Hochschule für Technik und Wirtschaft Berlin

Bachelorarbeit

zur Erlangung des akademischen Grades
Bachelor of Engineering
über das Thema

Hard- und Softwareentwicklung für eine autonome Omnirad Plattform auf Basis eines eingebetteten Linux-Systems

im Studiengang
Computer Engineering

am Fachbereich
Ingenieurwissenschaften - Energie und Information

Autor: Zelimir Simic
Matrikelnummer: 531292

Erstprüfer: Prof. Dr. Frank Bauernöppel
Zweitprüfer: Dipl.-Ing. Peter Puschmann

Berlin, den 24.08.2017

Kurzfassung

Seit den 1990er Jahren sind Serviceroboter im Einsatz. Die verbauten Mikroprozessoren stehen der Allgemeinheit in Form von Kleinstcomputern zur Verfügung. Programmierer, Entwickler und Ingenieure aus der ganzen Welt bilden eine große Gemeinschaft, die die Entwicklung von vielen Projekten rund um diese Kleinstcomputer vorantreiben und mit der Welt teilen.
Diese Bachelorarbeit hat als Ziel, einen solchen Kleinstcomputer, den Raspberry Pi, als eingebettetes System zu verwenden, um einen Roboter zu entwickeln, zu bauen und zu programmieren. Es soll eine steuerbare Omnirad Plattform verwendet werden, die über sogenannte Allseitenräder verfügt.
Der Benutzer soll die Möglichkeit haben den Roboter in verschiedene Richtungen, mit unterschiedlicher Strecklänge und Geschwindigkeit zu steuern.
Das Projekt bildet die Grundlage für weitere, zukünftige Entwicklungen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ABS:	Antiblockiersystem
ARM:	Advanced RISC Machine
CPU:	Central Processing Unit
CW:	clockwise
CCW:	counterclockwise
CSI:	Camera Serial Interface
DC:	Direct Current
DMM:	Digitalmultimeter
DSI:	Display Serial Interface
GB:	Gigabyte
GPIO:	General Purpose Input Output
GPU:	Graphics Processing Unit
HDMI:	High Definition Multimedia Interface
I^2C:	Inter-Integrated Circuit
IC:	Integrated Circuit
IoT:	Internet of Things
IP:	Internet Protocol
LAN:	Local Area Network
MB:	Megabyte
Mbit/s:	Megabits pro Sekunde
MHz:	Megahertz
ms:	Millisekunde
PWM:	Pulsweitenmodulation
RC:	Remote Controlled / Radio Controlled
RISC:	Reduced Instruction Set Computing
rootfs:	Root Filesystem
SD:	Secure Digital
SMD:	Surface Mounted Devices
SoC:	System on a Chip
SPI:	Serial Peripheral Interface
SSH:	Secure Shell
UART:	Universal Asynchronous Receiver/Transmitter
USB:	Universal Serial Bus
WLAN:	Wireless Local Area Network

1. Einleitung

In der heutigen Zeit der Informationstechnik ist das Aufgebot der rasanten Entwicklung ein immens voranschreitender Prozess. In der aufkommenden Flut an Technik, die unseren Alltag begleitet, rücken Mikroprozessoren in den Vordergrund. Der Kern bei der Entwicklung eines Roboters umfasst ein eingebettetes System, welches auf Mikroprozessoren basiert. Musste man hingegen vor Jahrzehnten hohe Summen Geld für Computer bezahlen, kann man heutzutage programmierbare Mikroprozessoren zu einem günstigen Preis erwerben. Massenproduktionen und die Marktvielfalt mit dem verbundenen Wettbewerb tragen zu den geringen Preisen bei.
Damit wurde ein Weg für Kinder, Studenten, sowie allen Interessierten geebnet, um Projekte in Eigenarbeit oder mit Hilfe der großen Gemeinschaft, die sich in den letzten Jahren gebildet hat, zu realisieren.

Die Hochschule für Technik und Wirtschaft sucht nach einer mobilen Plattform, die durch weiterführende Arbeiten verbessert und erweitert werden kann. Diese Bachelorarbeit befasst sich mit dem ersten Schritt ein autonomes, mikrocontrollergesteuertes Vehikel zu entwerfen und zu bauen. Das markanteste an der Fortbewegung sind die zum Einsatz kommenden drei Omniräder, die an der vorgegebenen Plattform befestigt sind. Die Omniräder sind auch als Allseitenräder bekannt.
Für eine Ansteuerung und Programmierung des Projekts kommt ein eingebettetes Linux System zum Einsatz. Im Vorfeld der Bearbeitung wurde unter Absprache der Dozenten ein spezifisches, eingebettetes System ausgewählt, der Raspberry Pi. Seit seinem Erscheinen ist der Raspberry Pi in unzähliger Literatur und durch seine kompakte Form in allerlei Projekten, wie der Hausautomatisierung, als Medien-Center oder als Spielekonsole wiederzufinden. In eigenem Ermessen wird eine geeignete Linux-Distribution als Betriebssystem ausgewählt. Dazu werden Vor- und Nachteile diskutiert und ob es sinnvoll wäre ein eigenes Betriebssystem zu erstellen.
Diese Bachelorarbeit fungiert als Dokumentation damit ein Nachbau ermöglicht wird. Ebenso zielt diese Arbeit darauf ab, ein Einstiegspunkt für weiterführende Erweiterungen zu sein.

1.1. Problemstellung/Zielsetzung

Wie der Titel des Bachelorthemas besagt, soll eine steuerbare Omnirad Plattform entwickelt werden. Die Auswahl an technischen Komponenten, die bei diesem Prototypen verbaut werden können, ist vielfältig und bedarf einer genauen Untersuchung, sodass ein Zusammenspiel aller Komponenten am Schluss gewährleistet ist. Damit eine sinnvolle Auswahl getroffen werden kann, wird für die nicht vorgeschriebenen Komponenten auf die Marktübersicht eingegangen, damit Vor- und Nachteile aufgezeigt werden können.
Nachdem die Roboterplattform praktisch aufgebaut wird, folgt im Anschluss eine Softwareentwicklung. Es muss ein Algorithmus programmiert werden, um den Roboter fahrtüchtig zu machen. Weiterhin muss der Algorithmus am Roboter bewerkstelligen, sich autonom fortzubewegen und dabei eine vorgegeben Strecke abfahren. Bereits im Vorfeld ist bekannt, dass eine Auswertung mit Hilfe von Encodern, welche an den Motoren verbaut sind, stattfindet. Dies ermöglicht, dass der Roboter präzise gesteuert werden kann. Mögliche Abweichungen in der Wiederholbarkeit werden jedoch nicht ausgeschlossen.
Weiterhin werden alle Schritte beim Bau, sowie dem Programmieren des Prototypen dokumentiert, sodass zukünftige Erweiterungen unkompliziert implementiert werden können.

1.2. Motivation

Die Motivation sich mit dem Thema in dieser Bachelorarbeit auseinanderzusetzen, ist das während der Studienzeit erworbenen Wissens in Vereinigung von Hard- und Software anzuwenden. Grundlagen der Elektronik werden aufgegriffen. Elektronische Bauelemente werden beleuchtet. Das Wissen um den Gebrauch praktischen Umgangs mit Bauteilen und wie diese miteinander zu verbinden sind wird angewendet. Das Erfassen von logischen Signalen entstammt der Digitaltechnik. Signale zu erzeugen und Algorithmen zu entwerfen, werden durch eine Programmierung realisiert.

Von einer Idee bis zur Realisierung einer technischen Entwicklung, in Form einer praktischen Arbeit, bis hin zur Programmierung umfasst diese Bachelorarbeit vieles des im Studium kennengelernten.

2. Grundlagen

Das folgende Kapitel befasst sich mit den verwendeten Komponenten für das Thema dieser Bachelorarbeit. Für ein besseres Verständnis werden die verwendeten Hardwarekomponenten näher beschrieben und von einander gesondert betrachtet. Die wohlmöglich aufkommenden Fragen, die durch die namensgebenden Begrifflichkeiten des Themas resultieren, um was es sich bei einem eingebetteten System handelt oder was es mit einer Omnirad-Plattform auf sich hat, werden im Detail erläutert. Der zum Einsatz kommende Einplatinencomputer Raspberry Pi wird vorgestellt und seine Anschlussmöglichkeiten aufgezeigt. Es wird auf den technischen und mechanischen Aspekt der Allseitenräder eingegangen, einen Einblick in die vorhandenen und verschiedenen Akkumulatorenmodelle auf dem Markt mit ihrer Funktion aufgezeigt, sowie elektrische Bauteile betrachtet. Zur Ansteuerung mittels Algorithmen, die für eine Bewegung der Plattform verantwortlich sind, wird ein Konzept dargestellt.

2.1. Omnirad Plattform

Abbildung 1 - Omnirad Plattform[1]

Das Hauptaugenmerk dieser Bachelorarbeit beruht auf der Omnirad Plattform. Sie ist in zwei Ebenen unterteilt, auf denen Komponenten Platz finden. Die Plattform ist nahezu dreieckig angeordnet und an jedem der abgeflachten Eckpunkte befindet sich je ein 12 Volt Gleichstrom-Coreless-Motor mit einer maximalen Lastromaufnahme von 500 mA. [10]

[1] https://nodna.de/media/image/product/3024/lg/3wd-48mm-omni-wheel-platform-chassis.jpg [04.07.2017]

Die auffallenden, speziellen Räder der Omnirad Plattform, werden Allseitenräder genannt. Die Lauffläche der Räder besteht aus Rollen, die im rechten Winkel angebracht sind. Diese ermöglichen es der Plattform sich nicht nur in den gewohnten Richtungen, nach vorne und hinten zu bewegen, sondern ebenso nach rechts und links zu rollen. Dies bietet den Vorteil aus dem Stand mehrere Richtungen ansteuern zu können. Für ein genaueres Verständnis der Fortbewegungsmöglichkeiten, die sich durch die Anordnung der Allseitenräder ergibt, dient die untere Abbildung.

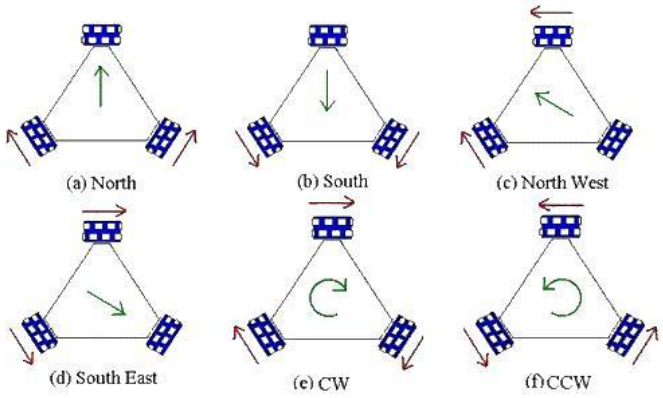

Abbildung 2 - Bewegungsmöglichkeiten der Omnirad Plattform[2]

2.1.1. Gleichstrommotor

Die Aufgabe eines Gleichstrommotors sieht vor, elektrisch zugeführte Energie in mechanische Energie umzuwandeln. Das Drehmoment entsteht durch die Lorentzkraft, die an stromdurchflossenen Drähten im magnetischen Feld entsteht.

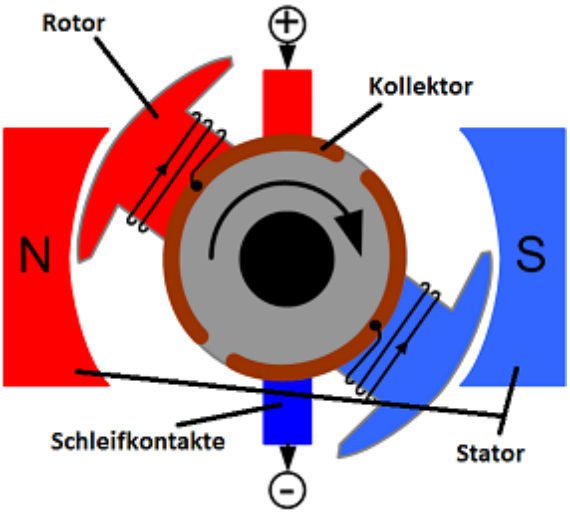

Abbildung 3 - Prinzip eines Gleichstrommotors[3]

Der Aufbau besteht aus feststehenden Magneten, Stator genannt und einem drehbarem Teil, Läufer oder Rotor. An den Kontakten der Stromquelle sind Schleifkontakte in Form von Kohlebürsten verbaut. Der Rotor ist mit Kupferlamellen behaftet (Kollektor), welche halbkreisförmig angeordnet sind. Der Rotor besteht je aus einer oder auch mehreren Wicklungen, die von Strömen durchflossen werden, sobald ein Kontakt der Kupferlamellen und Kohlebürsten stattfindet. Die stromdurchflossenen Wicklungen erzeugen durch die Lorentzkraft ein Magnetfeld, welches dem Magnetfeld des Stators gegenübersteht. Diese stoßen sich voneinander ab und der positiv geladene Rotor wird an die Schleifkontakte der negativen Stromquelle bewegt. Eine Umpolung findet statt. Der vormals positiv geladene Kollektor wird nun von der negativen Stromquelle gespeist. Die abermals auftretende Lorentzkraft führt zu einer permanenten Bewegung des Rotors. [38][39]

[2] http://sine.ni.com/cs/app/doc/p/id/cs-14839#prettyPhoto[gallery]/3/ [29.07.2017]
[3] http://technik-einkauf.de/wp-content/uploads/sites/13/2017/04/motor1.png [15.07.2017]

Bei den zum Einsatz kommenden Motoren handelt es sich um Coreless Motoren. Sie sind im deutschen Sprachgebrauch unter dem Begriff Glockenläufer- oder Glockenankermotor vertreten. Der Glockenläufermotor ist eine spezielle Bauform des Gleichstrommotors und gehorcht den gleichen Gesetzmäßigkeiten wie ein Gleichstrommotor.

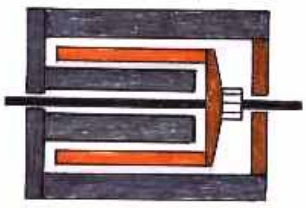

Abbildung 4 - Vereinfachte Darstellung eines Glockenläufer-Motors[4]

Die Glockenläufermotoren bestehen aus eisenlosen, selbstragenden Rotorwicklungen und umschließen den Stator wie eine Glocke. [41][S.37]
In der nebenstehenden Abbildung 4 sind diese orange gekennzeichnet.
Der Verzicht von Eisen ist ein Vorteil, da das Gesamtgewicht des Motors reduziert wird.

Die Bezeichnung der Motoren lautet: 22CL-3501PG

Der Motorenhersteller "Namiki Precision" stellt auf seiner Webseite kein Datenblatt für die Motoren von einem Durchmesser von 22 mm zur Verfügung. Lediglich auf der Versandseite der Omnirad Plattform werden Angaben über die Spezifikationen zur Verfügung gestellt. [10]

Die angegebene Leistung wird mit 6 Watt deklariert. Im Leerlauf beträgt eine Radumdrehung 9600 Umdrehungen pro Minute. Daraus resultieren 160 Radumdrehungen pro Sekunde.

[4] http://www.ferromel.de/teo/glock1.jpg [15.07.2017]

2.1.2. Encoder

Jeder Motor ist mit einem Quadraturencoder versehen. Weitere Bezeichnungen sind Rotary Encoder oder Drehgeber. Sie müssen unabhängig von den Motoren mit Strom und Spannung versorgt werden.

Bei Drehung des Motors erzeugt der Drehgeber an zwei Datenleitungen ein digitales Signal.
Verantwortlich für die Signalerzeugung ist ein optischer Sensor. Eine mit der Motorenachse verbundene Abdeckung wandert bei Bewegung des Motors über den optischen Sensor.
Der Encoder beginnt mit der Signalerzeugung eines High-Pegels, sobald sich der Sensor im abgedunkelten Zustand befindet.
Aus den Signalen lassen sich Drehrichtung, zurückgelegte Strecke und Geschwindigkeit ermitteln.

Eine Bestimmung der Drehrichtung erfolgt durch 90° phasenverschobene Signale A und B. [57] Bewegt sich der Motor mit dem Uhrzeigersinn (clockwise, Abk. CW), wird primär auf der Signalleitung A ein High-Pegel erzeugt und sekundär auf Leitung B. Umgekehrt verhält sich die Signalerzeugung, sobald sich der Motor gegen den Uhrzeigersinn (counterclockwise, Abk. CCW) dreht.

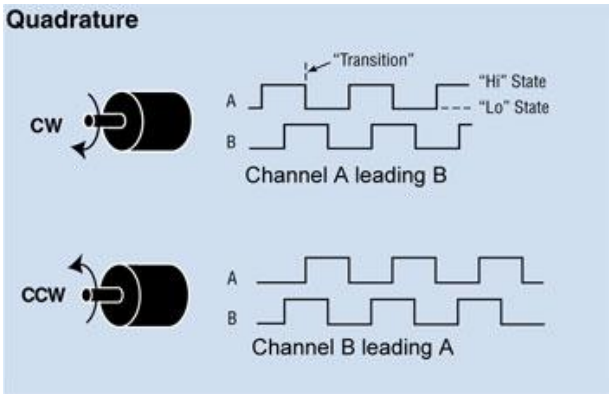

Abbildung 5 - Signalerzeugung eines Quadraturencoders[5]

Die Bestimmung der Geschwindigkeit erfolgt über die Zeitlänge des High-Pegels. Befinden sich die Motoren in einer langsamen Rotation, findet eine längere Zeitspanne der Abdunkelung des optischen Sensors statt. Eine längere Erzeugung des High-Pegels ist die Folge. Schnelle Motorenbewegungen hingegen erzeugen verhältnismäßig kurze High-Impulse.

Die Ansammlung aller erfassten Signale liefert die zurückgelegte Strecke.

[5] http://www.dynapar.com/uploadedImages/_Site_Root/Technology/Encoder_Basics/quadrature.jpg [29.07.2017]

2.2. Raspberry Pi

Der Raspberry Pi ist ein Einplatinencomputer im Kreditkartenformat und kostet ca. 35 Euro.

Seinen Ursprung hat der kleine Computer an der Universität Cambridge. [1][S.1] Der britische Informatiker Eben Upton entwickelte 2006 einen ersten Prototypen. [2][S.12] Die Intention dabei war es, ein kostengünstiges, einfach zu programmierendes und universell einsetzbares Produkt zu entwickeln. Die Grundidee war es, Kindern, Jugendlichen, sowie allen Interessierten, die nicht über die finanziellen Mittel für einen richtigen Computer verfügen, das Programmieren von Computern näher zu bringen und diese dafür zu begeistern, sowie es einst der Commodore 64 seinerzeit in den 1980er-Jahren vermochte. [6][S.13],[5][S.15]

Mit der Markteinführung von 2012, durch die Raspberry Pi Foundation, avancierte der Raspberry Pi schnell unter Elektronik- und Computer-Fans. Seinen Erfolg verdankt er zum einen durch die Wahl des kostenfreien Open-Source Betriebssystems Linux und zum anderen, seinen allgemeinen Allzweckeingabe und -ausgabe Anschlussmöglichkeiten, die in der Fachsprache General Purpose Input/Output (kurz GPIO)-Pins genannt werden. An dieser Steckleiste können externe, periphere Geräte oder Messinstrumente angeschlossen werden, um Daten zu verarbeiten oder Funktionen zu steuern. [5][S.15]

Im Lieferumfang des Raspberry Pis befindet sich kein Gehäuse oder sonstiges Zubehör. Ein Netzteil zur Spannungsversorgung und eine microSD-Karte, welches als Festplatte fungiert, sind unabdingbar für eine Nutzung und müssen separat erworben werden. Dies ist bereits ausreichend für eine minimale Ausstattungsform und sofern das System für eine spezielle Aufgabe programmiert wird, kann es als Embedded System bezeichnet werden. [1][S.1] Betrieben wird das System mit einer Software. Entsprechend der Aufgabenstellung, kann es sich dabei einerseits um ein selbstgeschriebenes Programm handeln oder um ein angepasstes Betriebssystem. Je nach Einsatzgebiet stehen dem Benutzer mehrere Betriebssysteme, zumeist Linux-basierte Distributionen, zur Verfügung. Die vom Hersteller entwickelte Linux-Distribution ist Raspbian. Sie ist die populärste Distribution und basiert auf einer gängigen Linux-Distribution namens Debian. Arch Linux ARM, Ubuntu, OpenELEC sind weitere Betriebssysteme, selbst Microsoft bietet eine kostenlose Windows 10 IoT-Version an. [5][S.33,34]
Durch seine Anschlussmöglichkeiten von USB-Geräten, beispielsweise Maus und Tastatur, einem Ethernet-Port für eine Internetverbindung und eine HDMI-Buchse für einen Monitor, kann der Raspberry Pi entweder als Computerersatz verwendet werden, um Textverarbeitungsprogramme zu verwenden, im Internet zu surfen oder um als Home-Media-Center zu dienen. Über die CSI-Schnittstelle kann eine Kamera und über eine DSI-Schnittstelle ein kompakter Monitor angeschlossen werden.

