, welche eine Reihe von Methoden definiert, welche von jeder Klasse, welche Motoren ansteuern soll, implementiert werden müssen. Dies stellt sicher, dass zu einem späteren Zeitraum eine neue Klasse zur Ansteuerung von Motoren erstellt werden kann, ohne dass andere Komponenten der Anwendung geändert werden müssen. Dieses Interface sieht wie folgt aus:

```
1  public interface ElectronicSpeedController {
2      //Interface for the Implementing ESC
3      //Speed should start from 0 to 1024 milliseconds
4
5      void calibrate();
6
7      void arm(boolean isCalibrated);
8
9      void disArm();
10
11     void maxSpeed();
12
13     void midSpeed();
14
15     void testSpeed(int testSpeed);
16
```

```
17      void minSpeed();
18
19      void updateSpeed(int speed);
20
21  }
```

Die Simon20ASpeedController Klasse implementiert alle im ElectronicSpeedController Interface definierten Klassen unter der Verwendung der wiringpi Bibliothek, welche im Pi4J Framework enthalten ist.
Dem Konstruktor der Simon20ASpeedController Klasse muss die Pinnummer, an dem die Motorsteuerungskomponente an den Raspberry Pi angeschlossen ist, übergeben werden. Zusätzlich ruft der Konstruktor die initializeEsc() Methode auf, in welcher die PWM Signallänge und die PWM Frequenz für das PWM Signal des Raspberry Pi gesetzt werden.

```
1
2
3   //Use bcm numbering
4       public Simon20ASpeedController(int pinNumber) {
5           this.pinNumber = pinNumber;
6           this.initializeEsc();
7       }
8
9       private void initializeEsc() {
10
11          Gpio.wiringPiSetupGpio();
12          Gpio.pinMode(this.pinNumber, Gpio.PWM_OUTPUT);
13          Gpio.pwmSetMode(Gpio.PWM_MODE_MS);
14          Gpio.pwmSetClock(192);//40Hz
15          Gpio.pwmSetRange(2000);//2000ms
16      }
```

Um nun das PWM Signal anpassen zu können muss die pwmWrite() Methode der wiringpi Bibliothek aufgerufen werden. Diese Methode benötigt die Pinnummer und die Lange, des zu sendenden PWM Signals, als Parameter. Da jede zuvor im ElectronicSpeedController Interface definierte Methode, welche die Geschwindigkeit des Motors beeinflussen soll, diese Methode benutzen muss, wurde der Aufruf in die updateSpeed() Methode ausgelagert:

```
@Override
    public void updateSpeed(int speed) {
        Gpio.pwmWrite(this.pinNumber, speed);
    }
```

Diese benötigt nun nur noch als Parameter die neue Geschwindigkeit für den Motor, welche in der PWM Signallänge angegeben wird, und nutzt die pinNumber Variable welche bei der Initialisierung des Objekts als Parameter dem Konstruktor übermittelt wird.

8.3.3 Konfiguration des Buildprozesses mit Maven

Mittels Maven kann der Build-Prozess der Java Anwendung, unter richtiger Konfiguration aber auch der Angular Anwendung, automatisiert werden. Maven kann so konfiguriert werden, dass die gebundelte Angular Anwendung automatisch in den "/static" Ordner der Java Anwendung kopiert wird und dann anschließend mit dem .war Archiv deployed wird. Dies hat den Vorteil, dass auf dem Raspberry Pi später nur ein Server laufen muss, was die Auslastung des Arbeitsspeichers und der CPU reduziert.

Hierzu muss das Angular Projekt eine eigene pom.xml Datei erhalten, über welche dann ihr Build-Prozess gestartet wird. Ein Ausschnitt aus der pom.xml der Angular Anwendung, welcher die nötigen Befehle beinhaltet um den besagten Build-Prozess zu starten sieht folgt:

```
<executions>
     <execution>
       <id>install node and npm</id>
       <goals>
         <goal>install-node-and-npm</goal>
       </goals>
     </execution>

     <execution>
       <id>npm install</id>
       <goals>
         <goal>npm</goal>
       </goals>
     </execution>

     <execution>
       <id>npm run build</id>
       <goals>
         <goal>npm</goal>
       </goals>

       <configuration>
         <arguments>run build</arguments>
       </configuration>
     </execution>
</executions>
```