Der Raspberry Pi hat sich seit seiner Einführung mehr als 12,5 Millionen Mal verkauft. [7] Seit Februar 2016 ist er in der dritten Generation verfügbar. [5][S.22]

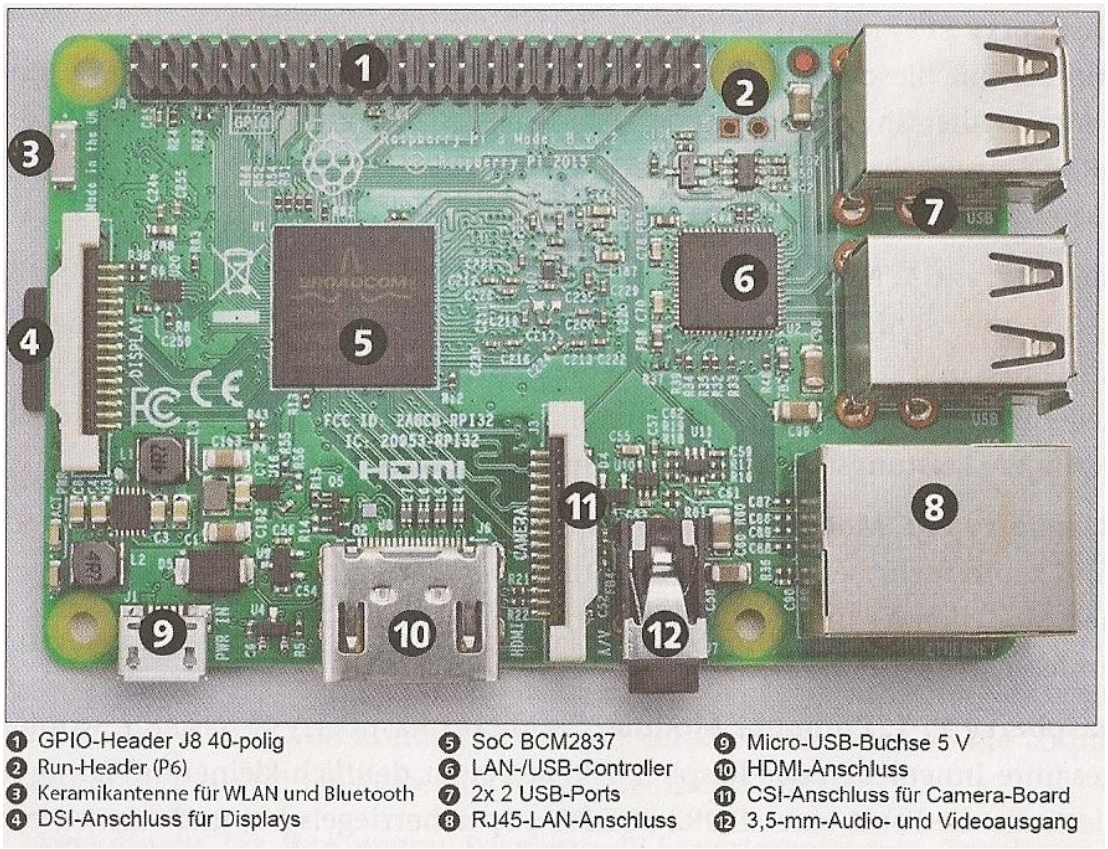

Abbildung 6 - Raspberry Pi 3 Modell B[6]

2.2.1. Datenblatt: Raspberry Pi 3 Modell B [8]

- Maße (Länge x Breite x Höhe): 85,6 mm x 56,0 mm x 20 mm
- Gewicht: 40 g
- SoC: Broadcom-BCM2837
- CPU
  - Typ: ARM Cortex-A53
  - Kerne: 4
  - Takt: 1200 MHz
  - Architektur: ARMv8-A (64 Bit)
- GPU: Broadcom Dual Core Videocore IV
- Arbeitsspeicher: 1024 MB
- Netzwerk
  - Ethernet: 10/100 MBit/s
  - WLAN: Broadcom BCM43143 2,4 GHz WLAN b/g/n
  - Bluetooth: 4.1 Low Energy
- Schnittstellen
  - GPIO-Pins: 40
  - CSI, DSI, I^2C, SPI, UART, microSD-Slot
- Videoausgabe: HDMI (Typ A), Composite Video
- Audioausgabe: HDMI (digital), 3,5-mm-Klinkenstecker (analog)
- Betriebsspannung: 5 Volt Micro-USB-Anschluss (Micro-USB-B

[6] Quellenverzeichnis [5][S.23]

Das Thema der Bachelorarbeit beinhaltet den Begriff "eingebettet". Eingebettete Systeme lassen sich als mikroprozessorgesteuerte Systeme bezeichnen. Diese übernehmen spezifische Aufgaben für übergeordnete Anwendungen oder sind als eigenständige Geräte oder Geräteteile konstruiert. Eingebettete Systeme sind häufig mit einem hohen Maß an Autonomie verbunden. Dies schließt sowohl eine mögliche Interaktion mit der Umwelt durch die Erfassung und Ausgabe analoger und digitaler Signale, als auch Schnittstellen zur Kommunikation mit anderen eingebetteten Systemen und Computern ein. Durch die zunehmende Leistungsfähigkeit der Prozessoren innerhalb der letzten Jahre, ist es möglich aufwendige Algorithmen und Kommunikationsprotokolle in solche kompakten Geräte zu integrieren. Sie sind in größeren Geräten „eingebettet“, wie z.B. das ABS in einem Bremssystem. [20][S.1]

Da das Einsatzgebiet dieser eingebetteten Systeme unter Anderem vorsieht, ununterbrochen in Betrieb zu sein, zeichnet sie ein geringer Leistungsverbrauch aus. Die sparsamsten Systeme haben einen Verbrauch von 2,5 Watt, die im Dauerbetrieb weniger als 5 € pro Jahr an Kosten verursachen. [3][S.XIII]

Das eingebettete System, welches in dieser Bachelorarbeit verwendet wird, ist der bereits vorgestellte Raspberry Pi. Streng genommen handelt es sich bei dem Raspberry Pi nicht um einen Mikroprozessor, sondern um einen Mikrocontroller. Ein weitere Bezeichnung, die in den letzten Jahren vermehrt in der Literatur verwendet wird, lautet SoC (Silicon on a Chip, Silizium auf einem Chip). [4][S.21]

Mikroprozessoren besitzen einen Systembus, welcher sich in Adress-, Daten- und Steuerbus aufteilt, damit externe Hardware, wie beispielsweise der Arbeitsspeicher, schnellstmöglich adressiert werden kann. [59]
Durch die voranschreitende Miniaturisierung im Fertigungsprozess von Mikroprozessoren, ist es möglich, neben dem Mikroprozessor, zusätzliche Peripherie auf dem Chip zu implementieren. Der Mikrocontroller beherbergt beispielsweise verschiedene, unabhängige, integrierte, serielle Schnittstellen, Timer, Arbeitsspeicher oder Flashspeicher. [4][S.21]

Somit ist der Mikrocontroller oder auch SoCs genannt, das Resultat einer Weiterentwicklung von Mikroprozessoren. [9]

Zu den bekanntesten eingebetteten Systemen zählen Raspberry Pi, Banana Pi, Beagle Bone Black und Cubietruck, um nur einige Systeme zu nennen. [4][S.22]

2.2.2. GPIO-Pins

Wie eingangs erwähnt, werden die General Purpose Input/Output-Pins dahingehend verwendet, um externe, periphere Geräte oder Messinstrumente anzuschließen, um Daten zu verarbeiten oder Funktionen zu steuern.
Damit ein fehlerloser Anschluss an einen gewählten Pin gewährleistet wird, dient ein Pfeil als Indikator der Orientierung. Der Pfeil ist in Abbildung 6 auf der oberen linken Seite neben den GPIO-Pins vorzufinden. Dieser markiert den ersten Pin der GPIO-Steckleiste.
Aus der Abbildung 7 ist zu erkennen, dass drei Nummerierungssysteme zur Bezeichnung der Pins existieren. Eine Auswahl der Namensbezeichnung ist bei einer softwaretechnischen Umsetzung zwingend erforderlich.
Die physische Position wird durch die Spalte Pin# repräsentiert. Die Bezeichnung NAME bezieht sich auf die offizielle Dokumentation des auf dem Raspberry Pi verbauten BCM2837-Chips. Ferner haben die Entwickler eine weitere Bezeichnung eingeführt, welche in der Abbildung 7 nicht weiter deklariert ist, aber zum Teil den Pin in seiner Funktion ausweist.
Zur Illustrierung sind die Pins in der Abbildung 7 farblich in Gruppen markiert, damit Zusammenhängende Funktionen der Pins deutlicher ausgemacht werden können.

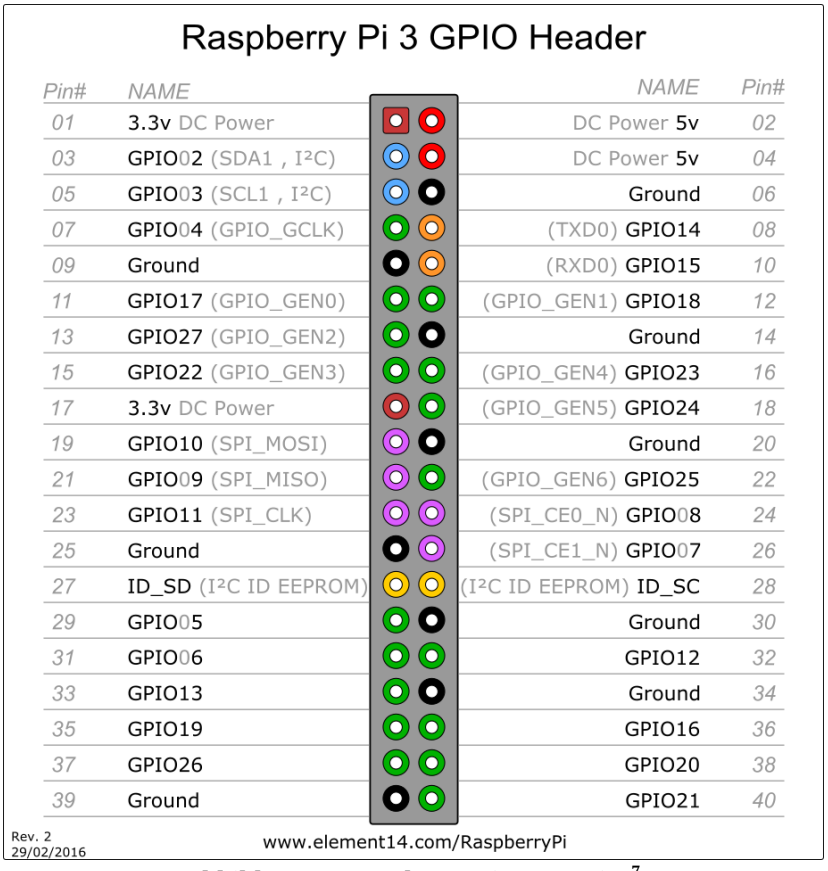

Abbildung 7 - Raspberry Pi GPIO-Pins[7]

Auf der nachfolgenden Seite werden die jeweiligen Funktionen näher erläutert.

[7] https://www.element14.com/community/servlet/JiveServlet/previewBody/73950-102-11-339300/pi3_gpio.png [14.07.2017]

Die GPIO-Steckleiste verfügt über insgesamt 40 Anschlussmöglichkeiten.

Zu den 40 Pins zählen jeweils zwei Gleichspannungs-Pins für eine 5 Volt bzw. 3,3 Volt Spannungsversorgung (rot). Die acht Ground-Pins sind Masse-Pins und werden verwendet, um Ströme aus Schaltkreisen abzuleiten und das Spannungspotential auf null zu setzen (schwarz).
Beim Startvorgang des Raspberry Pis werden die Pins 8 und 10 (orange) standardmäßig als universelle serielle Schnittstelle konfiguriert (UART). [35]

Die Pins 3 und 5 (blau) sind I^2C-Komponenten vorbehalten. Die Pins sind mit einem 1,8 kΩ Pull-Up-Widerstand verbunden und eignen sich, abgesehen vom I^2C-Bus, gut für Signaleingänge, beispielsweise Schalter oder Taster. [5][S.350] Pull-Up ist ein hochohmiger Widerstand, welcher eine Signalleitung auf ein höheres Spannungspotential verbindet, sollte es zu keiner aktiven Potentialerniedrigung kommen.
Zusätzlich gehören Pin 27 und 28 (gelb) ebenfalls den I^2C-Komponenten an. Diesen ist jedoch eine Kommunikation mit EEPROMs vorbehalten. EEPROMs sind nichtflüchtige, elektronische Bausteine, die programmiert werden.

Die Pins 19, 21 und 23 umfassen den SPI-Bus (violett).

Die übrigen GPIO-Pins (grün) können für allgemeine Zwecke eingesetzt werden. Die Funktionen dahinter ermöglichen es Signale mit High- und Low-Pegel zu generieren, um einfache Bauelemente, wie LEDs anzusteuern. Eine Pulsweitenmodulation (PWM), um eine LED zu dimmen oder einen Motor mit unterschiedlich schnellen Radumdrehungen anzusteuern, ist ebenfalls möglich. Hierfür existieren zwei PWM-Kanäle, die sich an den Positionen 12 und 35 oder alternativ an 32 und 33 befinden. [62][S.102] Mit einer PWM werden Signale erzeugt, die sich zwischen High- und Low-Pegel (1 = an und 0 = aus) befinden.
Verwendet man GPIO-Pins zur Ansteuerung von externen Komponenten, so sollte beachtet werden, dass der Steuerungsstrom pro Pin 16 mA bzw. 50 mA für alle GPIO- Pins beträgt. [5][S.349]

Mittels Software ist es möglich den nicht-grünen-Pins Funktionen von GPIO-Pins zuzuweisen.

2.3. Motortreiber

Der nutzbare Strom von 16 mA, der an einem GPIO-Pin des verwendeten Mikrocontrollers entnommen werden kann, ist unzureichend für eine Motorenansteuerung. Eine entsprechende Leistungsstufe wird benötigt, damit ein Motor mit einer ausreichenden Stromstärke von 500 mA und einer Spannung von 12 Volt versorgt werden kann.
Bei einem Motorentreiber handelt es sich um einen Treiberbaustein mit integrierter H-Brücke. Eine H-Brücke wird verwendet um Motoren anzusteuern, diese umzupolen und dadurch deren Drehrichtung umzukehren Daher eignet sich dieses Bauteil ideal für den Bau von kleinen Autos oder Robotern. [5][S.413,414]
Skizzenhaft dargestellt ist die H-Brücke in der Abbildung 8 - Prinzip einer H-Brücke.
Werden die Schalter S1 und S4 geschlossen, liegt an der linken Seite des Motors das positive Potential und an der rechten Seite das negative Potential an. Werden hingegen die Schalter S2 und S3 geschlossen, befindet sich das positive Potential auf der rechten und das negative Potential auf der linken Seite. So kommt es zu einer Veränderung der Drehrichtung des Motors. Ein Motorenstillstand, repräsentiert durch ein abruptes Abbremsen, kann durch einen Kurzschluss herbeigeführt werden, indem die Schalter S2 und S4 geschlossen werden.

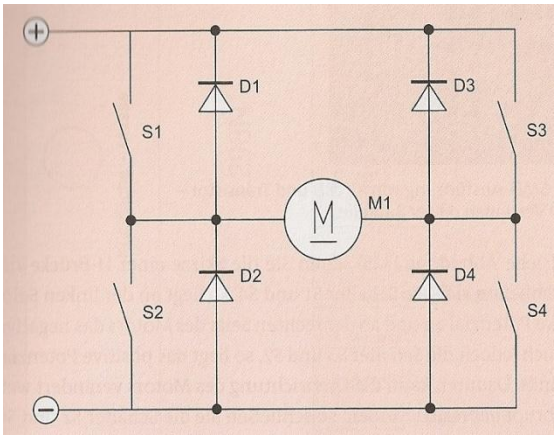

Abbildung 8 - Prinzip einer H-Brücke[8]

Die in dem Schaltplan vorzufindenden Dioden D1 - D4 sind Schutzdioden, auch als Freilaufdioden bekannt. Sie haben die Aufgabe die Ansteuerungselektronik vor Spannungsspitzen aus den Wicklungen des Motors zu schützen. [5][S.414]
Durch die Induktivität der Spulen wird Energie in dem dadurch aufgebautem Magnetfeld gespeichert. Wird ein Motor abgeschaltet, fließt weiterhin Strom, der den Treiberbaustein zerstören könnte.

An einer reichlichen Marktauswahl an geeigneten Motorentreibern mangelt es nicht. Die Unterschiede belaufen sich auf die Anzahl der verbauten H-Brücken, die damit einhergehenden, zur Verfügung stehenden Pins, der zugeführten Stromstärke und zulässigen Versorgungsspannung.

Bezeichnung	Versorgungs-spannung in Volt	max. Stromstärke in Ampere	Anzahl H-Brücken	Bemerkung
MC33926[11]	5 - 28 V	5 A	2	-
L298N[12]	5 - 46 V	4 A	2	SMD-Bauform: L298P[15]
STL9958[13]	4 - 28 V	8,6 A	1	SPI-Ansteuerung
IRLIZ44N[14]	55 V	47 A	-	MOSFET-Transistor hohe Leistung eine Drehrichtung

Tabelle 1 - Auszug der Marktübersicht von Motorentreibern

8 Quellenverzeichnis [5][S.413]
11 https://www.pololu.com/product/1213 [22.07.2017]
12 https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf [22.07.2017]
13 http://www.st.com/en/automotive-analog-and-power/l9958.html [22.07.2017]
14 https://www.mikrocontroller.net/articles/MOSFET-%C3%9Cbersicht [22.07.2017]
15 Quellenverzeichnis [5][S.414]

2.4. Akkumulator

Mit dieser Bachelorarbeit wird der Bau eines autonomen Vehikels angestrebt. Damit dieses auch autonom agieren kann und nicht durch eine konventionelle Stromversorgung durch eine Anbindung an eine Steckdose angewiesen ist, wird ein wiederaufladbarer Energiespeicher in Form eines Akkumulators verwendet.

Anhand einer, auf der nächsten Seite vorzufindenden Tabelle, wird die Markübersicht vorhandener Akkumulatoren aufgezeigt. Vor- und Nachteile und deren regulärer Einsatzort, der jeweiligen Akkumulatoren, werden ebenfalls in der Tabelle aufgeführt. Im Vorfeld werden einige Begrifflichkeiten erläutert, sodass die Tabelle verständlich ist.

Die **Energiedichte** ist ein Maß und gibt Aufschluss darüber, wie viel Kapazität ein Energieträger mit einer gewissen Masse hat. Die Einheit der Energiedichte wird erfahrungsgemäß in Joule pro Kilogramm angegeben. Die hier verwendete Maßeinheit erfolgt in Wattstunden pro Kilogramm. Umso mehr Wattstunden pro Kilogramm an Masse des Akkumulators geliefert werden können, desto geringer kann ein Akku im Vergleich zu anderen Akkutypen gebaut werden, um dennoch die gleiche Energiemenge zu liefern. [19]

Die **Zellspannung** gibt an wie viel Spannung eine einzelne Zelle beherbergt. Dieser Wert ist abhängig von den eingesetzten Materialien. Zudem sei angemerkt, dass durch "in Reihe" schalten von mehreren Zellen die Spannung zu einem Vielfachen erhöht werden kann.

Die **Lebensdauer** der Akkumulatoren hängt von seinen Lade- bzw. Entladezyklen ab. Die Lebensdauer wird in Jahren und Zyklen angegeben.

Der **Ladewirkungsgrad** sagt aus, wie hoch der Energieanteil ist, welcher durch Wärmeentwicklung beim Laden bzw. Entladen verloren geht.

Die **Selbstentladung** bezeichnet selbstablaufende Vorgänge, die zu einer Entladung führen ohne dass ein Verbraucher angeschlossen ist. Im verbauten Material führen Nebenreaktionen dazu, dass das elektrochemische aktive Material in den Elektroden verbraucht wird.
Aufgrund interner Kurzschlüsse, hervorgerufen durch einen mangelhaften Separator, gerät Material der positiven und negativen Pole miteinander in Kontakt.
Selbstentladungen treten in allen Akkumulatoren auf. [27]

Akkutyp / Material	Energiedichte in Wh/kg	Zell-spannung in Volt	Ladewirkungs-grad in %	Lebensdauer Jahre	Lebensdauer Zyklen	Selbstentladung mtl. in %	Vorteil	Nachteil	Einsatzort
Blei	**30 - 40 30 - 50**	**2,0 V**	**60 - 70 %**	**4 - 8**	**300 - 600 200 - 300**	**5 - 10 % 5 %**	**kurzzeitig hohe Ströme**	**sehr niedrige Energiedichte**	**vorzugsweise in KFZ**
Nickel-Cadmium (NiCd)	**40 - 50 45 - 80**	**1,2**	**70 %**	**15**	**800 - 1500 1500**	**10 - 15 % 20 %**	**hohe Toleranz gegenüber Tiefentladung**	**EU- Verbot, nur mit Ausnahmeregel, Cadmium ist umwelt-schädigend**	**Medizingeräte, Elektrowerkzeug, Alarmsysteme**
Nickel-Metallhydrid (NiMH)	**60 - 80 60 - 120**	**1,2**	**70 %**	**7 - 10**	**350 - 500**	**15 - 20 % 30 %**	**kein Memory Effekt**	**hohe Selbstentladung empfindlich gegenüber Überladung, Tiefentladung**	**Ersatz für handelsübliche Batterie**
Nickel-Metallhydrid mit geringer Selbstentladung (NiMH LSD)	**60 - 80**	**1,2**	**70 %**	**7 - 10**	**350 - 500**	**1 - 2 %**	**kein Memory Effekt**	**empfindlich gegenüber Überladung, Tiefentladung**	**Ersatz für handelsübliche Batterie (Nachfolger der NiMH Akkus)**
Lithium-Ionen (LiIon)	**120 - 180 110 - 160**	**3,6**	**90 %**	**10 - 15**	**500 - 800 500 - 1000**	**1 - 2 % 10 %**	**kein Memory Effekt**	**empfindlich gegenüber Tiefentladung Temperatur-abhängig**	**Mobiltelefon, Notebook**
Lithium-Polymer (LiPo)	**130 - 150 100 - 130 262**	**3,7**	**90 %**	**7 - 10**	**300 - 500**	**1 - 2 % 10 %**	**kein Memory Effekt**	**empfindlich gegenüber Tiefentladung Temperatur-abhängig**	**Modellbau**

Tabelle 2 - Vergleich: Akkumulatoren

1. Zeile in einer Zelle, Vor- und Nachteile, Einsatzort: http://www.akku-abc.de/akku-vergleich.php [13.07.2017]
2. Zeile in einer Zelle: Quellenverzeichnis [36]
3. Zeile (LiPo): https://web.archive.org/web/20131103220103/http://www.gebattery.com.cn/product/High_energy_battery.html [14.07.2017]

Aufgrund der vorliegenden Werte wird ersichtlich, dass sich Akkumulatoren mit verbautem Lithium-Material von den übrigen Akkumulatoren abheben.
Ausgezeichnet hat sich diese Form der Energiespeicherung durch seine hohe Energiedichte, bei einer verhältnismäßig geringen Masse und einer langen Lebensdauer, bei einer geringen Selbstentladung. [23] Ein weiterer Vorteil ist ein nichtvorhandener Memory-Effekt. Der Memory-Effekt äußert sich bei häufiger Teilentladung des Akkus, indem ein früher Spannungsabfall eintritt. Eine Verringerung der nutzbaren Kapazität ist die Folge. Mit der Zeit kann die benötigte Mindestspannung nicht länger erbracht werden und der Akku wird unbrauchbar.

Ein negativer Aspekt der Lithium Akkumulatoren besteht in ihrer Empfindlichkeit gegenüber der Tiefentladung bzw. der Überladung. Die Tiefentladung ist ein Faktor, der einen direkten Einfluss auf die Lebensdauer hat. Dabei kommt es zu einer vollkommenen Entladung einer Zelle durch Stromentnahme. Die Spannung sinkt dabei unter die Entladeschlussspannung, einem Grenzwert, bei dem keine nutzbare Energie mehr entnommen werden kann. In mehrzelligen Akkumulatoren, die in Reihe geschaltet sind, werden die Zellen mit der geringsten Ladekapazität in ihren Polaritäten umgepolt. [21]
Der Effekt der Überladung tritt ein, sofern die äußere, zugeführte Spannung nach vollständiger Aufladung weiterhin am Akkumulator anliegt. Interne chemische Prozesse resultieren zu einer Aufheizung des Akkumulators, Brandgefahr besteht!
Die Temperaturabhängigkeit ist ein weiterer Nachteil der Lithium-Akkumulatoren. Bei hohen Temperaturen tritt eine vermehrte Zell-Oxidation auf. Dabei oxidieren die Elektroden und verlieren ihre Eigenschaft Lithium-Ionen zu speichern, welche essentiell für den Stromfluss sind. Besonders hervorgerufen wird dieses Phänomen bei der Benutzung und des gleichzeitigen Ladevorgangs von elektronischen Geräten, wie beispielsweise Notebooks. [22]
In modernen Lithium-Akkumulatoren werden Präventivmaßnahmen ergriffen, um den negativen Effekten entgegenzuwirken. Sie sind in Form einer Überwachungselektronik im Akkumulator vorzufinden. Sie dienen dem Schutz gegenüber Tiefentladung, Überladung und thermischer Überlastung. [25]

Lithium-Polymer-Akkumulatoren, kurz LiPo-Akkus, sind eine Ausführung oder Weiterentwicklung des Lithium-Ionen Akkus.
Die Funktionsweise beider genannter Akkumulatoren ist identisch. Lediglich das verwendete Material im Inneren des Akkumulators, der Elektrolyt, unterscheidet sie. In Lithium-Ionen Akkus besteht dieser aus einer flüssigen, mit Salzen angereicherten Substanz.
LiPo-Akkus verwenden eine feste oder gelartige Folie auf Polymerbasis. Der Vorteil von besagtem Gel ist, dass es einerseits nicht auslaufen kann und andererseits, dass der Akku nicht länger zwingend ein Gehäuse benötigt, so können unterschiedliche Formen produziert werden, die sich den Geräten flexibel anpassen können. [24]

Anwendungsbereiche des Lithium-Ionen Akkus sind in sehr großem Maß in elektronischen, tragbaren Geräte mit hohem Energiebedarf vorzufinden, Mobiltelefone, Tablets, Digitalkameras, Camcorder und Notebooks sind einige davon. [25]

Anwendungsgebiete der Lithium-Polymer-Akkumulatoren lassen sich in der RC-Modellbauszene in aktuelleren Mobiltelefonen, beispielsweise dem Samsung Galaxy S7 oder auch im Bereich der Elektroautos wiederfinden. Die Kurzform RC steht für Radio Controlled oder auch Remote Controlled und kommt aus dem Englischen und lautet zu Deutsch funkferngesteuert. [24]

2.5. Spannungswandler

Laut Datenblatt benötigt der Raspberry Pi eine Versorgungsspannung in Höhe von 5 Volt. Der Tabelle 2- Vergleich: Akkumulatoren ist zu entnehmen, dass Lithium-Akkumulatoren eine Zellspannung von 3,6 - 3,7 Volt aufweisen. In Reihe geschaltete Zellen akkumulieren sich, d.h. ein 2 Zellen Lithium-Akkumulator weist die doppelte Spannung von 7,2 - 7,4 Volt auf. Damit sich eine Spannung von 5 Volt einstellt, werden Gleichspannungswandler eingesetzt. Weitere Begriffe sind DC/DC Auf- bzw. Abwärtswandler, Buck-Converter, Schaltregler, Step-up/down Spannungsregler oder Switching Mode Regulator. Diese können als Fertigmodule oder als Bausatz vorkommen.
Abwärtswandler regeln eine hohe Eingangsspannung in eine niedrige Ausspannung herunter. Umgekehrt verhalten sich Aufwärtswandler. Sie transformieren eine niedrige Eingangsspannung in eine höhere Ausgangsspannung um. Linearregler verbinden beide dieser Eigenschaften.
In der nachfolgenden Tabelle werden typischerweise verwendete integrierte Schaltkreise (ICs) zur Spannungsregulierung von 5 Volt vorgestellt. Eine im Anschluss folgende Betrachtung wird anhand eines Abwärtswandlers dargelegt.

Bezeichnung	verwendeter IC	vorgefertigte Platine	Vorteil	Nachteil
Abwärtswandler[16]	LM2596	Step-down Spannungsregler Modul 2-3A	höhere Eingangsspannung, durch großen Energiespeicher liefert mehr Spannung über längeren Zeitraum	Einsatz von Kühlkörper bei mehr als 2 Ampere
Linearregler[18]	LM7805*	-	nur ein Bauelement	schlechter Wirkungsgrad hohe Verlustleistung Abwärme am Transistor[19]
Aufwärtswandler[17]	TPS61090	PowerBoost 1000 Basic	kompakterer Enegiespeicher als Versorgungs-spannung	Stromstärke maximal 1,2 Ampere

Tabelle 3 - Vergleich: Spannungsregulierung

*Linearregler bestehen aus keinem integriertem Schaltkreis. Sie bestehen lediglich aus einem Bauelement, einem Transistor.

16 https://www.geras-it.de/step-down-spannungsregler-modul-2-3a-1-5-34v-einstellbar-lm2596-fuer-z-b-arduino/a-249/ [06.07.2017]
17 https://www.adafruit.com/product/2030 [24.07.2017]
18 https://www.conrad.de/de/spannungsregler-linear-typ78-stmicroelectronics-l7805cv-to-220ab-positiv-fest-5-v-15-a-179205.html [24.07.2017]
19 Quellenverzeichnis [5][S.360]

Ein Abwärtswandler besteht aus zwei Kondensatoren, einer Diode und zwei Widerständen, wovon ein Widerstand regelbar ist. Dabei handelt es sich um ein Potentiometer, mit welchem die Ausgangsspannung auf einen bestimmten Wert festgelegt wird. [5][S.360]

Der integrierte Schaltkreis des Abwärtswandlers erzeugt über Schaltvorgänge eine pulsierende Spannung, welche durch die verbauten Kondensatoren geglättet werden. [5][360] Dabei handelt es sich um Elektrolytkondensatoren (Abk. Elko). Diese verhindern, dass pulsierende Spannungssignale in die Schaltung einfließen. Die Glättungskondensatoren laden sich auf den Scheitelwert der pulsierenden Gleichspannung auf und überbrücken die Lücken. Dabei gibt der Kondensator einen Teil seiner gespeicherten Ladung an den Verbraucher ab und verliert dabei an Spannung.

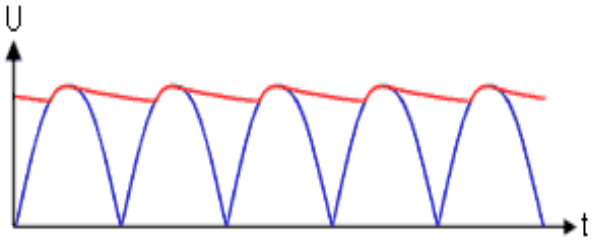

Abbildung 9 - Pulsierender Spannungsverlauf mit Glättungskondensator[9]

Zum besseren Verständnis dient die Abbildung 9, die den Spannungsverlauf über einer Zeitdauer abbildet. Die pulsierende Spannung ohne Einwirkung eines Kondensators ist in der Abbildung blau dargestellt. Der rote Verlauf stellt die Glättung eines Kondensators dar. [13]

[9] http://www.elektronikinfo.de/techpic/strom/netzteil2.gif [10.7.2017]

2.6. Software

Damit der Raspberry Pi betrieben und angesteuert werden kann ist eine Software notwendig. Diese kann in Form eines geschriebenen Programms vorkommen oder eines Betriebssystems, dass durch zusätzliche Programme ergänzt wird. Weiterhin besteht die Möglichkeit ein eigenes, angepasstes Betriebssystem zu erstellen.
Zunächst jedoch wird Begriff "Linux" erklärt und im Anschluss daran wird der bereits in einem gesonderten Kapitel eingeführte Begriff "Embedded" erneut aufgegriffen und im Zusammenhang mit Linux erläutert.
Abgeschlossen wird das Software-Kapitel mit dem Abschnitt "Anwendung". Eine Übersicht der zur Verfügung stehenden Programmiersprachen wird aufgezeigt, damit eine passende Anwendung entwickelt werden kann, um den Roboter fahrtüchtig zu machen.

Linux ist ein kostenfreies Betriebssystem. Es steht für embedded- oder Desktop-Varianten zur Verfügung. Für Desktop Betriebssysteme lautet die korrekte Bezeichnung Linux-Distribution.
Linux besteht aus einem Kernel und einem root-Dateisystem "rootfs". Der Linux-Kernel umfasst Funktionalitäten, wie das Initialisieren von Gerätetreibern, darunter Audio, Netzwerk, USB usw., Verwaltung von Speicher und Prozessen, sowie den Zugriff auf das Dateisystem. Der Kernel enthält allerding keine Programme, die das System benutzbar machen. Erst das rootfs, welches das höchste Verzeichnis einer Datenstruktur ist und auch Ausgangs- bzw. Wurzelverzeichnis genannt wird, ermöglicht beim Startvorgang des Betriebssystems, das Einbinden von Dateisystemen, wo sich letztendlich Programme befinden.
Umfangreicher setzt sich eine Linux-Distribution zusammen. Kernel und rootfs werden durch viele Anwendungen wie Office-Programme, Audio- und Videoanwendungen, Mail-Clients und Internet Browser erweitert. [4][S.19,23]

Da die Begrifflichkeiten "Embedded", aus dem vorherigen Kapitel 2.2 und "Linux" nun näher erläutert sind, kann zusammengefasst werden, dass ein Embedded Linux System ein mikroprozessor/controller gesteuertes System mit Anschlussmöglichkeiten ist, das auf einem Linux-Betriebssystem basiert. Da eingebettete Systeme im Vergleich zu herkömmlichen Standrechnern über weniger Hardwareressourcen verfügen, wie geringere Prozessorgeschwindigkeit oder weniger Arbeitsspeicher, kommt ein spezielles, für das Embedded System angepasstes Linux-basiertes Betriebssystem zum Einsatz.

Die Möglichkeit bestünde, sich gegen ein Betriebssystem zu entscheiden, sofern wenige Hardwareressourcen zur Verfügung stünden und lediglich einfache Funktionalitäten gebraucht würden. Jedoch verfügt der Raspberry Pi über genügend Ressourcen um ein Betriebssystem zu beherbergen.

Ein Betriebssystem ermöglicht den Zugriff auf ein Dateisystem, stellt eine Konnektivität zu Funkschnittstellen bereit und ermöglicht des Weiteren den Zugriff auf externe Peripherie.

Für den Raspberry Pi existieren je nach Einsatzgebiet verschiedenste, angepasste Betriebssysteme. Die für dieses Projekt benötigten Funktionen werden durch die vom Hersteller entwickelte und optimierte Linux-Distribution namens Raspbian, allesamt abgedeckt. Raspbians Entwicklung basiert auf der Open-Source-Initiative, deren Bestreben darin liegt, freie Software zu entwickeln ohne dass Lizenzgebühren anfallen.
Der Umgang mit weiteren Distributionen beruht auf persönlichen Erfahrungen. Dazu kann berichtet werden, dass Distributionen, wie Kali Linux oder Ubuntu Mate, in der Version 16.04.2, den gleichen Umfang an Funktionen bietet wie Raspbian, jedoch entscheidende Nachteile mit sich führen.
Der Bootvorgang mit Unterstützung einer graphischen Oberfläche dauert länger (Raspbian 17 Sekunden, Kali Linux 23 Sekunden, Ubuntu Mate 35 Sekunden). Der verwendete Arbeitsspeicher im Leerlaufzustand ist höher (Raspbian 90 MB, Kali Linux 130 MB, Ubuntu Mate 300 MB). Das Starten von Programmen benötigt mehr Zeit, z.B. das Laden eines Bildes in das Bildbearbeitungsprogramm GIMP (Raspbian 14 Sekunden, Ubuntu Mate 25 Sekunden)

Anstelle auf vorgefertigte Betriebssysteme zurückgreifen zu müssen, ermöglichen die Tools, Yocto und Buildroot, das Erstellen eines eigenen Betriebssystems.
Vorteile hierbei wären, eine individuelle Anpassung an das eingebettete System, das ausschließlich Funktionen zu Schnittstellen bereitstellt, die benötigt werden. Nichtbenutzte Treiberelemente fließen folglich nicht mit in das erzeugte Image ein, sodass das erzeugte Betriebssystem schneller hochgefahren werden kann und weniger Speicherplatz benötigt und dabei im Umfang sehr gering gehalten werden kann.
Da eingebettete Systeme meist über wenige Speicherressourcen verfügen, wird auf einem fremden System eine Cross-Compilation ausgeführt, d.h. Quellcode mit einer neuen Architektur wird generiert, bestehend aus einem Kernel und einem root-Dateisystem. [29]
Yocto verwendet für diesen Zweck eine sogenannte bitbake shell, in der spezielle Initialisierungsskripte, die auf dem fremden Host-Computer gestartet werden. Über das Internet werden aus Archiven tausende Zeilen an Quellcode heruntergeladen, entpackt, gepatcht und kompiliert. Über sogenannte Rezepte, bestehend aus einfachen Text-Dateien, kann das Image schrittweise nach Belieben zusammengestellt und erweitert werden. [30]

Neben den ganzen Vorteilen, die mit einem eigenen, angepassten Betriebssystem einhergehen, birgt es ebenso Nachteile. Der Zeitaufwand der betrieben werden muss, um sich in die Materie einzuarbeiten, ist immens. Ein weiterer negativer Aspekt ist die benötigte Zeit, die während des Erstellens des Images verstreicht. Dabei kann es sich, je nach verwendeten Funktionen, um mehrere Stunden handeln. Werden weitere Rezepte dem Image hinzugefügt, weil neue Treiberelemente oder Funktionen gebraucht werden, beginnt der Kompilierungsprozess von Neuem und es vergehen abermals mehrere Stunden.

2.6.1. Anwendung

Das Betriebssystem stellt Funktionalitäten der Schnittstellen zur Kommunikation bereit. Zusätzlich zum Betriebssystem muss eine Anwendung bzw. ein Algorithmus entwickelt werden, damit die Motoren über die Schnittstellen angesprochen werden können, sodass der Roboter fahrtüchtig wird. Hierfür stehen mehrere, vom Raspberry Pi, zur Verfügung gestellte Programmiersprachen bereit.

Der Name des Raspberry Pis knüpft an die Tradition an, Computer nach Früchten zu benennen. Bekannte Vertreter sind Apple, Blackberry und Acorn. Der Zusatz "Pi", ausgesprochen wie das englische Wort "Pie", übersetzt für Tortenstück, wird oft fälschlicherweise als solches interpretiert, steht jedoch für **P**ython **I**nterpreter, eine Andeutung für die Hauptprogrammiersprache des Raspberry Pis. [60]
Bei Pyhton handelt es sich um eine höhere Programmiersprache, die sich durch ihre Komplexität der Maschinensprache entfernt. Erst der Einsatz von Interpretern oder Compilern übersetzt Befehle des Programmiercodes in Maschinensprache, die der Mikroprozessor bzw. Mikrocontroller versteht. [61]
Das Spektrum an Programmiersprachen wird jedoch nicht auf Python begrenzt. Der Raspberry Pi unterstützt Scratch, C, C++, Java, Perl und weitere Skriptsprachen, wie HTML5, PHP und JavaScript. [55] Da der Raspberry Pi auf einem Linux-Kernel operiert, können über die angebotenen Paketquellen nach Belieben weitere Programmiersprachen hinzugefügt werden.
Mit Hilfe einer sinnvollen Programmiersprache wird ein Algorithmus entwickelt, welcher in der Lage sein muss, sowohl die Motoren, als auch die Encoder anzusteuern, damit der Roboter fahrtüchtig wird. Eine Ansteuerung der Motoren, verbunden mit einer Geschwindigkeitsregulierung, sowie einem Auslesen der Pulse der Quadraturencoder sind dafür notwendig.

3. Lösungsansätze

Bezogen auf die Grundlagen und das damit einhergehende, erlangte Wissen, mitsamt der erstellten Tabellen, werden in diesem Kapitel Entscheidungen bezüglich der Hard- und Softwarekomponenten getroffen, die letztendlich in den Bau des Roboters einfließen. Es wird auf jede einzelne Komponente zurückgegriffen und begründet, warum eine Auswahl als sinnvoll erachtet wird. Den Abschluss dieses Kapitels bilden ein Blockschaltbild und eine Bill of Materials. Das Blockschaltbild zeigt die Wechselwirkung aller Komponenten auf. Die Bill of Materials ist eine Auflistung aller verwendeten Komponenten, einschließlich ihres Preises, um die Gesamtkosten dieses Projekts zu verdeutlichen

3.1. Vorbetrachtung: Stromversorgung

Das empfohlene Netzteil zur Stromversorgung wird vom Hersteller des Raspberry Pis für die 3. Version mit 2,5 Ampere deklariert. [17] Die daraus resultierende maximale zur Verfügung stehende Leistung kann errechnet werden. 5 Volt bei bis zu 2,5 Ampere, ergeben eine Leistung von 12,5 Watt. Der tatsächliche Verbrauch hängt jedoch von der CPU-Auslastung ab, sowie dem Leistungsbedarf der angeschlossenen peripheren Geräte, beispielsweise Maus und Tastatur über die USB-Ports. [17] Der Raspberry Pi kann als eingebettetes System als solches auf eine Ansteuerung über Eingaben einer Tastatur verzichten und über das Netzwerk mittels einer SSH Verbindung angesprochen werden.

In einer Vorbetrachtung bezüglich der Stromversorgung wurde eine Messung mit einem Gleichspannungsmessgerät durchgeführt, welches die Ströme über den Micro-USB Port auslesen und erfassen kann. Ein angeschlossener Monitor und eine Kamera über den CSI Port des Raspberry Pis kamen als einzige Peripherie zum Einsatz. Nachdem der Raspberry Pi gestartet und das Betriebssystem Raspbian hochgefahren wurde, konnte über das Netzwerk eine SSH Verbindung aufgebaut werden. Diese Messung ergab, dass der Raspberry Pi in seinem Ruhezustand 320mA benötigt. Unterzog man den Raspberry Pi einem CPU-Stresstest, ergab sich durch die Messung ein Wert von 550mA. Vollführte man zusätzlich zum CPU-Stresstest eine Live Übertragung der Kamera auf dem angeschlossen Monitor, konnte ein Wert von 850mA dem Messgerät abgelesen werden.
Die Messungen wurden von dem Erstprüfer dieser Bachelorarbeit durchgeführt und im Internet veröffentlicht. [18]
Die Vorbetrachtung wurde unter dem Aspekt durchgeführt, um aufzuzeigen, welche Maximalströme der Raspberry Pi je nach Anwendung benötigt und in wie weit sich diese dem empfohlenen Wert von 2,5 Ampere annähern. Da das Projekt auf eine konventionelle Stromversorgung über eine Anbindung an eine Steckdose verzichtet, da sich das Gefährt autonom fortbewegen soll, kommen Akkumulatoren zum Einsatz. Die gewonnen Messwerte werden dazu verwendet einen geeigneten Akkumulator auszuwählen.

3.2. Stromversorgung

Einige Komponenten des Roboters benötigen unterschiedliche Spannungen. Zum einen müssen die drei Motoren mit 12 Volt und zum Anderen muss der Raspberry Pi mit 5 Volt betrieben werden. Mit Hilfe eines Spannungsteilers kann eine große Spannung aufgeteilt und dazu verwendet werden, mehrere Komponenten, unterschiedlicher Spannungen, zu versorgen. Des Weiteren besteht die Möglichkeit jede Komponente mit ihrer eigenen Spannungsversorgung zu versehen.

Aus dem vorherigen Abschnitt Vorbetrachtung: Stromversorgung, konnte durch Messungen der benötigte Strom für den Raspberry Pi 3 Modell B, aufgezeigt werden. Unter der Prämisse, dass dieses Bachelorarbeit lediglich der erste Schritt eines größeren Projekts ist, welches in der Zukunft weiter aufgegriffen wird und durch Erweiterungen ausgebaut werden soll, wird beschlossen, sich dem empfohlenen Wert des Herstellers von 2,5 Ampere, anzunähern, auch wenn der Raspberry Pi für dieses Projekt eine solche Stromstärke nicht benötigt. Es soll für die Zukunft gewährleistet sein, jedwede Anschlussmöglichkeit in Betracht zu ziehen, sodass dies durch die bereits jetzige verbaute Stromversorgung ermöglicht wird.

Wie der Tabelle 2 - Vergleich: Akkumulatoren zu entnehmen ist, haben sich Lithium-Akkumulatoren, durch ihre kompakte Form, bei einer hohen Energiedichte und dem nichtvorhandenem Memory-Effekt, ausgezeichnet. Die in Grundlagen bereits genannten Nachteile und die damit eventuell auftretenden Schäden, können aufgrund der verbauten Überwachungselektronik nahezu ausgeschlossen werden. Ein sorgsamer Umgang beim Laden mit einem speziellen Ladegerät für Lithium-Akkumulatoren ist dennoch ratsam.
Die Frage die sich nun ergibt, lautet: Der Einsatz welches Lithium-Akkumulators ist sinnvoller, Lithium-Ionen- oder Lithium-Polymer-Akkumulatoren?
Bereits in dem dazugehörigen Grundlagenkapitel wurde erwähnt, dass Lithium-Ionen Akkus mit einem flüssigen und Lithium-Polymer-Akkus mit einem festen oder gelartigen Elektrolyten ausgestattet sind. Der fertige Roboter wird über keine Außenkleidung verfügen, d.h. bei unsachgemäßem Gebrauch des Roboters übertragen sich eventuelle Kollisionen direkt auf verbaute Komponenten. Damit ein Auslaufen des Akkumulators unterbunden wird,
werden Energiespeicher in Form von Lithium-Polymer-Akkumulatoren verwendet.

Ein einzelner mit hoher Spannung ausgestattet Akkumulator könnte dazu verwendet werden, alle Komponenten zu versorgen, sofern eine Schaltung mit einem Spannungsteiler realisiert werden würde. Die Zeit, die in die Entwicklung und Realisierung einer Schaltung investiert werden muss, kann erspart werden, sofern jede Komponente ihre Spannungsversorgung erhält.
Demzufolge dient ein Energiespeicher der alleinigen Stromversorgung des Raspberry Pis. Zudem kommt ein weiterer, gesonderter Lithium-Polymer-Akkumulator zum Einsatz, welcher für die Ansteuerung der drei Gleichspannungsmotoren eingesetzt wird.

Abbildung 10 - Lithium-Polymer-Akkumulatoren

Bei den in diesem Projekt verwendeten Akkumulatoren handelt es sich einerseits um einen 2 Zellen (7,4 Volt) 20C 2400mAh und einen 4 Zellen (14,8 Volt) 20C 3000mAh Akku.
Beide Modelle besitzen einen gelben XT-60-Stecker als Anschlussmöglichkeit. Des Weiteren verfügen sie über einen weißen Balancer-Stecker. Es kann vorkommen, dass einzelne Zellen unterschiedlich stark entladen werden. Der Balancer in Kombination mit einem passenden Ladegerät ermöglicht ein ausgewogenes Aufladen aller Zellen. [11]
Die Kenngröße 20C ist für die maximal zulässige Dauerstromentnahme verantwortlich und errechnet sich aus der angegebenen Kapazität C multipliziert mit den angegebenen Milliampere. Demzufolge liefert der erwähnte 2 Zellen Akku einen Dauerstrom in Höhe von 20C x 2400 mA = 48 Ampere. [11]
Die einheitenbehaftete Zahl [mAh] gibt an, wie viel Strom bzw. Milliampere in einer Stunde geliefert werden können bis die Spannung des Akkus aufgebraucht ist.

Die Entscheidung einen 2 Zellen Akkumulators für den Raspberry Pi einzusetzen, wird im nachfolgenden Kapitel 3.3 Abwärtswandler begründet.

Unter maximaler Last beträgt die Stromaufnahme eines Motors 500mA auf. Eine gleichzeitige Ansteuerung aller drei Motoren mit maximaler Auslastung hat zusammen demnach eine Stromaufnahme in Höhe von 1500mA. Somit verbleiben weitere 1500 mA des Akkus unberührt. Für die Auswahl der Stromstärke des Akkumulators wurden erneut zukünftige Erweiterungen miteinbezogen, sodass spätere hinzukommende mechanische oder technische Komponenten ebenfalls von dieser Energiequelle gespeist werden können.
Einen 4 Zellen Akku mit einer Spannung von 14,8 Volt auszuwählen ist damit begründet, dass die Motorentreiber laut Datenblatt einen Spannungsabfall von 3,2 Volt bei einem Ampere aufweisen, bei zwei Ampere sogar 4,9 Volt. [64] Der Spannungsabfall, welcher sich bei 1,5 Ampere einstellen würde, liegt dementsprechend in etwa bei 4 Volt. Im aufgeladenen Zustand liefert der 4 Zellen Akku eine Spannung von 16,8 Volt (eine Zelle = 4,2 Volt). Abzüglich des Spannungsabfalls würden die Motoren mit 12,8 Volt versorgt werden.

3.3. Abwärtswandler

Wie bereits im vorhergehenden Abschnitt erwähnt, verfügt der verwendete Lithium-Polymer-Akkumulator für den Raspberry Pi über eine Spannung von 7,4 Volt.
Aus dem Datenblatt des Raspberry Pis ist zu entnehmen, dass dieser eine Eingangsspannung von 5 Volt benötigt. Da keine LiPo-Akkus mit einer Spannung von 5 Volt existieren, wird ein 2 Zellen Akku mit einer Spannung von 7,4 Volt ausgewählt, dessen Spannung abgesenkt und angepasst werden muss. Hierfür existieren die bereits im Grundlagen-Kapitel erwähnten Gleichspannungs-Abwärtswandler.
Ein Verzicht auf eine kompakte Platine mit Auf- bzw. Abwärtswandler, hätte sich durch einen Einsatz von einem Linearregler eingestellt. Die bereits in Tabelle 3 - Vergleich: Spannungsregulierung erwähnten Nachteile überwiegen jedoch und verdeutlichen, auf den Einsatz eines Linearreglers zu verzichten. Sich gegen den in Tabelle 3 aufgeführten Aufwärtswanlder zu entscheiden, ist damit begründet, dass dieser lediglich eine Stromstärke von maximal 1,2 gewährleistet, aber nicht für zukünftige Erweiterungen geeignet ist.