Zusätzlich muss eine Java Klasse erstellt werden um, welche das WebMvcConfigurer Interface implementiert, damit die Angular Anwendung später über ihre vorher festgelegte URL erreichbar ist:

```
@Configuration
public class WebAngularApplicationConfig implements
    WebMvcConfigurer {

    @Override
    public void addViewControllers(ViewControllerRegistry
    registry){
        registry.addViewController("/notFound").setViewName("
    forward:/index.html");
    }

    @Bean
    public WebServerFactoryCustomizer<
    ConfigurableServletWebServerFactory> containerCustomizer() {
        return container -> container.addErrorPages(new
    ErrorPage(HttpStatus.NOT_FOUND, "/notFound"));
    }
}
```

Eine weitere Voraussetzung, um den Build-Prozess für alle Module automatisiert abzuwickeln, ist es, dass diese eine gemeinsame Parent-pom Datei besitzen. Diese wird in Verzeichnis platziert, in dem dann die einzelnen Module platziert werden. Die einzelnen Module werden dann in der Parent-pom Datei aufgelistet:

```
<modules>
     <module>tank-control-center</module>
     <module>tank-control-server</module>
     <module>authorization-server</module>
</modules>
```

Anschließend kann über den sogenannten Maven-Lifecycle “install” ausgeführt werden, welcher dann alle Module zusammen in einer ausführbaren .war Datei zusammenfasst.

8.4 Zusammenbau der Physischen Komponenten

Der Zusammenbau der Physischen Komponenten, also der Motoren, der Motorsteuerungen, des Anschlusses für den Lithium-Polymer-Akku und des Raspberry Pi. Hierzu wird ein Lötkolben benötigt um eventuell fehlende Stecker und und Kabelverbindungen zu verlöten.

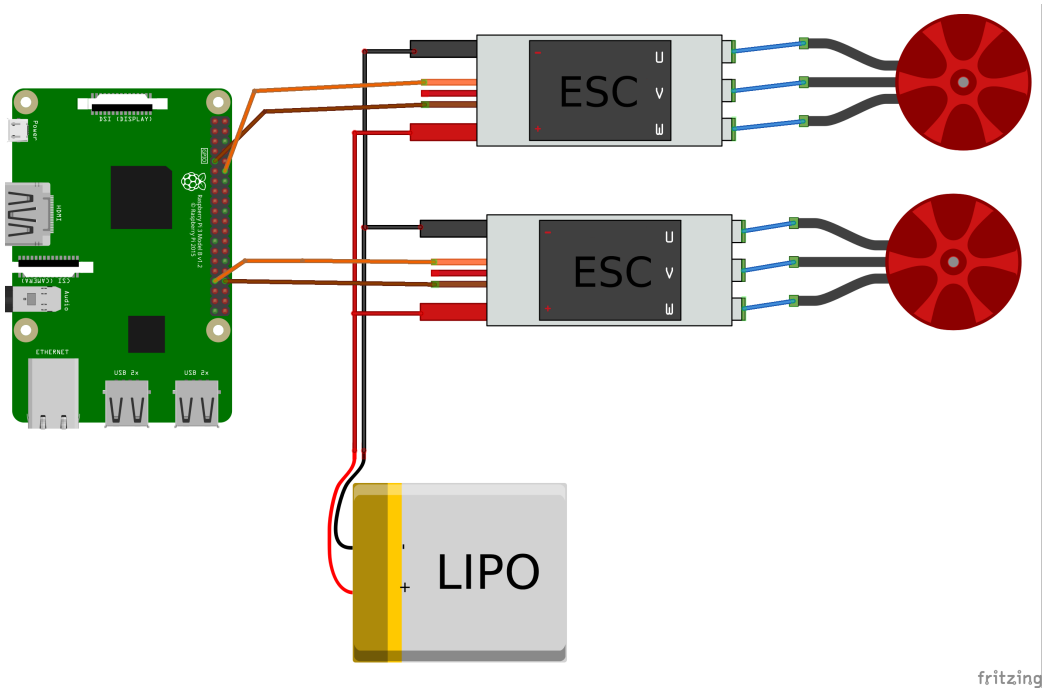

Abbildung 8.4.1: Schaltplan zur Verbindung der Komponenten
Quelle: eigene Darstellung; Hergestellt mit Fritzing

Abbildung 8.4.1 zeigt nach welchem Schema die Verkabelung erfolgen muss. Es empfiehlt sich dabei, die einzelnen Komponenten nicht dauerhaft aneinander zu löten, sonder die Verbindungen, wie auf Abbildung 8.4.2 gezeigten, EC3 Steckern oder ähnlichem zu verbinden.