Bei der Auswahl des Abwärtswandlers sollte die maximale durchlässige Stromstärke beachtet werden. Bei Recherchen nach einer geeigneten Spannungsregulierung, die in Tabelle 3 - Vergleich: Spannungsregulierung eingeflossen sind, hat sich ergeben, dass viele Module lediglich eine Stromstärke von maximal 2 Ampere zulassen. Dies hätte zur Folge, dass der Akkumulator mit seinen 2400mA in seiner Kapazität begrenzt würde. Für den Fall, dass der Raspberry Pi die komplette Bandbreite von 2400mA nutzen möchte, sollte das Modul die Anforderung haben eben jene Stromstärke zu gewährleisten.

Das zum Einsatzkommende Modul benutzt den Schaltregler LM2596 und kann bis zu maximal 3 Ampere liefern, sofern ein Kühlkörper auf dem IC verwendet wird [12].
Der verwendete Abwärtswandler ist als Fertigmodule vorhanden und muss nicht selber entwickelt, mit Bauelemente bestückt und verlötet werden.
Die Ein- bzw. Ausgänge sind hingegen in Form von Lötflächen vorhanden.

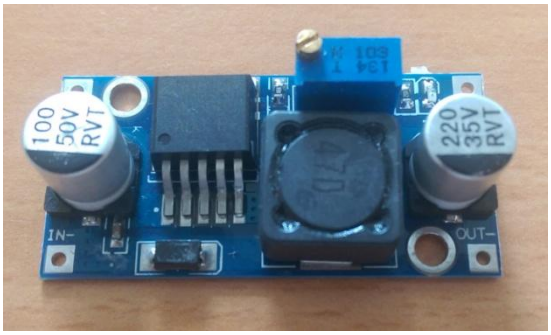

Abbildung 11 - Abwärtswandler

3.4. Motortreiber: L298N

Anhand der in den Grundlagen aufgestellten Tabelle 1 - Auszug der Marktübersicht von Motorentreibern wurde ein passender Motortreiber, mit der Bezeichnung L298N, ausgewählt.
Der MOSFET-Transistor mit der Bezeichnung IRLIZ44N wurde in die Tabelle aufgenommen, um aufzuzeigen, dass sehr hohe Leistungen mit Motortreibern ermöglicht werden. Da dieser über keine H-Brücke verfügt und somit keine Umpolung der Drehrichtung ermöglicht wird, fällt dieser aus der engeren Auswahl. Die restlichen Treibermodule verfügen allesamt über die benötigten Parameter, einer 12 Volt Versorgungsspannung der Motoren bei einer Stromstärke von 1,5 Ampere. Das Treibermodul STL9958 verfügt über eine H-Brücke und ist lediglich über einen SPI-Bus regelbar, ein Konfigurationsschritt, der erspart werden kann, sofern man einen der beiden verbliebenen Treibermodule MC33926 und L298N wählt.
Aus eigener Erfahrung hat sich der Motortreiber mit der Bezeichnung L298N durch seine vielfältigen Anschlussmöglichkeiten und einfach Handhabung in einem vorherigen Projekt bewährt und wird erneut in diesem Projekt verwendet.
Bei dem Motorentreiber L298N handelt es sich um einen dualen Treiberbaustein mit integrierter H-Brücke. Sie erlaubt es bis zu zwei Motoren (Motor 1/2) gleichzeitig und unabhängig voneinander anzusteuern, diese umzupolen und dadurch deren Drehrichtung umzukehren.
Es müssen drei Räder individuell mit Strom versorgt und angesteuert werden. Ein Einsatz von zwei Motorentreibern ist unabdingbar.
Die zulässige Versorgungsspannung liegt bei 46 Volt. Die maximale Stromstärke sollte nicht mehr als 2 Ampere betragen. [5][S.415]

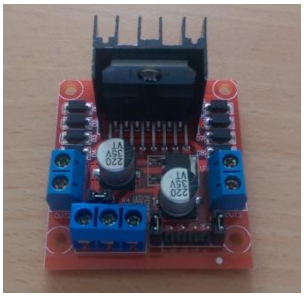

Abbildung 12 - Motortreiber: L298N

Die Motortreiberplatine besitzt insgesamt elf Anschlüsse. Dazu zählen fünf Anschlussmöglichkeiten in Form von blauen Ösen an denen Kabel eingesteckt und verschraubt werden können. Hierzu zählt eine Masseleitung und zwei Anschlüsse dienen der Versorgungsspannung. Ein 5 Volt Anschluss und ein weiterer, der darüber hinausgeht. Diese wird mit +12 deklariert. Des Weiteren befinden sich an jeder Seite Anschlüsse an denen jeweils ein Motor mit seinen zwei Kabeln angeschlossen wird.

Die sechs verbliebenen Anschlussmöglichkeiten sind als Pins auf der Platine vorhanden. Eine Verbindung zum Raspberry Pi und seinen GPIO-Pins erfolgt mit Jumper Wires (Drahtbrücken).
Die Bezeichnung der Pins lautet: Enable A und B, sowie Input 1 - 4.

3.4.1. Ansteuerung

Die Beschaltung der notwendigen Pins für eine Bewegungsteuerung der Motoren wird mit Hilfe einer Wahrheitstabelle für Motor 1 dargestellt. Ein High-Pegel wird durch eine 1 und ein Low-Pegel durch eine 0 repräsentiert. Die entstehende Zahlenkombination aus drei Werten entspricht einer Funktion.
Die Funktionen lassen sich für Motor 2 übertragen. Dabei werden die Input-Pins 3 und 4 verwendet, sowie der Enable B Pin.

Input 1	Input 2	Enable A	Funktion
0	**0**	**0**	**Motor dreht aus**
0	**0**	**1**	**Stopp**
0	**1**	**0**	**Motor dreht aus**
1	**0**	**0**	**Motor dreht aus**
0	**1**	**1**	**Rückwärts**
1	**1**	**1**	**Stopp**
1	**1**	**0**	**Motor dreht aus**
1	**0**	**1**	**Vorwärts**

Tabelle 4 - Wahrheitstabelle für Motor 1[20]

Hier kristallisiert sich heraus, dass es nur zu einer Motorenbewegung kommt, sofern der Enable-Pin ein High-Pegel erhält. Wird nach einer bestehenden Bewegung der Enable-Pin mit einem Low-Pegel beschaltet, deaktiviert sich die H-Brücke und der Motor dreht langsam aus.

Aus den entstandenen Beobachtungen wird eine verbesserte Wahrheitstabelle angefertigt, da es irrelevant ist, womit die Input-Pins beschaltet sind, sobald ein Low-Pegel an dem Enable-Pin anliegt. Die Enable A Spalte befindet sich der Übersicht wegen auf der linken Seite. Das *-Symbol repräsentiert einen High- oder Low-Pegel,

Enable A	Input 1	Input 2	Funktion
0	*****	*****	**Motor dreht aus**
1	**0**	**0**	**Stopp**
1	**0**	**1**	**Rückwärts**
1	**1**	**0**	**Vorwärts**
1	**1**	**1**	**Stopp**

Tabelle 5 - Verbesserte Wahrheitstabelle für Motor 1

[20] Quellenverzeichnis [5][S.417]

3.5. Software

Die Wahl eines Betriebssystems ist erforderlich, da es zum Einen eine Projektvorgabe ist ein eingebettetes Linux-System zu verwenden und des Weiteren wird auf Schnittstellen sowie ein Dateisystem zugegriffen.
Aufgrund des Zeitaufwandes, der betrieben werden müsste, um ein eigenes Betriebssystem zu entwickeln, wird auf das vorgefertigte Raspbian Betriebssystem zurückgegriffen.
Aufgrund der schnellen und stabilen Performance zählt Raspbian zu den beliebtesten und in Projekten am meisten verwendeten Linux-Distributionen. Durch tägliche Updates wird das Betriebssystem auf dem aktuellsten Stand gehalten. Der Hersteller bietet Raspbian in zwei unterschiedlichen Varianten an. Eine im vollen Umfang bereitgestellte Version vereinnahmt 2500 MB auf der microSD-Karte, wohingegen eine reduzierte Lite-Version lediglich 925 MB beansprucht. Eine graphische Oberfläche und zusätzliche Programme unterscheiden den Umfang. Die Lite-Version konzentriert sich dabei auf wesentliche Pakete, damit die gesamten Hardwarefunktionen des Raspberry Pis angesteuert werden können. Die Verwendung ohne graphische Oberfläche zeichnet sich anhand des verwendeten Arbeitsspeichers, sowie weniger Zeit beim Bootvorgang aus. Die Benutzung erfolgt über eine Kommando-Shell.
Die Arbeitsspeicherauslastung im Leerlaufzustand beläuft sich in der vollen Version auf 90 MB und in der Lite-Version auf 30 MB. [42][46] Benötigte Programme, welche nicht in der Lite-Version vorhanden sind, können durch die mittlerweile von über 35000 bereitgestellten Paketen nachinstalliert werden. [46] Der derzeitige lauffähige Linux-Kernel unter Raspbian ist Version 4.9.
Das Betriebssystem eines eingebetteten Systems konzentriert sich auf die nötigsten Funktionalitäten. Eine graphische Oberfläche oder Programme, wie ein Browser für einen Internetzugriff werden für dieses Projekt nicht benötigt, daher bietet sich die Verwendung der Lite-Version Raspbians an.

Eine hardwarenahe Programmierung ist für dieses Projekt essentiell, denn es müssen Motoren angesteuert, sowie die Pulse der Encoder bei der Entstehung einer Radumdrehung ausgelesen werden. Es bieten sich die Programmiersprachen Python und C an. Das Haupteinsatzgebiet von C liegt in der Systemprogrammierung. Bei der Erstellung von Betriebssystemen und Anwendungen findet es ebenfalls Verwendung. Der Vorteil C in eingebetteten Systemen einzusetzen liegt in der Effizienz Hardware direkt anzusprechen ohne dabei hohe Anforderungen an das System zu stellen. Zu Testzwecken wird zunächst die Programmiersprache Python benutzt, um erste Ansteuerungen der Motoren und Encoder zu erreichen, da bereits aus einem Vorprojekt Erfahrungen gesammelt werden konnten. Eine anschließende Umsetzung in der Programmiersprache C wird angestrebt.

Dem Benutzer soll die Möglichkeit in Aussicht gestellt werden die Geschwindigkeit der Motoren zu handhaben. Für eine Geschwindigkeitsregulierung bietet sich eine Pulsweitenmodulation an. In der Digitaltechnik existieren High- und Low-Pegel, repräsentiert durch 1 und 0, um Zustände oder Vorgänge ein- bzw. auszuschalten. Eine PWM bedient sich eines solchen Impulses bei einer Größe von 8-Bit und teilt diesen in 256 Schritte auf, sodass ein größerer Bereich zur Verfügung steht, der genauer abgetastet werden kann. [63] Eine Ansteuerung mittels PWM über die zur Verfügung gestellten Hardware-Pins an der GPIO-Steckleiste wird nicht angestrebt, da lediglich zwei PWM-Kanäle existieren, jedoch drei Motoren unabhängig voneinander angesteuert werden müssen. Eine Umsetzung findet mittels einer Softwarelösung statt. Ein Parameter zwischen 0 und 100 wird an den Motorentreiber und seinen dazugehörigen Motor übermittelt. 0 bis 100 repräsentieren in einer prozentuellen Skala die zur Verfügung stehende Motorenspannung. Anhand des übergebenen Wertes findet eine Reduzierung oder Erhöhung eben jener Spannung statt. Eine niedrige Spannung sorgt für ein langsames Rotieren des Motors. Eine hohe Spannung hingegen lässt den Motor vergleichsweise schneller drehen.

Die Encoder werden für das Auswerten der Streckenlänge eingesetzt. Der Benutzer soll die Länge einer Strecke in cm eingeben können. Das Problem das sich hierbei ergibt, ist die Schräglage der angebrachten Motoren und den verbundenen Encodern. In der Abbildung 1 bzw. Abbildung 2 ist dies deutlich zu erkennen. Ein im Leerlauf betriebener Motor ohne Bodenkontakt wird dementsprechend in geradliniger Bewegung bei weniger Pulsen mehr Strecke zurücklegen, als ein schräg angebrachter Motor.
Zunächst muss Software entwickelt werden, damit entstehende Pulse bei Radumdrehungen erfasst werden können. Anschließend muss eine Strecke mit vordefinierter Länge vom aufgebauten Roboter abgefahren werden und die in der Zeit gemessenen Pulse notiert werden.

3.6. Blockschaltbild: Komponenten

Die Wechselwirkungen zwischen den verwendeten Komponenten mit ihren Anschlüssen werden skizzenhaft dargestellt.

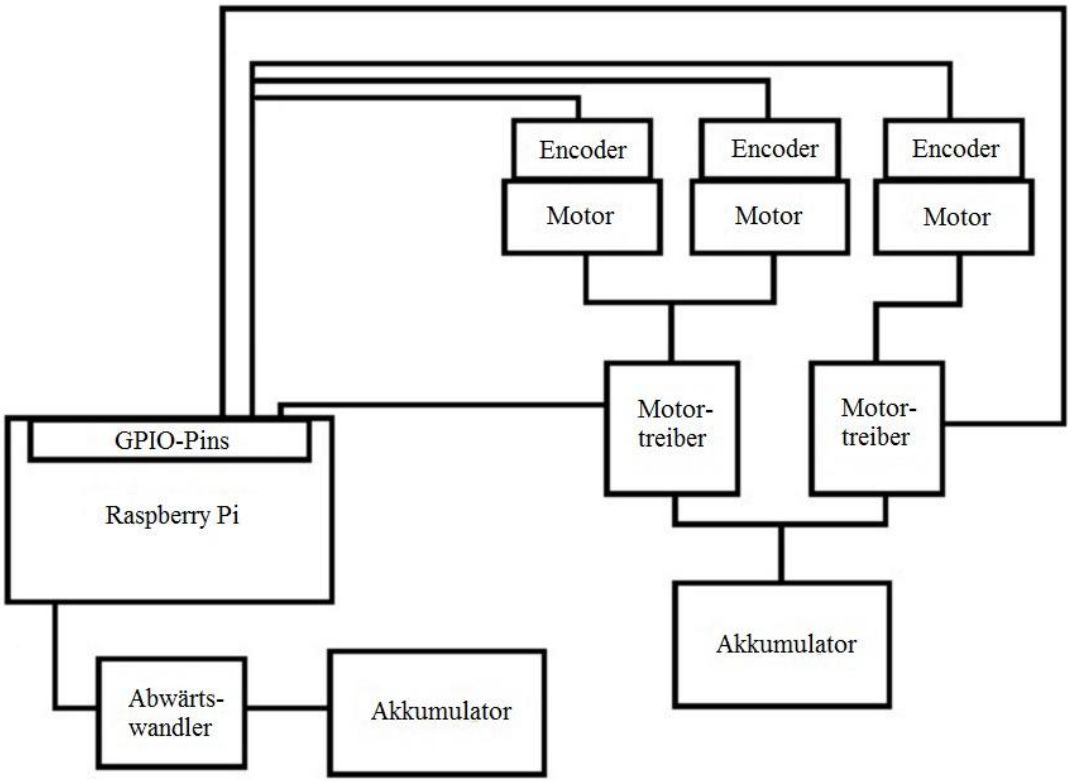

Abbildung 13 - Blockschaltbild der Komponenten

Der Abwärtswandler passt die Spannung des 2 Zellen Akkumulators auf die benötigte 5 Volt Versorgungsspannung des Raspberry Pis an. Der 4 Zellen Akkumulator liefert zwei Motorentreibern genügend Spannung, damit drei Motoren versorgt werden. Über die GPIO-Pins des Raspberry Pis werden die Motorentreiber und die drei Encoder angesteuert. Die Versorgungsspannung der Encoder liefert ein 5 Volt GPIO-Pin.

3.7. Bill of Materials

Um eine preisliche Vorstellung für dieses Projekt aufzuzeigen, werden anhand einer Bill of Materials (Stückliste) alle benötigten Komponenten in einer Tabelle zusammengefasst.

Bauteil	Lieferant	Preis in Euro
Omnirad Plattform	**https://nodna.de**	**141,80 €**
Raspberry Pi 3 Modell B	**https://www.cyberport.de**	**36,00 €**
Raspberry Pi Gehäuse	**https://www.amazon.de**	**7,99 €**
micro SD-Karte 64 GB	**https://www.cyberport.de**	**25,90 €**
2x L298N Motortreiber	**https://www.amazon.de**	**14,99 € (5 Stück insgesamt)**
Abwärtswandler	**https://www.geras-it.de**	**3,90 €**
2 Zellen 2400mAh LiPo Akku	**https://www.conrad.de**	**15,99 €**
4 Zellen 3000mAh LiPo Akku	**https://www.conrad.de**	**36,99 €**
LiPo Aufladegerät	**https://hobbyking.com**	**27,71 €**
XT60 Ladekabel	**https://www.amazon.de**	**6,74 €**
Drahtbrücken	**https://www.amazon.de**	**7,09 €**
4x XT60 Buchse	**https://www.rctech.de**	**2,40 € (gesamt)**
doppelseitiger Klettverschluss	**Spielwarenladen**	**2, 99 €**

Tabelle 6 - Bill of Materials

Die Gesamtkosten betragen **330,49 €**.

4. Realisierung

Dieses Kapitel umfasst den praktischen Teil der Bachelorarbeit. Das erste Unterkapitel "Pflichtenheft" gliedert sich in drei Kategorien "Muss", "Wunsch" und "Abgrenzung" auf. In Stichpunkten wird aufgezählt, welche notwendigen Schritte in der Realisierung vorhanden sein müssen, welche Wunschkriterien eventuell realisiert werden, aber nicht zwingend erforderlich sind und explizite Abgrenzungen, die nicht umgesetzt werden.
Im Anschluss daran folgt der Roboteraufbau. Schrittweise wird jede einzelne Komponente in Unterkapitel gegliedert und mit Fotos der Fortschritt des Zusammenbaus dokumentiert. Nachdem das Gerüst der Omnirad Plattform aufgebaut ist, werden die Komponenten nacheinander auf den Ebenen der Plattform platziert, Kabelverbindungen werden aufgetrennt, es wird gelötet und eine abschließende Verbindung aller Komponenten mit den GPIO-Pins des Raspberry Pi findet statt. Das Unterkapitel endet mit einer Illustration der Pinbelegung am Raspberry Pi.
Ein Software-Entwurf, anhand eines Blockschaltbilds, beschreibt den Ablauf der fertigen Anwendung. Zusätzlich werden einige Funktionen näher beschrieben.

4.1. Pflichtenheft

Musskriterien:

- Aufbau der Hardware
    - Aufbau der Omnirad-Plattform
    - Spannungsversorgung: Raspberry Pi
    - Spannungsversorgung: Motoren und Encoder
    - Kommunikation zwischen Motorentreiber und Raspberry Pi
- Inbetriebnahme Raspberry Pi
    - Aufspielen des Betriebssystems,
    - Einstellung der Grundkonfigurationen
- Anwendung
    - Motorenansteuerung mittels PWM
    - Auslesen der Encoder
    - Benutzereingaben

Wunschkriterien:

- C-Programmierung
- Geometrische Figuren abfahren
- WLAN-Einbindung ins Hochschulnetzwerk
- Analyse von:
    - Motoren:
        - Drehrichtung, Geschwindigkeit,
    - Encoder:
        - Streckenlänge

Abgrenzungskriterien:

- autonomes Fahren
- Aufteilung der Anwendung auf mehrere Klassen

4.2. Roboteraufbau

4.2.1. Omnirad Plattform

Die Omnirad Plattform besteht aus einem Bausatz von Metallplatten und Schrauben, die nach einem Zusammenbau zwei Ebenen besitzt, auf denen Komponenten Platz finden. Drei Motoren und ihre dazugehörigen Encoder sind bereits vormontiert. An jedem Motor wird ein Allseitenrad verschraubt. Metallstreben werden auf der unteren Ebene mit Schrauben befestigt, sodass die zweite Ebene darauf angebracht werden kann.

Abbildung 14 - Omnirad Plattform: Unterseite

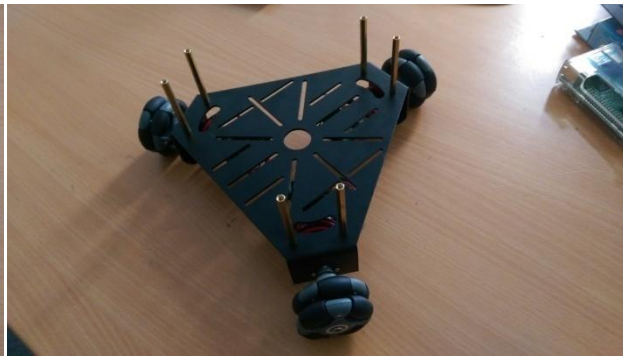

Abbildung 15 - Omnirad Plattform: untere Ebene

Die Öffnungen auf der Metallplatte ermöglichen es Komponenten zu verschrauben. Da einige der verwendeten Komponenten über freiliegende Kontakte auf der Platine verfügen und möglicherweise in Kontakt mit der Metallplatte geraten könnten, werden die Elemente nicht verschraubt, sondern mit doppelseitigem, selbstklebendem Klettverschluss befestigt. Eine vorbeugende Maßnahme der Isolierung, um eventuelle übergreifende Ströme auf der Metallplatte zu unterbinden.

Der Vorteil der mit doppelseitigem Klettverschluss einhergeht, ist die einfachere Entfernung, als bei verschraubten Komponenten. Beispielhaft aufgezeigt an den verwendeten Akkumulatoren, die nach verbrauchter Ladung mühelos entfernt und am Stromnetz aufgeladen werden können.

Anhand der Abbildung 16 sind die freiliegenden Lötstellen deutlich zu erkennen. Die nebenstehende Abbildung 17 zeigt den mit Klettverschluss versehenen 4 Zellen Akkumulator.

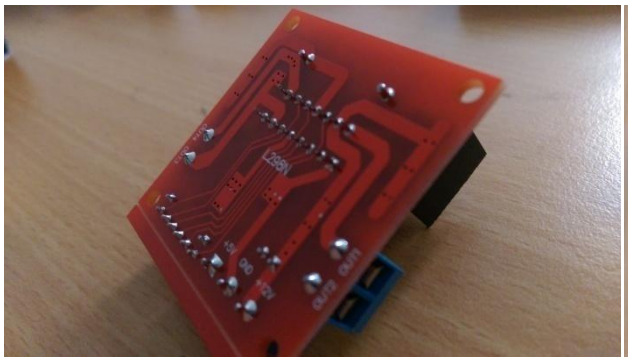

Abbildung 16 - Treibermodul L298N: Unterseite

Abbildung 17 - 4 Zellen Akku mit Klettverschluss

Die Omnirad Plattform wird ebenfalls mit Klettverschluss ausgestattet und das erste Element, der 4 Zellen Akku, wird platziert.

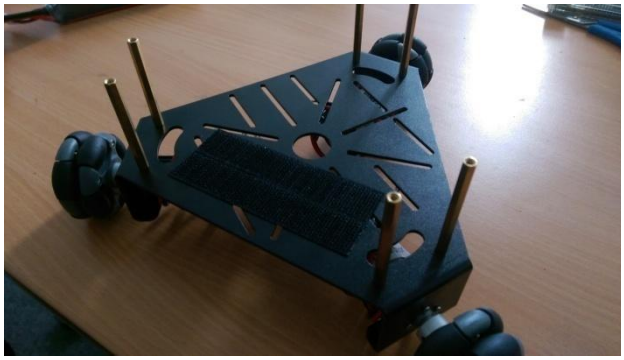

Abbildung 18 - Omnirad Plattform: Klettverschluss

Abbildung 19 - Befestigung des 4 Zellen Akkus

4.2.2. Spannungswandler: Raspberry Pi

Die nächsten Elemente, die auf der unteren Ebene der Omnirad Plattform platziert werden, umfassen die Spannungsversorgung des Raspberry Pis.
Der Abwärtswandler verfügt über Ein- bzw. Ausgangsöffnungen in Form von freiliegenden Kontaktstellen. Der Eingang des Abwärtswandlers soll durch den 2 Zellen Akku gespeist werden. Der Akku besitzt einen männlichen XT-60 Stecker.