8.5 Einrichtung des Raspberry Pi

Zuletzt muss nur noch der Raspberry Pi eingerichtet werden. Hierzu empfiehlt es sich, an den Raspberry Pi einen Bildschirm, eine Maus und eine

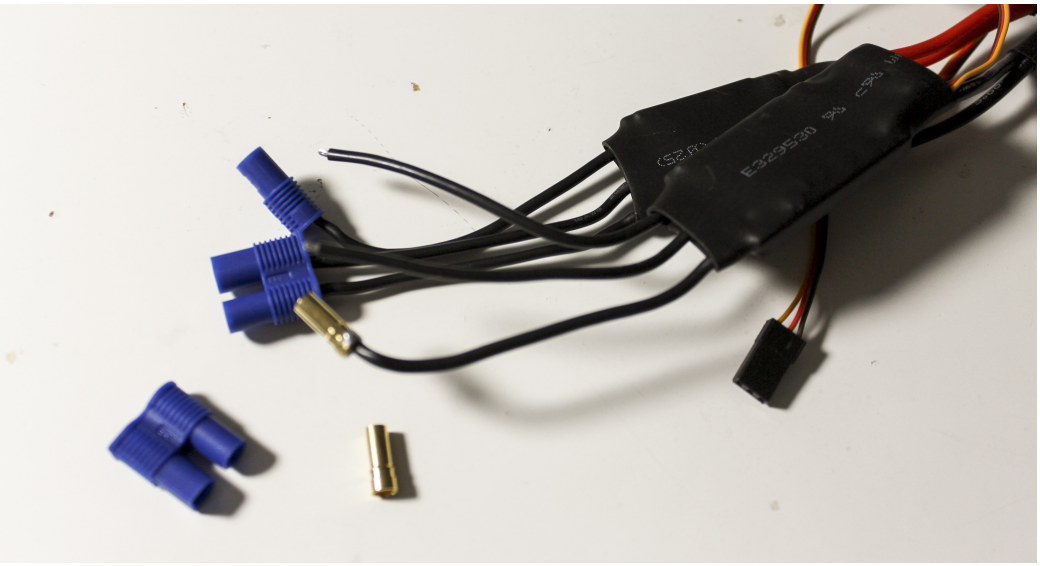

Abbildung 8.4.2: Arbeitsschritt beim Verlöten von EC3 Steckern
Quelle: eigene Darstellung

Tastatur anzuschließen. Dann muss zuerst ein Betriebssystem aufgesetzt werden. Da Raspbian, ein auf Debian basierendes Betriebssystem, welches speziell für den Raspberry Pi entwickelt worden ist, mit vorinstallierter Software, welche für die Nutzung der zuvor erstellten, Programme nötig ist, ausgeliefert wird, macht es das, zu einer guten Wahl, um manuelles nachinstallieren von Software zu vermeiden.

Nach der Installation von Raspbian muss der Raspberry Pi noch mit dem WLAN verbunden werden, dazu muss mit folgendem Befehl die wpa_supplicant.conf Datei geöffnet werden:

```
sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

Und anschließend am Ende der Datei folgender Eintrag ergänzt werden:

```
network={
    ssid="Name"
    psk="Passwort"
}
```

Dabei muss “Name” durch den Namen des WLAN Netzes ersetzt werden und “Passwort” durch dessen Passwort.

Anschließend muss in Unterkapitel 8.3.3 erstellte .war Datei noch auf den Raspberry Pi kopiert werden. Hierzu empfiehlt sich die Verwendung eines USB-Sticks.

Zuletzt muss nur noch eine .desktop Datei im Ordner "/.config/autostart" angelegt werden, welche nach dem Starten des Raspberry Pi automatisch ausgeführt wird. Die Aufgabe der Datei ist es die in Unterkapitel 8.3.3 erstellte .war Datei auszuführen. Das Datei selbst, beinhaltet dabei den nötigen Befehl, um die .war Datei zu starten:

```
1 [Desktop Entry]
2 Type=Application
3 Exec=java -jar -Dpi4j.linking=dynamic /home/pi/Desktop/tank-
     control-server.war
4 Terminal=true
```

Dabei ist darauf zu achten, dass die Datei die Endung .desktop hat, da diese sonst nicht bei Systemstart ausgeführt wird.

9 Bewertung

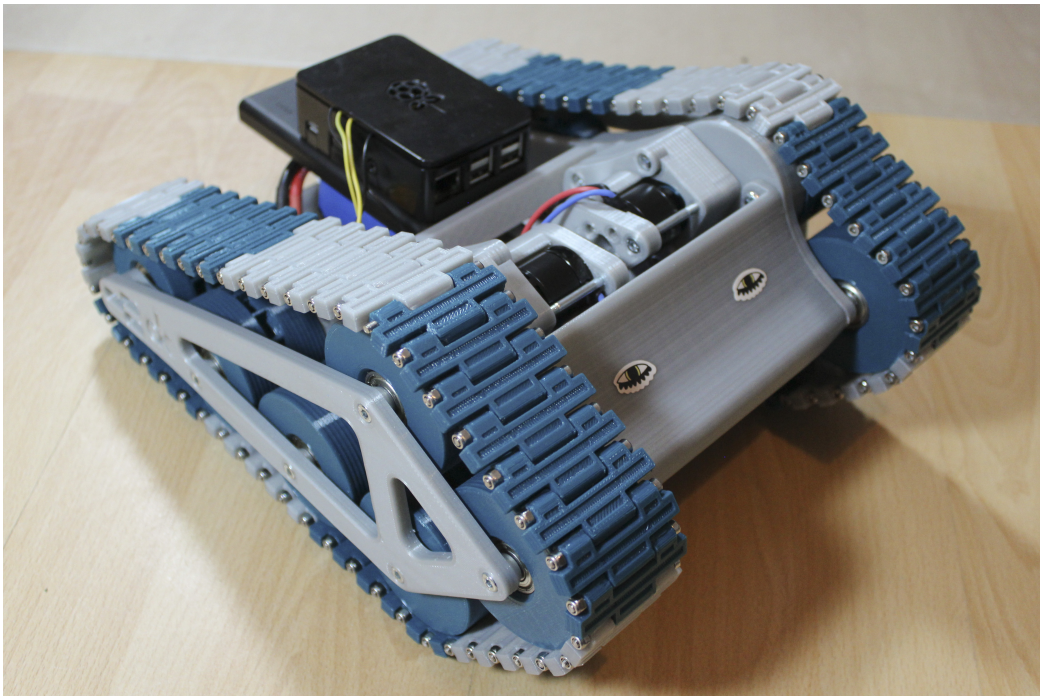

Abbildung 9.0.1: Fertiger Prototyp des Kettenfahrzeugs
Quelle: eigene Darstellung

Die prototypische Entwicklung eine Kettenfahrzeugs auf der Basis eines Einplatinencomputers hat gezeigt, dass diese Äußerst flexibel einsetzbar sind. Auch die Verwendung von eher schwergewichtigen Technologien, wie Java oder auch Angular, funktionierte überraschen gut. Allerdings kann man doch leicht erhöhte Ladezeiten der Webseite feststellen, wenn diese über eine Andere Maschine aufgerufen wird.
Die Steuerung mittels WebSocket-Verbindung ist überraschend performant, welche auch bei schnelleren Manöver des Kettenfahrzeugs für gute Fahreigenschaften sorgt. Es zeigt sich also, dass auch ohne großes Vorwissen mit der Materie Einplatinencomputer, in moderater Zeit kleine bis mittlere Projekte bewerkstelligen lassen. Mitverdienst daran hat auch die große und immer weiter wachsende Raspberry Pi Community, welche eine große Menge an Ressourcen für die Arbeit mit dem Raspberry Pi zur Verfügung stellt.

Einzig die Bedienung der Motorsteuerungen machte anfänglich Probleme, ohne das im Internet Hilfe gefunden werden konnte. Dies wurde dann durch eine Manuelle Eingabe des PWM-Signals gelöst, wodurch die richtigen Werte für die Signale mittels Trial and Error Vorgehen gefunden werden konnte.

10 Ausblick

Eine Verbesserung, derer Implementation aus zeitlichen Gründen leider nicht erfolgte, ist die Einbindung der JavaScript Bibliothek Hammer.js. Diese erlaubt es Aktionen mittels Touch-Gesten auszulösen. Dies würde sich bestens eignen, um das Steuerungs-Front-End auch über ein Smartphone bedienen zu können.

Aber auch weitere Ausbaumöglichkeiten für das Kettenfahrzeug sind durchaus vorhanden. Zum einen wäre die Installation einer Kamera inklusive Videoübertragung an das Steuerungsinterface möglich. Aus dieser Erweiterung würde die Möglichkeit hervorgehen, das Kettenfahrzeug auch ohne Sichtkontakt richtig fahren zu können. Auch wäre die Verwendung einer SIM-Karte möglich, um nicht mehr an eine WLAN-Verbindung gebunden zu sein.

Eine weitere sehr interessante Erweiterung wäre, die später einzubauende Kamera an einen kognitiven Service, wie Beispielsweise dem IBM Watson, anzubinden, welcher dann, neue Möglichkeiten in Form von Bilderkennung oder der Bedienung des Kettenfahrzeuges durch Spracherkennung.