Damit ein einfaches Verbinden der beiden Komponenten ermöglicht wird, wird an den negativen und positiven Kontaktstellen am Eingang des Abwärtswandler das Gegenstück, ein weiblicher XT-60 Stecker, verlötet. Mit Schrumpfschläuchen werden die Lötstellen elektrisch isoliert.

Der 2 Zellen Akku wird mit dem Abwärtswandler verbunden. Ein Digitalmultimeter misst die Ausgangsspannung. Der regelbare Widerstand, das Potentiometer, auf dem Abwärtswandler wird auf die benötigten, konstanten 5 Volt des Raspberry Pis eingestellt.
Damit der Raspberry Pi die Versorgungsspannung erhält, wird ein Micro-USB-Kabel Typ B an dem Ausgang des Abwärtswandlers verlötet. Hierfür wird ein handelsübliches USB-A zu Micro-USB-Kabel Typ B, welches auch bei Smartphones zum Einsatz kommt, zerteilt.

Von den zum Vorschein kommenden vier Signalleitungen, in den Farben schwarz, rot, weiß, grün, werden ausschließlich die schwarze und rote Leitung benötigt.
Bei der weißen und grünen Signalleitung handelt es sich um Datenleitungen. [58]
Für eine reine Versorgungspannung werden sie nicht benötigt. Um dennoch einen Kontakt der beiden Datenleitungen zu unterbinden, werden sie in unterschiedlicher Länge getrennt und voneinander isoliert.
Die schwarze Masseleitung und die rote Versorgungsspannungsleitung, aus dem aufgetrennten USB-Kabel, werden an der negativen sowie positiven Kontaktfläche des Ausgangs am Abwärtswandler verlötet.

Eine erneute Überprüfung der Ausgangsspannung stellt sicher, dass während des Lötvorgangs keine Änderungen am Potentiometer entstanden sind.

Abbildung 20 - weiblicher XT-60 Stecker

Abbildung 21 - Micro-USB-Kabel Typ B

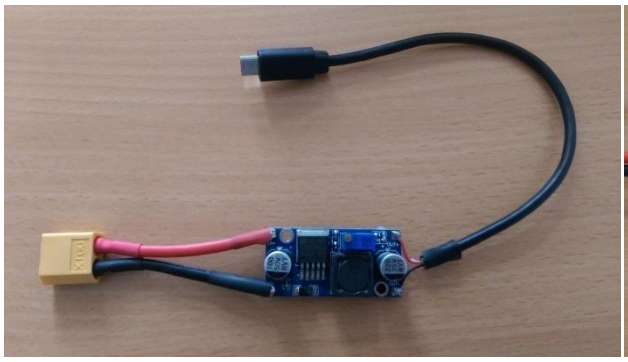

Abbildung 22 - Abwärtswandler mit XT-60 Stecker und Micro-USB-Kabel Typ B

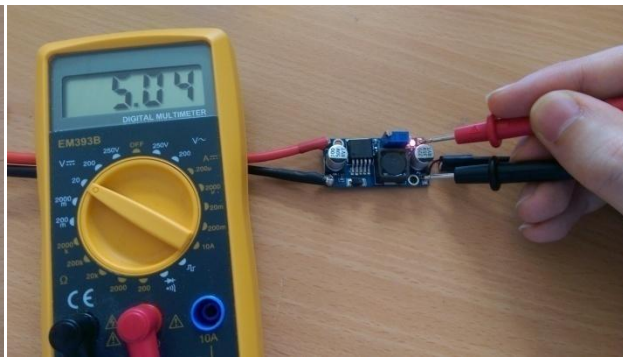

Abbildung 23 - Messung der Ausgangsspannung

Abbildung 24 - 2 Zellen Akku, Abwärtswandler, Raspberry Pi

Der Abwärtswandler und der Akku werden mit selbstklebendem Klettverschluss versehen und auf der unteren Ebene der Omnirad Plattform platziert, wobei der 2 Zellen Akku, aufgrund der begrenzten Fläche, auf dem 4 Zellen Akku angebracht wird.

4.2.3. Motortreiber

Als nächstes werden die beiden Treibermodule auf der unteren Ebene angeordnet. Damit diese eine gleichzeitige Versorgungsspannung durch den 4 Zellen Akkumulator erhalten, wird eine Parallelverkabelung, welche auch unter dem Begriff Y-Kabel bekannt ist, realisiert. Die erneute Verwendung eines weiblichen XT-60 Steckers dient der Verbindung des 4 Zellen Akkus. Die Masse- und Versorgungsleitung des weiblichen XT-60 Steckers werden mit Schrumpfschläuchen überzogen und mit jeweils zwei gleichfarbigen Kabeln verlötet. Die Schrumpfschläuche werden über die erkalteten Lötstellen gezogen und erhitzt, sodass diese sich zusammenziehen.

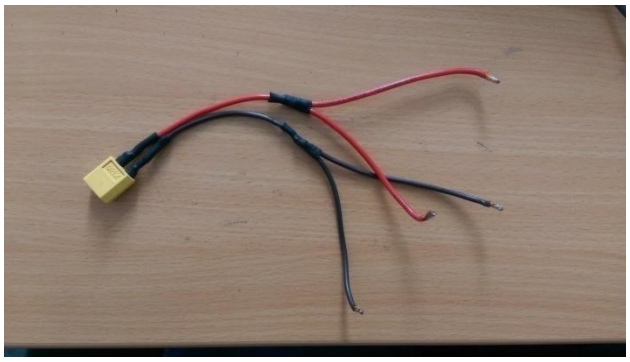

Abbildung 25 - Parallelverkabelung XT-60 Stecker (ohne Jumper Wire)

Des Weiteren wird an den Enden der beiden schwarzen Masseleitungen zusätzlich jeweils ein schwarzes Jumper Wire (Drahtbrücke) verlötet. Die Jumper Wire werden auf den Masseleitung-Pins des Raspberry Pis aufgesteckt. Es ist essentiell diese anzuschließen, damit der Motorentreiber zwischen High- und Low-Pegeln unterscheiden kann. Findet keine Verkabelung der Masseleitung mit dem Raspberry Pi statt, lassen sich die Motorentreiber nicht bedienen!

Die freiliegenden Y-Kabelenden werden mit heißem Lötzinn bestrichen, sodass eine optimale Verbindung der einzelnen Aderpaare, aus dem Kabelinneren, entsteht. Anschließend werden jeweils ein rotes und ein schwarzes Kabelende in die vorgesehenen Motorentreiberösen eingesteckt und verschraubt.

Das rote Kabelende wird in die +12 Volt-Öse und das schwarze Kabelende in die GND-Öse eingesteckt.
Die beiden Treibermodule werden, wie die anderen Komponenten, auf der Unterseite mit selbstklebendem Klettverschluss versehen und auf der Omnirad Plattform positioniert.

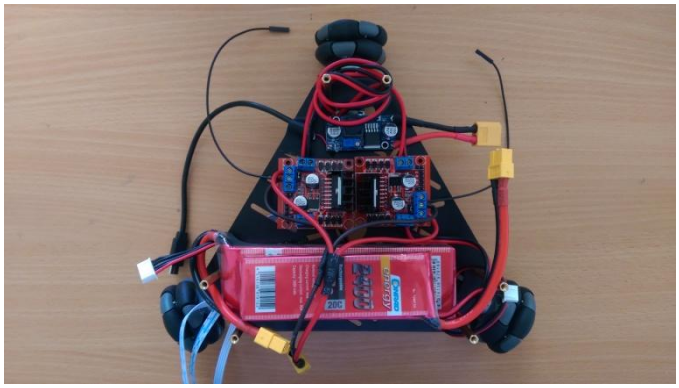

Abbildung 26 - Omnirad Plattform: untere Ebene

Die drei Motoren, mit ihren jeweils zwei Kabeln, schwarz und rot, werden in die Motorentreiberösen OUT 1 - OUT 4 eingesteckt und verschraubt.

Motor 1 ist mit dem ersten Motortreiber an OUT 1, OUT 2 und Motor 2 an OUT 3, OUT 4 verbunden. Motor 3 ist mit dem zweiten Treibermodul an OUT 1, OUT 2 verbunden. Primär wird das schwarze und sekundär das rote Kabel eingesteckt.
(schwarz = OUT 1, OUT 3, rot = OUT 2, OUT 4)

Eine Ansteuerung der Motorentreiber erfolgt durch die GPIO-Pins am Raspberry Pi. Für eine Kommunikation der beiden Komponenten werden Jumper Wire benutzt. Jeder Motor erhält eine farbliche Zuordnung anhand der verwendeten Kabel. Motor 1, in der Abbildung 27 im Südosten lokalisiert und verwendet violette Kabel. Motor 2 befindet sich südwestlich und wird durch gelbe Kabel angesteuert. Der letzte verbliebende Motor 3 ist im Norden vorzufinden und wird durch orangene Kabel angesteuert. Abschließend werden die farblichen Kabel mit Kabelbindern befestigt, sodass eine gewisse Ordnung entsteht.

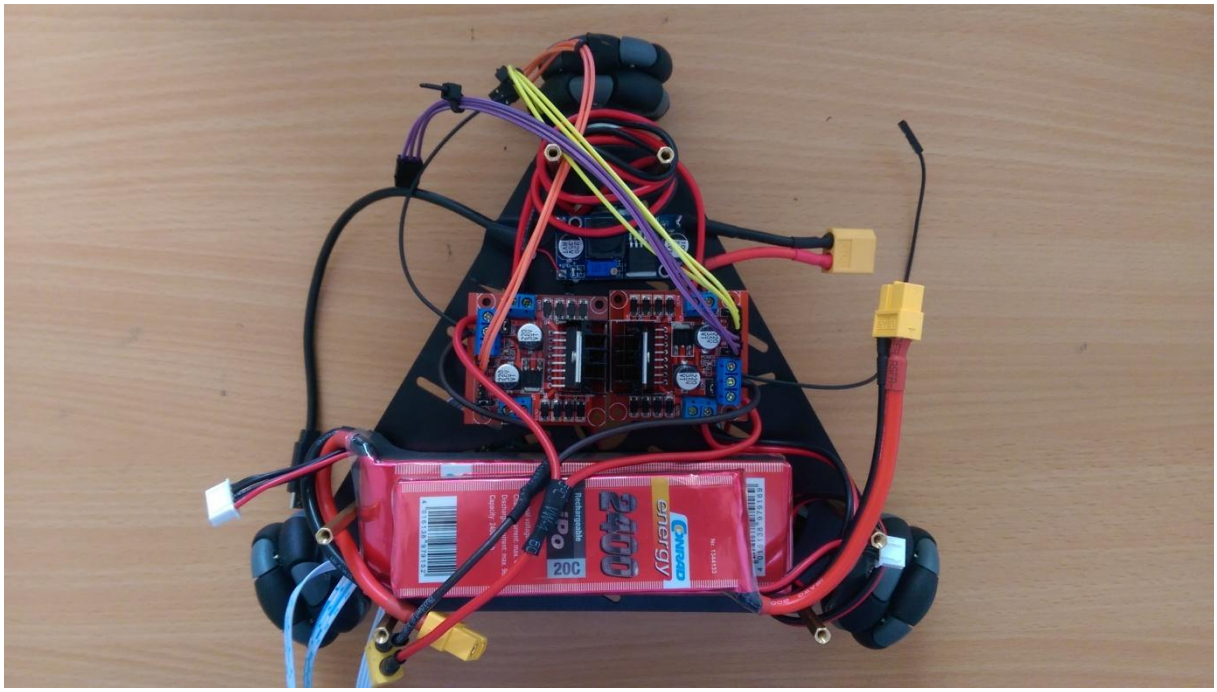

Abbildung 27 - Omnirad Plattform: untere Ebene, Pinverkabelung

4.2.4. Quadraturencoder

Die Quadraturencoder sind mit einem weißen Flachbandkabel, bestehend aus vier Leitungen, dessen Enden aus Jumper Wires besteht, ausgestattet. Die vier Leitungen teilen sich einerseits in Signalleitungen A und B, sowie einer Versorgungs- und Masseleitung auf.

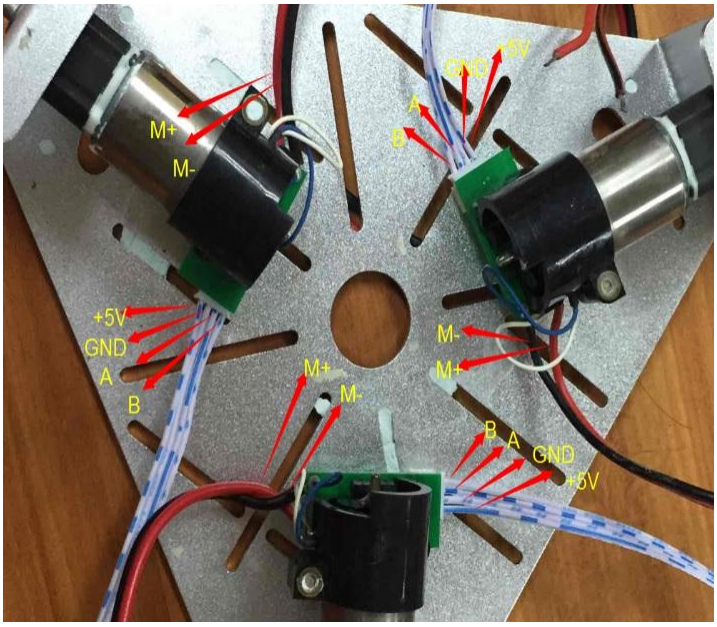

Abbildung 28 - Pinbelegung der Quadraturencoder[10]

Eine gleichzeitige Versorgung der drei Encoder erfolgt durch einen 5 Volt Pin am Raspberry Pi. Das 5 Volt Spannungspotential liegt an allen drei Ausgängen an, lediglich die Ströme teilen sich auf. In zweifacher Ausführung wird eine Parallelverkabelung, wie in der Abbildung 30 angefertigt. Eine schwarze Ausführung dient der gleichzeitigen Abführung des Stroms an einen gemeinsamen Masse-Pin auf dem Raspberry Pi. Eine blaue Ausführung stellt die 5 Volt Versorgungsspannung der drei Quadraturenencoder bereit.

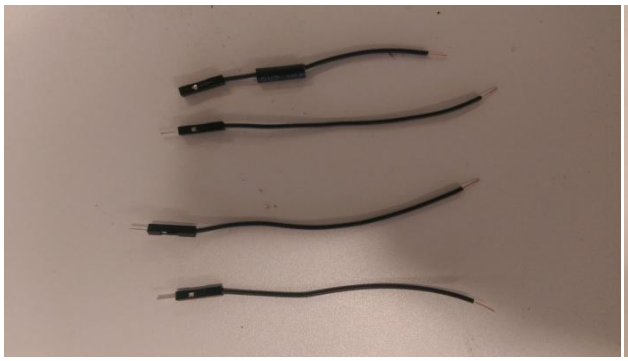

Abbildung 29 - Jumper Wire (einzeln)

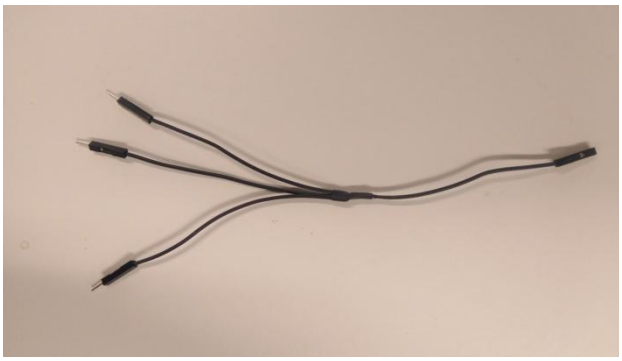

Abbildung 30 - Jumper Wire (verlötet)

[10] https://pm.f1.htw-berlin.de/redmine/attachments/download/6143/lALOL3NCA80C2c0DXg_862_729.png [16.08.2017]

4.2.5. Raspberry Pi

Die selbstverlöteten Parallelverkabelungen für die Encoder werden auf den GPIO-Pins aufgesteckt. Weiterhin werden sechs Jumper Wires für die Encoder-Datenleitungen, in den Farben weiß, grau und braun, mit den GPIO-Pins verbunden.

Abbildung 31 - Raspberry Pi: Encoder Pinverkabelung

Die obere Ebene der Omnirad Plattform wird angebracht und verschraubt. Die Motorentreiber- und die Encoderflachbandkabel werden aus der unteren Ebene, an den dafür vorgesehenen Öffnungen, zu der oberen Ebene durchgeführt. Der Raspberry Pi wird mit Klettverschlüssen versehen und auf der oberen Ebene platziert. Die Jumper Wires der Motorentreiber, sowie der Encoder, werden mit der GPIO-Steckleiste des Raspberry Pis verbunden. Der Aufbau ist abgeschlossen.

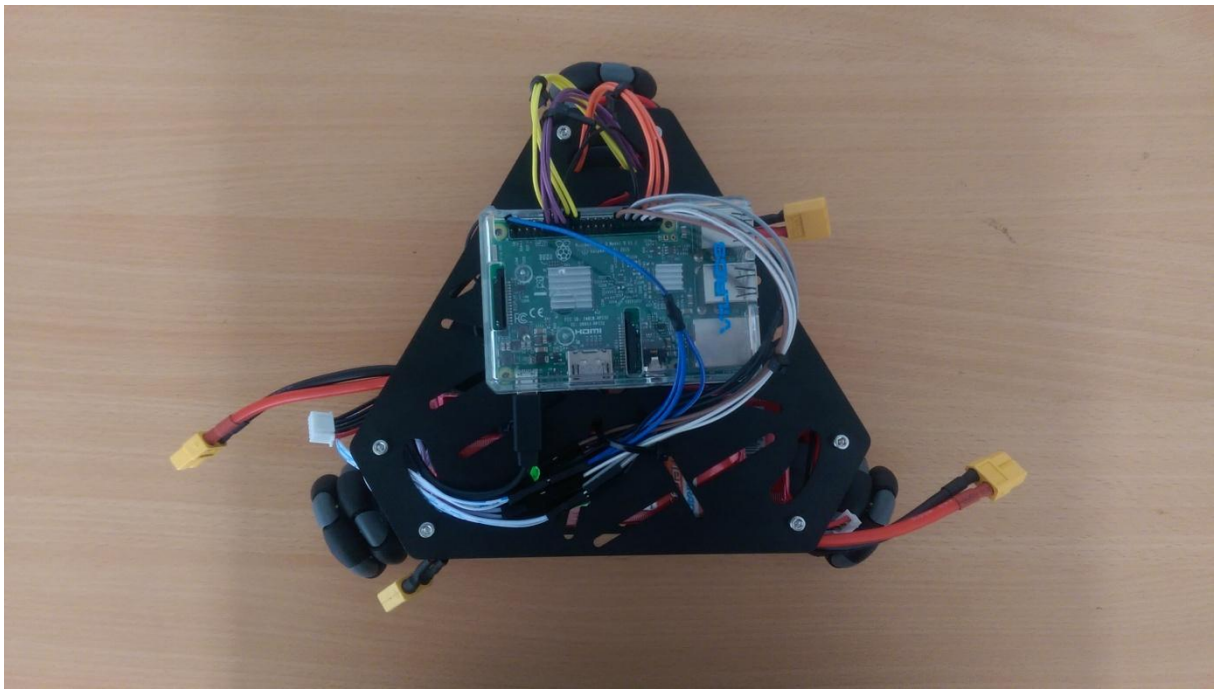

Abbildung 32 - Omnirad Plattform: obere Ebene

4.3. Raspberry Pi: GPIO-Pinbelegung

Mittels der Software "Fritzing"[21] wird ein Diagramm erstellt, welches der Veranschaulichung der GPIO-Pins dient. In einer darauffolgenden Tabelle werden die Pin-Nummern sowie deren Funktionen dargelegt. Die in dem Diagramm verwendeten Kabelfarben stimmen mit denen des Realaufbaus überein.

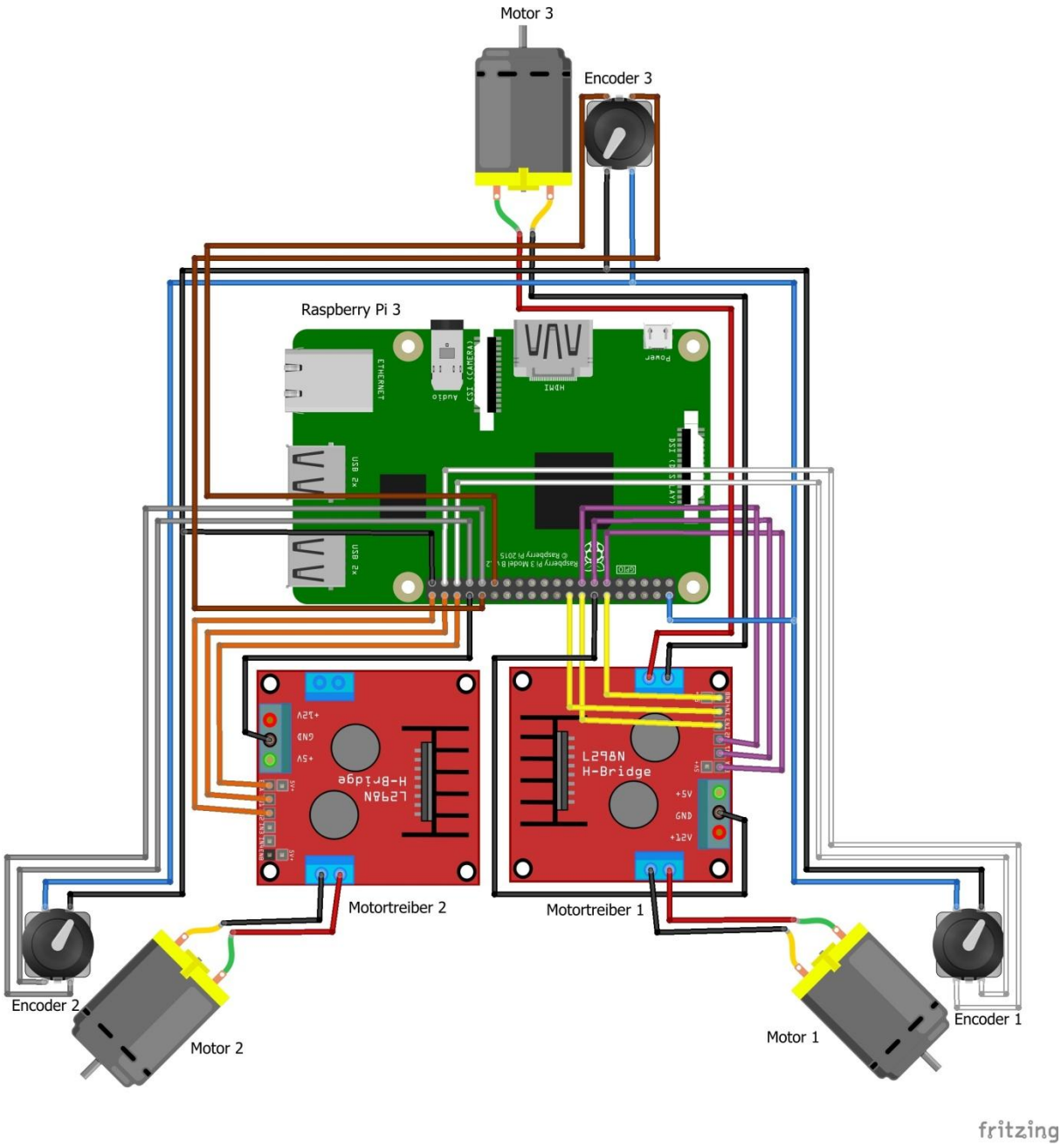

Abbildung 33 - GPIO-Pinbelegung als Fritzing-Diagramm

Im Hardwareaufbau sind die Motoren und die Encoder als ein Bauteil vorhanden. Die Software Fritzing enthält allerdings kein derartiges zusammenhängendes Bauteil. Aus diesem Grund sind in diesem Diagramm die Motoren und Encoder voneinander getrennt vorzufinden.

[21] fritzing.org/download

GPIO- Pinnummer	Anschluss an	Funktion	Kabelfarbe
2	Encoder 1, 2, 3	Spannungsversorgung	blau
11	Motortreiber 1 (Motor 1)	Enable A	violett
12	Motortreiber 1(Motor 2)	Enable B	gelb
13	Motortreiber 1(Motor 1)	Input 1	violett
14	Motortreiber 1	Masse	schwarz
15	Motortreiber 1(Motor 1)	Input 2	violett
16	Motortreiber 1 (Motor 2)	Input 3	gelb
18	Motortreiber 1 (Motor 2)	Input 4	gelb
29	Encoder 3	Kanal A	braun
31	Encoder 2	Kanal A	grau
32	Encoder 3	Kanal B	braun
33	Encoder 2	Kanal B	grau
34	Motortreiber 2	Masse	schwarz
35	Encoder 1	Kanal A	weiß
36	Motortreiber 2 (Motor 3)	Enable A	orange
37	Encoder 1	Kanal B	weiß
38	Motortreiber 2 (Motor 3)	Input 1	orange
39	Encoder 1, 2, 3	Masse	schwarz
40	Motortreiber 2 (Motor 3)	Input 2	orange

Tabelle 7 - GPIO-Pinbelegung

4.4. Software-Entwurf : Programmablauf

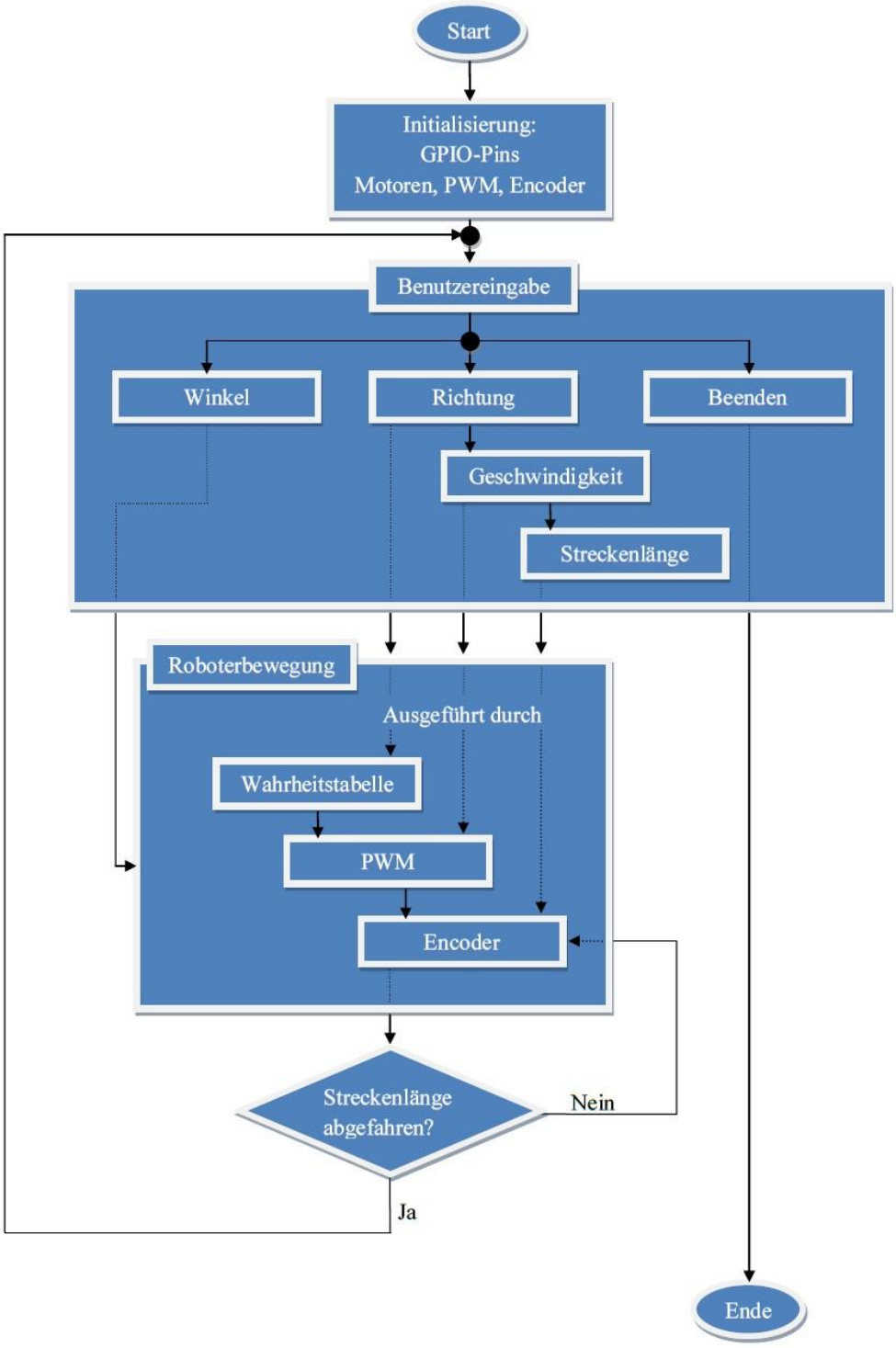

Abbildung 34 - Programmablauf

Zunächst wird der Programmablauf der fertigen Anwendung beschrieben. Im Anschluss folgt eine Betrachtung einzelner erwähnenswerter Funktionen. Der komplette Quellcode wird nicht aufgeführt.

Mit dem Starten der Anwendung erfolgt im ersten Schritt eine Initialisierung der GPIO-Pins. Die mit den Motorentreibern verbundenen Pins werden als Output-Pins initialisiert, sodass Zustandsänderungen vorgenommen werden können. Die mit den Encodern verbundenen Pins werden hingegen als Input-Pins initialisiert. Die Zustände der Encoder müssen nicht verändert, sondern lediglich ausgelesen werden. Eine zusätzliche Initialisierung der drei Enable-Pins für die Motoren findet mittels einer PWM statt.
Nach dem Initialisierungsvorgang startet die main()-Funktion. Dem Benutzer wird auf der Konsole eine grafische Oberfläche dargestellt, die drei Auswahlmöglichkeiten anbietet, welche durch Eingabe einer Zahl ausgewählt werden.
Bei Beendigung der Anwendung wird die quit()-Funktion aufgerufen, die Motoren werden mit einer Funktion gestoppt, die initialisierten GPIO-Pins werden freigegeben und die Anwendung terminiert. Die verbleibenden Auswahlmöglichkeiten betreffen die Fortbewegungsart des Roboters. Bei Auswahl einer Richtung erfolgen im Anschluss zwei weitere Auswahlmöglichkeiten, Geschwindigkeit und Streckenlänge. Bei Auswahl eines Winkels sind Geschwindigkeit und Streckenlänge vordefiniert und können nicht bestimmt werden.
Die Roboterbewegungen werden durch mehrere Funktionen ausgeführt.
Eine Ansteuerung der Motoren und deren Richtung erfolgt durch geschriebene Funktionen, die sich aus der Tabelle 5 - Verbesserte Wahrheitstabelle für Motor 1 ergeben. In sogenannten "Sets" findet eine Zuweisung der Benutzereingabe und der damit verbundenen Richtungsfunktion statt. Die Funktionen erhalten über die Parameterliste einen Geschwindigkeitswert.
Die Geschwindigkeit wird aus drei vordefinierten Werten ausgewählt. Zur Auswahl stehen langsam, mittel und schnell. In einem Set wird anhand der Benutzereingabe der dafür vorgesehene Geschwindigkeitswert zugeordnet. Der Wert wird an die Richtungsfunktion über die Parameterliste übergeben und über die PWM wird die Geschwindigkeit angepasst.
Als Streckenlänge stehen dem Benutzer fünf vordefinierte Werte zur Auswahl. Die kürzeste Streckenauswahl liegt bei 10 cm, die längste Strecke bei 50 cm.
Anhand eines weiteren Sets wird die eingegebene Streckenlänge einem spezifischen Pulswert zugeordnet. Polling wird dazu verwendet die Encoder zyklisch abzufragen, um deren generierte Pulse zu ermitteln. Eine Roboterbewegung findet so lange statt bis die während der Fahrt generierten Encoderpulse mit dem aus dem Set übereinstimmen. Sobald die Strecke zurückgelegt wurde, stoppen die Motoren.
Nach jeder vollendeten Roboterbewegung erfolgt ein Aufruf der grafischen Oberfläche zur erneuten Fortbewegung des Roboters.

Zu Beginn der Programmierung werden einzelne Klassen mit spezifischen Funktionen geschrieben. Dazu zählen, das Ansteuern jedes einzelnen Motors in beide Drehrichtungen für eine bestimmte Zeit, das unabhängige Stoppen eines jeden Motors, eine Geschwindigkeitsregulierung mittels der PWM, das Auslesen beider Encoderkanäle eines jeden Motors und eine fehlerüberprüfende Benutzereingabe. Nacheinander werden die einzelnen, getesteten Funktionen in eine umfangreiche Hauptklasse – `robot.py` – übernommen und als einzelne Funktionen implementiert.

Anhand der Tabelle 5 kann ein Motor in vier unterschiedlichen Varianten angesteuert werden. Er kann sich nach vorne und hinten drehen, stoppen und langsam ausdrehen. Für ein erfolgreiches Ansteuern eines Motors werden allerdings nur die ersten drei genannten Varianten als Funktionen implementiert. Der Benutzer soll eine festgelegt Strecke auswählen, die der Roboter abfährt. Bei einer Fortbewegung würde ein langsames Ausdrehen der Motoren dazu führen, dass zusätzliche Strecke zurückgelegt würde.
Es ergeben sich insgesamt neun Funktionen für eine Ansteuerung aller Motoren.
Bei den Funktionsnamen für die Drehrichtung wird darauf verzichtet, die Begriffe vorne und hinten zu verwenden. Ein Rad dreht sich entweder mit dem Uhrzeigersinn (rechts) oder gegen den Uhrzeigersinn (links). Eine Bewegung nach vorne oder nach hinten kommt zustande, indem sich zwei der drei Motoren in gegensätzliche Richtungen bewegen. Damit sich der Roboter in einem bestimmten Winkel dreht, müssen alle drei Motoren simultan und in die gleiche Richtung gesteuert werden.

Mit Hilfe der aufgestellten Tabelle 5 wird durch das Setzen von High- und Low-Pegeln an den Output-Pins eine Drehrichtung eingestellt und die Funktion entsprechend benannt. Zusätzlich verfügen die Funktionen, abgesehen von der stop-Funktion, über einen Parameter (speed) für die Geschwindigkeitsregulierung mittels PWM. Der in der Tabelle 5 aufgeführte Enable-Pin fungiert als PWM-Pin. Sobald die Funktion aufgerufen wird, wird über die Parameterliste ein Integer-Wert mit übergeben, der über die Funktion ChangeDutyCycle() die Geschwindigkeit reguliert. Anschließend werden der Enable- bzw. der PWM-Pin und die Input-Pins des Motors mit High- oder Low-Pegeln versehen. Es stellt sich eine Drehrichtung mit vordefinierter Geschwindigkeit ein.
Die drei Auszüge aus dem Quellcode beziehen sich auf Motor 1 und verdeutlichen das eben Erläuterte.

```
def M1_left(speed):
    PWM1.ChangeDutyCycle(speed)
    GPIO.output(M1_pwm, 1)
    GPIO.output(M1in1, 1)
    GPIO.output(M1in2, 0)
```

```
def M1_right(speed):
    PWM1.ChangeDutyCycle(speed)
    GPIO.output(M1_pwm, 1)
    GPIO.output(M1in1, 0)
    GPIO.output(M1in2, 1)
```

```
def M1_stop():
    GPIO.output(M1_pwm, 1)
    GPIO.output(M1in1, 1)
    GPIO.output(M1in2, 1)
```

Eine Roboterbewegung nach vorne erfolgt, indem die Funktion für Motor 1 aufgerufen wird, die den Motor nach links drehen und die Funktion, die den Motor 2 nach rechts drehen lässt. Umgekehrt drehen müssen sich die Motoren, sofern eine Rückwärtsbewegung angestrebt wird.

```
def forward_12(speed):
    M1_left(speed)
    M2_right(speed)
```

Eine weitere Funktion in einer separaten Klasse wird angefertigt, um einen Quadraturencoder anzusteuern und seine gemessenen Pulse auszugeben. In der Hauptklasse ist die Funktion "measure_encoder()" benannt. Über die Parameterliste wird ein Impulswert übergeben, der einer Streckenlänge entspricht. Mittels eines Vorgangs namens Polling findet eine zyklische Abfrage der generierten Zustände am Encoder statt.
Innerhalb einer Dauerschleife wird der Zustand des gelieferten Impulses durch den Encoder abgefragt und mit dem zuvor geliefertem Wert verglichen. Unterscheiden sich die Werte, wird eine Counter-Variable hochgezählt, der Zählstand auf der Konsole ausgegeben und der aktuelle Zustand zurückgesetzt.
Bei einer Roboterbewegung erfolgt die Ermittlung der Impulse lediglich durch einen Encoder.

Sobald ein Benutzer aufgefordert wird Eingaben zu tätigen können Fehler auftreten. Sofern eine Benutzereingabe ausschließlich Zahlen vorsieht, sollte eine Überprüfung der Eingabe stattfinden. Die Funktion fordert den Benutzer auf, einen Integer-Wert, in einem festgelegten Bereich, einzugeben. Misslingt die Eingabe, weil Buchstaben oder Sonderzeichen eingegeben werden, erfolgt eine erneute Eingabe solange bis eine Zahl eingegeben wird.
Eine Realisierung im Quellcode findet mittels Dauerschleife und try-Block statt. Die Einhaltung des Zahlenbereichs wird mit einer if-else-Bedingung gelöst.
Fehlerhafte Eingaben werden durch das Werfen einer Exception abgefangen und dem Benutzer auf der Konsole mitgeteilt.
Über die Parameterliste der Funktion `get_int(prompt, min, max)` wird ein String übergeben, welcher dem Benutzer den gültigen Bereich übermittelt, sowie Mindest-und Maximalgrenze als Integer-Werte.

Die in dem Programmablauf erwähnten Sets werden anhand des Geschwindigkeits-Sets verdeutlicht.
Bei der Benutzereingabe erfolgt ein Zahlenwert im Bereich von 1 - 3. Im SPEED-Set sind diese auf der linken wiederzufinden. Die Werte auf der rechten Seite repräsentieren den Geschwindigkeitswert, welche später an die Motorenfunktion übergeben und durch die Funktion `ChangeDutyCycle()` ausgeführt werden. Eine Zuweisung der Benutzereingabe (speed_in) in den entsprechenden Geschwindigkeitswert erfolgt über die unten aufgeführte if-Anweisung.

```
SPEED = {
    1: 10,
    2: 50,
    3: 100
}
if speed_in in SPEED:
    speed_value = SPEED[speed_in]
```

5. Implementierung

In dem ersten Kapitel wird der Raspberry Pi in Betrieb genommen. Ein Betriebssystem wird installiert und grundlegende Einstellungen werden vorgenommen. Eine darauffolgende Anleitung zeigt, wie der Raspberry Pi in das Hochschulnetzwerk integriert wird, sodass eine Benutzung des Systems über eine drahtlose Verbindung erfolgen kann.
Das folgende Kapitel "Anwendung" beschreibt zunächst, welche Bedingungen erfüllt sein müssen, damit die Anwendung lauffähig wird. Eine anschließend Anleitung beschreibt die Vorgehensweise, um die Anwendung zur Steuerung des Roboters zu starten.
Abgeschlossen wird das Kapitel mit einer genauen Beschreibung anhand von Quellcodeauszügen, an denen zukünftige Änderungen vorgenommen werden könnten.

5.1. Raspberry Pi: Inbetriebnahme

Eine Inbetriebnahme des Raspberry Pis setzt ein Betriebssystem voraus, welches vorerst auf der microSD-Karte installiert werden muss. Unter Windows wird eine 64 GB microSD-Karte im FAT-Format formatiert.
Raspbian Lite (2017-07-05-raspbian-jessie-lite.zip) wird von der Herstellerseite[22] heruntergeladen und anschließend aus dem Archiv entpackt. Ein zusätzliches Programm namens "Win32DiskImager"[23] wird verwendet, um das Abbild des Betriebssystems auf die microSD-Karte zu kopieren.
Die Speicherkarte wird in den dafür vorgesehenen Einschub am Raspberry Pi eingeführt. Eine Tastatur wird mit einem der vorhandenen USB-Ports verbunden. Ebenso werden HDMI-Kabel für einen Monitor und eine Stromversorgung angeschlossen.
Für zukünftige Benutzungen wird eine SSH-Verbindung angestrebt. Hierbei wird von einem Computer eine Verbindung zum Raspberry Pi aufgebaut, sodass der Raspberry Pi selbst auf Tastatur und Monitor verzichten kann.

Der erste Startvorgang des Raspberry Pis beginnt.

Mit Eingabe des Benutzernamens "pi", verbunden mit dem Passwort "raspberry", erfolgt der Login. Dazu sei angemerkt, dass das Tastaturlayout in der Werkseinstellung nicht einer deutschen Tastatur entspricht, sodass die Buchstaben "y" und "z" vertauscht sind.
Abhilfe hierfür sorgt eine Änderung der Grundeinstellungen. Diese und weitere Änderungen werden vorgenommen.
Der Befehl `sudo raspi-config` öffnet eine Übersicht der Grundeinstellungen. Der vorangehende Befehl `sudo` ist notwendig, damit der Standardbenutzer "pi" erweiterte Rechte erhält, da Veränderungen am System durchgeführt werden.

[22] https://www.raspberrypi.org/downloads/raspbian/ [16.07.2017]
23 https://sourceforge.net/projects/win32diskimager/ [16.07.2017]

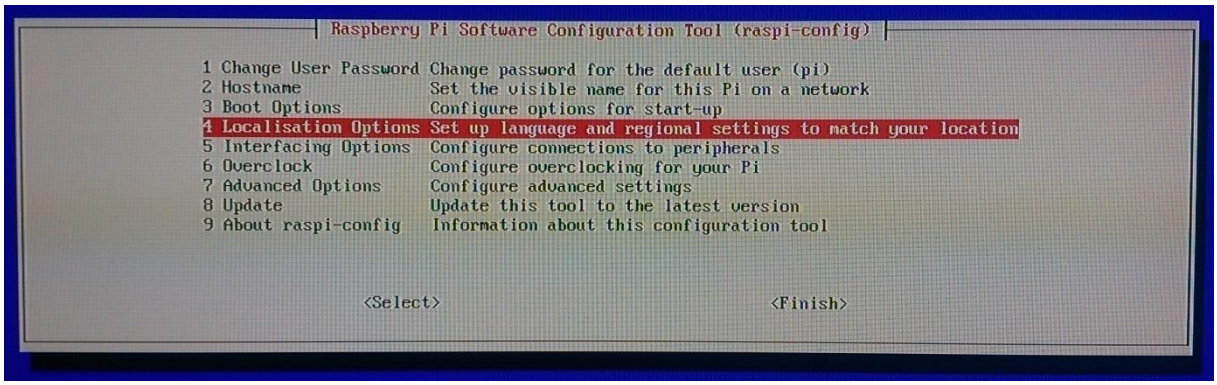

Abbildung 35 - Raspberry Pi Konfiguration: raspi-config

1. Change User Password Das Standardpasswort "raspberry" wird in "BaSS2017" geändert

4. Localisation Options Änderungen bezüglich des deutschen Tastaturlayouts, Zeitzone, Betriebssystemsprache, sowie Landesauswahl für die Benutzung des WLANs

5. Interfacing Options SSH-Aktivierung

7. Advanced Options Bereitstellung des gesamten Speicherplatzes der microSD-Karte für das Betriebssystem, Abschaltung des Overscans, für volle Benutzung des Monitors, ohne schwarze Ränder

Mit Fertigstellung der Konfiguration ist ein Neustart des Systems erforderlich.

5.1.1. WLAN-Verbindung: Aufbau zum Hochschulnetz

Der Raspberry Pi 3 Modell B verfügt über ein integriertes WLAN-Modul. Es wird eine drahtlose Verbindung in das Universitätsnetzwerk "eduroam" aufgebaut. Unter dem Link[24] findet man eine Anleitung des HTW-Rechenzentrums, um eine Verbindung in das Universitätsnetzwerk, mit Hilfe einer Linux-Distribution namens Ubuntu, aufzubauen.
Auf Basis dieser Grundlage wird eine für den Raspberry Pi angepasst Anleitung unter Raspbian erstellt. Zunächst muss auf einem Computer mit bereits bestehender Internetverbindung unter dem Link ein Zertifikat **deutsche-telekom-root-ca-2.pem** heruntergeladen werden. Mit Hilfe eines USB-Sticks wird das Zertifikat auf den Raspberry Pi übertragen und unter dem Pfad `/etc/wpa_supplicant` abgelegt. Der Kommandozeilenbefehl dafür lautet:

```
sudo cp /media/pi/USB-Stick/deutsche-telekom-root-ca-2.pem
/etc/wpa_supplicant
```


[24] https://anleitungen.rz.htw-berlin.de/de/wlan/linux/manuell/index [10.08.2017]

Anschließend wird die Datei `wpa_supplicant` bearbeitet
`sudo nano /etc/wpa_supplicant/wpa_supplicant.conf`
und mit folgendem Inhalt beschrieben:

```
country=DE
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
# eap-ttls
network={
        ssid="eduroam"
        mode=0
        proto=WPA
        key_mgmt=WPA-EAP
        auth_alg=OPEN
        eap=TTLS
        identity="s05*****@htw-berlin.de"
        anonymous_identity="anonymous@htw-berlin.de"
        password="HTW-Passwort"
        ca_cert="/etc/wpa_supplicant/deutsche-telekom-root-ca-2.pem"
        phase2="auth=PAP"
        priority=5
}
# eap-peap
network={
        ssid="eduroam"
        mode=0
        proto=WPA2
        key_mgmt=WPA-EAP
        auth_alg=OPEN
        eap=PEAP
        identity="s05*****@htw-berlin.de"
        password="HTW-Passwort"
        ca_cert="/etc/wpa_supplicant/deutsche-telekom-root-ca-2.pem"
        phase1="peapver=0"
        phase2="auth=MSCHAPV2"
        priority=10
}
```

Angemerkt sei, dass Benutzername und Passwort in der Datei einsehbar sind.

Nach einem Neustart des Raspberry Pis verbindet sich dieser nun automatisch mit dem Hochschulnetzwerk.

5.2. Anwendung

Die Anwendung zur Steuerung des Roboters wird mit der Programmiersprache Python, in der Version 2.7.9, umgesetzt. Die Entwicklungsumgebung ist ein einfacher Editor namens nano.
Für eine Ausführung werden zwei Bibliotheken benötigt. Zum einen ermöglicht **RPi.GPIO** den Zugriff auf die GPIO-Pins und zum anderen wird auf eine **sleep**-Funktion aus der Bibliothek **time** zurückgegriffen, sodass die Anwendung angehalten wird und sich die Motoren für eine festgelegte Zeitdauer drehen. Python, der Editor nano und die Bibliotheken sind auf dem Raspbian Betriebssystem bereits vorhanden. Sollten sie aus unerfindlichen Gründen nicht vorhanden sein, können sie mit dem folgenden Befehl nachinstalliert werden:
`sudo apt-get install python python-dev python-rpi.gpio nano`
Programme, welche in Python geschrieben sind, haben die Dateiendung .py. Sie werden in der Kommandozeile ausgeführt, indem der Programmname mit dem vorangestellten Begriff "python" aufgerufen wird. Beispiel: `python motortest.py`
Der Quellcode wird zunächst auf mehrere Klassen aufgeteilt um einzelne Funktionen zu testen. Anschließend werden die Funktionen der einzelnen Klassen in einer Hauptklasse zusammengeführt.
Der komplette Quellcode aller Klassen ist auf einer beigefügten CD im Ordner "Anwendung" vorzufinden.

Eine Anleitung für die Benutzung der Anwendung wird im Folgenden beschrieben.