Auch währen Näherungssensoren, wie man sie von modernen Einparkhilfen kennt, eine Möglichkeit die bidirektionalen Fähigkeiten der WebSocket-Verbindung zu nutzen.

Abbildungsverzeichnis

Literatur

Alles, was man über RC-Motoren wissen sollte (2014). URL: http://www.d-edition.de/blog/ratgeber/alles-was-man-ueber-rc-motoren-wissen-sollte/ (besucht am 09. 03. 2019).

Baichtal, John (2018). *THINGIVERSE.COM LAUNCHES A LIBRARY OF PRINTABLE OBJECTS*. URL: https://www.wired.com/2008/11/thingiversecom/ (besucht am 10. 03. 2019).

Brushless vs. Brushed – Motoren (2015). URL: http://www.d-edition.de/blog/ratgeber/brushless-vs-brushed-motoren/ (besucht am 10. 03. 2019).

Cellan-Jones, Rory (2011). *A 15 pound computer to inspire young programmers*. URL: http://www.bbc.co.uk/blogs/thereporters/rorycellanjones/2011/05/a_15_computer_to_inspire_young.html (besucht am 06. 03. 2019).

Dorn, Sebastian (2018). *Warum Angular?* URL: http://www.virtual7.de/blog/de/2018/01/warum-angular/ (besucht am 11. 03. 2019).

Getting started (2018). URL: https://angular.io/guide/quickstart (besucht am 13. 03. 2019).

Gleichstrommaschine (2019). URL: https://de.wikipedia.org/wiki/Gleichstrommaschine (besucht am 09. 03. 2019).

Miller, Paul (2017). *Raspberry Pi sold over 12.5 million boards in five years*. URL: https://www.theverge.com/circuitbreaker/2017/3/17/14962170/raspberry-pi-sales-12-5-million-five-years-beats-commodore-64 (besucht am 06. 03. 2019).

Piksa, Peter (2013). *Wie mein Raspberry Pi meinen Sonnenstrom zu Bitcoins macht*. URL: https://web.archive.org/web/20131004230837/http://www.piksa.info/blog/2013/06/15/wie-mein-raspberry-pi-meinen-sonnenstrom-zu-bitcoins-macht/ (besucht am 06. 03. 2019).

SafeDrone Team (2017). *Electronic Speed Controller – Sie regeln die Geschwindigkeit*. URL: https://www.safe-drone.com/de/electronic-speed-controller-sie-regeln-die-geschwindigkeit/ (besucht am 10. 03. 2019).

SALES SOAR AND RASPBERRY PI BEATS COMMODORE 64 (2018). URL: https://blog.adafruit.com/2018/12/21/23-million-raspberry-pi-computers-sold-raspberry_pi-raspberrypi/ (besucht am 08. 03. 2019).

Storm, Ingo T. (2018). *Der neue Raspberry Pi 3B+ - schneller zum gleichem Preis*. URL: https://www.heise.de/ct/artikel/Der-neue-Raspberry-Pi-3B-schneller-zum-gleichem-Preis-3994036.html (besucht am 09. 03. 2019).

Torrone, Phillip (2018). *23 million + Raspberry Pi computers sold*. URL: https://blog.adafruit.com/2018/12/21/23-million-raspberry-pi-computers-sold-raspberry_pi-raspberrypi/ (besucht am 08. 03. 2019).

Upton, Eben (2016). *Raspberry Pi 3 on sale now at* 35. URL: https://www.raspberrypi.org/blog/raspberry-pi-3-on-sale/ (besucht am 09. 03. 2019).