Am Raspberry Pi werden Tastatur und Monitor angeschlossen. Der 4 Zellen Akkumulator wird mit dem Y-Kabel der Motorentreiber verbunden. Der 2 Zellen Akkumulator wird mit dem Abwärtswandler verbunden und das Micro-USB-Kabel in den Raspberry Pi eingesteckt.
Nach dem Startvorgang des Betriebssystems wird der Benutzername **pi** und das Passwort **BaSS2017** eingegeben. Der Befehl `ifconfig` wird in der Kommandozeile eingegeben, um die IP-Adresse herauszufinden, welche sich im Abschnitt `wlan0` befindet. Auf einem weiteren Computer, der sich ebenfalls im Hochschulnetzwerk befindet, wird mit der zuvor ausgelesenen IP-Adresse eine SSH-Verbindung zum Raspberry Pi aufgebaut. Unter Windows wird erfahrungsgemäß das Programm "PuTTY"[25] verwendet. Linux-Benutzer können dies über die Kommandozeile erledigen. Tastatur und Monitor können nun vom Raspberry Pi getrennt werden.
Der Umstand Tastatur und Monitor anschließen zu müssen rührt daher, dass der Raspberry Pi von dem Hochschulnetz eine dynamische IP-Adresse zugewiesen bekommt und diese sich bei jedem Neustart des Systems ändern kann. Die Zuweisung einer festen, statischen IP-Adresse würde das anfängliche Einstecken von Tastatur und Monitor obsolet machen.

In der Kommandozeile wechselt man aus dem Home-Verzeichnis in den Ordner "project".
`cd project`
Das Starten der Anwendung erfolgt mit der Eingabe:
`python robot.py`

[25] www.putty.org

5.2.1. Anwendungsänderungen

Damit zukünftige Änderungen am Quellcode einfacher durchgeführt werden können, werden markante Stellen anhand der folgenden Anwendung "motor.py" erläutert. Die Anwendung wurde zu Beginn der Softwareumsetzung geschrieben, um grundlegende Ansteuerungen der Motoren und der Pulsweitenmodulation zu testen. Die im Folgenden behandelten Quellcodeauszüge sind in der fertigen Anwendung in ähnlicher Form vorhanden. Die Anwendung ist im Unterordner "test" des Ordners "project" vorzufinden. Weitere Testanwendungen sind in diesem Unterordner vorzufinden. Dazu zählen die Benutzereingabe "input.py", das Auslesen der Encoderpulse "encoder.py" und die erweiterte Version "encoder_pwm.py", welche Benutzereingabe, PWM und das Auslesen eines Encoders beinhaltet.

Im ersten Schritt wird eine Datei mit dem Befehl `nano motortest.py` erzeugt und geöffnet. Zunächst werden die bereits erwähnten Bibliotheken importiert.

```
import RPi.GPIO as GPIO
from time import sleep
```

Im Grundlagenkapitel 2.2.2 GPIO-Pins wird darauf hingewiesen, dass die Ansteuerung der GPIO-Pins einer Namenskonvention unterliegt. Die Position der GPIO-Pins wird anhand ihrer Nummer ausgewählt (GPIO.BOARD). Ferner besteht die Möglichkeit die Pinbelegung des BCM2837-Chips auszuwählen (GPIO.BCM).

```
GPIO.setmode(GPIO.BOARD)
```

Der Tabelle 7 - GPIO-Pinbelegung werden die Anschlussbelegungen für Motor 1 entnommen. Durch das Ändern der Pins können an dieser Stelle auch Motor 2 und 3 getestet werden. Globale Variablen werden als Platzhalter für die verwendeten GPIO-Pins angelegt, sodass diese bei Bedarf nur einer Stelle im Quellcode geändert werden müssen.

```
M1_pwm = 11
M1in1 = 13
M1in2 = 15
```

Die GPIO-Pins werden als Output initialisiert, sodass der angeschlossene Motor angesteuert werden kann. Die Encoder hingegen liefern Werte zurück und müssen als Input initialisiert werden. (GPIO.IN)

```
GPIO.setup(M1_pwm, GPIO.OUT)
GPIO.setup(M1in1, GPIO.OUT)
GPIO.setup(M1in2, GPIO.OUT)
```

Eine Initialisierung der Pulsweitenmodulation zur Regulierung der Drehgeschwindigkeit erfolgt. Der Maximalwert der PWM wird auf 100 festgelegt. Anschließend wird der PWM-Pin an eine Variable übergeben und mit dem Anfangswert 0 gestartet.

```
PWM1 = GPIO.PWM(M1_pwm, 100)

PWM1.start(0)
```

Anhand der aufgestellten Wahrheitstabelle, Tabelle 5 - Verbesserte Wahrheitstabelle für Motor 1, werden mehrere Funktionen geschrieben, die den Motor vorwärts und rückwärts bzw. nach links und rechts drehen und stoppen lässt. Über die Parameterliste wird ein Integer-Wert (speed) im Bereich von 0 - 100 übergeben, der die Geschwindigkeit mit Hilfe der Pulsweitenmodulation (ChangeDutyCycle) steuert.

```
def links(speed):
    PWM1.ChangeDutyCycle(speed)
    GPIO.output(M1_pwm, 1)
    GPIO.output(M1in1, 1)
    GPIO.output(M1in2, 0)
```

Die Geschwindigkeit wird festgelegt. Die Funktion den Motor nach links zu drehen wird aufgerufen und mit Hilfe der sleep-Funktion wird die Anwendung für 10 Sekunden angehalten, damit der Motor sich in dieser Zeit drehen kann.

```
speed = 20

links(speed)
sleep(10)
```

Jeder Benutzung eines GPIO-Pins zieht nach fertigem Ablauf der Anwendung eine Beendigung nach sich. Die verwendeten GPIO-Pins müssen freigegeben werden. Wird dies vergessen, treten bei erneuter Benutzung der Pins Warnungen und Fehlern auf.

```
GPIO.cleanup()
```

Der Hauptklasse "robot.py" sind die folgenden Quellcodeauszüge entnommen und können verändert werden.
Damit Geschwindigkeiten angepasst werden können, müssen im SPEED-Set auf der rechten Seite die gewünschten Zahlenwerte im Bereich von 10 - 100 eingegeben werden. Bei Eingabe von Werten, die kleiner sind als 10 kann nicht gewährleistet werden, dass eine Motorenbewegung erfolgt, da die anliegende Motorenspannung zu gering ist.

```
SPEED = {
    1: 10,
    2: 50,
    3: 100
}
```

Der Roboter kann sich theoretisch in drei Richtungen nach vorne und hinten bewegen. An jeder geraden Seite der Plattform werden zwei Motoren dafür genutzt. Davon ist jedoch nur eine Vorwärts- und eine Rückwärtsbewegung implementiert. Entschließt man sich dazu weitere Bewegungen nach vorne und hinten zu implementieren, muss man eine weitere Funktion erstellen, die sich an der unten aufgeführten Funktion orientiert.

```
def forward_12(speed):
    M1_left(speed)
    M2_right(speed)
```

Die neue Funktion könnte man `forward_23(speed)` benennen, indem die bereits implementierten Funktionen `M2_left(speed)` und `M3_right(speed)` aufgerufen werden.
Die Abbildung 2 kann für ein näheres Verständnis hinzugezogen werden.

6. Test

In diesem Kapitel finden Analyse und Auswertung der geschriebenen Anwendungen statt.
Wie bereits zuvor erwähnt, setzt sich die Hauptklasse "robot.py" aus mehreren verschiedenen Funktionen zusammen, die während der Softwareumsetzung aus einzelnen Klassen bestanden. Während der ganzen Testphase befinden sich die Motoren des Roboters im Leerlauf, d.h. der Roboter und dessen Räder haben keinen Bodenkontakt.

Mit Fertigstellung des Roboters und beginnender Programmierung ist der erste vorgenommene Test eine Überprüfung der Drehrichtungen jedes einzelnen Motors. Eine Klasse wird erstellt und erhält den Namen "motor.py". Die bereits vorgestellten Funktionen aus Kapitel 4.4 `M1_left()`, `M1_right()` und `M1_stop()` werden implementiert und anschließend ausgeführt.
Die GPIO-Pins der übrigen Motoren werden der Tabelle 7 entnommen und im Quellcode ersetzt, sodass nacheinander alle Motoren einzeln getestet werden. Der Test soll verdeutlichen, ob Verkabelungsfehler, die beim Anschließen der Motoren an den Motorentreiber oder beim Verbinden des Motorentreibers mit Jumper Wires mit dem Raspberry Pi entstanden sein könnten. Für eine selbstgewählte Zeitdauer von einigen Sekunden, drehen sich die Motoren.
Der Test zeigt, dass alle Motoren in die korrekte Richtung drehen.

Anschließend wird die Pulsweitenmodulation der Klasse ergänzt. Der gleiche Vorgang, jeden einzelnen Motor für eine bestimmte Zeitdauer anzusteuern, findet erneut statt. Es werden Geschwindigkeiten im Bereich von 10 - 100 getestet.
Der Test zeigt, dass alle Motoren mit einer Pulsweitenmodulation gesteuert werden können.

Der nächste ausgeführte Test befasst sich mit dem Auslesen der Encoderpulse. Eine neue Klasse wird geschrieben und erhält den Namen "encoder.py". In einem zyklischen Intervall werden die generierten Encoderzustände ausgelesen und dem Benutzer auf der Konsole mitgeteilt. Der Test soll zeigen, ob die angegebenen Radumdrehungen von der Versandseite tatsächlich 160 Pulsen entsprechen.
Bevor die Anwendung gestartet wird, erhält der getestete Motor 1 eine Markierung am Rad. Die Anwendung wird gestartet. Behutsam wird solange an dem Rad gedreht bis eine Umdrehung vollzogen wird. Zeitgleich werden die generierten Pulse auf der Konsole überwacht. Nachdem die Markierung seinem ursprünglichen Ort erreicht, können 160 Pulse auf Konsole abgelesen werden. Damit ist Angabe der Versandseite korrekt.

Im nächsten Schritt wird eine weitere Klasse geschrieben, die sich mit sinnvollen Benutzereingaben befasst. Der Name der Klasse lautet "input.py". Damit der Benutzer lediglich Zahlenwerte in vorgegebenen Grenzen eingeben kann, findet während der Eingabe solange eine Überprüfung statt, bis eine Zahl eingegeben wird.
Getestet wird die Klasse, indem Sonderzeichen, Buchstaben oder auch Zahlen, die außerhalb der vordefinierten Grenzen liegen, eingegeben werden. Der Test läuft erfolgreich. Sobald der Benutzer eine falsche Eingabe tätigt wird er auf der Konsole daraufhin gewiesen.

Die letzte Klasse die geschrieben wird "encoder_pwm.py", umfasst alle Funktionen der bisher genannten Klassen. Es soll getestet werden, ob Geschwindigkeiten durch Benutzereingaben korrekt durch die Pulsweitenmodulation ausgeführt werden und ob eine bestimmte, im Vorfeld vorgegebene Streckenlänge erreicht wird. Die Streckenlänge wird auf 160 Pulse festgelegt, sodass eine Radumdrehung angestrebt wird.
Bevor die Anwendung ausgeführt wird, erhält Motor 1 erneut eine Markierung, sodass eine Radumdrehung ausgemacht werden kann.
Die Anwendung startet und die Aufforderung zur Benutzereingabe erfolgt auf der Konsole. Die Zahl 1 wird eingegeben, sodass eine langsame Drehung des Motors stattfindet.
Die Konsole wird überwacht bis der vorgegebene Wert von 160 Pulsen erreicht ist. Es sollte eine Radumdrehung stattfinden, sodass die Markierung am Rad wieder an seinem ursprünglichen Ort landet.
Die Markierung am Rad landet weiter hinter seiner Ausgangsposition. D.h. der Encoder liefert mehr Pulse als 160, sobald es zu einer Motorenbewegung und einer Geschwindigkeitsregulierung mittels einer PWM kommt.
Der Test ist fehlgeschlagen.

Das Problem wird analysiert. Unter Zuhilfenahme eines Oszilloskops werden die Pulse des Encoders bei unterschiedlichen Geschwindigkeiten ausgelesen und dargestellt. In der Anwendung werden 30 Pulse vorgegeben. Sobald dieser Wert erreicht ist stoppt der Motor. Auf dem Oszilloskop werden die tatsächlich generierten Pulse dargestellt. 30 Pulse werden gewählt, da diese noch problemlos auf dem Oszilloskop abgelesen werden können.
Er werden mehrere Messungen vorgenommen. In der ersten Messung wird die Geschwindigkeit auf langsam eingestellt (PWM = 10).

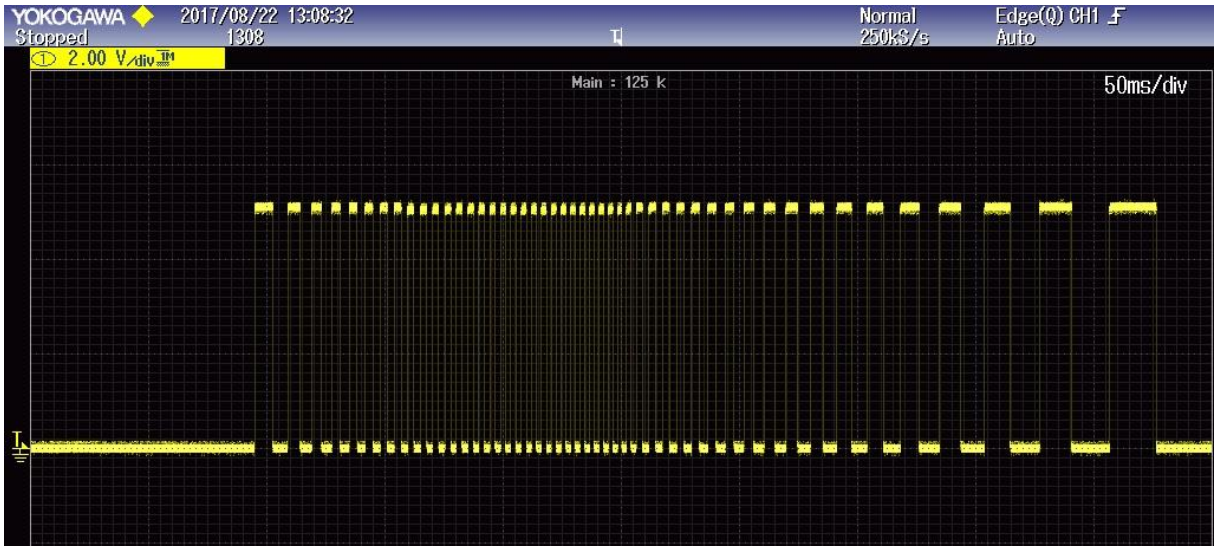

Abbildung 36 - Oszilloskop: 30 erwartete Pulse, PWM = 10 (langsam)

Anstelle der erwarteten 30 Pulse, liefert das Oszilloskop 48 Encoderpulse.

Im zweiten Test wird die Pulsweitenmodulation auf 50 angehoben, bei gleichbleibenden, vorgegebenen 30 Pulsen.

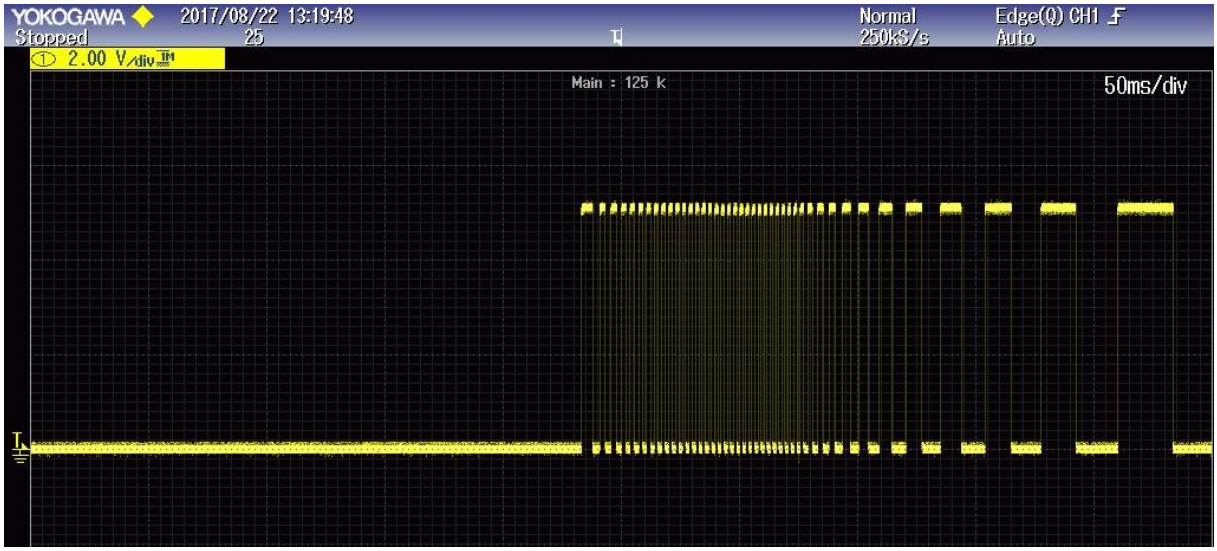

Abbildung 37 - Oszilloskop: 30 erwartete Pulse, PWM = 50 (mittel)

Nach mehrmaliger Ausführung des Tests, werden anstelle der zu erwartenden 30 Encoderpulse, 42 Pulse durch das Oszilloskop festgehalten.

Da die tatsächlich gemessenen Encoderpulse mit steigender Geschwindigkeit sinken, wird ein weiterer Test ausgeführt. 30 Encoderpulse werden erneut vorgegeben und die Geschwindigkeit wird durch die PWM auf 100 erhöht.

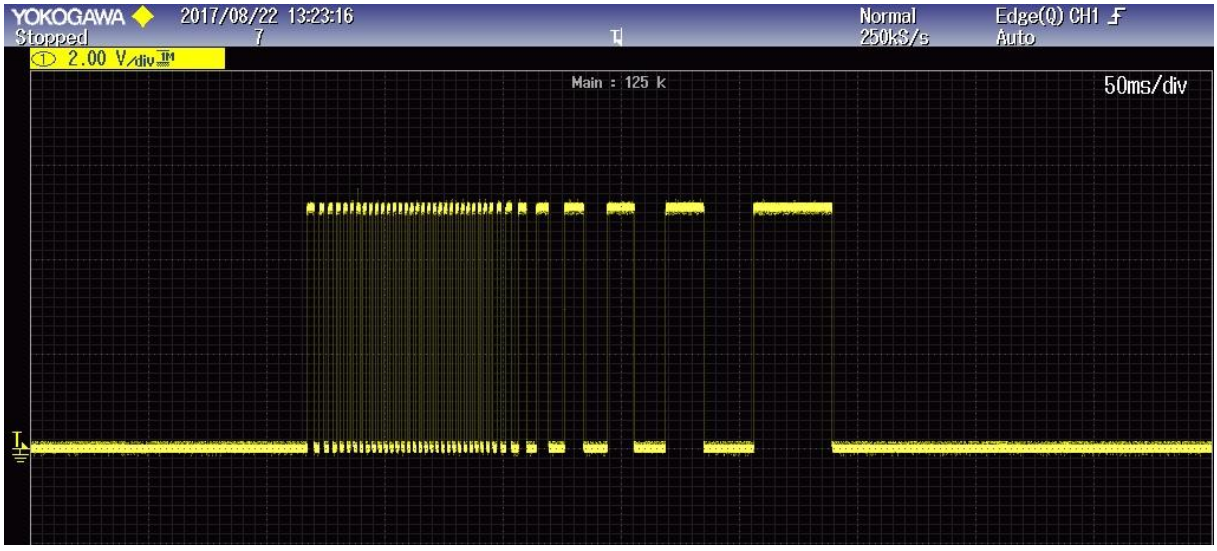

Abbildung 38 - Oszilloskop: 30 erwartete Pulse, PWM = 100 (schnell)

Der letzte durchgeführte Test mit seinen 38 gemessenen Pulsen zeigt, dass bei einer 100%igen Ausnutzung der Maximalgeschwindigkeit, der Wert von 30 Pulsen nicht erreicht werden kann. Mit steigender Geschwindigkeit nähert man sich den 30 Pulsen lediglich an.

Anhand der drei aufgenommen Bilder durch das Oszilloskop wird deutlich, dass die letzten aufgenommenen Pulse eine längere Zeitdauer aufweisen. In dem Grundlagenkapitel bezüglich der Encoder wurde bereits erwähnt, dass durch ein längeres Abdunkeln des optischen Sensors längere Pulse erzeugt werden. Die Vermutung liegt nahe, dass das Aufrufen der stop-Funktion des Motors im Quellcode kein abruptes Stoppen hervorruft. Da die Messungen im Bereich von Millisekunden stattfinden, kann es sein, dass in der Anwendung, sobald die Bedingung von 30 Pulsen erreicht ist, einige Millisekunden verstreichen bis die stop-Funktion aufgerufen wird.

Damit das Projekt noch zu einem Abschluss kommt und Streckenlängen durch den Benutzer eingegeben werden können, werden die Pulsabweichungen vorerst hingenommen.

Da bei einer Ausreizung der maximalen Geschwindigkeit die annehmbarsten Pulswerte geliefert werden, werden für diese Geschwindigkeit bestimmte Pulswerte ermittelt, welche vordefinierten Streckenlängen entsprechen. Der Benutzer soll die Möglichkeit haben, bei einer Vor- oder Rückwärtsbewegung zwischen verschiedenen Streckenlängen wählen zu können.

Die Hauptklasse "robot.py" wird dazu verwendet, um herauszufinden wie viele Pulse 10 cm entsprechen. In dem LENGTH-Set wird ein willkürlicher Wert für 10 cm gewählt. Der Roboter wird auf einem glatten Untergrund abgesetzt. Neben dem Roboter und auf dem Boden ausgebreitet befindet sich ein Zollstock. Die Mitte des linken, vorderen Rads (Motor 1) dient dem Abgleich mit dem Zollstock.
Die Anwendung "robot.py" wird gestartet. Es wird die Vorwärtsbewegung, die Geschwindigkeit schnell (PWM = 100) und die Streckenlänge 10 cm gewählt.
Nach der Testfahrt wird die zurückgelegte Strecke auf dem nebenliegenden Zollstock abgelesen und der anfänglich willkürliche Wert im LENGTH-Set wird angepasst, sodass sich 10 cm mehr und mehr genähert wird. Der Roboter wird auf seine Anfangsposition zurückgebracht. Der Test wird wiederholt.
Nach einigen Testfahrten kann der Pulswert von 95 ermittelt werden. Dieser Wert wird im LENGTH-Set für 10 cm übernommen.

Weitere Werte für die Streckenlängen von 20, 30, 40 und 50 cm werden benötigt. Anstatt willkürliche Werte zu wählen, werden die ermittelten 95 Pulse für die übrigen Werte aufaddiert. Beispiel: $30\ \text{cm} = 95\ \text{Pulse} \cdot 3 = 285\ \text{Pulse}$

Im LENGTH-Set werden folgende Pulswerte für die Streckenlängen notiert:

- 10 cm = 95 Pulse
- 20 cm = 190 Pulse
- 30 cm = 285 Pulse
- 40 cm = 380 Pulse
- 50 cm = 475 Pulse

Es folgen eine Reihe von Messungen, um die Genauigkeit der Streckenlängen zu ermitteln. Der Vorgang wird beibehalten. Der Roboter wird auf einem glatten Untergrund abgesetzt und die tatsächlich zurückgelegte Streckenlänge mit Hilfe des Zollstocks notiert.

Bevor die Messungen erfolgen, wird die derzeitige Spannung des 4 Zellen Akkumulators notiert, um zu überprüfen, wie stark der Akku während der Tests beansprucht wird.
Die Spannung beträgt 15,37 Volt.

Die schwarz hinterlegte, erste Tabellenzeile gibt die zu erreichende Streckenlänge an. (Sollwert) Die in der Spalte folgenden Werte sind die tatsächlich zurückgelegten Streckenlängen des Roboters. (Istwert)
Anhand der Tabellenbezeichnungen kann einerseits die Bewegungsrichtung und andererseits die Geschwindigkeit anhand der PWM abgelesen werden.

10 cm	20 cm	30 cm	40 cm	50 cm
10,3 cm	20,0	31,0	40,8	51,0
9,8 cm	19,8	30,5	41,0	50,5
10,0 cm	20,0	30,0	40,5	50,5
10,0 cm	20,5	30,5	40,5	50,8
10,3 cm	20,0	30,5	40,5	51,0

Tabelle 8 - Auswertung Streckenlänge - Vorwärtsbewegung, PWM = 100 (schnell)

10 cm	20 cm	30 cm	40 cm	50 cm
11,0	21,0	31,5	41,5	51,5
10,8	20,8	31,3	41,3	52,0
10,8	21,0	31,5	41,3	52,3

Tabelle 9 - Auswertung Streckenlänge - Vorwärtsbewegung, PWM = 50 (mittel)

10 cm	20 cm	30 cm	40 cm	50 cm
11,5	22,5	32,2	42,8	52,3
11,7	21,8	32,3	42,3	52,8
11,5	22,2	32,5	42,5	53,0

Tabelle 10 - Auswertung Streckenlänge - Vorwärtsbewegung, PWM = 10 (langsam)

10 cm	20 cm	30 cm	40 cm	50 cm
11,5	21,7	32,0	42,2	52,2
10,7	21,2	32,0	41,5	51,8
11,5	21,2	32,0	42,0	51,7

Tabelle 11 - Auswertung Streckenlänge - Rückwärtsbewegung, PWM = 100 (schnell)

Tabelle 8 zeigt, dass die anfängliche Überlegung, 95 Pulse für die längeren Strecken einfach aufzuaddieren, sich bewährt hat. Es tritt eine Abweichung von maximal einem Zentimeter auf. Durch eine Feinjustierung der jeweiligen Pulse könnte ein noch besseres Ergebnis erzielt werden. Dabei sollte jedoch berücksichtigt werden, dass die gemessenen Werte der übrigen Tabellen 9, 10 und 11 mit beeinflusst werden.
Durch die Aufnahmen des Oszilloskops war bereits bekannt, dass niedrige Geschwindigkeiten dazu führen, dass vermehrt Encoderpulse generiert werden, sodass längere Strecken zurückgelegt werden. Anhand der Tabellen 9 und 10 wird dies ersichtlich. Umso niedriger die Geschwindigkeit, desto größer die Abweichung vom Sollwert.
Fragwürdig ist es warum schnelle Rückwärtsbewegungen Abweichungen von über zwei Zentimetern aufweisen.

Der letzte Test der durchgeführt wird, befasst sich mit der Rotation des Roboters, damit ein Winkel von 45° nach links und rechts erreicht wird. Es müssen alle drei Motoren in die gleiche Richtung drehen, damit eine Rotation erfolgt.
Durch die Ungenauigkeiten der Encoderpulse, bei niedrigen Geschwindigkeiten, erfolgt eine Drehung um 45° ausschließlich mit einem PWM-Wert von 100. Somit stehen dem Benutzer im Hauptprogramm keine Auswahlmöglichkeiten zur Verfügung, die Geschwindigkeit bei einer Rotation von 45° zu beeinflussen.
Der Test wird auf die gleiche Art durchgeführt, wie es bereits bei der Ermittlung der Streckenlänge geschehen ist.
Der Roboter wird auf einem glatten Untergrund positioniert und nacheinander der günstigste Wert ermittelt, der sich bei einem Winkel von 45° einstellt. Der Unterschied zum Test davor, liegt darin, dass auf einen Zollstock verzichtet wurde. Per Augenmaß werden Winkelabweichungen verbessert. Nacheinander werden im Hauptprogramm 45° ausgewählt, bis der Roboter eine 360° Umdrehung absolviert. Es werden 90 Pulse als Streckenlänge für eine 45° Drehung identifiziert.
Der Test verläuft erfolgreich.

Die Spannung des 4 Zellen Akkumulators weist am Ende aller Test eine Spannung von 15,31 Volt auf. Während der ganzen Testphase ist die Spannung des 4 Zellen Akkumulators um 0,06 Volt gesunken.

Für zukünftige Betrachtungen und weitere Auswertungen, bezüglich der aufgetretenen Ungenauigkeiten der Encoderpulse, könnten die folgenden Oszilloskopaufnahmen dienlich sein. Die Aufnahmen zeigen einzelne Pulse bei unterschiedlichen Geschwindigkeiten.

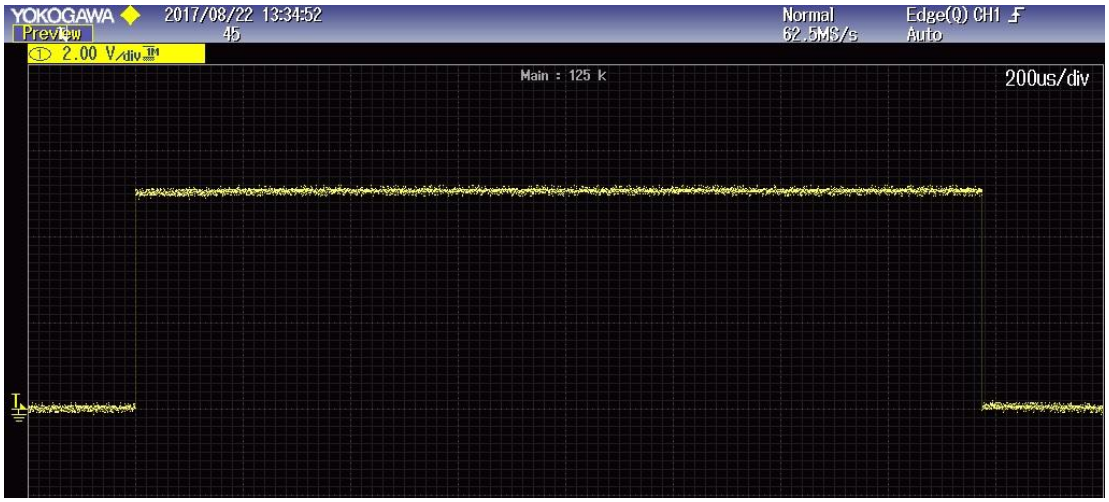

Abbildung 39 - Oszilloskop: einzelner Encoderpuls, PWM = 10 (langsam)

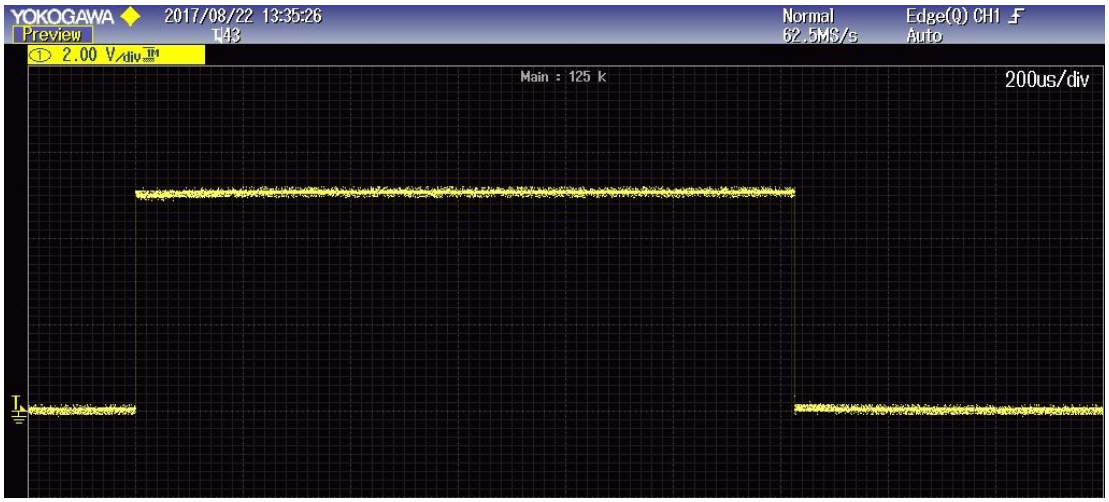

Abbildung 40 - Oszilloskop: einzelner Encoderpuls, PWM = 50 (mittel)

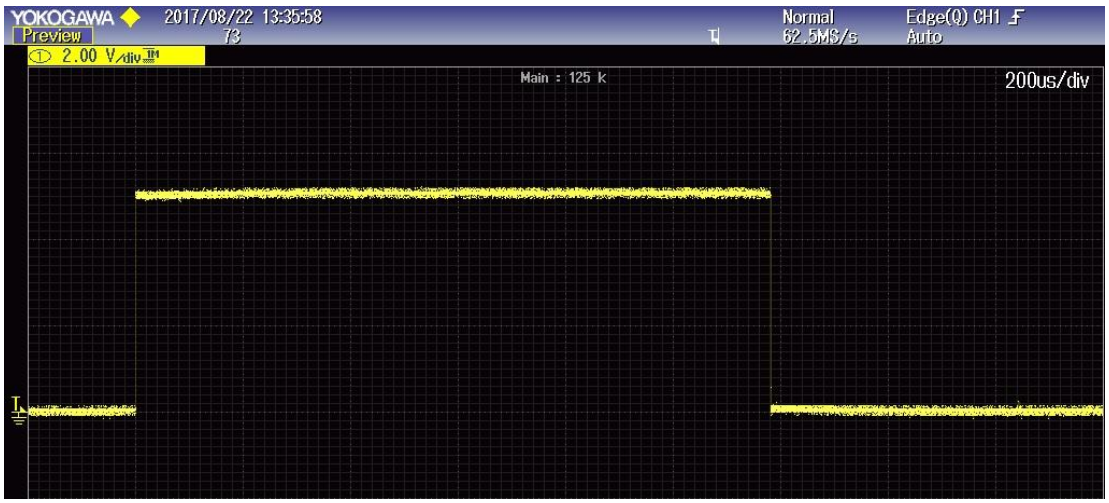

Abbildung 41 - Oszilloskop: einzelner Encoderpuls, PWM = 100 (schnell)

7. Schlussbetrachtung

7.1. Erreichter Stand

Anhand der Kriterien, die im Pflichtenheft in Kapitel 4.1 aufgeführt werden, wird der erreichte Stand aufgezählt.
Die Omnirad Plattform wurde aufgebaut und mit allen in den Lösungsansätzen aufgeführten Hardwarekomponenten bestückt.
Der Raspberry Pi wurde mit einer eigenen Versorgungsspannung versehen, die über einen Abwärtswandler angepasst werden musste.
Die drei Motoren wurden über zwei Motorentreiber mit einer eigenen Versorgungsspannung versehen. Über die Motorentreiber wurde für jeden einzelnen Motor eine Verbindung zum Raspberry Pi erstellt, sodass die Motoren über den Raspberry Pi individuell angesteuert werden konnten.
Für eine erfolgreiche Ansteuerung musste der Raspberry Pi mit einem Betriebssystem ausgestattet werden, welches im Anschluss noch konfiguriert wurde.
Für eine gezielte Fortbewegung des Roboters durch Benutzereingaben musste eine Anwendung programmiert werden, die Geschwindigkeiten, Streckenlängen, Richtungen und Benutzereingaben ermöglicht.
Alle genannten Kriterien sind in den Musskriterien des Pflichtenhefts aufgeführt. Sie konnten allesamt während der Bearbeitungszeit des Projekts realisiert werden.

Eines der wünschenswerten Kriterien sah vor, den Raspberry Pi in das WLAN-Netz der Hochschule einzugliedern. Für gewöhnlich besteht die Eingliederung in ein WLAN-Netz beim Raspberry Pi aus fünf Zeilen der wpa_supplicant.conf Datei. Da die Hochschule nur authentifizierten Benutzern einen Zugang gewährt, gestaltete sich die Aufgabe aufwendiger.
Internetrecherchen erzielten anfänglich keine Erfolge, bis eine Anleitung auf der eigenen Hochschulseite ausfindig gemacht werden konnte. Wäre eine Eingliederung in das Hochschulnetz erfolglos geblieben, hätte die Möglichkeit bestanden, den Raspberry Pi mit dem eigenen Smartphone eine Verbindung ins Internet zu ermöglichen.
Ein weiteres Wunschkriterium sah vor mit dem Roboter geometrische Figuren abfahren zu können. Sobald das Hauptprogramm gestartet wird, werden dem Benutzer keine Möglichkeiten angeboten direkt eine geometrische Figur abzufahren. Der Benutzer kann aber mit Hilfe seines Verstandes durch die gegebenen Auswahlmöglichkeiten, einer geraden Strecke und des 45° Winkels, selber geometrische Figuren bestimmen. Somit kann man sagen, dass das Wunschkriterium erfüllt wurde.
Das nächste Wunschkriterium sah vor, dass die Programmierung der Anwendung mit der Programmiersprache C umgesetzt werden sollte. Aus einem unabhängigen, aber ähnlichem Projekt konnten bereits Erfahrungen mit der Programmiersprache Python gesammelt und gute Erfolge erzielt werden. Es kamen in dem vorherigen Projekt allerdings keine Messungen im Millisekundenbereich vor. Die Sprache C wäre an dieser Stelle durchaus die sinnvollere Variante gewesen, da die Programmiersprache C in der Hardwareprogrammierung eingesetzt wird. Durch die straff ausgelegte Bearbeitungsdauer der Bachelorarbeit, konnte dieses Wunschkriterium nicht umgesetzt werden.

Aufgrund der verbliebenen Zeitdauer konnte nicht sichergestellt werden, ob das Testen der programmierten Anwendungen stattfinden wird. In Kapitel 6 konnten allerdings noch viele Tests durchgeführt werden, damit es zu einer präzisen Bewegung des Roboters kommt.
Allerdings hätte noch ein genaueres Auswerten der Encoderpulse stattfinden sollen, damit die Ursache festgestellt ermittelt werden könnte, warum bei niedrigen Geschwindigkeiten, so hohe Ungenauigkeiten auftreten.
Eine Vermutung wäre, dass das Betriebssystem des Raspberry Pis kein Echtzeitsystem ist und dementsprechend die Abtastraten nicht genau genug sind. Die Vermutung sollte in zukünftigen Realisierungen genauer betrachtet werden.

Aus den Wunschkriterien konnten drei von vier Kriterien umgesetzt werden.

Die Abgrenzungskriterien wurden im Vorfeld als nichterreichbar deklariert.
Diese sahen unter anderem vor, den Roboter autonom agieren zu lassen. Wie man dem Umfang der Bachelorarbeit entnehmen kann, reicht es bereits aus einen Roboter überhaupt erst ein mal fahrtüchtig machen zu können. Ein autonom agierender Roboter müsste über weitere Komponenten verfügen, beispielsweise eine Kamera oder einen Ultraschallsensor, damit die Umgebung erfasst werden kann. Weitere Anwendungen müssten programmiert werden, damit eine Interaktion mit der Umgebung, beispielsweise einem Ausweichen eines Objekts, stattfinden kann. Aufgrund der begrenzten Zeitdauer hätte dies unmöglich realisiert werden können.

Das letzte Abgrenzungskriterium, den Quellcode in unterschiedliche Klassen aufzuteilen, wäre durchaus sinnvoll gewesen. Zum einen dient die Aufteilung der Funktionen auf mehrere Klassen der Übersicht und zum anderen können die Klassen dynamisch eingesetzt werden. Sollte eine Klasse beispielsweise in einem anderen Projekt eingesetzt werden, muss diese nicht erst umständlich aus einer großen zusammenhängenden Klasse herausgesucht werden.

7.2. Fazit & Ausblick

Das Bachelorthema zu wählen, um einen eigenen Roboter nach seinen Vorstellungen zu entwickeln, zu bauen und zu programmieren, sodass er am Ende fahrtüchtig ist und gesteuert werden kann, war eine sinnvolle Entscheidung. Im Verlauf der Realisierung wurde neues Wissen angeeignet, dass in der Zukunft sicherlich von Vorteil sein wird.
Im Vorfeld mussten Überlegungen angestellt werden, damit am Ende ein sinnvolles Projekt realisiert werden konnte. Vieles aus dem im Studium kennengelernten wurde angewendet und half bei der Umsetzung den fertigen Roboter zu entwickeln.

Der Grund sich für LiPo Akkus zu entscheiden war, dass sie über eine hohe Energiedichte verfügen. Es sollte in der Zukunft darauf geachtet werden, den Akkumulator während des Betriebs zu überprüfen, sodass es zu keiner Tiefentladung kommt. Dies sollte durch eine zukünftige Erweiterung implementiert werden.
Die Akkumulatoren mussten während der ganzen Bearbeitungsphase kein einziges mal aufgeladen werden. Testweise wurde der 4 Zellen Akkumulator aufgeladen um das von der Universität zur Verfügung gestellte Ladegerät zu prüfen. Das Nutzerhandbuch weist den Anwender daraufhin, einen Akkumulator nur unter Aufsicht zu laden. Während des Aufladevorgangs kam es beim 4 Zellen Akkumulator zu Schwierigkeiten. Das beiliegende Netzteil konnte nach einer gewissen Zeit von einigen Minuten nicht länger die genügende Spannung aufbringen um den Akku zu laden. Es wird dazu geraten ein passendes Netzteil zu suchen.

Eine genaue Auswertung der genutzten Spannungen an den Motoren sollte in zukünftigen Realisierungen analysiert werden. Da die genaueste Ansteuerung für eine Streckenlänge eine 100%ige PWM vorsieht, sollte überprüft werden, dass die Motoren nicht über einen längeren Zeitraum übersteuert werden, da dies Schäden verursachen könnte.

Es wird dazu geraten, das Problem der Abtastrate des Raspberry Pis zu untersuchen. Per Ausschlussverfahren könnte im ersten Schritt die Anwendung in C programmiert werden. Ein anschließender Test würde aufzeigen, ob der Einsatz der Programmiersprache C bessere Encoderpulse liefert. Im nächsten Schritt könnte das jetzige Betriebssystem durch ein eigenes Betriebssystem ersetzt werden. Yocto würde sich dafür anbieten. Im nächsten Schritt könnte der Kernel gepatcht werden, sodass schnellere Abtastraten ermöglicht werden.

Durch des Raspberry Pis Vielfalt an Anschlussmöglichkeiten sind Erweiterungen kaum Grenzen entgegen zu setzen, mit denen der Roboter in der Zukunft weiter bestückt werden kann. Durch die großzügige Auswahl des Akkumulators können weitere Hardwareelemente mit Strom und Spannung versorgt werden.
Es könnte beispielsweise eine Platine entwickelt werden, die alle benutzen Hardwareelemente vereint und einfach auf den GPIO-Pins aufgesteckt wird.
Abschließend kann gesagt werden, dass bei der Wahl der Hardwarekomponenten eine gute Wahl getroffen wurde, da es während der Benutzung des Roboters zu keinerlei Problemen kam.

Quellenverzeichnis

[1] Dembowski, Klaus (2013): Raspberry Pi - Das Handbuch, Springer Vieweg
[2] Engelhardt, E.F. (2015): Coole Projekte mit Raspberry Pi, 4. Auflage, Franzis Verlag
[3] Follmann, Rüdiger (2014): Das Raspberry Pi Kompendium. Springer Vieweg
[4] Jesse, Ralf (2016): Embedded Linux mit Raspberry Pi und Co., 1. Auflage, mitp
[5] Kofler, Kühnast, Scherbeck (2016): Raspberry Pi - Das umfassende Handbuch, 3. Auflage, Rheinwerk Computing
[6] Kampert, Daniel (2015): Raspberry Pi - Der praktische Einstieg, 2015, Rheinwerk Computing
[7] http://t3n.de/news/computer-raspberry-pi-c64-806761/ [4.7.2017]
[8] Raspberry Pi Geek (05-06/2017): RasPi Zero W, S.67
[9] https://de.wikipedia.org/wiki/Mikroprozessor [4.7.2017]
[10] https://nodna.de/3WD-48mm-Omni-Wheel-Platform-Chassis#&gid=1&pid= [4.7.2017]
[11] http://wiki.mikrokopter.de/LiPo-Grundlagen [4.7.2017]
[12] https://www.geras-it.de/step-down-spannungsregler-modul-2-3a-1-5-34v-einstellbar-lm2596-fuer-z-b-arduino/a-249/ [6.7.2017]
[13] http://www.elektronikinfo.de/strom/kondensatoranwendung.htm [10.7.17]
[17] - https://www.raspberrypi.org/documentation/hardware/raspberrypi/power/README.md [10.7.17]
[18] https://github.com/FrankBau/raspi-repo-manifest/wiki/power-consumption [10.7.17]
[19] http://www.akku-abc.de/akku-vergleich.php [13.07.2017]
[20] Nauth, Peter (2005): Embedded Intelligent Systems, De Gruyter Oldenbourg
[21] http://www.itwissen.info/Tiefentladung-deep-discharche.html [13.07.2017]
[22] https://www.elektronik-kompendium.de/sites/bau/0810281.htm [13.07.2017]
[23] https://patona.de/blog/akkus/lithium-ion-vs.-lithium-polymer-akku-die-unterschiede [13.07.2017]
[24] https://de.wikipedia.org/wiki/Lithium-Polymer-Akkumulator [13.07.2017]
[25] https://de.wikipedia.org/wiki/Lithium-Ionen-Akkumulator [13.07.2017]
[27] https://de.wikipedia.org/wiki/Selbstentladung [13.07.2017]
[29] http://www.webopedia.com/TERM/C/cross_compiler.html [14.07.2017]
[30] https://github.com/FrankBau/raspi-repo-manifest/wiki/Yocto-Project [14.07.2017]
[35] https://pinout.xyz/pinout/# [14.07.2017]
[36] Babiel, Gerhard (2009): Elektrische Antriebe in der Fahrzeugtechnik Lehr- und Arbeitsbuch, 2. Auflage, Vieweg + Teubner Verlag, S. 23
[38] Moeller, Franz (1936): Elektromotor und Arbeitsmaschine, Springer Verlag, S. 1
[39] https://www.youtube.com/watch?v=6kJuuPxo4Q0 [15.07.2017]
[41] Stölting, Hans-Dieter (2011): Handbuch Elektrische Kleinantriebe, 4. Auflage, Carl Hanser Verlag GmbH & Co. KG
[42] https://www.raspberrypi.org/forums/viewtopic.php?f=66&t=133691 [16.07.2017]
[46] http://elinux.org/RPi_Distributions#Comparison [16.07.2017]
[55] http://www.dummies.com/computers/raspberry-pi/top-10-programming-languages-ported-to-the-raspberry-pi/ [26.7.2017]

[57] http://www.dynapar.com/Technology/Encoder_Basics/Quadrature_Encoder/ [29.7.17]
[58] https://www.elektronik-kompendium.de/sites/com/0902081.htm [30.7.17]
[59] https://www.elektronik-kompendium.de/sites/com/0309221.htm [02.08.2017]
[60] https://de.wikipedia.org/wiki/Raspberry_Pi#Namensgebung_und_Logo [04.08.2017]
[61] https://de.wikipedia.org/wiki/H%C3%B6here_Programmiersprache [04.08.2017]
[62] https://www.raspberrypi.org/app/uploads/2012/02/BCM2835-ARM-Peripherals.pdf [15.08.2017]
[63] https://de.wikipedia.org/wiki/Pulsweitenmodulation [15.08.2017]
[64] https://www.rugged-circuits.com/the-motor-driver-myth/ [15.08.2017]

Anhang

Der Bachelorarbeit ist eine CD beigelegt. Folgende Elemente sind dort in Ordnern zu finden:

- Bachelorarbeit: Die Bachelorarbeit in PDF- und DOCX-Format
- Anwendung: Der Quellcode der Hauptanwendung "robot.py"
    - Test: "encoder.py", "encoder_pwm.py", "input.py", "motor.py"

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass

- ich die vorliegende wissenschaftliche Arbeit selbstständig und ohne unerlaubte Hilfe angefertigt habe,
- ich andere als die angegebenen Quellen und Hilfsmittel nicht benutzt habe,
- ich die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe,
- die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfbehörde vorgelegen hat.

Berlin, den _____ _____

(Unterschrift)